Diameter-constrained network reliability : properties and computation

Pablo Enrique Sartor del Giudice

▶ To cite this version:

HAL Id: tel-00945265

https://theses.hal.science/tel-00945265

Submitted on 12 Feb 2014

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

en cotutelle internationale avec
PEDECIBA - Université de la République, Uruguay

pour le grade de
DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique
Ecole doctorale Matisse

présentée par

Pablo Enrique SARTOR DEL GIUDICE

préparée à l'unité de recherche UMR 6074 - IRISA
Institut de Recherche en Informatique et Systèmes Aléatoires
Equipe de Recherche DIONYSOS

Propriétés et
méthodes de calcul
de la fiabilité
diamètre-bornée
des réseaux

**Thèse soutenue à Montevideo
le 18 décembre 2013**

devant le jury composé de :

Kishor S. TRIVEDI
Directeur de Département, Duke University, USA / rapporteur

Guillermo DURÁN
Directeur d'Institut, Université de Buenos Aires, Argentine / rapporteur

Sergio NESMACHNOW
Professeur, PEDECIBA Informatique, Uruguay / examinateur

Reinaldo VALLEJOS
Chercheur, Université Technique Federico Santa María, Chili / examinateur

Gerardo RUBINO
Directeur de Recherche, INRIA Rennes, France / examinateur

Bruno TUFFIN
Directeur de Recherche, INRIA Rennes, France / examinateur

À José Nasazzi.

Remerciements

Cette thèse, ainsi que toute les réalisations professionnelles et académiques qui peuvent se produire dans l'avenir, est une conséquence directe de la vision et de l'éducation que j'ai reçue de mes parents. À eux, donc, bonne part du mérite.

Mes derniers remerciements, les plus spéciaux, vont à Vanessa. Elle a été une partie essentielle de ce processus à tout moment. J'ai vu son véritable bonheur à chaque cours terminé, chaque article accepté, chaque pas en avant que j'ai donné, et je sais bien comment elle a caché le fardeau de mes voyages, mon temps limité, ma fatigue ... Vraiment, c'est ta réussite aussi, et je t'en serai éternellement reconnaissant.

Abstract

Consider a communication network whose links fail independently and a set of sites named terminals that must communicate. In the classical stochastic static model the network is represented by a probabilistic graph whose edges occur with known probabilities. The classical reliability (CLR) metric is the probability that the terminals belong to a same connected component. In several contexts it makes sense to impose the stronger condition that the distance between any two terminals does not exceed a parameter d. The probability that this holds is known as the diameter-constrained reliability (DCR). It is an extension of the CLR. Both problems belong to the $\mathcal{N}$P-hard complexity class; they can be solved exactly only for limited-size instances or specific network topologies.

In this thesis we contribute a number of results regarding the problem of DCR computation and estimation. We study the computational complexity of particular cases parameterized by the number of terminals, nodes and the parameter d. We survey methods for exact computation and study particular topologies for which computing the DCR has polynomial complexity. We give basic results on the asymptotic behavior of the DCR when the network grows as a random graph. We discuss the impact that the diameter constraint has in the use of Monte Carlo techniques. We adapt and test a family of methods based on conditioning the sampling space using structures named d-pathsets and d-cutsets. We define a family of performability measures that generalizes the DCR, develop a Monte Carlo method for estimating it, and present numerical evidence of how these techniques perform when compared to crude Monte Carlo. Finally we introduce a technique that combines Monte Carlo simulation and polynomial interpolation for reliability metrics.

Résumé

Soit un réseau comprenant des lignes de communication qui échouent indépendamment, dans lequel tous ou certains sites, appelés terminaux, doivent être capables de communiquer entre eux. Dans le modèle stochastique statique classique le réseau est représenté par un graphe probabiliste dont les arêtes sont présentes selon des probabilités connues. La mesure de fiabilité classique (CLR) est la probabilité que les terminaux appartiennent à la même composante connexe. Dans plusieurs contextes il est utile d'imposer la condition plus forte que la distance entre deux terminaux quelconques soit bornée supérieurement par un paramètre d. La probabilité que ća se produise est connue comme la fiabilité diamètre-bornée (DCR). Il s'agit d'une extension de la CLR. Les deux problèmes appartiennent à la clase $\mathcal{N}$P-difficile de complexité; le calcul exact n'est possible que pour les instances de taille limitée ou topologies spécifiques.

Dans cette thèse, nous contribuons des résultats concernant le problème du calcul et l'estimation de la DCR. Nous étudions la complexité de calcul de cas particuliers,

paramétré par le nombre de terminaux, nœuds et le paramétre d. Nous passons en revue des méthodes pour le calcul exact et étudions des topologies particulières pour lesquelles le calcul de la DCR a une complexité polynomiale. Nous présentons des résultats de base sur le comportement asymptotique de la DCR lorsque le réseau se développe comme un graphe aléatoire. Nous discutons sur l'impact de la contrainte de diamétre dans l'utilisation des techniques de Monte Carlo, et adaptons et testons une famille de méthodes basées sur le conditionnement de l'espace d'échantillonnage en utilisant des structures nommées d-pathsets et d-cutsets. Nous définissons une famille de mesures de performabilité qui généralise la DCR, développons une méthode de Monte Carlo pour l'estimer, et présentons des résultats experimentaux sur la performance de ces techniques Monte Carlo par rapport à l'approche naïve. Finalement, nous proposons une nouvelle technique qui combine la simulation Monte Carlo et l'interpolation polynomiale pour les mesures de fiabilité.

Contents

List of Figures

Notations and glossary

Standard mathematical symbols

$\mathbb{Z}$	Set of integers
$\mathbb{R}$	Set of real numbers
$\Pr(\cdot)$	Probability
$\mathrm{E}(\cdot)$	Expected value
$\mathrm{Var}(\cdot)$	Variance

Graphs

$G = (V, E)$	Undirected, simple graph
V	Node set, representing the sites of the network
$E = \{\{u, v\} : u, v \in V \wedge u \neq v\}$	Edge set, representing the existing links between sites
$n = |V|$	Number of nodes
$m = |E|$	Number of edges
uv	Simple notation for edge $\{u, v\}$, mainly used for subindexing
$u_1 - u_2 - \cdots - u_\ell$	Path with node sequence $u_1, u_2, \ldots, u_\ell$

Reliability models

$G = (V, E, \vec{r})$	Undirected, simple probabilistic graph
$\vec{r} = (r_e)_{e \in E}$	Vector of edge reliabilities
r_e	Reliability of edge e
$K \subseteq V$	Set of terminal nodes
$R(G, K)$ or $R_K(G)$	Classical K-reliability
$R(G, K, d)$ or $R_K(G, d)$	Diameter-constrained d-K-reliability

Abbreviations

CLR	Classical reliability (the problem, the measure or the context)
DCR	Diameter-constrained reliability (idem)
K-CLR, K-DCR	CLR and DCR for terminal set K
d-DCR	DCR with distance parameter d
d-K-DCR	DCR with distance parameter d and terminal set K
DFS	Depth-first search
BFS	Breadth-first search

Résumé en Français

Introduction

Cette thèse porte sur certaines propriétés qui sont définies pour les réseaux de télécommunication. Ici, nous travaillons avec des modèles de réseaux où les composants élémentaires peuvent échouer selon un certain modèle probabiliste que l'on suppose connu. Même si nous parlons en général de réseaux de télécommunication, les modèles, mesures, résultats théoriques et les algorithmes que nous utilisons et introduisons peuvent être appliqués à d'autres contextes, comme les réseaux de transport, les architectures informatiques parallèles, les réseaux électriques, le contrôle de systèmes, la propagation du feu, etc. La raison pour laquelle nous nous concentrons sur les réseaux de télécommunications, c'est qu'ils ont été le principal moteur de la recherche et développement de mesures de fiabilité, le sujet principal de cette thèse.

Ici, nous commençons par discuter de plusieurs types de propriétés des réseaux qui peuvent être englobées sous le concept général connu comme " fiabilité ". Ensuite, nous nous intéressons spécifiquement au rôle joué par l'évaluation des indicateurs de fiabilité (network reliability) ; cette thèse est principalement concerné sur l'une de ces mesures, connue sous le nom de " fiabilité diamètre-bornée ". Une fois cette introduction contextuelle est donnée, nous présentons nos contributions et la liste correspondante des articles publiés ou acceptés.

Propriétés de fiabilité

Un grand nombre de mesures pour quantifier le service que les réseaux de télécommunication offrent aux utilisateurs peut être conçu. Certaines d'entre elles peuvent être simplement exprimées comme indicateurs binaires (par exemple, le réseau est soit " active ou hors service "). D'autres mesures peuvent avoir un caractère non - binaire (par exemple, le débit du réseau). On peut même penser à des mesures très complexes définies par exemple en attribuant des poids différents à différentes situations du réseau et en tenant compte de la vitesse, perte de paquets, délais, etc. En général, nous ferons référence comme performance du réseau au niveau qu'un certain réseau atteint en ce qui concerne certains métriques quantitatives bien définies. Durant les premières décennies du développement des réseaux de télécommunication, l'accent a été mis sur les mesures qui manifestent la capacité d'un réseau de continuer à fonctionner [1]. Comme nous le

verrons plus tard, en fonction du modèle et de la définition, ces mesures sont appelées disponibilité et fiabilité (*availability* et *reliability*). Quand les premiers réseaux à commutation de paquets ont été déployés sur les lignes téléphoniques, les principaux efforts visaient à maintenir les lignes (et la topologie sous-jacente du réseau) opératives. Dans ce contexte, une simple différenciation entre les états opérationnel et hors service était suffisante, donc motivant intenses efforts de recherche dans le domaine de l'analyse de la fiabilité. L'amélioration des technologies, et la progressive réduction du coût des dispositifs qui permettait l'introduction de redondance, a conduit à la conception et déploiement des réseaux qui étaient beaucoup plus fiables en termes de connectivité. Au même temps, avec plus trafic de données et applications plus complexes, le focus s'est déplacé de la connectivité aux indicateurs de performance plus complexes, capables de distinguer entre plusieurs niveaux de service. Puis, dans les années 80 et 90, il est venu le temps pour le déploiement massif de lignes à fibres optiques très fiables. L'extraordinaire capacité de ces lignes rend possible qu'un réseaux ayant un seul chemin pour relier deux points offre le niveau de performance requis, à condition que ses lignes soient opérationnelles. Dans ce contexte, les modèles de fiabilité ont récupéré leur pertinence. Actuellement, la connectivité a récupéré aussi la pertinence principalement à cause du déploiement massif des réseaux sans fil, dont le liens point-à-point sont moins fiables pour des raisons telles que les facteurs environnementaux et d'économie d'énergie. On peut dire que l'accent historiquement placé sur la connectivité a été nivelé par le concept plus global de performabilité (*performability*), qui englobe à la fois questions de fiabilité et performance [1]. Le service offert par les réseaux réels peut être affectée par plusieurs types d'échecs. Des erreurs humains peuvent entraîner des dysfonctionnements, coupures de service, le déploiement des routes erronées pour paquets de commutation, etc. Les humains peuvent également provoquer des dommages intentionnels (sabotage). Des défauts de conception peuvent entraîner la perte de données, des performances insuffisantes, des coupures de courant, saturation de liens saturation, ainsi que d'autres problèmes plus subtils comme le *jitter*. Les facteurs environnementaux peuvent causer des interférences et du bruit dans des lignes filaires, atténuation de signal, des interruptions intermittentes dans les liaisons sans fil, ainsi que des dommages plus généraux en cas d'inondations, de tremblements de terre et d'autres situations semblables. L'usure aléatoire peut provoquer le dysfonctionnement ou stoppage de l'activité d'un composant. Lors de la conception des réseaux de communication, les ingénieurs doivent anticiper les sources possible d'échec et de dégradation de la performance. Une fois un modèle mathématique pour le réseau a été adopté, deux approches principaux peuvent être suivies, souvent en collaboration itérative. D'une part, le réseau peut être conçu avec le but d'atteindre certaines propriétés et seuils de performance désirés ; les algorithmes de conception, les topologies utilisés, etc. sont telles que la réalisation de ces propriétés est garantie. D'autre part il existe des techniques d'analyse de réseau qui, étant donné un certain réseau, permettent le calcul ou estimation de mesures qui quantifient les propriétés dont on est intéressé. Par exemple, on peut d'abord analyser un réseau existant, et une fois que ses propriétés actuelles sont connus, on peut essayer de les améliorer grâce à des techniques de conception, jusqu'à ce que le réseau atteigne les niveaux souhaités, plus exigeants que ceux atteints par le

réseau initial. Cette approche hybride peut également être suivie quand les techniques de conception proposent des probables améliorations (techniques heuristiques). En ces cas, il faut effectuer des analyses sur le réseau augmenté afin de déterminer si les propriétés souhaitées ont été atteintes. Dans cette thèse, nous nous intéressons aux techniques d'analyse des réseaux.

Lors de la conception de systèmes avec certains seuils de fiabilité, l'effort peut être orienté dans deux directions principales (et compatibles). D'une part nous pouvons essayer d'améliorer la fiabilité des composants élémentaires. Cela peut être considéré comme une approche " d'optimisation locale ". D'autre part nous pouvons introduire des composants redondants dans le système. Cela peut être vu comme une approche " systémique ". Il y a plusieurs raisons pour lesquelles la deuxième approche a traditionnellement joué un rôle relativement important dans la conception de réseaux. Par exemple, cela est particulièrement vrai par rapport aux principes de fabrication de qualité totale comme celles du " Toyota Production System "[2] ou Six-Sigma [3], où le focus est placé dans la production d'unités fiables à travers de tâches atomiques qui soient autant libres d'erreur que possible. Tout d'abord, il est très difficile de prévoir toutes les sources possibles d'échec qui menacent les composants du réseau. Deuxièmement, le majeur coût de déploiement entraîné par la redondance est justifiée normalement par les économies significatives obtenues en utilisant des composants de qualité ou capacité inferieures. Pensez par exemple au cas des clusters commerciaux d'équipement informatique, où des grandes économies sont réalisées à l'aide de serveurs basiques et identiques (*commodity servers*), malgré la nécessité de quelques unités supplémentaires pour avoir de la redondance.

Évaluation des mesures de fiabilité

Une famille de métriques développées au cours des 50 dernières années qui a été très étudiée est celle de la fiabilité. D'une manière générale, elles expriment la probabilité qu'un réseau qui peut être dans deux états du point de vue de l'utilisateur (opérationnel / échoué), ne tombe pas dans l'état d'échec lors d'un intervalle de temps donné (modèles dynamiques), ou à tout moment (modèles statiques) - voir la Section 2.2. Les réseaux de télécommunication physiques, indépendamment de leur taille et leur complexité, sont construites utilisant quelques composants élémentaires. La plupart des réseaux peuvent être bien représentés en fonction de deux grandes classes de composants : (i) des composants qui envoient, reçoivent, traitent et stockent des données (sites) et (ii) des composants qui emmènent les données d'un site à un autre site voisin (liens). Pour l'analyse de la fiabilité, ces réseaux sont souvent modélisés sous forme de graphes probabilistes (voir Section 2.1). Les échecs aléatoires de sites et liens sont modélisés comme des nœuds et des arêtes qui sont étiquetés avec leurs probabilités d'échec. Le complément de la fiabilité (*unreliability*), est alors la probabilité que se produise une combinaison de défauts élémentaires qui rende le réseau dans l'état d'échec. Des efforts considérables ont été consacrés à l'étude des mesures de fiabilité. D'une part, cela est dû à la pertinence que la connectivité avait notamment pour les réseaux réels qui ont été développés dans

la seconde moitié du siècle passé. D'autre part parce que, même avec de nombreuses hypothèses simplificatrices, les problèmes qui se posent sont complexes, difficiles et en général loin d'avoir des solutions connues satisfaisantes (voir Section 3.2). Au cours des dernières décennies, l'accent mis sur la connectivité a été complémenté par celui mis sur la performance, créant la notion de performabilité, qui englobe à la fois la fiabilité et la performance. Dans cette thèse, nous travaillons avec des mesures de fiabilité définies par un modèle stochastique statique qui est constitué d'un graphe probabiliste et un ensemble de nœuds terminaux que l'on appelle " distingués "ou " terminaux ". Dans le Chapitre 2, nous formalisons le modèle et les définitions. La fiabilité classique (CLR = Classical Reliability) est la probabilité que les terminaux soient connectés. Des résumés de littérature peuvent être trouvés dans [4, 5, 6]. Cette mesure a été généralisée [7] pour considérer non seulement la connectivité mais aussi la distance entre les terminaux, imposant un borne supérieur. Un résumé de littérature peut être trouvée dans [8]. La nouvelle mesure, appelée fiabilité diamètre-bornée (DCR = Diameter-constrained Reliability), permet modéliser des situations où l'on impose des limites au nombre acceptable de sauts que les paquets peuvent subir. Cette contrainte est pertinente dans plusieurs contextes qui sont de plus en plus importantes depuis les années 90. Par exemple, en applications de peer-to-peer (P2P) et voix sur IP (VoIP), la qualité perçue est affecté par la latence, déterminée à son tour par le nombre de liaisons parcourues par les paquets. La qualité se détériore à mesure que le réseau présente plus souvent des états avec haute distances entre terminaux. Dans les applications Web avec interfaces riches, la qualité perçue par l'utilisateur final est lié à la réactivité, où la latence détermine le délai entre les actions de l'utilisateur et leur effet sur l'interface de sortie.

L'hypothèse d'indépendance

La plupart des méthodes qui ont été développées pour le calcul exact, de bornes ou estimation de la fiabilité de réseaux assume que les composants échouent indépendamment. Cela est dû à plusieurs raisons. D'abord, comprendre et modéliser la dépendance des échecs est généralement une tâche très complexe. Deuxièmement, l'hypothèse d'indépendance simplifie considérablement le traitement mathématique du problème, grâce au fait que les probabilités peuvent être écrites facilement en utilisant des expressions factorisées . Troisièmement, plusieurs sources d'erreurs sont bien modélisés comme des événements indépendants. Un exemple typique est l'usure naturelle. Une autre famille est celle des erreurs humaines, comme par exemple la mauvaise configuration d'une interface ou la coupure accidentelle d'une ligne de fibre due aux activités de mouvement de terre dans un campus. La dépendance des échecs de composants peut être importante et donc non négligeable par exemple lorsque l'on considère les catastrophes naturelles (qui peuvent affecter à la fois plusieurs liens proches) et en particulier pour l'analyse de performabilité. Par exemple, l'échec de quelques liens provoque que le trafic qu'ils conduisaient transite par d'autres liens, rendant ces derniers plus susceptibles d'être surchargés et échouer. Dans cette thèse, sauf spécifiée localement, on assume que l'hypothèse d'indépendance est valide.

Definitions fondammentales

Systèmes multi-composants

Les réseaux de communication peuvent être considérés comme des systèmes multi-composants. En général, un tel système est intégré de plusieurs composants (M). Chaque composant peut être dans l'un de deux états possibles. Chaque combinaison des états du composant entraîne à la fois le système entier dans deux états aussi. L'un de ces états binaires est normalement dit opérationnel (*operational, up, working*); l'autre état binaire est normalement dit échoué (*failed, down*). Classiquement, les deux états sont représentés par les nombres entiers 1 et 0 respectivement. Un vecteur d'état, aussi dit une configuration du système, est un vecteur $\vec{x} = (x_1, \ldots, x_M) \in \{0,1\}^M$. La valeur de $x_i : i \in 1, \ldots, M$ indique l'état de la i-ème composante du système (suivant un certain ordre conventionnel). Comme nous l'avons dit, chaque configuration $\vec{x}$ entraîne un état 1 ou 0 pour le système. Par conséquent, il existe une fonction de structure Φ définie comme suit :

$$\Phi : \{0,1\}^M \to \{0,1\}$$

$$\Phi(\vec{x}) = \begin{cases} 1 & \text{si le système est opérationnel lorsque la configuration est } \vec{x}, \\ 0 & \text{si le système est échoué lorsque la configuration est } \vec{x}. \end{cases}$$

Observez qu'il y a 2^M fonctions différentes qui respectent cette définition. Celle qu'on utilise dépend de l'aspect spécifique de la situation réelle que l'on veut modéliser. Par exemple, deux systèmes à M composantes peuvent être telles que l'un est considéré opérationnel lorsque (et seulement lorsque) tous ses composants sont opérationnels, tandis que l'autre est dite opérationnel lorsque (et seulement lorsque) il y a au moins un composant qui fonctionne. En outre, il peut y avoir différent aspects à évaluer pour la même situation réelle, qui se traduisent par différents fonctions de structure. Considérons par exemple un bureau ayant M employés. On pourrait dire que le bureau " travaille "s'il y a au moins un employé présent. Alternativement, on pourrait aussi dire que le bureau " fonctionne "toutefois que la totalité de certains employés essentiels sont présents.

La fiabilité classique

Le modèle classique pour l'analyse de la fiabilité du réseau est un modèle statique stochastique où le réseau est représenté par un graphe probabiliste $G(V, E, \vec{r})$ avec $n = |V|$, $m = |E|$ et $\vec{r} = (r_e)_{e \in E}$ le vecteur de fiabilité des arêtes. Les nœuds modélisent les sites et les liens modélisent les lignes de communication. S'il existe au moins un lien qui est unidirectionnel, on doit utiliser un graphe orienté. Si toutes les liens sont bi-directionnels, un graphe non orienté peut être utilisé. Si plusieurs liens relient la même paire de sites, ils peuvent être remplacés dans le modèle par un lien unique dont la fiabilité est égal à la probabilité qu'au moins l'un de ces liens soient opérationnels (voir Section 5.1.1). Dans cette thèse, nous travaillons avec des réseaux dont les nœuds sont parfaits (ils n'échouent jamais, c'est à dire, leur fiabilité est égal à 1) et dont les nœuds ont des fiabilités connues

et échouent de manière indépendante. Nous allons donc modéliser le réseau comme un graphe probabiliste simple et non orienté. Une fois établi le modèle de graphe, la fonction de structure Φ doit être définie. Cela se fait selon une propriété de connexité du graphe ; la fiabilité du réseau est alors définie comme la probabilité que cette propriété soit vraie pour le graphe. La littérature classique distingue trois principaux problèmes d'analyse de fiabilité selon la définition precise de ladite propriété :

- *two terminal* : que deux nœuds bien définis, connus comme source et terminal, souvent notés s et t, appartiennent à une même composante connexe du graphe (i.e. qu'il y ait un chemin qui relie les deux nœuds) ;

- *all-terminal* : que le graphe soit connexe (c'est à dire qu'il y ait une seul composante connectée) ;

- *K-terminal* : un sous-ensemble de nœuds K appartienne à une même composante connexe (c'est à dire, que le graphe soit K-connexe). Ce dernier modèle englobe les deux précédents.

Voici alors la définition de nos problèmes principaux.

Définition. Etant donné un graphe probabiliste $G = (V, E, \vec{r})$, deux nœuds $s, t \in V$ et un sous-ensemble $K \in V$, nous définissons comme $\{s, t\}$-CLR le problème de calculer la probabilité $R(G, \{s, t\})$ que les nœuds s et t soient reliés ; comme *all-terminal*-CLR le problème du calcul de la probabilité $R(G, V)$ que le graphe soit connecté ; et comme K-CLR le problème du calcul de la probabilité $R(G, K)$ que le graphe soit K-connexe.
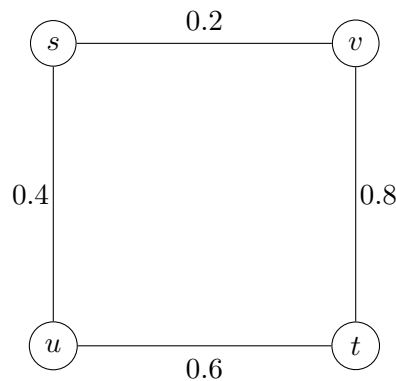
Figure 1: Un réseau avec 4 sites et 4 liens.

Nous allons illustrer les définitions précédentes au moyen d'un exemple. La Figure 1 présente le modèle d'un certain réseau avec quatre sites et quatre liens. Chaque arête

dans le graphe est étiquetée conformément à sa fiabilité (dans l'exemple, les valeurs sont de 0,2 , 0,4, 0,6 et 0,8). Plusieurs mesures de fiabilité peuvent être calculées selon la définition choisie pour K. La valeur de la fiabilité pour certains ensembles K de tailles 2 , 3 et 4 est montré dans la Figure 2 à droite. Il y a seize différentes configurations qui peuvent résulter de la combinaison des états des arêtes. Chaque événement aléatoire X_i correspondant à une configuration $\vec{x}_i$ a une probabilité d'occurrence $\Pr(X_i)$ associé, qui en vertu de l'indépendance d'échec est calculée comme le produit des fiabilités des arêtes opérationnelles fois le produit du complément de la fiabilité des arêtes échoués. Les trois colonnes à droite identifient les configurations qui appartiennent aux événements " le graphe est $\{s, t\}$-connecté ", " le graphe est connexe "et " le graphe est $\{s, t, u\}$-connecté ". Totalisant chaque colonne on obtient respectivement les trois mesures de fiabilité. En ce qui concerne le réseau, chaque configuration représente un événement aléatoire élémentaire, avec une probabilité connue d'occurrence. Alors les trois mesures correspondent à des événements qui englobent plusieurs événements élémentaires, c'est à dire plusieurs configurations. Notez que la procédure suivie ici pour calculer la CLR est simple et directe mais présente un inconvénient majeur : elle nécessite un effort, mesurée par le nombre d'opérations mathématiques élémentaires à effectuer, qui grandit exponentiellement avec le nombre d'arêtes. D'où l' intérêt pour le développement de méthodes de calcul et d'estimation plus efficaces.

La fiabilité diamètre-bornée

La fonction la structure Φ est définie seulement en termes de connectivité pour le CLR. Mais comme nous avons déjà vu, dans plusieurs contextes, la distance entre les nœuds peut être aussi pertinente. Dans [7], les auteurs ont introduit un modèle qui englobe le modèle CLR en ajoutant une limite supérieure pour la distance entre les terminaux. Voici alors les définitions suivantes pour la DCR, analogues à celles de la CLR.

Définition. Etant donné un graphe probabiliste $G = (V, E, \vec{r})$, un entier positif d, deux nœuds $s, t \in V$ et un sous-ensemble $K \in V$, nous définissons comme $\{s, t\}$-DCR le problème de calculer la probabilité $R(G, \{s, t\}, d)$ que la distance entre les nœuds s et t soit au maximum d ; comme *all-terminal*-DCR le problème du calcul de la probabilité $R(G, V, d)$ que la distance entre deux nœuds quelconques ne dépasse pas d ; et comme K-DCR le problème du calcul de la probabilité $R(G, K, d)$ que la distance entre deux nœuds de K ne dépasse pas d.

Selon ces définitions il est évident que tout problème DCR avec $d \geq n - 1$ est équivalent au problème CLR correspondant. Mais, lorsque $d < n - 1$, il peut y avoir des configurations où les terminaux sont connectés mais il y a quelques terminaux dont la distance dépasse d. Par conséquence nous avons que pour chaque graphique probabiliste $G = (V, E, \vec{r})$, ensemble $K \in V$ et entiers $1 \leq d \leq d'$, il se vérifie que $R(G, K, d) \leq R(G, K, d')$. Aux fins d'illustration, nous ramenons le réseau exemple de la Figure 1 et calculons les mesures de DCR correspondantes pour une valeur de $d = 2$ dans la Figure 2. Remarquez que, dans ce cas, la DCR et CLR coïncident pour les problèmes

Config. X	Pr(X)	Node Set K {s,t}	V	{s,t,u}
1	0.0384			
2	0.0096			
3	0.0256			
4	0.0064			
5	0.0576			
6	0.0144			
7	0.0384	✔		✔
8	0.0096	✔	✔	✔
9	0.1536			
10	0.0384	✔		
11	0.1024			
12	0.0256	✔	✔	✔
13	0.2304			
14	0.0576	✔	✔	✔
15	0.1536	✔	✔	✔
16	0.0384	✔	✔	✔
	1.0000	0.3616	0.2848	0.3232

Config. X	Pr(X)	Node Set K {s,t}	V	{s,t,u}
1	0.0384			
2	0.0096			
3	0.0256			
4	0.0064			
5	0.0576			
6	0.0144			
7	0.0384	✔		✔
8	0.0096	✔		✔
9	0.1536			
10	0.0384	✔		
11	0.1024			
12	0.0256	✔		
13	0.2304			
14	0.0576	✔		
15	0.1536	✔		✔
16	0.0384	✔	✔	✔
	1.0000	0.3616	0.0384	0.2400

Figure 2: À gauche : un exemple de trois mesures CLR (même réseau, différents terminaux) ; à droite, idem pour DCR.

à deux terminaux, mais le DCR est inférieur au CLR à la fois pour les autres cas.

Contributions

Dans cette thèse, nous avons présenté plusieurs résultats concernant le problème DCR. Nous avons présenté une introduction à plusieurs types des propriétés des réseaux qui peuvent être englobés sous le concept général de fiabilité. Ensuite, nous nous sommes concentrés spécifiquement sur le rôle joué par l'évaluation des métriques de fiabilité et en particulier la DCR. Nos contributions se réfèrent aux sujets qui suivent :

- la complexité computationnelle de certain cas particuliers de la DCR ;

- des topologies pour lesquelles le calcul de la DCR a une complexité polynomiale ;

- le comportement asymptotique de la DCR lorsque le réseau grandit selon certains règles probabilistes ;

- des méthodes de simulation de Monte Carlo appliquées à l'estimation de la DCR ;

- une métrique de performabilité qui généralise la DCR et son estimation par Monte Carlo ;

- l'utilisation de techniques d'interpolation polynomiales associés à Monte Carlo pour la CLR et DCR.

Ensuite, nous résumons nos résultats obtenus pour chaque sujet.

Complexité computationnelle

Le problème DCR généralise le problème CLR, en ajoutant une contrainte pour la distance entre les terminaux. La complexité de plusieurs problèmes augmente lors de l'ajout d'une telle contrainte, pour la plupart des valeurs du paramètre de distance d (voir par exemple [9]). Nous avons vu que la même chose s'applique lorsque l'on passe de CLR à DCR. Dans cette thèse, nous avons démontré que le calcul de la 2-K-DCR (i.e. K-DCR avec $d = 2$) pour quelconque ensemble K de taille fixe est une tache de complexité polynomiale (en plus, linéaire en n). Nous avons présenté deux démonstrations, l'une avec une formule fermée, l'autre avec une formulation recursive. Ce dernier a été implémenté et deux exemples ont été présentés qui attestent de la linearité du calcul.

On savait déjà que le K-CLR était un problème $\mathcal{NP}$-difficile en général, et aussi pour le cas particulier où $d \geq 3$ et $|K| \geq 2$ est fixe. La classification de la complexité de calcul est un problème ouvert pour le d-K-DCR lorsque $|K|$ est considérée comme une entrée libre, pour tout $d \geq 2$. Comme nous commentons dans la Section 4.4, ce problème est au moins aussi complexe que celui de compter le nombre de graphes partiels du réseau ayant un diamètre donné. Si l'on prouvait que ce dernier est $\mathcal{NP}$-difficile, ça suffirait pour

prouver la $\mathbb{N}$P-difficulté du d-V-KDCR aussi (et donc, pour quelconque K libre). Nous adressons actuellement nos efforts dans cette direction. La Figure Figure 3 reproduit les classes de complexité connues et non connues des sous-problèmes que l'on obtient pour différentes combinaisons de $|K|$ et d.

		$|K|$ (fixe)		$|K| = n$ ou libre
	2		3...	
2	$O(n)$ [10]		$O(n)$ [Cette thèse]	Non connue
3				
d ⋮		$\mathbb{N}$P-hard [10]		Non connue
$n-2$				
$n-1$ ⋮		$\mathbb{N}$P-hard [11]; voir Table 3.1 ici		

Figure 3: Complexité des sub-problèmes DCR selon d et $|K|$

L'analyse exacte et asymptotique des mesures DCR

Pour plusieurs catégories particulières de topologies le CLR et DCR peuvent être résolus en temps polynomial en le nombre de nœuds ou arêtes. Ici, nous avons illustré l'utilisation des techniques de programmation dynamique pour calculer le DCR pour deux familles simples de réseaux. Nous avons également caractérisé une vaste gamme de topologies pour lesquelles la $\{s,t\}$-DCR est polynomiale. Les graphes de ces familles doivent être telles que l'ensemble de nœuds puisse être partitionné en sous-ensembles dont le cardinal est borné par un certain paramètre h. La partition définit quels sont les liens permis dans cette topologie. La Figure 4 illustre cette famille de graphes. L'ordre de temps est polynomiale en la cardinalité de la partition (ainsi qu'en le nombre total de nœuds) bien qu'encore exponentielle en le paramètre h. Cela permet de calculer la valeur exacte de la DCR pour graphes de grande taille qui appartiennent à ces familles, ce qui peut être utile pour calculer des bornes (grâce à des méthodes basées sur l'identification de graphes partiels avec lesdites topologies), ou pour connaître des valeurs exactes auxquelles comparer les résultats de méthodes d'estimation.

Lorsque les réseaux se développent selon certains patrons probabilistes, il est naturel de se demander comment évoluent leurs propriétés de fiabilité. Nous avons introduit la question de l'analyse du comportement asymptotique du CLR et DCR, dans les réseaux
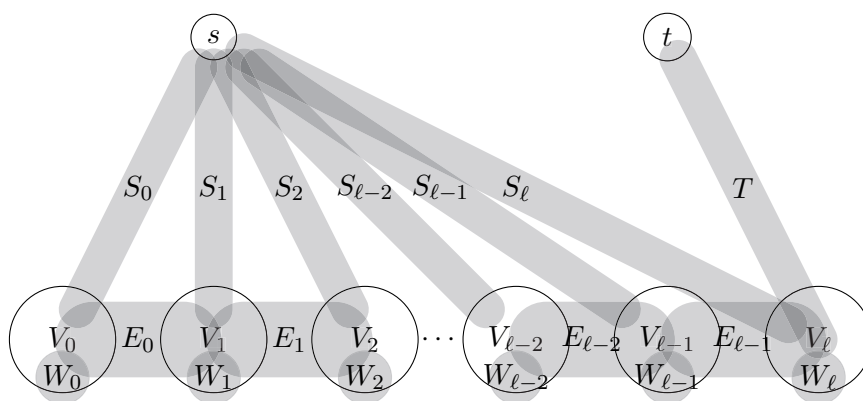
Figure 4: La famille de graphes $\mathcal{G}_h$

qui se développent selon les modèles extraits de la Théorie des Graphes Aléatoires. Nous avons présenté des résultats pour le mode de l' Gilbert selon lequel chaque paire de nœuds a la même probabilité p d'être lié par une arête. Nous avons prouvé que la K-DCR d'un tel réseau tend vers 1 quand $p > 1$. Par ailleurs, nous avons montré que, sous certains conditions, la K-DCR tend vers 1 voire avec $p \underset{n}{\to} 0$ (c'est à dire, avec une dégradation progressive de p sans borne inférieur). Nous visons à prolonger ce travail par l'étude du comportement asymptotique sur autres modèles aléatoires comme ceux d'Albert-Barabasi et Watts-Strogatz.

Méthodes de Monte Carlo pour l'estimation de la DCR

La complexité de calcul élevée de la CLR a motivé le développement de procédures pour l'estimation et le calcul de bornes. Plusieurs méthodes Monte Carlo ont été proposés. Dans les réseaux de communication réels, la fiabilité est généralement très proche de 1, et donc l'échantillonnage d'un réseau en état d'échec est un événement rare. Si on utilise la méthode Monte Carlo naïve il faut utiliser des échantillons de grande taille pour atteindre des résultats statistiquement significatifs (voir [12]). En outre, donné un certain intervalle de confiance, la taille nécessaire des échantillons augmente sans borne à mesure que la fiabilité du réseau devient plus proche à 1. Par conséquent, plusieurs méthodes de Monte Carlo avec des techniques de réduction de variance ont été développées pour la CLR. Dans le Chapitre 8, nous avons adapté une famille de telles méthodes au contexte DCR et montré comment la transition de CLR à DCR affecte chaque étape de la simulation. Nous avons étudié leur efficacité relative à Monte Carlo naïve. Cette famille méthodes est basée sur le conditionnement de l'espace d'échantillonnage, utilisant ensembles d'arêtes nommés *pathsets* et *cutsets* pré-calculées. Nous avons présenté quatre familles de tests qui attestent de l'amélioration de performance obtenue par rapport aux méthodes naïves.

Measures de performabilité dependentes du K-diamètre

Outre les mesures de fiabilité, qui mettent l'accent sur la connectivité du réseau, les mesures de performabilité reflètent mieux les aspects comme la performance globale et la qualité perçue. Ces mesures sont souvent définies comme la moyenne pondérée de la probabilité que le réseau soit dans certains états, si ceux-ci sont discrètes, ou l'intégrale de la densité de probabilité des états fois la valeur d'une fonction de l'état. De nombreuses méthodes qui sont appliquées pour calculer et estimer les mesures de performabilité ont leurs racines dans des travaux antérieurs développés pour les métriques de fiabilité. Dans le Chapitre 10, nous avons defini une métrique définie en termes du K-diamètre du réseau et adapté la technique Monte Carlo du Chapitre 8 pour l'estimer. Nous supposons que les diamètres possibles sont divisées en un certain nombre de gammes et chaque gamme reçoit un poids pour la construction de la métrique. Ces mesures de performabilité qui considèrent le K-diamètre peuvent être utiles dans des contextes où les latences affectent l'expérience de l'utilisateur, comme les réseaux P2P, les applications VoIP ou les applications Web. Nous avons montré qu'on atteint des réductions très significatives de variance par rapport à Monte Carlo naïve en exécutant des tests sur des réseaux de type maille. Nous avons introduit aussi une famille d'heuristiques qui génèrent les *pathsets* et *cutsets* à être utilisés pour le conditionnement de l'espace d'échantillonnage ; les tests montrent avantages significatives de performance quand on enchaîne ces heuristiques avec la méthode Monte Carlo proposée.

Interpolation polynomiale pour les métriques de fiabilité

Dans le Chapitre 10 nous avons introduit un algorithme pour l'estimation du polynôme de fiabilité. Quand tous les arêtes ont la même fiabilité p, la fonction $R(p)$ qui donne la fiabilité du réseau selon la valeur de p est un polynôme dont le degré est le nombre d'arêtes. Bien que l'exposition soit faite pour la all-terminal-CLR, la méthode peut être également appliquée à la DCR et quelconque ensemble K. L'algorithme utilise Monte Carlo pour estimer la fiabilité du réseau pour différentes valeurs de p, suivie d'une interpolation polynomiale de Newton. La métrique d'Hilbert induite dans $L^2[0,1]$ est utilisée pour projeter et retourner un polynôme à coefficients entiers qui satisfait certains propriétés élémentaires partagés par tous les polynômes de fiabilité. L'exemple que nous avons présenté illustre que les coefficients des majeures degrés sont estimés avec plus de précision que le reste. Cela est prometteur car les coefficients dominants sont les plus pertinents dans le contexte de défaillances rares. Un avantage important de connaître le polynôme de fiabilité est que cela permet l'analyse de sensibilité immédiat de la fiabilité du réseau par rapport à p.

Publications

Les travaux effectués pour cette thèse et les résultats obtenus ont abouti aux publications suivantes (articles publiés ou acceptés pour publication) :

- [13] E. Canale, H. Cancela, F. Robledo, G.Rubino, and P. Sartor. "On computing the 2-diameter-constrained K-reliability of networks", *International Transactions in Operational Research*, vol. 20, no. 1, pp. 49-58, 2013.

- [14] H. Cancela, F. Robledo, G. Rubino, and P. Sartor. "Monte Carlo estimation of diameter-constrained network reliability conditioned by pathsets and cutsets", *Computer Communications*, vol. 36, no. 6, pp. 611-620, 2013.

- [15] H. Cancela, F. Robledo, G. Rubino, and P. Sartor. "A Monte Carlo sampling plan for estimating diameter-dependent network parameters", in *Proceedings of the 4th International Workshop on Reliable Networks Design and Modeling*, pp. 129-134, 2012.

- [16] F. Robledo, P. Romero, and P. Sartor. "A novel interpolation technique to address the Edge-Reliability Problem", in *Proceedings of the 5th International Workshop on Reliable Networks Design and Modeling*, pp. 77-82, 2013.

- [17] H. Cancela, F. Robledo, G. Rubino, and P. Sartor. "Efficient estimation of distance-dependent metrics in edge-failing networks", to appear in *ITOR (International Transactions in Operational Research)*, 2013.

- [18] F. Robledo and P. Sartor. "A simulation method for network performability estimation using heuristically computed pathsets and cutsets", to appear in *IJMH (International Journal of Metaheuristics)*, 2013.

Part I

Introduction

Chapter 1

Context and Motivation

This thesis is about certain properties that are defined in the context of telecommunication networks. We work with models of networks where the basic components can fail and we assume that we know the probabilistic behavior of such failures. Although in general we talk about telecommunication networks, the models, metrics, theoretical results and algorithms that we employ and introduce can be applied to other contexts as transportation networks, parallel computer architectures, electrical networks, control systems, fire spreading, etc. The reason why we focus in telecommunication networks, is that they have been the main driver for the development of reliability measures, the main metrics we are concerned with in the thesis.

In this chapter we start by discussing several types of network properties that can be encompassed under the broad concept of *network dependability*. Then, we focus specifically on the role played by the evaluation of reliability metrics; this thesis is mainly concerned about one of such metrics, known as the *diameter-constrained reliability*. Once this contextual introduction is given, we finally summarize our contributions and the corresponding list of published and accepted articles.

1.1 Dependability properties of telecommunication networks

Communication networks are increasingly present in our daily lives in different ways. A vast number of metrics to quantify the service that they deliver to users can be conceived. Some of these can be simply expressed as binary metrics (e.g. the network is either "up" or "down"). Other metrics can have a non-binary nature (e.g. network throughput). One can even think of very complex metrics defined e.g. by assigning different weights to different classes of "actors" and taking into account throughput, packet loss, delays, etc. In general we will refer to network performance as the level that a certain network achieves regarding some well-defined quantitative metric.

During the first decades of telecommunication networks development, the emphasis was put on metrics that expressed the capacity of a network to keep operating [1]. As we see later, depending on the model and definition, these metrics are referred to as *availability* and *reliability*. When the first packet switched networks were deployed using

3

telephone lines, the main efforts aimed at keeping the lines (and the underlying network topology) operating. Within this context, a simple differentiation between an "up" and a "down" state was useful enough, therefore motivating a big deal of research in the field of reliability anaysis. Improved technologies and reduced costs of networking equipment, that enabled affordable redundancy, led to the design and deployment of networks that were much more reliable in terms of connectedness. At the same time, higher data traffics and more complex applications led to a focus shift from conectedness to more complex performance metrics able to distinguish between several levels of service [19]. Then, in the late 80's and 90's, there came the time for massive deployment of highly reliable optic fiber lines. The extraordinary capacity of these lines made possible that single-path networks deliver the required performance levels, provided that the lines were operational. In this context, reliability models for those lines gained relevance again. At present, the massive deployment of wireless networks, whose point-to-point links are less reliable because of reasons like environmental factors and energy-saving concerns, made connectivity properties gain relevance again. We can say that the historical emphasis on connectedness has been leveled by the more comprehensive concept of *performability*, which encompasses both reliability and performance concerns [1, 20].

The service that real networks offer can be affected by several kinds of failures. Human errors can cause operational faults, black-outs, the deployment of wrong routes for packet switching, etc. Humans can also purposely damage networks (sabotage). Design flaws can result in data loss, insufficient performance, power outages, link saturation or more subtle problems like *jitter*. Environmental factors can cause interferences and noise in wired lines, signal atenuation, intermittent interruptions in wireless links, as well as more general damages in case of floods, earthquakes and other situations alike. Random wearout can cause a component to malfunction or to stop operating.

When designing communication networks, engineers must anticipate the possible sources of failure and performance degradation. Basically, once a mathematical model for the network has been adopted, two major approaches can be followed, often collaborating through several iterations. On the one hand the network can be designed with the aim of achieving certain desired properties and performance thresholds; the design algorithms, topologies used, etc. are such that fulfilling these properties is guaranteed. On the other hand there are network analysis techniques that, given a certain network, allow to compute, bound or estimate metrics that quantify the properties and performance metrics we are interested in. An example of interaction can be that of analyzing first a legacy network, and once its current properties are known, improving it through design techniques until it reaches the desired levels, more demanding than those achieved by the original network. This hybrid approach can be also followed when design techniques are only able to suggest improvements that can probably improve the desired properties (e.g. heuristics); analysis must be performed on the augmented network to determine if these properties were reached. In this thesis we are concerned with network analysis techniques.

When designing systems with the goal of having a failure probability that lies below a threshold goal, investments can be oriented in two major (compatible) directions. We

can spend resources trying to improve the individual reliability of single components. This can be seen as a "local optimization" approach. We can also spend resources by introducing redundant components in the system (without an extreme focus on their individual reliability). This can be seen as a "systemic" approach. There are several reasons why the second approach has traditionally played an important relative role in network design. For example, this is particulary true when compared to total-quality manufacturing principles like those of Toyota Production System [2] or Six-Sigma [3], where major stress is put in producing the reliable units through atomic tasks that are error-free as much as possible. Firstly, it is very difficult to anticipate all possible sources of failure that threaten network components. Secondly, the costs of deploying more components for having redundancy is often justified by the significant savings achieved using "just acceptable" components. Think for example of the case of commercial computer clusters, where great economies are achieved by using *commodity servers*, at the expense of some additional units being needed for improved redundancy.

1.1.1 A classification of concepts underlying dependability

Dependability is a broad generic concept that includes attributes such as reliability, availability, safety, integrity, maintenability and others. Boundaries in the definition of these concepts have been historically blurred. Some attempts have been made to agree on precise definitions for helping communication and cooperation among the research community. What follows is based on [21].

The following definitions, yet focusing mainly on computing and communication systems, are in general applicable to systems that include human and organizations as part of their environment. This said, a *system* is defined as an entity that interacts with other entities that build its *environment*. The *function* of the system is what it is intended to do. It is described, in terms of functionality and performance, by a *functional specification*. The *behavior* of the system is the sequence of states by which the system implements its function. It is enabled by the system's *structure*, which is a recursive set of *components* that can be further structurally decomposed or be considered as *atomic*. The *service* delivered by a system is its behavior as perceived by its users, through a *service interface*. The *total state* of a system is the set of the following states: computation, communication, stored information, interconnection and physical condition. The part of the provider's total state that is (is not) perceivable at the service interface is its *external (internal) state*.

Of particular importance is the precise definition of failures, errors and faults. A *failure* is any deviation from *correct service*, which is defined as the condition where the system implements the system function. The period that the failure spans is a *service outage* and the transition back to correct service is a *service restoration*. The occurrence of a failure implies that the external state has deviated from the correct state; such deviation is called an *error*. An error is the part of the total state that may lead to a subsequent failure. Errors are caused by *faults*. A fault is said to be *active* when it causes an error, otherwise it is *dormant*. Note that a fault can cause an error that affects the state of a component that is part of the internal state of the system; in that case,

the external state might not be immediately affected and no error would occur (at least immediately). For example, if a single hard drive wich makes part of a RAID-5 array in a server malfunctions, there is a dormant fault (considering the server as "the system"). If a second drive starts also to malfunction before the first one is replaced, an error occurs. This can lead to a failure if the system's users are affected by this error before the array is repaired and rebuilt. Note that in general the system provides more than one specific function to the users. When some (but not all) of them fail, the system is said to be in a *degraded mode* and there is a *partial failure*.

Dependability was originally defined as the ability to deliver a service that could justifiably be trusted; an alternate definition is whether the system presents or not the ability to avoid service failures that are more frequent and more severe than is acceptable. It encompasses the following attributes:

- *availability*: readiness for correct service (at a given instant t);

- *reliability*: continuity of correct service (during a given time interval $[t_1, t_2]$);

- *safety*: absence of catastrophic consequences on the user and the environment;

- *integrity*: absence of improper system alterations;

- *maintainability*: ability to undergo modifications and repairs.

An additional attribute is *confidentiality* i.e. the absence of unauthorized disclosure of information. Confidentiality, together with availability and integrity, define the concept of *security*. There has been a trend towards considering at the same time issues related to dependability and security. Finally, the dependability and security specification of a system must disclose what the requirements are for the above attributes, in terms of the acceptable frequency and severity of service failures for specified classes of faults and given use environments. When designing dependable and secure systems, the means to achieve the desired levels for each dependability and security attributes can be grouped in four categories:

- *fault prevention*: preventing the occurrence or introduction of faults;

- *fault tolerance*: avoiding service failures in the presence of faults;

- *fault removal*: reducing the number and severity of faults;

- *fault forecasting*: estimating the present number, the future incidence, and the likely consequences of faults.

1.2 Evaluation of reliability metrics

A family of metrics developed in the last 50 years that has received vast consideration is that of the *reliability* metrics. Generally speaking, they express the probability that a

given network that can be in two states from the user's standpoint (operational/failed), avoids the failed state during a given time interval (dynamic models) or at any moment (static models) - see Section 2.2. Some well-defined property of the network determines the difference between both states. Physical telecommunication networks, regardless of their size and complexity, are built on top of few basic components. Most networks can be well represented in terms of two major classes of component: (i) components that send, receive, process and hold data (sites) and (ii) components that channel data from one site to another neighbor site (links). For reliability analysis these networks are often modeled as probabilistic graphs (see Section 2.1). The random failures of sites and links are modeled as nodes and edges that are labeled with their failure probabilities. A failing site is a failure for the atomic subsystem "site"); but it is a fault for the system "network". Therefore, reliability is related with the fault tolerance category in the context of dependable systems. The complement of the reliability, called *unreliability*, is then the probability that a combination of faults occurs that makes the network be in a failure state. Considerable effort has been devoted to the study of reliability metrics. On the one hand this is due to the relevance that connectedness had in particular for real networks that were developed in the second half of past century. On the other hand because, even with many simplifying assumptions, the problems that arise are complex, challenging and far from having reached definitively satisfactory solutions (see Section 3.2). During the last decades, the focus on connectedness has been broadened to consider performance concerns, giving rise to the concept of *performability*, which encompasses both reliability and performance.

In this thesis we work with reliability metrics defined by a static stochastic model that consists of a probabilistic graph and a set of distinguished nodes called *terminals*. In Chapter 2 we formalize the model and definitions. The classical reliability (CLR) analysis studies the probability that the terminals are connected. Surveys can be found in [4, 5, 6]. This metric has been generalized [7] to consider not only connectedness but also to enforce that the distance between terminal nodes is not above an acceptable threshold. A survey can be found in [8]. The new metric, called diameter-constrained reliablility (DCR), allows for modeling contexts where there are limits in the acceptable number of hops that packets can undergo. This constraint is relevant in several contexts that are increasingly important since the 90's. For example, in peer-to-peer (P2P) and voice-over-IP (VoIP) applications, perceived quality is affected by latency, which is in turn determined by the number of links traversed by packets. Quality deteriorates as high hop-distance states occur more frequently in the network. In web applications with rich interfaces, the quality perceived by the end user is related to responsiveness, where latency determines the delay between the user actions and their effect on the output interface.

1.2.1 The independence hypothesis

Most of the methods that have been developed for computing, bounding and estimating the network reliability assume that the failure of components are independent. This is due to several reasons. First, predicting and modeling the way that failures depend

on each other is usually a very complex task. Second, the independence hypothesis simplifies significantly the mathematical treatment of the problem, thanks to the fact that probabilities can be conveniently written using factorized expressions. Third, several sources of faults are well-modeled as independent events. A typical example is component wearout. Another family is that of human errors, like bad configuration of a network interface card or accidentally chopping a fiber line due to land-moving activities in a campus. Dependency between the failure of components can be significant and thus non-negligible e.g. when considering natural catastrophes (that can affect at the same time several links), and in particular for performability analysis. For example, the failure of several links causes that a lot of traffic be channeled through other links, which at this time makes them more prone to be overloaded and go down. In this thesis, unless locally specified, we will assume that the independence hypothesis holds.

1.3   Contributions of the thesis

The list below summarizes the contributions of this thesis. We include references to the chapters where they can be found, as well as to the articles that were written under the research process leading to the completion of the thesis.

- *Results on the computational complexity of the DCR.* We survey previous results on computational complexity for the CLR and DCR problems. Then we prove that the DCR problem with diameter parameter $d = 2$ and any fixed number of terminal nodes can be solved in a time linear in the number of nodes. We give two alternative proofs for this result. See Chapters 3 and 4, as well as [13].

- *Characterization of a family of topologies that has polynomially-computable DCR.* We characterize a broad family of topologies for which the two-terminal DCR can be computed in polynomial time in the number of nodes of the network. We give the corresponding algorithm. The family is defined in terms of a certain way to partition the nodes in subsets with bounded cardinality. Restrictions on the sets to which both nodes of any edge can belong must be respected. See Chapters 5 and 6.

- *Asymptotic analysis of the DCR and basic results.* We introduce the question of analyzing the asymptotic behaviour of CLR and DCR metrics in networks that grow following precise probabilistic rules. Some basic results are given for the Gilbert's model. We show that the DCR tends to 1 provided that the link reliabilities and diameter parameter d respect relationships, even when a certain degradation of those reliabilities is allowed. See Section 6.4.

- *Monte Carlo methods for DCR metrics.* We discuss the differences that arise when moving from CLR to DCR context, regarding the use of Monte Carlo techniques for network reliability estimation. A family of methods based on conditioning the sampling space by pathsets and cutsets is adapted to DCR. We present numerical

examples that evidence important gains in efficiency relative to crude Monte Carlo simulations. See Chapters 7 and 8, as well as [14].

- *Monte Carlo estimation of performability measures defined in terms of distances.* We define a family of performability measures as a weighted average based on K-diameter intervals. Then we estimate it, by introducing a Monte Carlo method that generalizes the one presented for DCR in the previous section. We present numerical examples that evidence important gains in efficiency relative to crude Monte Carlo. See Chapter 8 as well as [15, 17, 18].

- *Reliability estimation by simulation and polynomial interpolation.* We introduce a novel interpolation algorithm to estimate the reliability polynomial in networks with uniform edge reliability p. We develop an example for the all-terminal CLR, but the technique can be applied for DCR and any set K in general. Knowing the reliability polynomial is useful for performing sensitivity analysis on the edge reliabliity. The example shows that the dominant coefficients are estimated with more precision, which is good particularly in the context of high edge reliabilities (rare failures). See Chapter 10.

The work carried on during this thesis and the results obtained have resulted in the following articles, either published or accepted for publication:

- [13] E. Canale, H. Cancela, F. Robledo, G.Rubino, and P. Sartor. "On computing the 2-diameter-constrained K-reliability of networks", *International Transactions in Operational Research*, vol. 20, no. 1, pp. 49-58, 2013.

- [14] H. Cancela, F. Robledo, G. Rubino, and P. Sartor. "Monte Carlo estimation of diameter-constrained network reliability conditioned by pathsets and cutsets", *Computer Communications*, vol. 36, no. 6, pp. 611-620, 2013.

- [15] H. Cancela, F. Robledo, G. Rubino, and P. Sartor. "A Monte Carlo sampling plan for estimating diameter-dependent network parameters", in *Proceedings of the 4th International Workshop on Reliable Networks Design and Modeling*, pp. 129-134, 2012.

- [16] F. Robledo, P. Romero, and P. Sartor. "A novel interpolation technique to address the Edge-Reliability Problem", in *Proceedings of the 5th International Workshop on Reliable Networks Design and Modeling*, pp. 77-82, 2013.

- [18] F. Robledo and P. Sartor. "A simulation method for network performability estimation using heuristically computed pathsets and cutsets", to appear in *IJMH (International Journal of Metaheuristics)*.

- [17] H. Cancela, F. Robledo, G. Rubino, and P. Sartor. "Efficient estimation of distance-dependent metrics in edge-failing networks", to appear in *ITOR (International Transactions in Operational Research)*, 2013.

1.4   Structure of the thesis

This document is organized as follows. Part I includes an introduction to the concept of dependability in telecommunication networks and, in particular, to the evaluation of the reliability metrics that this thesis focuses on. Basic concepts from Graph Theory that are used throghout the thesis are defined in Chapter 2. Also, in Chapter 2, the classical and diameter-constrained reliability metrics are formally defined, grounding the foundation for the remaining chapters.

Part II is devoted to the analysis of the computational complexity of the DCR metric. A survey of previously available results for the CLR and DCR is done in Chapter 3. Then, in Chapter 4 we contribute results for a particular case that was not yet classified into complexity classes.

Part III is devoted to the exact calculation of the DCR metric. Chapter 5 includes a brief survey on exact methods both for the CLR and the DCR. Then, in Chapter 6, we illustrate how the exact computation of the CLR and DCR metrics can be done polynomially for certain topologies using dynamic-programming techniques. This chapter is intended also to illustrate some concepts related to the DCR metric e.g. the distribution of probabilities for the diameter of the network. We start by developing algorithms for two simple topologies. We generalize them into a broader family of topologies and present an algorithm that computes the two-terminal reliabilities in polynomial time. Finally, in Section 6.4 we introduce the problem of analyzing the asymptotic behaviour of the DCR metric in networks that grow following strict probabilistic rules drawn from Random Graph Theory. We introduce basic results for the simplest type of random graphs (Gilbert's model).

Part IV is devoted to the estimation of the DCR and related performability measures by means of Monte Carlo techniques. Chapter 7 motivates and introduces to the use of this simulation technique in the context of reliability metrics and failures that are rare events. Chapter 8 describes a family of methods previously introduced for the CLR, based on sampling conditioned by pathsets and cutsets and adapts it to DCR context. The chapter is intended to illustrate the differences that arise when moving from CLR to DCR as well as presenting numerical examples. In Chapter 9 we generalize the method developed in Chapter 8 for estimation of a performability measure that we define in terms of the network K-diameter. In Chapter 10 we introduce an algorithm for estimating reliability measures that combines Monte Carlo simulation with polynomial interpolation. It is based on estimating the reliability polynomial through Newton's interpolation based on estimates of the network reliability for different edge reliability values. The example developed in the chapter is based on the all-terminal reliability although the technique can be applied straightforwardly for DCR and any set of terminal nodes.

Finally, Part V summarizes the conclusions that can be drawn from this thesis, further aspects to analyze, and discusses the current research lines that we are pursuing.

Chapter 2

Mathematical background and metric definitions

The main purpose of this chapter is to lay out the mathematical basis of the reliability metrics used in this thesis. We also illustrate the classical and diameter-constrained network reliability metrics. We start by introducing the necessary definitions and notation from Graph Theory. Next we introduce the basic concepts we need of multi-component systems. We finally employ these definitions and concepts to define the classical and diameter-constrained reliability measures and give an example of each one.

2.1 Graph concepts

In this section we formally define several mathematical objects related to Graph Theory that we employ throughout this thesis. This is not intended to be a comprehensive corpus of definitions, but to introduce the particular definitions and notation that we employed in this work. Observe that there is no complete consensus in Graph Theory literature about the exact definition of certain concepts, so this section should be useful for disambiguation in the context of the thesis.

A *graph* is a mathematical object that represents a set of objects where some pairs of such objects are related (connected) by links. This is usually represented as $G = (V, E)$, where G is the graph, V is the set of objects (called *vertices* or *nodes*) and E is a family of pairs called *edges* or *arcs*. The edges can be directed (when the pair is ordered) or undirected (when they are defined by an unordered pair of nodes). A graph can have *loops* i.e. edges that relate a node with itself. It can also have *multiple edges* i.e. more than one edge connecting the same pair of nodes (in the same order if they are directed). A *simple graph* is a graph with no loops and no multiple edges. In the context of this thesis, unless locally specified, we work always with simple, undirected graphs. Therefore, we introduce the following definition.

Definition 1. *A* graph *is a pair $G = (V, E)$ where V are called nodes and $E \subseteq \{\{x, y\} : x \in E \wedge y \in E\}$. In general we will denote the cardinalities of V and E as $n = |V|$ and*

$m = |E|$. *We will also denote the edge corresponding to the nodes u and v as both uv or $\{u, v\}$.*

Graphs will allow us to model networks, where sites are represented by nodes and links between them are represented by edges. We will assign probabilities of failures to the edges and be interested in certain properties that the resulting graphs can have. Therefore we introduce the definition of probabilistic graphs.

Definition 2. *A* probabilistic graph *is a graph whose nodes and edges are labeled with probabilities (that can be different in general). That is, $G = (V, E, \vec{r})$ where $r_i \in [0, 1]$ for all $i \in V \cup E$.*

We will work with perfect nodes and edges that can fail, so (unless otherwise specified locally) the vector $\vec{r}$ will be indexed just by the set E. In general, r_{uv} and r_e will denote the probability that the edge $e = \{u, v\}$ is operational. Then, every probabilistic graph $G = (V, E, \vec{r})$ yields random events that consist of the partial graphs $G' = (V, E' \subseteq E)$. Each of these random events occurs with a certain discrete probability. As we will see, the final goal is to calculate or estimate the probability accumulated by the random events for which a certain connectivity property holds.

Definition 3. *A* walk *is a sequence of edges $e_1, \ldots, e_\ell$ such that $e_i = u_i v_i$ for $i = 1 \ldots \ell$ and $v_i = u_{i+1}$ for $i = 1 \ldots \ell - 1$. The walk is said to* connect *the nodes u_1 and v_ℓ and have length equal to ℓ. We say that this is an $(u_1 - v_\ell)$-walk.*

Observe that the definition of walks allow for visiting the same nodes several times. A walk where this does not happen is a *simple walk*. Although there is no consensus, most authors adhere to the following definition (and so do we in this thesis).

Definition 4. *A* path *is a simple walk. We say that the path corresponding to an $(u_1 - v_\ell)$-walk is an $(u_1 - v_\ell)$-path and denote it by giving the sequence of visited nodes as $u_1 - u_2 - \cdots - u_\ell - v_\ell$.*

As we said we are interested in evaluating the probability that the probabilistic graphs that model networks have certain connectivity properties. Next we define the concepts related to connectivity.

Definition 5. *A graph is said to be* connected *if and only if there is an $(s - t)$-path for every pair of nodes $s, t \in V$. Connectedness between nodes is an equivalence relationship that induces a partition of the nodes in* connected components.

We will distinguish certain nodes $K \in V$ and call them *terminals*. In general we will be interested in connectivity properties that apply only for the nodes of K, and consider the nodes in $V \setminus K$ as optional ones (nodes that can be optionally used if they help to achieve the desired properties). This results in the following definitions.

Definition 6. *A graph is said to be K-connected if and only if there is an $(s - t)$-path for every pair of nodes $s, t \in K$. Observe that the nodes outside of K can optionally be part of such paths.*

In this thesis we are also concerned with the length of paths that connect terminal nodes. In general we will accept only path whose length does not exceed a certain threshold d. Next we introduce several definitions related to this.

Definition 7. *The* distance *between two nodes s and t is the length of the shortest $s-t$-path, i.e. the length of the $s-t$-path with the lowest length (it can be either a non-negative integer, or infinity if s and t are not connected).*

Definition 8. *The* diameter *of a graph (or a certain connected component) is the maximum of the set of distances between all pairs of nodes of the graph (or the connected component). Again, it can be a non-negative integer of infinity.*

Definition 9. *Given a non-negative integer d, two nodes s,t are said to be d-connected if and only if their distance does not exceed d.*

Definition 10. *The K-diameter of a graph (being $K \subseteq V$) is the maximum of the set of distances between all pairs of nodes in K.*

Definition 11. *A graph is said to be d-K-connected if and only if its K-diameter is not above d.*

The definitions above allow for a short introduction to the problem of computing the reliability metrics we deal with in this thesis. Assume that a probabilistic graph G modeling a network with edges that can fail, a set of terminals K and a distance threshold d are given. Computing the classical reliability (CLR, Section 2.3) and the diameter-constrained reliability (DCR, Section 2.4) consist of computing the probability that the graph is K-connected or d-K-connected respectively. Next we define some additional objects that play an important role in the techniques used for computing and estimating these measures.

Definition 12. *Given a graph $G = (V, E)$ and a set of terminals $K \subseteq V$, a subset of edges $E' \subseteq E$ is said to be a* pathset *if and only if the partial graph $G' = (V, E')$ is K-connected. Conversely, the subset E' is said to be a* cutset *if and only if the partial graph $G' = (V, E \setminus E')$ is not K-connected.*

The context will always make clear what the set K is. Taking into account the condition of useful paths not being longer than a certain threshold d, results in the following definition.

Definition 13. *Given a graph $G = (V, E)$, a set of terminals $K \subseteq V$ and a positive integer d, a subset of edges $E' \subseteq E$ is said to be a* d-pathset *if and only if the partial graph $G' = (V, E')$ is d-K-connected. Conversely, the subset E' is said to be a* d-cutset *if and only if the partial graph $G' = (V, E \setminus E')$ is not d-K-connected.*

In general we will refer to pathsets and d-pathsets (alternatively, cutsets and d-cutsets) just as "pathsets" ("cutsets"). The context will make clear if there is a parameter d involved and its value in that case. Note that pathsets and cutsets (with no parameter

d) can be seen as *d*-pathsets and *d*-cutsets respectively for any $d \geq n - 1$. Consider a probabilistic graph $G = (V, E, \vec{r})$ and one of the random events $G' = (V, E'' \subseteq E)$. Then, a pathset E' is said to operate in G' if and only if $E' \subseteq E''$. Similarly, a cutset E' is said to fail in G' if and only if $E' \cap E'' = \emptyset$. Pathsets and cutsets are concepts widely employed in reliability analysis because they account for sets of edges whose presence (respectively absence) guarantees (respectively deters) connectivity.

Observe that adding an edge to a pathset results in another pathset. The same applies to cutsets. But there are pathsets (cutsets) for which the suppression of any edge results in an edge set that is not a pathset (cutset). Formally this leads to the following definition.

Definition 14. *A set of edges $E' \subseteq E$ is a* minpath *(respectively* mincut*) if and only if it is a pathset (cutset) and for any $e \in E'$ it holds that $E' \setminus \{e\}$ is not a pathset (cutset).*

Minpaths and mincuts are thus minimal-cardinality edge sets, whose presence and absence respectively determines that the network is connected or disconnected, in a certain context given by a well-defined terminal set K and an eventual length threshold *d*.

The above definitions are illustrated in Figure 2.1. The modeled network has 6 nodes and 9 edges. Two nodes, *s* and *t*, are considered as terminals. Different pathsets and cutsets are represented by thick, dashed and double lines, whereas dotted lines represent the remaining links. Figures (a), (b) and (c) show respectively two pathsets for $K = \{s, t\}$, one 3-pathset and one 4-pathset. These are also minpaths; observe that when $|K| = 2$, as in the example, all minpaths are paths. Similarly, Figures (d), (e) and (f) show respectively three cutsets for $K = \{s, t\}$, one 3-cutset and one 4-cutset. Suppressing the edges of the 4-cutset in (f) does not disconnect *s* and *t* but breaks every possible path whose length is lower or equal to 4. Analogously as for (a), (b) and (c), the cutsets illustrated are also mincuts. When working with no length constraint (or $d \geq n - 1$), a minpath is always a tree whose leaves are exactly the nodes of K. When there is a length constraint $d < n - 1$, this is not necessarily true and minpaths can in general have cycles.

2.2 Multi-component systems

Communication networks can be seen as multi-component systems. In general, a multi-component system is a system built by several (*M*) components. Each component can be in one of two possible states. Each combination of the component's states results in the overall system having one of two states too. One of these binary states is normally referred to as *operational*, *working* or *up*. The other binary state is normally referred to as *failed* or *down*. Conventionally, both states are represented by the integers 1 and 0 respectively. A *state vector* or *configuration* of the system is a vector $\vec{x} = (x_1, \ldots, x_M) \in \{0, 1\}^M$. The value of $x_i : i \in 1, \ldots, M$ indicates the state of the *i*-th component of the system (following a certain conventional order). As we said, every possible configuration $\vec{x}$ results

two different pathsets

(a)

one 3-pathset

(b)

one 4-pathset

(c)

three different cutsets

(d)

one 3-cutset

(e)

one 4-cutset

(f)
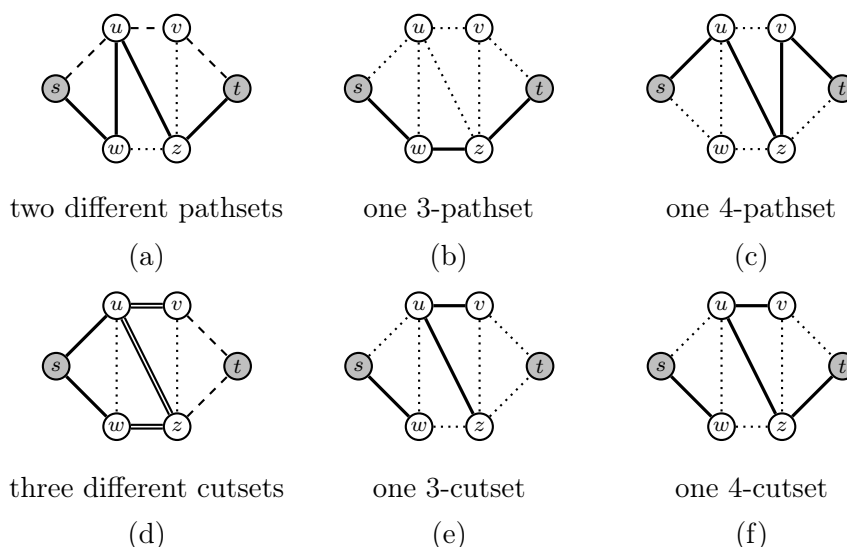
Figure 2.1: Pathsets and cutsets.

in the system being in a state represented by 1 or 0. Hence, there exists a *structure function* Φ defined as follows:

$$\Phi : \{0,1\}^M \to \{0,1\}$$

$$\Phi(\vec{x}) = \begin{cases} 1 & \text{if the system is } \textit{operational} \text{ when the configuration is } \vec{x}, \\ 0 & \text{if the system is } \textit{failed} \text{ when the configuration is } \vec{x}. \end{cases}$$

Observe that there are 2^M different functions that respect this definition. The particular one to be used depends on the specific aspect of the real situation that is being modeled. For example, two systems with M components may be such that one is said to work when (and only when) all of its components work, while the other is said to work whenever there is at least one component that works. Moreover, there may be different aspect to evaluate for the same real situation, that result in different Φ functions. Consider for example an office with M employees. One could say that the office "works" if there is at least one employee present. Alternatively, one could also say that the office "works" if certain essential employees are present, regardless of the remaining. Both result in different Φ functions for the same reality (the office).

Frequently, structure functions are assumed to be *coherent*, meaning that the following three properties hold for them:

- (i) $\Phi(\vec{0}) = 0$;

- (ii) $\vec{x} \leq \vec{y} \to \Phi(\vec{x}) \leq \Phi(\vec{y})$;

- (iii) $(\forall i)(\exists \vec{x}) : \Phi(\vec{x}, 0_i) \neq \Phi(\vec{x}, 1_i)$;

15

where $\vec{0}$ denotes a configuration where all entries are 0, $\vec{x} \leq \vec{y}$ denotes the fact that $x_i \leq y_i$ for all i (similarly for $\vec{x} < \vec{y}$) and $(\vec{x}, 0_i)$ denotes the configuration obtained from $\vec{x}$ by setting x_i to 0 (similarly for $(\vec{x}, 1_i)$). Condition (i) states that the system cannot work if no component works. Condition (ii), monotonicity, states that bringing up components can never result in the system switching from the up state to the down state. Condition (iii) states that all components are meaningful regarding the system state, since for each one there is at least one configuration for which changing its state affects the state of the system. It follows from condition (iii) that there must exist configurations that result in the system being up. In the field of reliability analysis, systems are usually coherent. This is the case of the models that we study here. Nevertheless there are also non-coherent systems, as the classical model where nodes can fail and the system is considered to be operational if and only if the surviving nodes are connected; in this case, the monotonicity condition does not hold.

Once the function Φ is defined, a framework must be set for its probabilistic analysis. Let us see the main ones and how the *reliability* (as a function of the time t) and *availability* (as a function of time intervals $[0, t]$ or $[t_1, t_2]$) are defined. There are stochastic models where all components start in operational state ($t = 0$), fail independently and then remain failed forever (there is no repairing capability). Defining as *reliability* the probability that a component/the system is up in time t (*reliability*) and as *availability* the probability that a component/the system is always up in the interval $[0, t]$ (or $[t_1, t_2]$), we observe that in these models the reliability and availability of the system coincide. Other stochastic models allow for repairing components, which takes a certain time. In this case, the system can switch between the operational and failed state as components fail and are repaired. Observe that when allowing repairs, failure of a set of components whose simultaneous failure implies a system failure does not guarantee a system failure. If the components are repaired in times such that there is no time when all of them are in a failed state simultaneously, the system will not necessarily fail. A comprehensive set of definitions related to the more general concept of *dependability* (that subsumes more specific concepts such as reliability, availability, safety, integrity, maintainability and others) can be seen in [21]. When performing asymptotic analysis, the availability of a component can be estimated [11] as the ratio between the mean time to failure and the sum of it and the mean time to repair. This definition is extensible to the system availability of the system. But the most usual framework in classical literature for reliability analysis is a static stochastic model, where the state of the i-th component is a Bernoulli random binary variable X_i and the M random variables are independent. This is the model we work with in this thesis. In this time-independent model, both the terms *availability* and *reliability* R are used for referring to the probability that the network system is operational, that is,

$$R = \Pr(\Phi(X) = 1) = \mathrm{E}(\Phi(X))$$

with $X = (X_1, \ldots, X_M)$.

2.3   Classical reliability metrics

As said before, the classical model for network reliability analysis is a stochastic static model where the network is represented by a probabilistic graph. Nodes model the sites and edges model the communication lines. If there is at least one line that is uni-directional then a directed graph must be used. If all lines are bi-directional then an undirected graph can be used. If several lines exist that connect the same pair of sites, they can be replaced in the model by a single line whose reliability is equal to the probability that at least one of such lines are operational (see Section 5.1.1). In the context of this thesis we will work with networks that have perfect nodes (they never fail, i.e. their reliability is equal to 1) and edges that fail independently with known reliabilities. Therefore we will model the network as a simple, undirected and probabilistic graph. Once the graph model is chosen, the structure function Φ must be defined somehow. This is done by means of some connectedness property of the graph; the reliability of the network is then defined as the probability that this property holds for the graph. Classical literature distinguishes three main reliability analysis problems for which the corresponding property states that:

- *two-terminal*: two well-defined nodes, known as "source and terminal", often denoted as s and t, belong to the same connected component of the graph (there is a path that connects both nodes);

- *all-terminal*: the graph is connected (i.e. there must be one single connected component);

- *K-terminal*: only a certain subset of nodes K are required to belong to the same connected component (the graph is K-connected). This model subsumes the two previous ones.

Recall that we refer throughout this thesis as *CLR* both to the classical reliability context as well as the specific problems that we define next (the context will make it clear, in each occurrence, what the meaning is).

Definition 15. *Given a probabilistic graph $G = (V, E)$ and two nodes $s, t \in V$, we define as $\{s, t\}$-CLR the problem of computing the probability $R(G, \{s, t\})$ that the nodes s and t are connected.*

Definition 16. *Given a probabilistic graph $G = (V, E)$ we define as* all-terminal-CLR *the problem of computing the probability $R(G, V)$ that G is connected.*

Definition 17. *Given a probabilistic graph $G = (V, E)$ and a subset $K \in V$, with $|K| \geq 1$, we define as K-CLR the problem of computing the probability $R(G, K)$ that the graph is K-connected.*

Let us illustrate the previous definitions by means of an example. Figure 2.2 presents the model for a certain network with four sites and four links. Each edge in the graph is

labeled according to its reliability (in the example the values are 0.2, 0.4, 0.6 and 0.8). Depending on what the terminal set is, several reliability measures can be computed. In Figure 2.3 we compute the reliability for terminal sets of sizes 2, 3 and 4. There are sixteen different configurations that can result from the combination of the edge states. Each configuration x_i has a probability of occurrence associated, which by virtue of the independence of failures is computed as the product of the reliabilities of operational edges times the product of the unreliabilities of failed edges. The three rightmost columns identify the configurations that belong to the events "the graph is $\{s, t\}$-connected", "the graph is connected" and "the graph is $\{s, t, u\}$-connected". Summing each of these rightmost columns yields respectively the two-terminal (s, t) CLR, the all-terminal CLR and the K-terminal CLR for $K = \{s, t, u\}$. This is known as state enumeration method. Regarding the network, each configuration represent an elementary random event, with a known probability of occurrence. The events that we are interested in (connectedness of a certain set of nodes) are events that encompass several elementary events, i.e. several configurations. Observe that the procedure followed here to compute the CLR is simple and straightforward yet has a major drawback; it requires an effort, measured as the number of elemental mathematical operations to compute, that grows exponentially with the number of edges. Hence the interest in developing more efficient methods for computing, bounding and estimating these measures.
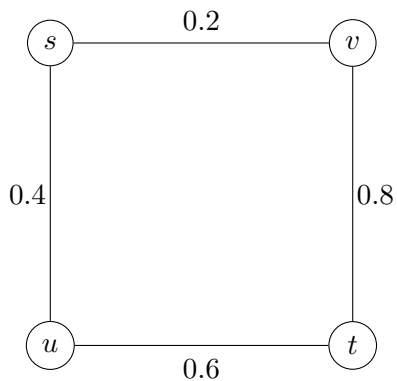
Figure 2.2: A network with 4 nodes and 4 edges.

2.4 Diameter-constrained reliability metrics

The structure function Φ is defined just in terms of connectedness for the CLR. But in several contexts, the distance between nodes can be relevant too and affect the way Φ is defined. In [7] the authors introduced a model that subsumes the CLR model by adding an upper limit for the distance between terminals. Similarly to what we did for the CLR, we next define the DCR (diameter-constrained reliability). Again, recall that we refer as DCR both to the context where such upper limit is imposed as well as to the specific problem of computing the diameter-constrained network reliability.

Config. X		Pr(X)	Node Set K		
			{s,t}	V	{s,t,u}
1		0.0384			
2		0.0096			
3		0.0256			
4		0.0064			
5		0.0576			
6		0.0144			
7		0.0384	✓		
8		0.0096	✓		✓
9		0.1536	✓		
10		0.0384	✓	✓	✓
11		0.1024	✓		
12		0.0256	✓	✓	✓
13		0.2304	✓	✓	
14		0.0576	✓	✓	✓
15		0.1536	✓	✓	✓
16		0.0384	✓	✓	✓
		1.0000	**0.3616**	**0.2848**	**0.3232**

Figure 2.3: An example of three CLR measures (same network, different terminal set).

Definition 18. *Given a probabilistic graph $G = (V, E)$, two nodes $\{s, t\} \in V$ and an integer $d > 0$, we define as $\{s, t\}$-DCR the problem of computing the probability $R(G, \{s, t\}, d)$ that the nodes s and t are d-connected.*

Definition 19. *Given a probabilistic graph $G = (V, E)$ and an integer $d > 0$, we define as* all-terminal-DCR *the problem of computing the probability $R(G, V, d)$ that G is d-connected (i.e. it has a diameter value that is lower or equal to d).*

Definition 20. *Given a probabilistic graph $G = (V, E)$, a subset $K \in V$ with $|K| \geq 1$ and an integer $d > 0$, we define as K-DCR the problem of computing the probability $R(G, K, d)$ that the graph is d-K-connected.*

From these definitions, it follows that a DCR problem where $d \geq n - 1$ is equivalent to the corresponding CLR problem. But, when $d < n - 1$, there can be configurations where the terminal set is connected but not d-connected. This results in the following

Property 1. *For every probabilistic graph $G = (V, E)$, set $K \in V$ and integers $1 \leq d \leq d'$, it holds that $R(G, K, d) \leq R(G, K, d')$.*

For illustration purposes, we recall the example network of Figure 2.2 and compute the corresponding DCR measures for a value of $d = 2$ in Figure 2.4.

Observe that, in this case, the DCR and CLR coincide for the two-terminal problems, but the DCR is lower than the CLR both for the all-terminal as well as the K-terminal problems.

		Node Set K		
Config. X	Pr(X)	{s,t}	V	{s,t,u}
1	0.0384			
2	0.0096			
3	0.0256			
4	0.0064			
5	0.0576			
6	0.0144			
7	0.0384	✔		✔
8	0.0096	✔		✔
9	0.1536			
10	0.0384	✔		
11	0.1024			
12	0.0256	✔		
13	0.2304			
14	0.0576	✔		
15	0.1536	✔		✔
16	0.0384	✔	✔	✔
	1.0000	**0.3616**	**0.0384**	**0.2400**

Figure 2.4: An example of three DCR measures (same network, different terminal set).

Part II

Computational complexity

Chapter 3

The computational complexity of DCR metrics

3.1 About the analysis of computational complexity

3.1.1 Computational problems

In this part of the thesis we are concerned with concepts like the "effort" or "difficulty" of computing the DCR; as well as several subproblems derived from it (specific cardinalities of K, diameters d, topologies, etc.). Let us start by making these concepts precise. The computational complexity theory sets a formal framework to classify computational problems into several classes, according to their inherent difficulty. Computational problems are formalizations of the concept of "problem". A computational problem can be viewed as a collection of pairs <instance-of-the-problem, solution-for-it>. There are several ways for specifying the instances and their solutions, whose nature are deeply related; in general they consist of strings defined over a certain alphabet. The simplest one (in terms of the alphabet) is the binary string defined over the alphabet $\{0, 1\}$. An example of a computational problem is that of determining wheter a given positive integer number is prime or not. In this case, the instances are integer numbers (that can be represented as binary strings using base-two notation), and the solutions are {yes, no} (also, representable as single-symbol binary strings). There are different categories of computational problems according to the possible anwers or outcomes. The simplest ones are the *recognition problems*, also known as *decision problems*, for which the answer is binary (yes/no), as in the "is it prime?" example. Decision problems can be seen as formal languages, where the strings that belong to the language correspond to (and only to) the problem instances for which the answer is "yes". A broader category of problems is that of the *function problems*, where the set of possible solutions has more than two members. Consider the problem of finding the minimum-cost spanning tree of a given connected, undirected graph, with cost labels in each edge. In this case each instance can be specified by a binary string that encodes the adjacency matrix of the graph and each edge cost; the solution to the instance will be some binary string representing the

edges that build the optimal spanning tree. Observe that the function problem of finding the square of an integer i, is equivalent to the decision problem of determining wheter (i, j) are such $i^2 = j$ for every pair of integers i, j, in the sense that once an algorithm is available that solves one of these problems, we can easily cast an algorithm for solving the other problem (although nothing is yet said about the relative efforts). *Optimization problems* are not different from function problems; the mentioned problem of the minimum-cost spanning tree is equivalent to the following decision problem: determine if a (graph, subtree) pair (G, T) is such that no subtree T' of G exists whose cost is lower than that of T. A very important question about computational problems is that of the *tractability*. A problem is said to be tractable if there is an algorithm that returns a solution for every instance with a finite amount of effort (time, number of steps, amount of memory or any other measure of complexity). Note that nothing is said about the amount of effort (other than being finite for all instances). It is well known that not all problems are tractable. Kurt Gödel [22] proved that there exist problems for which no such algorithm exist via his two famous incompleteness theorems. The first one states that no consistent system of axioms can prove all arithmetic truths that hold for the natural numbers; whilst the second one extends the first one to prove that such a system cannot prove its own consistency.

3.1.2 Computational complexity

Besides the theoretical question of whether a certain problem is tractable or not, the effort undergone to solve a given problem is very relevant for practical applications; in particular, how it evolves as the instances of the problem become larger. One possible definition for the concept of "instance size" is the length of the binary string that represents the instance. Then, computational complexity theory deals with analyzing the solution time, space and other resource-consumption metrics that a machine spends to solve a problem, as a function of the instance size. Several models of theoretical machines have been defined to formalize such metrics; the most famous one is that of the *Turing machine*, introduced in [23]. It is a device that reads and writes symbols contained on a tape with discrete slots. Each operation performed (reading a symbol, moving the tape one slot, etc.) adds one unit to the total "time". The Turing machine is then a theoretical model that allows to make precise the definition of resource metrics as time and space, and are used to classify problems into several classes of computational complexity. The maximum number $t(x)$ of basic steps that the Turing machine performs, to answer yes/no for a decision problem instance of size x, defines the time complexity of the problem; therefore it is said that the problem *takes time $t(x)$ to be solved*. The nature of the function $t(x)$ defines the most usual complexity classes. The most prominent classes of decision problems are:

- P: These are the decision problems that can be solved, in polynomial time, by a deterministic Turing machine (a machine where there are precise and deterministic rules of what to do according to the tape current position and the symbol stored on it). Because of their polynomial time, these problems are usually seen as "effi-

ciently solvable"; at least by contrast to those problems for which the best-known algorithms have exponential times.

- $\mathcal{N}P$: These are the decision problems that can be solved, in polynomial time, by a non-deterministic Turing machine (an imaginary machine where there can be many transitions associated with each tape position and symbol combination; all possible courses of action are taken at the same time, "forking" the execution flow into many simultaneous replicas of the machine). This class of problems contain the class P. There are many common $\mathcal{N}P$ problems for which no polynomial-time algorithm is known.

It is believed, although not yet proved, that the class $\mathcal{N}P$ is different than P, i.e. that there are problems that belong to $\mathcal{N}P$ but not to P. To compare different classes, the notions of *hardness* and *completeness* are used. A problem X is said to be *hard* for a class C (or C-hard) if all problems in C can be reduced to X. Here, "reduced" means that the problems in C can be transformed into X with a number of steps that respects certain restrictions; the most used one (and throughout the rest of this work) is that of being polynomial in the instance size. If X is hard for the class C and at the same time X is a member of C, then X is said to be *complete* for C (or C-complete). One can think of these problems as the most difficult to solve within the class C. There are problems that are $\mathcal{N}P$-complete, as first shown in [24] with the boolean-satisfiability SAT theorem. Observe that, if a problem X that is C-complete for a certain class C can be reduced to another problem Y, then Y is also C-complete; this is a recurrently used procedure to classify problems. Several compendia of known $\mathcal{N}P$-complete problems exist (and for other classes too); the first one published was Karp's 21 $\mathcal{N}P$-complete problems in [25].

Other commonly used complexity classes are those for counting problems. A problem X is said to be a *counting problem*, if its solution is a non-negative integer, that corresponds to the number of alternate paths that a certain non-deterministic Turing machine can follow that lead to a solution. In other words, they can be stated as "given an instance x, determine how many y exist such that (x, y) is recognized by a certain non-deterministic Turing machine". For example, counting the number of spanning trees of a given undirected graph, or counting the number of different prime factors of a given integer, are both counting problems. It follows from its definition that every counting problem is associated to a decision problem (in the examples, determining if a certain graph is a subtree of the input graph, and determining if a certain integer is a prime factor of the input integer). The complexity class of the counting problems whose underlying decision problem is $\mathcal{N}P$, is known as #P (sharp-P); similarly, #P-complete denotes those counting problems whose underlying decision problem is $\mathcal{N}P$-complete.

3.2 Known results on the complexity of reliability metrics

3.2.1 Known results for CLR

The problems of reliability analysis can be related to discrete combinatorial problems, by means of the special case where all edges have the same reliability r. Observe that in this case, computing $R(G, K)$ consists of evaluating the following *reliability polynomial*:

$$R(G, K) = \sum_{i=0}^{m} F_i r^{m-i} (1 - r)^i.$$

Finding this polynomial is denoted as the f-Rel problem in [11]. Each coefficient F_i is the number of operational configurations that have exactly i failing edges. So, the determination of each coefficient F_i is a counting problem, thus at least as difficult as their associated decision problem - determining whether a certain configuration with exactly i failing edges is operational or not. It is easy to see that the latter belongs to the P class for CLR and DCR problems; e.g. it suffices with doing breadth-first-searches between every pair of terminals, which can be done in $|K|(|K| - 1)/2O(m) = O(m^3)$ time. But, as we see below, counting many F_i is an $\mathbb{N}$P-hard problem. When edge probabilities can differ, yet being rational numbers, the problem is denoted as r-Rel in [11]; there it is proved by polynomial-time reduction that, if f-Rel is $\mathbb{N}$P-hard, then r-Rel is $\mathbb{N}$P-hard too. As a generalization of r-Rel, the same applies then to the CLR problem (where edge reliabilities can be different and any real number in $[0, 1]$). The authors then present the following theorem from which they later derive several results for CLR complexity.

Theorem 3.1. *(Ball, Colbourn, Provan 1992): If any of the following hold, then f-Rel and r-Rel are $\mathbb{N}$P-hard:*

- *The minimum cardinality pathset recognition problem is $\mathbb{N}$P-hard;*

- *The minimum cardinality pathset counting problem is $\mathbb{N}$P-hard;*

- *The minimum cardinality cutset recognition problem is $\mathbb{N}$P-hard;*

- *The minimum cardinality cutset counting problem is $\mathbb{N}$P-hard;*

- *The problem of determining a general coefficient of the reliability polynomial is $\mathbb{N}$P-hard.*

Table 3.1, taken from [11], summarizes the known results for each condition in Theorem 3.1 and for the two-terminal, K-terminal and all-terminal CLR problems. Therefore, by virtue of the theorem, the three versions of the CLR happen to be $\mathbb{N}$P-hard. The case "min. card. pathset rec." vs. "all-term", not justified in [11], consists of counting the number of spanning trees of the graph G. This can be done in polynomial time by virtue of Kirchoff's matrix tree theorem using the non-zero eigenvalues of the Laplacian matrix; see [26].

Table 3.1: Complexity results related to Theorem 3.1; (* means polynomial, ! means NP-hard

.	two-term	K-term	all-term
min. card. pathset rec.	*[25]	![27]	*
min. card. cutset rec.	*[28, 29]	*[28, 29]	*[28, 29]
min. card. pathset count.	*[25]	![30]	*[31]
min. card. cutset count.	![32]	![32]	*[30, 33]
gen. term. count.	![25]	![34]	![32]

The problem of computing the CLR remains NP-hard even when restricted to certain stuctured networks. Provan [35] shows that the two-terminal CLR problem remains NP-hard for the (very restrictive) condition that the graph is planar and has node degrees bounded by three. The high complexity of the CLR problem has motivated intense research on developing bounds and estimators; exact computations are feasible for networks that belong to certain topologies only, or with sizes that allow applying the best available methods yet having exponential times.

3.2.2 Known results for DCR

As a generalization of the CLR problem, the two-terminal, K-terminal and all-terminal DCR problems are NP-hard too. The addition of a bounded-length constraint increases considerably the complexity of several problems on graphs. For example, the min-max equality of Menger's theorem and the Dantzig-Fulkerson's max-flow min-cut theorem do not hold anymore, when bounds are imposed to the path lengths. Also, in [9], the authors summarize the complexity of the decision problems asking for the existence of edge-disjoint pathsets and cutsets with a given cardinality j for a given diameter constraint d. They show that these problems are NP-hard when one or both of j and d are considered as inputs, and belong to the FTP complexity class[1] when both are parameters. Regarding particular cases of the DCR, it is proved in [10] that computing the d-K-DCR is NP-hard when the diameter $d > 2$ and $|K| \geq 2$ is any fixed number of nodes. They also show that for complete topologies where the edge set are partitioned using a certain pattern that allows at most four equi-reliable classes, the two-terminal DCR can be computed in polynomial time.

The following list resumes what was known about the complexity of the DCR when restricting $k = |K|$ and d to have certain values.

- $k = 2$ and $d = 2$: polynomially solvable [10];

- any k and $d \in [3, n-1]$: NP-hard [10];

---

[1]FTP is the complexity class FTP of the parameterized problems taking time $O(f(p)g(i))$, where f is any function on the parameters p and g is any polynomial on the inputs i

- any k and $d \geq n - 1$: $\mathcal{N}$P-hard [11] since the problem is equivalent to the CLR.

The following cases were not yet covered by existing literature:

- The DCR subproblem with K-diameter two and any fixed $|K|$;

- The DCR subproblem with $|K|$ as an input for the complexity analysis.

In the remaining sections of Part I, we contribute to the knowledge on the complexity of computing the DCR, by introducing results for the first case.

Chapter 4

The DCR subproblem with K-diameter two and any fixed $|K|$

In this section we prove that the 2-K-DCR belongs to the FTP complexity class when considering $|K|$ as a parameter; in particular it is linear in $\min\{|V|, |E|\}$ when $|K|$ is fixed. Recall that FTP is the complexity class of the parameterized problems taking time $O(f(p)g(i))$, where f is any function on the parameters p and g is any polynomial on the inputs i. First, in Section 4.1 we develop the case where $|K| = 3$. Then we generalize it in Section 4.2 for any $|K|$, by finding a closed formula for computing the $R(G, K, 2)$ and showing that it can be computed in a time linear in n. Then, in Section 4.3, we present an alternative demonstration, based on finding a recursive formulation for $R(G, K, 2)$ and showing that it is indeed linear in n, i.e, $O(n)$. Finally, in Section 4.4 we comment on the case where K is a free input for the complexity analysis, whose complexity has not yet been determined.

An article based on the recursive formulation above mentioned for the case $d = 2$ and fixed K was published in the journal International Transactions in Operational Research; see [13].

4.1 Computing the 2-DCR with $|K| = 3$

Let us start analyzing the subproblem of the DCR where the terminal set is $K = \{1, 2, 3\}$ and the diameter parameter $d = 2$. The only edges whose state is important when computing $R(G, K, 2)$ are those that link two nodes of K and those that link a vertex of K to a vertex of $V \setminus K$. Let us call *central nodes* the set of nodes $V \setminus K$ and denote it as $V \setminus K = \{b_1, b_2, \ldots, b_s\}$, being $s = n - 3$. Note that we can consider that every central node is linked to every terminal node by adding to G edges with probability of operation equal to zero. Recall from Section 2.2 that every network configuration x defines a random event $(X = x)$. Then, let us define:

- for every central node b_h and subset $T \subseteq K$, we define $e_{h,T}$ as the event where T is exactly the set of terminals that are linked to b_h by operational edges. We define

31

also $e_{h,*}$ as the conjunction of all states $e_{h,T}$ with $|T| \leq 1$. Then, in what follows, we can partition the events for each central node h as $\{e_{h,123}, e_{h,12}, e_{h,13}, e_{h,23}, e_{h,*}\}$;

- for every edge (u,v) linking two nodes of K we define $e_{uv,1}$ and $e_{uv,0}$ as the events where the edge is operational or failing respectively.

Observe that the events above defined for different central nodes are independent, and so are the events defined for different edges within K. Let us introduce the following expression F to generate all possible combinations of these events. For ease of notation, $(+)$ and $(\cdot)$ denote the disjunction and conjunction operators respectively.

$$
\begin{aligned}
F =& (e_{12,0} + e_{12,1})(e_{13,0} + e_{13,1}) \cdot \\
& \cdot (e_{23,0} + e_{23,1})(e_{1,*} + e_{1,12} + e_{1,13} + e_{1,23} + e_{1,123}) \cdots \\
& \cdots (e_{m-1,*} + e_{m-1,12} + e_{m-1,13} + e_{m-1,23} + e_{m-1,123}) \cdot \\
& \cdot (e_{m,*} + e_{m,12} + e_{m,13} + e_{m,23} + e_{m,123}) = \\
& \prod_{u \neq v \in K} (e_{uv,0} + e_{uv,1}) \prod_{h=1}^{s} (e_{h,*} + e_{h,12} + e_{h,13} + e_{h,23} + e_{h,123})
\end{aligned}
$$

The function F is a multinomial whose development indicates every possible event of the system (among those defined in the above bullet points), that is relevant for computing $R(G, K, 2)$. By virtue of the independence hypothesis, the probability of any monomial will be the product of the probabilities of its factors.

Now let us analyze the events where the network has K-diameter two. Let us denote as $t(u, v)$ the lowest central node that is connected to both u, v by operational edges (where "lowest" refers to any arbitrary labeling of the central nodes $b_1 < b_2 < \cdots < b_s$). Then we can partition those states as follows:

- (a) at least two edges (u, v) with $u, v \in K$ are operational;

- (b) one and only one edge (u, v) with $u, v \in K$ is operational
  - (b.1) there is at least one central node h for which it holds that $e_{h,123}$;
  - (b.2) it holds that $e_{12,1} \wedge t(13) < t(23)$;
  - (b.3) it holds that $e_{12,1} \wedge t(23) < t(13)$;
  - (b.4) it holds that $e_{13,1} \wedge t(12) < t(23)$;
  - (b.5) it holds that $e_{13,1} \wedge t(23) < t(12)$;
  - (b.6) it holds that $e_{23,1} \wedge t(12) < t(13)$;
  - (b.7) it holds that $e_{23,1} \wedge t(13) < t(12)$;

- (c) no edge (u, v) with $u, v \in K$ is operational
  - (c.1) there is at least one central node h for which it holds that $e_{h,123}$;

- (c.2) $t(12) < t(13) < t(23)$;
- (c.3) $t(12) < t(23) < t(13)$;
- (c.4) $t(13) < t(12) < t(23)$;
- (c.5) $t(13) < t(23) < t(12)$;
- (c.6) $t(23) < t(12) < t(13)$;
- (c.7) $t(23) < t(13) < t(12)$.

The events (a) will be:

$$(a) = (e_{12,1}e_{13,1}e_{23,1} + e_{12,1}e_{13,1}e_{23,0} + e_{12,1}e_{13,0}e_{23,1} + e_{12,0}e_{13,1}e_{23,1})$$
$$\prod_{h=1}^{s}(e_{h,*} + e_{h,12} + e_{h,13} + e_{h,23} + e_{h,123})$$

The events (b.1) can be obtained by the following aggregation of disjoint terms, based on partitioning according to the first central node that is connected to the three nodes of K:

$$(b.1) = (e_{12,1}e_{13,0}e_{23,0} + e_{12,0}e_{13,1}e_{23,0} + e_{12,0}e_{13,0}e_{23,1})$$
$$\sum_{k=1}^{s}\left\{\prod_{h=1}^{k-1}(e_{h,12} + e_{h,13} + e_{h,23} + e_{h,*})e_{k,123} \prod_{h=k+1}^{s}(e_{h,123} + e_{h,12} + e_{h,13} + e_{h,23} + e_{h,*})\right\}$$

The events (b.2) can be obtained by another aggregation of disjoint terms, based on partitioning according to the possible pairs (i, j) of central nodes that are the lowest ones connected respectively to 13 and 23 in K:

$$(b.2) = e_{12,1}e_{13,0}e_{23,0}$$
$$\sum_{1 \leq i < j \leq s}\left\{(\prod_{h=1}^{i-1}(e_{h,*} + e_{h,12})e_{i,13}\right.$$
$$\left.(\prod_{h=i+1}^{j-1}(e_{h,*} + e_{h,12} + e_{h,13}))e_{j,23}(\prod_{h=j+1}^{s}(e_{h,*} + e_{h,12} + e_{h,13} + e_{h,23}))\right\}$$

The events (b.3) to (b.7) can be obtained analogously to (b.2). The events (c.1) are similar to (b.1):

$$(c.1) = (e_{12,0}e_{13,0}e_{23,0})$$
$$\sum_{k=1}^{s}\left\{\prod_{h=1}^{k-1}(e_{h,12} + e_{h,13} + e_{h,23} + e_{h,*})e_{k,123} \prod_{h=k+1}^{s}(e_{h,123} + e_{h,12} + e_{h,13} + e_{h,23} + e_{h,*})\right\}$$

The events (c.2) can be obtained by another aggregation of disjoint terms, based on partitioning according to the possible triads (i, j, k) of central nodes that are the lowest

ones connected respectively to 12, 13 and 23 in K:

$$(c.2) = e_{12,0}e_{13,0}e_{23,0}$$

$$\sum_{1 \leq i < j < k \leq s} \left\{ (\prod_{h=1}^{i-1} e_{h,*})e_{i,12}(\prod_{h=i+1}^{j-1} (e_{h,*} + e_{h,12}))e_{j,13} \right.$$

$$\left. (\prod_{h=j+1}^{k-1} (e_{h,*} + e_{h,12} + e_{h,13}))e_{k,23}(\prod_{h=i+1}^{j-1} (e_{h,*} + e_{h,12} + e_{h,13} + e_{h,23})) \right\}$$

Finally the events (c.3) to (c.7) can be obtained analogously to (c.2). Now, to compute $R(G, K, 2)$ we must add the probability of the events (a), (b.1) ... (b.7) and (c.1) ... (c.7). Translating the previous expressions into probabilities is straightforward since the unions are disjoint and the intersections involve independent events. There are no more than $2^3 = 8$ combinations of events $e_{uv,01}$. Regarding each central node b_h, there are no more than 2^5 non-elementary events determined by the union of some of $e_{h,*}$, $e_{h,12}$, $e_{h,13}$, $e_{h,23}$ and $e_{h,123}$. So there is a number of building blocks for the previous expressions that can be computed with an effort that is $O(s)$. Then, the probabilities for (a), (b) and (c) can be computed with efforts that are:

- (a) $O(1)$ (computing the product operator is needless since it will always be equal to one);

- (b.1) $O(s^2)$ (dominated by a summation of s terms, each involving a product of s factors);

- (b.2) $O(s^2)$ (dominated by a summation of $s(s-1)$ terms, each involving s factors);

- (c.1) $O(s^2)$ (similar to b.1);

- (c.2) $O(s^3)$ (dominated by a summation of $\binom{s}{3} = s(s-1)(s-2)/6$ terms with s factors).

We have therefore shown that $R(G, K, 2)$, with $|K| = 3$, can be computed in a time that is polinomial in $s = |V \setminus K| = n - 3$ i.e. polinomial in n. In the next two sections, we generalize these ideas to demonstrate that this holds true for any value of $|K|$ that is fixed, i.e. where $|K|$ is not an input but a parameter for the complexity analysis; and that it is indeed linear in n.

4.2   A proof of linearity for any (fixed) $|K|$

Let us start by introducing some notation elements. In this section we work with a complete graph $G = <V, E>$, where the edges xy that correspond to real links have a probability of operation $p(xy)$, and those that do not correspond to real links are assigned $p(xy) = 0$. Additionally:

- given $K \subseteq V$, a *2-pathset* is any $E' \subseteq E$ such that the partial graph (V, E') is 2-K-connected;

- we denote by s the number of nodes not in K, that is, $s = |V \setminus K| = n - |K|$;

- we denote by $\mathcal{O}_K^2(E)$ the set of 2-pathsets determined by E and K (following the notation of [8]);

- we denote by $X^{\{m\}}$ the set of all subsets of a certain set X that have m different elements, that is, $X^{\{m\}} = \{Y \subseteq X : |Y| = m\}$;

- we denote by $X^{(m)}$ the set of all m-tuples built with different elements of a certain set X, that is, $X^{(m)} = \{Y \in X^m : i \neq j \rightarrow Y_i \neq Y_j\}$;

- we denote by $\otimes$ the binary operator defined as:
  $(a_1, a_2, \ldots, a_n) \otimes (b_1, b_2, \ldots, b_n) = \{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$;

- we call *requirement* to any set of two nodes $\{x, y\} \subseteq K^{\{2\}}$ and denote it by xy. We say that a requirement xy is *satisfied* if there is a path of length at most two, composed only by operational edges, that connects the two nodes xy;

- we denote by $\mathcal{P}(A)$ the powerset of a certain set A.

4.2.1 Demonstration plan

We start by finding an analytical expression for computing $R_K(G, 2)$. To do so, we partition $\mathcal{O}_K^2(E)$ in disjoint components whose probabilities are then computed and totaled. We build a function $f : \mathcal{O}_K^2(E) \rightarrow A$ (with a certain discrete codomain A conveniently chosen) and then compute the probability of the domain by totaling the probabilities of all preimages of A:

$$R_K(G, 2) = \Pr(\mathcal{O}_K^2(E)) = \sum_{a \in A} \Pr(f^{-1}(a)) \tag{4.1}$$

The set A is defined as

$$A = \bigcup_{\ell=0}^{s} \left((V \setminus K)^{(\ell)} \otimes \mathcal{P}(K)^{(\ell)} \right).$$

Each element of A is a set of pairs (t, C), where t is a node of $V \setminus K$ and C is a subset of K. We see each such element of A as a collection of sets of edges between nodes of $V \setminus K$ and K that belong to a 2-pathset (besides edges between nodes of K). Observe that any edge linking two nodes of $V \setminus K$ is irrelevant for the 2-K-connectivity, since they can not be part of a path of length below three connecting two nodes of K. The function f is defined later in Eq. (4.2). We first show that A can be built such that totaling the probabilities $\Pr(f^{-1}(a))$ involves a number of elementary operations that is polynomial in n; finally, we show that it is indeed linear in n.

4.2.2 Partitioning the pathsets $\mathcal{O}^2_K(E)$

We assume that there is a certain strict ordering within V. We say that a family $\mathcal{C} \subseteq \mathcal{P}(K)$ *covers (or is a cover of)* K for $F \subseteq K^{\{2\}}$, and denote it by $\mathcal{C} \sqsupset_F K$, if and only if for every requirement xy at least one of the following applies: (i) $xy \in F$; (ii) $\exists z \in K : \{xz, zy\} \subseteq F$; (iii) $\exists C \in \mathcal{C} : \{x, y\} \subseteq C$.

$$K = \{a, b, c, d\}$$
$$F = \{ab, ac, cd\}$$
$$\mathcal{C} = \{\{a, c\}, \{a, d\}, \{b, c, d\}\}$$
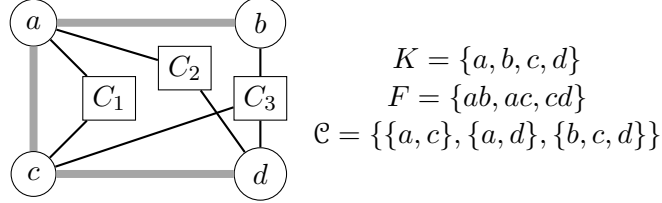
Figure 4.1: Example of cover set

Figure 4.1 illustrates the concept of covers. Thick lines represent the elements of F. Nodes connected by thin lines to the square nodes represent each element of $\mathcal{C} = \{C_1, C_2, C_3\}$. Condition (i) applies for the pairs ab, ac and cd. Condition (ii) applies for bc and ad. Condition (iii) applies for ac, ad, bc, bd and cd. Thus all pairs satisfy at least one of (i), (ii), (iii) and then $\mathcal{C} \sqsupset_F K$.

Observe that if $E' \in \mathcal{O}^2_K(E)$ then the family $\mathcal{C}_{E'}$ defined as the set

$$\mathcal{C}_{E'} = \{C_{E'}(t) : t \in V \setminus K\} \quad \text{being } C_{E'}(t) = \{x \in K : xt \in E'\},$$

covers K for $F = E' \cap K^{\{2\}}$, and we say that E' *generates* $\mathcal{C}_{E'}$. Conversely, given a cover $\mathcal{C}$ of K for F, if $|\mathcal{C}| \leq n - |K|$ then there is some E' that generates $\mathcal{C}$; for example $E' = \{xy \in E : xy \in F \vee (\exists C \in \mathcal{C} : \{x, y\} \subseteq C)\}$ is a 2-pathset. The preceding definitons relate with the operational states of the network in the following way. Assume that the elements of F correspond exactly to the edges linking nodes of K that operate. Assume also that each element $C \in \mathcal{C}$ represents a node in $V \setminus K$ whose neighbors, ignoring failing edges, are exactly the elements of C (e.g. C_1, C_2, C_3 in Figure 4.1). Then, $\mathcal{C} \sqsupset_F K$ implies that every requirement is satisfied. The powerset $\mathcal{P}(K)$ has $2^{|K|}$ elements (i.e. there are $2^{|K|}$ different subsets of K). Its powerset $\mathcal{P}(\mathcal{P}(K))$ has $2^{(2^{|K|})}$ elements that are sets of subsets of K. So, for the number of families $\mathcal{C}$ for which $\mathcal{C} \sqsupset_F K$, there is an upper bound $2^{(2^{|K|})}$ that depends only on $|K|$. Now, observe that when considering a family $\mathcal{C}_{E'}$, if there are two nodes $t \neq t' \notin K$ with $C_{E'}(t) = C_{E'}(t')$, one of t and t' can be removed from the graph G, yet obtaining the same family $\mathcal{C}_{E'}$. For example, in Figure 4.1, the addition of a node C_4 with exactly a and c as neighbors in K, would make no difference when producing $\mathcal{C}_{E'}$, due to the existence of a node C_1 with exactly the same neighbors in K. In general, given a path E' and one element C of the cover $\mathcal{C}_{E'}$ there are one or more $t \in V \setminus K$ such that $C_{E'}(t) = C$. Let us define $t_{E'}(C)$ as the minimum of them according to the ordering of V, that is

$$t_{E'}(C) = \min\{t \in V \setminus K : C_{E'}(t) = C\}.$$

Then, for every 2-pathset E', there is exactly one set $\{t_1, t_2, \ldots, t_\ell\}$ and one cover $\mathcal{C}_{E'} = \{C_1, C_2, \ldots, C_\ell\}$ for $F = E' \cap K^{\{2\}}$, such that $t_i = t_{E'}(C_i) \ \forall i = 1, \ldots, \ell$. Now we can define our function f as follows:

$$f(E') = \{(t_1, C_1), (t_2, C_2), \ldots, (t_\ell, C_\ell)\}. \tag{4.2}$$

4.2.3 Computing the probabilities

Given $x \in V \setminus K$ and $C \subseteq K$, let us denote as $P(x, C)$ the event where the set of neighbors of node x that belong to K and are connected by operational links is exactly C. Its probability, that we denote $p(x, C)$, is:

$$p(x, C) = \prod_{y \in C} p(xy) \prod_{y \in K \setminus C} (1 - p(xy)).$$

Now, given $C_1, \ldots, C_\ell \subseteq K$ and $t_1, \ldots, t_\ell \in V \setminus K$ such that $t_i < t_{i+1} \forall i = 1, \ldots, l-1$, let us define the event $P(t_1, \ldots, t_\ell, C_1, \ldots, C_\ell)$ as the event where the following statements hold true:

- the nodes $t_1, \ldots, t_\ell$ are connected exactly to $C_1, \ldots, C_\ell$ in K respectively;

- the nodes t with $t < t_1$ have no neighbors in K;

- the nodes t with $t_i < t < t_{i+1}$ have no neighbors in K or have exactly as neighbors one of $C_0, \ldots, C_i$ for $i \in \{0, \ldots, \ell - 1\}$;

- the nodes t with $t_\ell < t$ have no neighbors in K or have exactly as neighbors one of $C_1, \ldots, C_\ell$.

We denote its probability as $p(t_1, \ldots, t_\ell, C_1, \ldots, C_\ell))$, and so we have that

$$p(t_1, \ldots, t_\ell, C_1, \ldots, C_\ell) =$$

$$\left[\prod_{t < t_1} p(t, \emptyset) \right] p(t_1, C_1) \left[\prod_{t_1 < t < t_2} [p(t, \emptyset) + p(t, C_1)] \right] p(t_2, C_2)$$

$$\left[\prod_{t_2 < t < t_3} [p(t, \emptyset) + p(t, C_1) + p(t, C_2)] \right] p(t_3, C_3) \cdots$$

$$\left[\prod_{t_{\ell-1} < t < t_\ell} [p(t, \emptyset) + p(t, C_1) + \cdots + p(t, C_{\ell-1})] \right] p(t_\ell, C_\ell)$$

$$\left[\prod_{t_\ell < t} [p(t, \emptyset) + p(t, C_1) + \cdots + p(t, C_\ell)] \right]$$

To build a compact expression, let us define t_0 and $t_{\ell+1}$ as "virtual nodes" such that $t_0 < t_1$ and $t_\ell < t_{\ell+1}$; and $C_0 = \emptyset$. Then we have that:

$$p(t_1, \ldots, t_\ell, C_1, \ldots, C_\ell) = \left[\prod_{i=1}^{\ell} p(t_i, C_i) \right] \left[\prod_{i=0}^{\ell} \prod_{t_i < t < t_{i+1}} \sum_{j=0}^{i} p(t, C_j) \right]. \qquad (4.3)$$

Note that the terms in the addition $p(t, \emptyset) + p(t, C_1) + \cdots + p(t, C_j)$ correspond to the probabilities of events that are pairwise disjoint due to the "is exactly" condition in the definition of $P(x, C)$. Therefore, this addition results in the probability of the union event. It follows that the probability that a 2-pathset E' generates the cover $\{C_1, C_2, \ldots, C_\ell\}$ and the ℓ-tuple $(t_1, t_2, \ldots, t_\ell)$ with $t_{E'}(C_i) = t_i \quad \forall i \in 1, \ldots, \ell$ is:

$$p(\{E' : \mathcal{C}_{E'} = \{C_1, \ldots, C_\ell\} : t_{E'}(C_i) = t_i\}) = p(t_1, \ldots, t_\ell, C_1, \ldots, C_\ell). \qquad (4.4)$$

Finally, from Eq. (4.1) and Eq. (4.4) we have that:

$$R_K(G, 2) = \sum_{F \subset K^{\{2\}}} \left[\prod_{e \in F} p(e) \right] \left[\prod_{e \in K^{\{2\}} \setminus F} (1 - p(e)) \right]$$
$$\sum_{\substack{(C_1, \ldots, C_\ell) \in \mathcal{P}(K)^{(\ell)}: \\ \{C_1, \ldots, C_\ell\} \sqsupset_F K}} \sum_{\substack{(t_1, \ldots, t_\ell) \in (V \setminus K)^{(\ell)}: \\ t_i < t_{i+1}, i=1, \ldots, l-1}} p(t_1, \ldots, t_\ell, C_1, \ldots, C_\ell). \qquad (4.5)$$

where the second summation ranges over all covers of K for F having a number ℓ of elements between 0 and $\max(s, 2^{(2^{|K|})})$ (recall that the cover is a set and thus its elements are different).

4.2.4 Computational complexity

The first summation in Eq. (4.5) has $2^{\binom{|K|}{2}}$ terms. The second summation has no more than $2^{(2^{|K|})}(\ell!)$ terms with $\ell \leq 2^{(2^{|K|})}$. The third summation has $\binom{s}{\ell}$ terms. The product operands involve $\binom{|K|}{2}$ products. Computing $p(t_1, \cdots, t_\ell, C_1, \cdots, C_\ell)$ involves n products and a number of additions bounded by $(\ell+1)n$ since, denoting as τ_i the position of t_i' within the ordering of $V \setminus K$, the number of additions is equal to $2(\tau_2 - \tau_1 - 1) + 3(\tau_3 - \tau_2 - 1) + \cdots + \ell(\tau_\ell - \tau_{\ell-1} - 1) + (\ell+1)(n - \tau_\ell) = -2 - 3 - \cdots - \ell - 2\tau_1 - \tau_2 - \tau_3 - \cdots - \tau_\ell + (\ell+1)n < (\ell+1)n$. Hence the number of elemental operations (additions and products) needed to compute $R_K(G, 2)$ has order

$$2^{\binom{|K|}{2}} \binom{|K|}{2} 2^{(2^{|K|})} 2^{(2^{|K|})}! s^{2^{(2^{|K|})}} n(2^{(2^{|K|})} + 2) \qquad (4.6)$$

which is a polynomial in n of degree $2^{(2^{|K|})} + 1$, thus proving that the complexity is polynomial (recall that $s = n - |K|$). It is easy to see that an enumeration of the sets $\mathcal{P}(K^{(2)})$, $\{\mathcal{C} \sqsupset_F K\}$ and $(V \setminus K)^{(\ell)}$ can be done in a number of steps linear in

their cardinality; then the computational complexity order of the three summations is the number of terms involved. Now let us see that the complexity is, indeed, linear in n. To do so, it is enough to show that the rightmost summation of Eq. (4.5) can be computed in a time linear in n. This is because the remaining summations and product operators in Eq. (4.5) multiply the order by factors that depend only on $|K|$, thus being constant with regard to n. For ease of notation let us assume that $V \setminus K = \{1, \cdots, s\}$. Following Eq. (4.3), the rightmost summation of Eq. (4.5) coincides with the summation of all products that have the form

$$p(1, C_{a_1})p(2, C_{a_2})\ldots p(s, C_{a_s})$$

where $a_t \in \{0, 1, \ldots, \ell\}$ (recall that $C_0 = \emptyset$) and there are ℓ integers t_i with $0 < t_1 < \cdots < t_\ell \le s$ such that:

- $a_{t_i} = i \quad \forall i \in \{1, \ldots, \ell\}$,

- if $t < t_1$ then $a_t = 0$,

- if $t_i < t < t_{i+1}$ with $i \in \{1, \ldots, \ell - 1\}$ then $a_t \in \{0, \ldots, i\}$,

- if $t > t_\ell$, then a_t is any integer between 0 and ℓ.

These products can be associated to directed paths in the directed graph defined by:

$$\begin{aligned}
\vec{G} &= (\{1, \ldots, s\} \times \{0, \ldots, \ell\}^2, \vec{E}) \\
\vec{E} &= \{((t, a, b), (t+1, a', b')) : t \le s - \ell, t \le b + 1, 0 \le a' \le b + 1, b' = \max(a', b)\} \cup \\
&\quad \{((t, a, b), (t+1, a', b')) : s - \ell \le t \le s - \ell + b, \ell - (s - t) < a' \le b + 1, b' = \max(a', b)\}
\end{aligned}$$

that go from vertex $(1, 0, 0)$ to $(s, \cdot, \ell)$. Each vertex (t, a, b) is associated to the probability $p(t, C_a)$. The variable b cumulates the number of nodes of $t_1, \ldots, t_\ell$ already visited when t moves from $t = 1$ to $t = s$; while a represents the possible sets involved in the events $p(t, C_a)$. An example can be seen in Figure 4.2. Computing the rightmost summation of Eq. (4.5) can therefore be done by dynamic programming, proceeding from the vertices with the form $(s, \cdot, \ell)$ downwards to reach $(1, 0, 0)$. In each step, a value $\nu(t, a, b)$ is assigned to the vertex (t, a, b) as follows:

$$\nu(t, a, b) = \begin{cases} p(t, C_a) & \text{if } t = s, \\ p(t, C_a) \sum_{(t,a,b) \rightsquigarrow (t+1,a',b')} \nu(t+1, a', b') & \text{if } 0 < t < s. \end{cases}$$

Hence the number of operations for computing the rightmost summation of Eq. (4.5) will not exceed the number of edges of the graph $\vec{G}$, which is bounded by $(s-1)\ell^2$ times the maximum possible degree ℓ^2, that is $(s-1)\ell^4$. This is linear in n, which completes the proof.
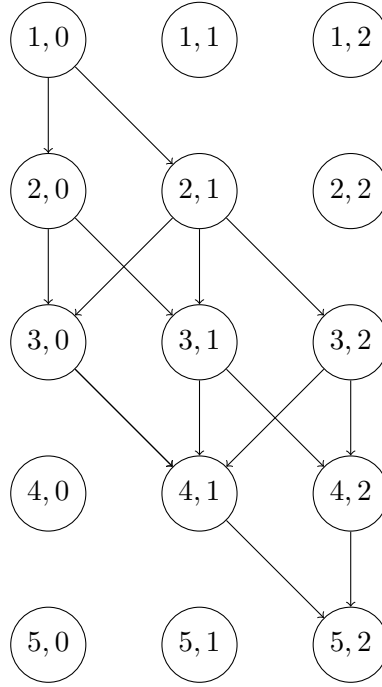
Figure 4.2: Auxiliary graph for computing $p(t_1, \ldots, t_5, C_0, \ldots, C_2)$

4.3 An alternative proof and algorithm: recursive formulation

In this section we present an alternative demonstration that the problem 2-DCR belongs to the FTP complexity class when considering $|K|$ as a parameter; in particular it is linear in $\min\{|V|, |E|\}$ when $|K|$ is fixed. We do so by means of a different approach than in Section 4.2; here we derive a recursive formulation for the 2-DCKR and show that computing it is a linear task when $|K|$ is fixed. The remainder of the section is organized as follows. Section 4.3.1 introduces the needed definitions and notation. Sections 4.3.2, 4.3.3 and 4.3.4 derive a formula to compute $R(G, K, 2)$ and introduce the corresponding algorithms. Section 4.3.5 provides two numerical examples.

4.3.1 Definitions and notation

We employ the following definitions and notations throughout the remaining of this section.

- As in the previous section, we define $s = |V \setminus K| = n - |K|$.

- $k = |K|$ is the number of non-terminal nodes.

- Given a set of partial graphs W of G, we denote as $\Pr(W)$ the probability of the

random event where the partial graph induced by the operational edges belongs to W.

- Given any set S, we denote by $S^{(2)} = \{\{u, v\} : u, v \in S \wedge u \neq v\}$ the set of all the unordered pairs of different elements of S.

- We say that two nodes are D2C (distance-two-connected) if they are connected by a path with length at most two. We also say that a node D2C (distance-two-connects) a pair of nodes if it is adjacent to both.

- We call *requirement* any element $p \in K^{(2)}$. A *met requirement* is one whose nodes are D2C.

- If T is a set of requirements ($T \subseteq K^{(2)}$) we denote as $V(T)$ the set of all nodes that occur in the definition of at least one requirement in T.

Consider the set of requirements being D2C in the partial graph obtained after discarding all edges that are incident to a non-terminal with index above a given j. Given $T \subseteq K^{(2)}$ we denote as A_T^j the event defined by such a set being exactly T. Formally,

- $\Phi_{u,v}^j$ is the binary random variable whose value is one if the nodes u and v are connected through a path of length at most two avoiding nodes $\{b_{j+1}, ..., b_s\}$, and zero otherwise;

- A_T^j is the event: "$(\forall \{u, v\} \in K^{(2)})(\Phi_{u,v}^j = 1 \iff \{u, v\} \in T)$".

So, the event A_T^j occurs whenever: (i) each requirement in T is met by a path of length one or whose central node is either a terminal or a non-terminal with index not above j; and (ii) the requirements in T are the only ones for which this holds true. We need also to consider the events in which, given sets $X \subseteq K$ and $Y \subseteq K^{(2)}$, a given non-terminal b_j is adjacent to all $u \in X$ and at the same time it does not D2C any requirement of Y. We denote this event as $B_{X,Y}^j$ and formally define it as follows:

- $B_{X,Y}^j = \{G' = <V, E'>:$
  $((\forall u \in X)(b_j, u) \in E') \wedge ((\forall \{v, w\} \in Y)(b_j, v) \notin E' \vee (b_j, w) \notin E')\}$.

As we see next these definitions allow us to recursively compute $R(G, K, 2)$. Note that the event $A = A_{K^{(2)}}^s$ includes all (and only) the configurations where all requirements are D2C in G, thus being exactly the event "the network is operational" in the 2-DCR context. So we have that $R(G, K, 2) = \Pr(A_{K^{(2)}}^s)$.

4.3.2 Computing $R(G, K, 2)$

Figure 4.3 shows the graph that correspond to a network G with four (bold) terminal nodes K and three (empty) non-terminal nodes $V \backslash K$. We can first of all reduce G by removing all edges connecting two non-terminals, since they will never be part of a
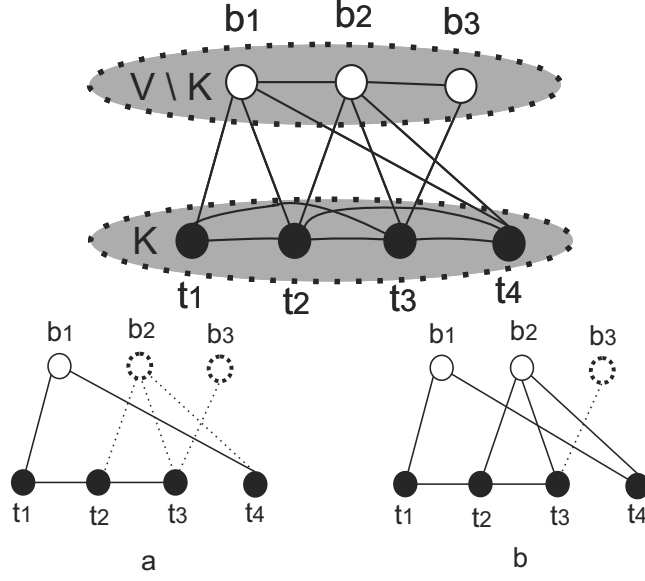
Figure 4.3: How the recursion works

path of length at most two connecting two terminals; edges incident to non-terminals with degree one can be also removed. Figure 4.3(a) depicts a possible partial graph G' yielded by a configuration x with only eight operational edges. As an example, let $T = \{\{t_1, t_2\}, \{t_1, t_3\}, \{t_1, t_4\}, \{t_2, t_3\}\}$; its elements are all (and only) the terminal pairs that are D2C in the subgraph of G' induced by $K \cup \{b_1\}$ (solid lines). So $G' \in A_T^1$. Similarly, if $T' = \{\{t_1, t_2\}, \{t_1, t_3\}, \{t_1, t_4\}, \{t_2, t_3\}, \{t_2, t_4\}, \{t_3, t_4\}\}$, observe that $G' \notin A_{T'}^1$ because there are no paths of length at most two in the subgraph of G' induced by $K \cup \{b_1\}$ connecting neither t_2 to t_4 nor t_3 to t_4. But as Figure 4.3(b) shows, when allowing also to employ the edges incident to b_2 the requirements in T' are met. In other words, $G' \in A_{T'}^2$.

Let us consider a certain partial graph G' that belongs to the event A_T^j. Some of its met requirements T might also be met after suppressing all edges incident to b_j, while other requirements might not be met anymore. Let Q be the set of the requirements that keep met after suppressing those edges. Then we have that $G' \in A_Q^{j-1}$ and thanks to the edges that connect b_j to the nodes $V(T \backslash Q)$ we also have that $G' \subseteq A_T^j$. This enables a way to recursively express A_T^j in terms of several events A_Q^{j-1} (being $Q \subseteq T$), whose unmet requirements must be met when considering also edges connecting K to b_j.

Lemma 4.1. *For all $0 < j \leq s$, the following recursive identity in A holds:*

$$A_T^j = \bigcup_{Q \subseteq T} \left(A_Q^{j-1} \cap B_{V(T \backslash Q), K^{(2)} \backslash T}^j \right) \qquad \text{(for $0 < j \leq s$)} \qquad (4.7)$$

Proof. Let us first prove that the left-hand side (LHS) is included within the right-hand

side (RHS). Let G' be an element of A_T^j. The requirements met by G' avoiding non-terminals above b_j can be partitioned as $Q, T \backslash Q$ being Q the set of requirements still met by G' when also avoiding b_j. So it is clear that $G' \in A_Q^{j-1}$. At the same time, G' has edges linking b_j to every node in $T \backslash Q$ (otherwise it would not belong to A_T^j due to unmet requirements), and b_j does not D2C any requirement in $K^{(2)} \backslash T$ (otherwise G' would not belong to A_T^j due to an excess of met requirements). Thus, G' belongs to the RHS. Let us now prove that the RHS is included within the LHS. Let G' be a configuration in the RHS with a certain Q. It meets the requirements Q as it belongs to A_Q^{j-1}. And as it also belongs to $B_{V(T \backslash Q), K^{(2)} \backslash T}^j$, it meets the requirements in $T \backslash Q$ without meeting any requirement out of T. Thus the set of requirements met by G' avoiding non-terminal nodes above b_j is exactly T, and then G' belongs to the LHS. $\qquad\square$

The following lemma is important for computing the RHS of Eq. 4.7.

Lemma 4.2. *The terms of the union in the RHS of Equation 4.7 are pairwise disjoint; and within each term, the events A and B are independent.*

Proof. Any two given terms of the union will involve two subsets of T (Q and Q') with $Q \neq Q'$. Then, for any configuration G', the set of requirements met by G' avoiding non-terminal nodes above $j - 1$ can be different from both Q and Q' (in which case G' does not belong to any of both terms) or coincide with only one of Q and Q' (in which case G' belongs to the corresponding term). Thus no configuration belongs to both terms and these are disjoint. Regarding independence, the term A_Q^{j-1} imposes conditions only on edges linking nodes within $K \cup \{b_1...b_{j-1}\}$ whereas any term with the form $B_{X,Y}^j$ imposes conditions only on edges incident to b_j. Independence follows from disjointness of both edge sets. $\qquad\square$

The leaves in the recursion tree are given by $j = 0$ i.e. when only considering edges that connect nodes of $K^{(2)}$. Denoting by E_K the subset of E involving only terminal nodes then we have that

$$A_T^0 = \{G' = < V, E_K' > / E_K' \subseteq E_K \wedge \\ \text{all and only the requirements in } T \text{ are met by } G'\}. \quad \text{(for } j = 0\text{)} \tag{4.8}$$

On one hand there are $2^{k(k-1)/2}$ different sets $T \subseteq K^{(2)}$ (which is a fixed number if k is a fixed parameter of the problem). On the other hand, computing the probabilities $\Pr(A_T^0)$ (that we will denote as P_T^0) involves adding the occurrence probability of each configuration of the subgraph of G induced by K. Then we have that the time τ_0 spent to compute all the probabilities P_T^0 for all T is bounded by some constant dependent on k and independent of n and m. Going back to Equation 4.7, by virtue of Lemma 4.2, the probability (denoted as P_T^j) that the event A_T^j occurs (with $0 < j \leq s$) can be computed as

$$P_T^j = \sum_{Q \subseteq T} \left(P_Q^{j-1} \Pr(B_{V(T \backslash Q), K^{(2)} \backslash T}^j) \right) \quad \text{(for } 0 < j \leq s\text{)} \tag{4.9}$$

Procedure B(j, X, Y)

1: **if** $\exists \{a, b\} \in Y/\{a, b\} \subseteq X$ **then**
2:     **return** 0
3: **else**
4:     $\hat{X} \leftarrow \{u \in V(Y)/(\exists v \in X/\{u, v\} \in Y)\}$
5:     $Y \leftarrow Y \setminus \{\{u, w\} \in Y/u \in X \cup \hat{X}\}$
6:     **return** $\left(\prod_{x \in X} r_{b_j x}\right) \left(\prod_{\hat{x} \in \hat{X}} (1 - r_{b_j \hat{x}})\right) C(j, Y)$
7: **end if**

Procedure C(j, Y)

1: **if** $Y = \emptyset$ **then**
2:     **return** 1
3: **else**
4:     $z \leftarrow$ any node from $V(Y)$
5:     $Y_1 \leftarrow Y \setminus \{\{z, v\} \in Y\}$
6:     $\hat{X} \leftarrow \{u \in V(Y)/(\{z, u\} \in Y)\}$
7:     $Y_2 \leftarrow Y_1 \setminus \{\{u, v\} \in Y/v \in \hat{x}\}$
8:     **return** $(1 - r_{b_j z}) C(j, Y_1) + r_{b_j z} \left(\prod_{\hat{x} \in \hat{X}} (1 - r_{b_j \hat{x}})\right) C(j, Y_2)$
9: **end if**

Figure 4.4: Pseudo-code for computing $B_{X,Y}^j$

and thus the sought reliability $R(G, K, 2) = P_{K^{(2)}}^s$. The event $B_{V(T\setminus Q), K^{(2)} \setminus T}^j$ involves a number of states bounded by 2^k and its probability can be computed in a time depending only on k. Figure 4.4 suggests a possible algorithm to compute $B_{X,Y}^j$. The value returned in line 6 is a product of three factors. The first factor involves edges that must be operational. The second factor involves edges that must necessarily fail. The third factor, obtained by means of an auxiliary algorithm $C(j, Y)$, computes the probability of the event C_Y^j defined as "the node b_j does not D2C the nodes of any requirement of Y". The latter is computed by sequentially considering the probability of operation and failure for each edge linking b_j and $V(Y)$ and invoking itself recursively with diminished sets Y_1, Y_2. It could be improved by first partitioning Y into components $Y_1 \cdots Y_h$ without common nodes pairwise and then invoking the algorithm for each one, finally returning the product of $C(j, Y_1) \cdots C(j, Y_h)$, with worst-case time $O(|Y_1|! \cdots |Y_h|!)$ (recall that $\sum_{i=1..h} |Y_i| \leq k$). The implementation used in our tests stores each $C(j, Y)$ in a map indexed by (j, Y) in order to compute it only the first time it is requested.

Now, note that to compute P_T^1 for a certain set T, a number of probabilities P_Q^0 (bounded by $2^{k(k-1)/2}$) are needed. The same holds true for all $j \in \{1 \cdots s\}$ when it comes to computing P_T^j (just knowing every P_Q^{j-1} is needed). Since the number of possible sets $T \subseteq K^{(2)}$ depends only on k, then the time τ_j spent computing these probabilities for a

---

Procedure Level0$(G' =< K, F >, \pi)$

1: $i \leftarrow 1; G'' \leftarrow Graph < K, \emptyset >; \rho \leftarrow \prod_{h=1..|F|}(1 - \pi_h)$
2: $(\forall Q \in 2^{K^{(2)}})(P_Q^0 \leftarrow 0)$
3: $level0Rec(G'', \rho, i)$
4: **return** P^0

Subprocedure Level0Rec$(G'' =< K, E' >, \rho, i)$

1: **if** $i \geq |F|$ **then**
2: $\quad Q \leftarrow$ all requirements met by G
3: $\quad P_Q^0 \leftarrow P_Q^0 + \rho$
4: **else**
5: $\quad Level0Rec(G'', \rho, i + 1)$
6: $\quad Level0Rec(G'' \cup \{f_i\}, \rho\pi_i/(1 - \pi_i), i + 1)$
7: **end if**

Figure 4.5: Level0 pseudo-code for computing P^0

certain j depends only on k. Thus, the algorithm will proceed downwards from $j = s$ to $j = 0$, computing $R(G, K, 2)$ in a time which is $O(n\tau)$ (linear in n, with $\tau = \max\{\tau_0...\tau_j\}$ depending only on k).

4.3.3 Computing the base cases

The algorithm `Level0` shown in Figure 4.5 computes the probabilities $\Pr(A^0_{Q \subseteq K^{(2)}})$ returning them in a map P^0 indexed by $Q \subseteq K^{(2)}$. When invoked it receives the subgraph $G' =< K, F >$ of G induced by the node set K and the vector $(\pi)_{1 \cdots |F|}$ of probabilities of operation of the edges of $F = \{f_1, \cdots, f_{|F|}\}$. An edgeless graph G'' is initialized with node set K and a variable ρ is initialized as its corresponding probability of occurrence (all edges failed). Line 2 initializes P^0 with all components equal to zero, ready to accumulate the probabilities of all configurations meeting each set of requirements Q. Then in line 3 the recursive subprocedure `Level0Rec` is invoked to perform a depth-first-search on G'', prior to returning P^0. This procedure completes a certain network configuration when it enters line 2, where the probability ρ of getting that particular configuration G'' is added to P_Q^0, being Q the subset of all requirements $(V(E'))^{(2)}$ that are D2C in G''. Lines 5 and 6 invoke recursively the same subprocedure, respectively for the cases where f_i is operational or failed.

4.3.4 The main algorithm

Our algorithm for computing $R(G, K, 2)$, shown in Figure 4.6, is a straightforward implementation of Equation 4.9. It receives the graph G, the set K and the edge's probabilities

---

Procedure RGK2($G =< V, E >, K, r$)

1: **for all** $(u \in K, v \in V \backslash K)/(u,v) \notin E$ **do**
2:     $E \leftarrow E \cup \{(u,v)\}$
3:     $r_{uv} \leftarrow 0$
4: **end for**
5: $G' =< K, F > \leftarrow$ subgraph of G induced by $K; \pi \leftarrow (r_{uv})_{(uv) \in F}$
6: $P_*^0 \leftarrow Level0(G', \pi)$
7: $p \leftarrow LeveljRec(K^{(2)}, |V| - |K|)$
8: **return** p

Subprocedure LeveljRec(T, j)

1: $p = 0$
2: **if** $j = 0$ **then**
3:     **return** P_T^0
4: **end if**
5: **for all** $Q \subseteq T$ **do**
6:     $\alpha \leftarrow LeveljRec(Q, j-1)$
7:     $\beta \leftarrow B(j, V(T \backslash Q), K^{(2)} \backslash T)$
8:     $p \leftarrow p + \alpha\beta$
9: **end for**
10: **return** p

Figure 4.6: Main algorithm pseudo-code for computing $R(G, K, 2)$

of operation r. In lines 1-4 all non-existent edges linking nodes among K and $V \backslash K$ are added to G with probabilities of operation equal to zero. Lines 5-6 invoke `Level0` to build P^0 with the probability of the base cases. Line 7 invokes the recursive subprocedure `LeveljRec` to compute $R(G, K, 2) = P_{K^{(2)}}^s$ as seen above (recall that $s = n - k$). Lines 2-4 of `LeveljRec` refer to the base cases where $(j = 0)$. When $j > 0$, lines 5-9 of the latter procedure accumulate the probability for each $Q \subseteq T$ making use of a recursive call on the set Q and non-terminals b_1 to b_{j-1}. As we did with $C(j, Y)$, in our implementation each P_Q^j is computed only the first time it is needed and stored for subsequent uses.

4.3.5   Numerical results

This section presents numerical results for two tests. The algorithms were implemented in `C++` and tested on an Intel Core2 Duo T5450 CPU machine with 2 GB RAM.

In the first test, the algorithm was run on the complete graphs $K_5 \cdots K_{20}$, both with four and five terminals, with all edges having failure probability 0.5. The results are

shown in Figure 4.7. As expected, both reliabilities grow when more non-terminal nodes are added and are higher when $|K|$ is lower (since less requirements must be met). Note, on one hand, the clearly linear progression of times following $|V|$; and on the other hand, the significant increase in times when dealing with one additional terminal node. The latter is due to the complexity inherent to a much larger number of subsets $T, Q \subseteq K^{(2)}$. The size of K affects more-than-exponentially this number of sets, thus significantly affecting the total time. This can be seen also in the closed formulae of Section 4.2, in particular in Eq. (4.6). This strong dependence on $|K|$ can severely limit the application of the algorithms in practice, when the set K is large.

The second test was run on a graph G with the Arpanet topology, with a set K of five nodes (square) distinguished as terminal, shown in Figure 4.8. Observe that some pairs of terminals (but not all) are at distance above two thus making $R_K(G, 2) = 0$. In order to allow communication in less than three "hops" between all terminals, we added from one to ten "virtual nodes" and linked them to every real node (the latter are those labeled $0 \cdots 19$). In this test the reliability of all edges was set to 0.9. Figure 4.9 shows how the network reliability increases as more virtual nodes are added, as well as the fairly linear progression of times that was obtained.

4.4 The case where $|K|$ is an input and $d \geq 2$

The case where $|K|$ is also considered as an input for the complexity analysis seems to be also an $\mathcal{N}$P-hard problem; although this has not yet been demonstrated. Even for the all-terminal particular case ($K = V$) the complexity remains unknown for all values of the distance parameter $d \geq 2$.

Given any $d \geq 2$, observe that the problem P_1 of computing the d-DCKR with $K = V$ and all edge reliabilities equal to 0.5 is equivalent to the problem P_2 of counting all the partial graphs $(V, E' \subseteq E)$ of G with diameter up to d. This is due to the fact that under such hypothesis every partial graph has the same probability of occurrence (2^{-n}) and each partial graph is $d - V-$connected if and only if it has diameter up to d; it follows that $R_V(G, d)$ equals the number of such partial graphs divided by 2^n. Then, proving that the problem P_2 is $\mathcal{N}$P-hard would suffice to prove that the d-DCKR with $K = V$ is $\mathcal{N}$P-hard too; since P_2 is a special case of the latter (where all edges are assigned reliabilities equal to 0.5). Furthermore, it would follow that the d-DCKR with K as a free input for the complexity would be $\mathcal{N}$P-hard, since so it is its particular case d-DCKR with $K = V$. Currently our efforts are headed towards proving that P_1 is an $\mathcal{N}$P-hard problem for different values of d. Unfortunately, we have not found yet results on the complexity of counting partial graphs with a given diameter.

4.5 Computational complexity: summary and conclusions

This section presented our contributions regarding the computational complexity of the DCR. We proved that the cases with diameter parameter $d = 2$ and fixed size $|K|$ for
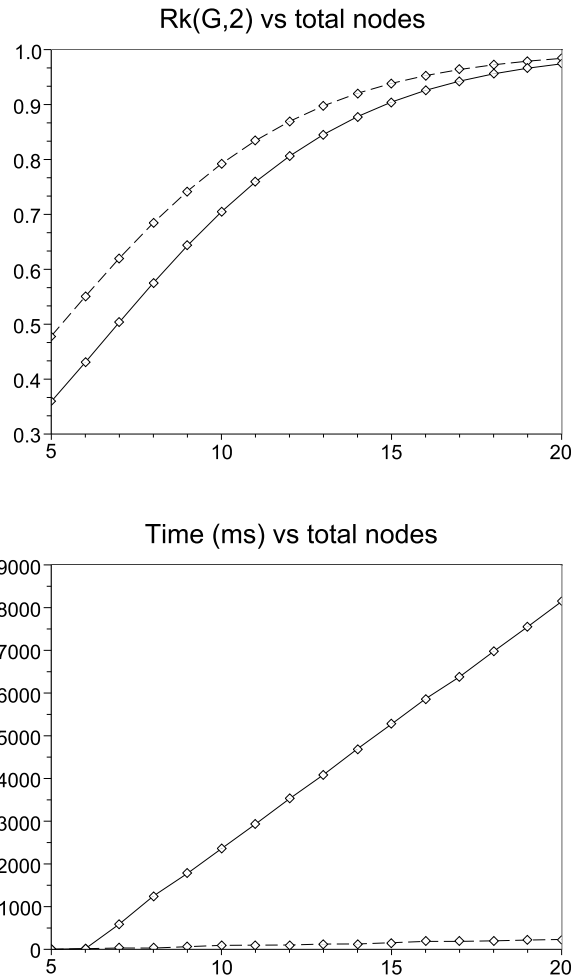
Figure 4.7: Numerical test I (dashed:$|K| = 4$; solid:$|K| = 5$)
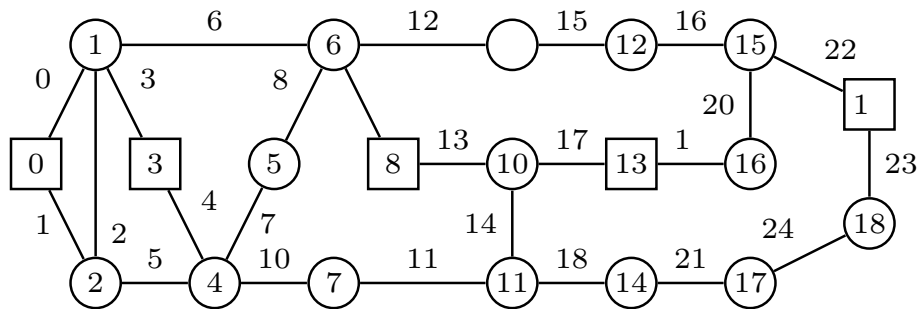
Figure 4.8: Numerical test II - Arpanet network
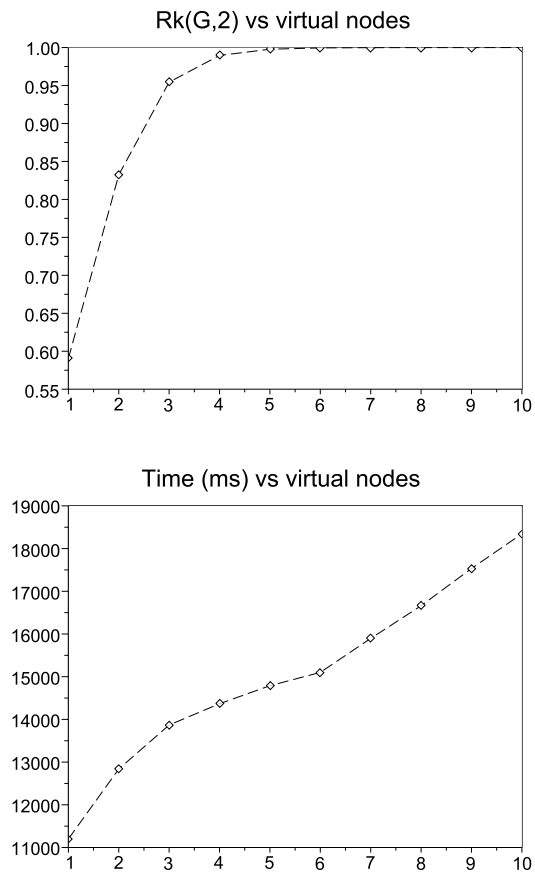
Figure 4.9: Numerical test II - Results

49

the terminals set can be computed in a time which is linear on the number of nodes $|V|$. Recall that it was already known that for $d > 2$ the problem is $\mathcal{N}$P-hard. Two proofs were given. The first one is based in combinatorial structures that we named covers and lead to a closed expression for computing $R(G, K, 2)$. The second one is based on a recursive formulation that allows for a dynammic programming algorithm; we presented an algorithm based on this recursive approach. The algorithm is valid also for non-fixed $|K|$ (i.e. considering $|K|$ as an additional input rather than a parameter) though taking exponential times in those cases. We implemented the recursive approach and run two families of numerical tests. They behaved as predicted, showing a clearly linear time progession when n grows. Although being linear in n when $|K|$ is fixed, the time is strongly dependent on $|K|$ (more than exponentially, as evidenced by its corresponding factors in the first proof), which can limit the applicability in instances with large sets K. It remains pending to classify the complexity of the case where $d \geq 2$ and K is either equal to V or free. Figure 4.10 resumes the complexity classes of the different subproblems for any value of d and fixed $|K|$, including references for tracking the works where each result was first presented.

Figure 4.10: Complexity of DCR subproblems for any d and fixed $|K|$

Part III

Exact and asymptotic analysis of DCR metrics

Chapter 5

Exact methods for reliability metrics

As we have already seen, computing CLR and DCR metrics is an $\mathcal{N}$P-hard task except for particular subclasses of these problems. In this section we intend to resume the basic algorithmic approaches that can be applied for computing the exact value of such metrics. These, in general, result in exponential-time algorithms for general cases, and polynomial ones for particular classes of the problems, given by specific topologies, terminal sets, distance parameters, uniformity of edge reliabilities, etc. We start by surveying known methods for CLR and DCR contexts, in Sections 5.1 and 5.2 respectively. Then, in Chapter 6, we illustrate, by means of simple examples, how dynamic programming techniques can be used to compute two-terminal metrics in polynomial time. We extend the examples to define a broader family of topologies, giving a polynomial algorithm for it too.

5.1   Exact methods for CLR

The most obvious way to compute $R(G, K)$ is to enumerate the possible operational configurations and total their probabilities, that is,

$$R(G,K) = \sum_{x:\Phi(x)=1} \Pr(X = x) = \sum_{x:\Phi(x)=1} \left(\prod_{i:x_i=1} r_i \right) \left(\prod_{i:x_i=0} (1 - r_i) \right).$$

This method can only be applied to small instances since the number of configurations grows exponentially with the number of edges m.

Recall the definition of *minpaths* and *mincuts* given in Section 2.1. The event "the network operates" is the (in general non-disjoint) union of all the events "minpath x operates" (the same applies to mincuts). Non-disjointness implies that we cannot simply add the probability of operation for all minpaths or mincuts. Several methods have been developed that compute the network reliability by means of a sum of disjoint products (SDP). They are based on expressing the event "the network operates" as a union of disjoint events, where each event is a conjunction of independent events, which allows to compute the overall probability as a SDP. A survey can be found in [36]. For example,

Poincare's formula of inclusion-exclusion can be applied either over all minpaths or all mincuts. Denoting as $\{\mu_1, \ldots, \mu_{MP}\}$ the list of all ordered minpaths, and as P_k the event "all links in μ_k are operational", we have that

$$R(G,K) = \sum_{i=1}^{MP} (-1)^{i-1} \sum_{1 \leq h_1 \leq \ldots \leq h_i \leq MP} \Pr(\bigcap_{j=1}^{m} P_{h_j}).$$

This formula contains 2^{MP-1} terms, and MP itself is usually exponential in the number of nodes or edges. But, as many terms cancel out, algorithms can be devised that generate only the remaining terms and then proceed simply totaling their probabilities; yet, in general, the number of these terms remains exponential in the size of the network. Several methods have been proposed that require the generation of all minpaths (e.g. [37, 38, 39, 40, 41, 42]). Other SDP methods that do not require generating all minpaths have also been proposed (e.g. [43]). A particular family of SDP methods is the multiple variable inversion (MVI) technique ([44, 45, 46]) that allows for further reduction on the number of summands. Another family is that of the binary decision diagram (BDD) based methods [47]. Dual approaches, based on mincuts instead of minpaths, can be applied, with the advantage that usually the number of minpaths in a graph exceeds that of its mincuts. For example, observe that in complete topologies and all-terminal context, the number of minpaths and mincuts are respectively n^{n-2} and $2^{n-1}-1$. Besides this, a drawback of using mincuts is that no current theory analogous to *domination* (see Section 5.1.2) exists that is based on mincuts.

5.1.1 Reduction techniques

Many techniques can be applied to reduce the size of the original instance of the problem. When these techniques involve low cost in terms of computational effort, they are usually worth to be embedded within general approaches. Some polynomial-time reductions that have been suggested[11, 48] include:

- deletion of *irrelevant* edges (those that do not belong to any minpath);

- contraction of *mandatory* edges (those that belong to all minpaths, e.g. bridges);

- *series-parallel* reductions [49];

- *polygon-to-chain* and *delta-Y* reductions [50, 51, 48];

- *s-attached* subgraphs replacement [52, 53];

- other divide-and-conquer approaches (e.g. decomposition in triconnected components [54]).

Although being unable to reduce general instances to polynomially-computable subinstances, there are several reasons why these reduction techniques are relevant. First, they can be embedded within general algorithms to reduce, when possible, the inherent subproblems, as we exemplify later for a factoring algorithm. Second, they can completely

solve certain topologies, which in practice is useful for building large instances whose reliability is exactly computable, that server for later analysis of the performance of bounding or estimation methods. Finally, they can be applied to compute lower and upper reliability bounds.

5.1.2 Factoring-based algorithms

The most powerful family of algorithms for the general CLR problem so far introduced exploit a property known as *factoring* or *pivotal decomposition*, first introduced in [55, 56]. Given a graph G and any of its edges $e = (i, j)$, let G_c be the graph obtained after replacing i and j by one single node, and after replacing every edge end incident to i or j by an edge end incident to the contracted node. This operation is called *contraction*; observe that it may generate loops and multi-edges. Similarly, let G_d be the graph obtained after dropping the edge e from G. Then, the factoring property states that, for any edge $e \neq (s, t)$ with reliability r_e,

$$R(G, \{s, t\}) = r_e R(G_c, \{s, t\}) + (1 - r_e) R(G_d, \{s, t\}).$$

In Section 5.2 we illustrate how the factoring algorithms proceed with an example in DCR context. The contraction operation can easily be defined for sets K with more than two elements. The straightforward recursive application of the pivotal decomposition leads to a binary tree with $2^m - 1$ nodes and 2^{m-1} leaves, that are trivial instances with reliability equal either to 1 or 0. An important property of this method is that it requires storage space that is $O(m^2)$, instead of the exponential order implied by precomputing all minpaths or mincuts for Boolean methods. The procedure to choose the edge to contract/delete (the *pivot*) heavily influences the performance of the algorithm, since it can lead to binary trees with different sizes (being $2^m - 1$ nodes the worst case). For example, the contracted graph G_c might be such that K gets reduced to one single node; its reliability is then equal to 1, and no further recursion is needed from G_c. Similarly, if G_d disconnects K, then its reliability is equal to 0 and no further recursion is needed from G_d. Intense research has been carried on to find good pivot selection strategies. Strategies based on the graph invariant known as the reliability domination were introduced by Satyanarayana and Prabhakar in [57] and later applied to the two-terminal reliability with series-parallel reductions in [58]. See also [59, 60, 61, 62, 63]. A variation of the dominant called *minimal domination* was used in [51] for polygon-to-chain reductions.

5.2 Exact methods for DCR

As a generalization, finding a general algorithm for computing the DCR metric is at least as difficult than doing the same for CLR. There are few particular cases for which exactly computing $R(G, K, d)$ is trivial. When the parameter $d = 1$, a configuration is operational if and only if every pair of terminals $\{i, j\} \in K^{(2)}$ corresponds to an

operational edge (i, j). Thus, the DCR can be computed in time $O(|K|^2)$ as follows:

$$R(G, K, 1) = \prod_{\{i,j\} \in K^{(2)}} r_{ij}.$$

Things are a bit more complex when $d = 2$. For two-terminals (s, t), every operational path consists of either a single edge $\{s, t\}$, or a two-edge path $s - u - t$ being u any node other than s and t. Hence, $R(G, K, d)$ can be computed as the probability of the complement of the event where all such paths fail [64]. Denoting by $\nu(w)$ the neighbors of node w in $G = < V, E >$, then

$$R(G, \{s, t\}, 2) = \begin{cases} 1 - (1 - r_{st}) \prod_{u \in \nu(s) \cap \nu(t)} (1 - r_{su} r_{ut}) & \text{if } \{s, t\} \in E \\ 1 - \prod_{u \in \nu(s) \wedge \nu(t)} (1 - r_{su} r_{ut}) & \text{if } \{s, t\} \notin E \end{cases}$$

which takes time $O(n)$. We have already seen that the same applies for $d = 2$, whatever the (fixed) size of K, although the constants involved grow exponentially with $|K|$ (at least for the methods we developed). Observe that the enumeration of paths with lengths up to d has order $O(|K|(|K|-1)/2\Delta^d)$, where Δ is the maximal node degree of the graph. So, for small values of d, the number of such paths could be handled with reasonable effort for problem instances that would be intractable in CLR context, by means of inclusion-exclusion or other Boolean procedures.

5.2.1   Reduction techniques in DCR context

Many reduction techniques that were presented in Section 5.1.1 cannot be applied in DCR context (at least without modifications), since they would affect the distance between terminals. Next we present some reduction techniques that can be applied to the two-terminal DCR; see [65, 66].

- *Deletion of irrelevant links.* An irrelevant link is any edge that does not belong to any d-pathset; see [66] for an efficient identification procedure.

- *Path-terminal reduction.* It can be applied whenever a terminal node s has degree one and is connected to a node v by means of a path $s - w_1 - w_2 - \cdots - w_\ell - v$ such that $w_1 \ldots w_\ell$ are non-terminals with degree two. It is easy to see that

$$R(G, \{s, t\}, d) = r_{s,w_1} \prod_{i=1\ldots\ell-1} r_{w_i, w_{i+1}} r_{w_\ell, v} R(G - \{s, w_1 \ldots w_\ell\}, \{v, t\}, d - \ell - 1).$$

- *Parallel links.* Two edges e, f that link the same pair of nodes, can be replaced by one single edge (linking the same nodes) with reliability $r_e + r_f - r_e r_f$.

- *Edges incident to a terminal.* Whenever all edges that are incident to a certain terminal u have reliability equal to one (i.e. are perfect), the neighbors of u can be collapsed into u. If the new graph is denoted by G', then $R(G, K, d) = R(G', K, d - 1)$.

Procedure FACT$(G = <V, E>, K, d, \vec{r})$

1: reduce G if possible
2: drop all $e \in E : \vec{r}_e = 0$
3: **if** the subgraph induced by $\{e \in E : \vec{r}_e = 1\}$ has K-diameter d or lower **then**
4:     return 1
5: **end if**
6: **if** G has K-diameter higher than d **then**
7:     return 0
8: **end if**
9: choose any $e \in E : \vec{r}_e < 1$
10: $\vec{r'} \leftarrow \vec{r}; \vec{r'}_e = 1$
11: **return** $\vec{r}_e \cdot \texttt{FACT}(G, K, d, \vec{r'}) + (1 - \vec{r}_e) \cdot \texttt{FACT}(G \setminus \{e\}, K, d, \vec{r})$

Figure 5.1: Pseudocode for the `FACT` algorithm in DCR context.

- *Junction points.* In two-terminal s, t context, a junction point is a node u that belongs to all paths between s and t. In [67], it is shown how $R(G, \{s, t\}, d)$ can be computed by solving two subproblems on a 2-partition of G (defined by u). Each subproblem consists of determining the distribution of probability of the d'-DCR for $d' \in 0, \ldots, d$ and then computing the convolution of both distributions obtained.

5.2.2 Factoring-based algorithms in DCR context

Pivotal decomposition techniques can also be applied in DCR context, resulting in the best performing algorithms currently available for the general problem. Next we describe how this family of algorithms proceed and comment on the differences arising when working in DCR context. Similar to the CLR case, an edge e is selected and the configuration space of the graph G is partitioned into two sets, according to whether the edge is in operational or failed state. When the edge is set to fail, which happens with probability $(1 - r_e)$, the algorithm is recursively invoked on the graph $G \setminus \{e\}$, exactly as we did for CLR. But, when the edge is set to operate, which happens with probability r_e, the algorithm is recursively invoked on the same graph, but setting the reliability of e to one, i.e. it is further considered as a "perfect" edge. Figure 5.1 shows the pseudocode of the basic `FACT` algorithm. It receives the graph G, terminals K, diameter threshold d and the vector of edge reliabilities $\vec{r}$. Although taking exponential time, the step where possible reductions are tried, toghether with the early detection of operational or failed partitions of the configuration space (lines 3 to 8), result in significant performance improvements over alternative methods. Again, choosing the pivot edge can significantly affect the depth of the recursion tree [64]. Results for the reliability domination in DCR context can be found in [68].

Let us see an example of how the `FACT` algorithm proceeds. Figure 5.2 shows an instance of the two-terminal DCR problem. Nodes s and t are the terminals and we

consider the network as operational if and only if they are at distance three or lower, i.e. $K = \{s, t\}$ and $d = 3$. Figure 5.3 illustrates the execution flow of the recursion. It results in a tree with 35 nodes, each one corresponding to a new instance of the DCR. Numbers 1 to 35 correspond to the order in which the recursive subinstances are invoked. Terminals are represented by gray nodes and non-terminal as empty ones. Dotted edges represent edges of the original graph that have not yet been used for pivoting. Full edges represent edges that were already considered for pivoting and conditioned as operational in the particular branch. In each step, the edge considered for pivoting is shown as a double line. For example, the first edge used for pivoting is $\{b, c\}$, resulting in the recursive steps 2 (where it is conditioned to be operational) and 21 (where it is conditioned to fail). Instances surrounded by an empty square correspond to "the network operates" recursion leaves; there are eight such instances (4, 7, 9, 14, 17, 23, 26, 32). Instances surrounded by a gray square correspond to "the network fails" recursion leaves; there are 10 such instances (10, 11, 18, 19, 20, 27, 28, 33, 34, 35). The probability that each "network operates" leave is reached is totaled. Each of those probabilities correspond to the product of the reliabilities or unreliabilities of the pivots used, according to the states each pivot was conditioned to have (operational or failed). Two important differences between CLR and DCR contexts arise with regards to the `FACT` algorithm. First, in DCR context there is no node contraction operation; instead, the pivot edge is set to operate, but it must be kept, in order not to lose one unit in the length of the paths it belongs to. Second, there are more chances of reducing the graph and pruning branches than in CLR context. This is due to the K-diameter constraint. For example, in instance 5, it can be seen that node c has no further chance of belonging to an $s - t$ path with fewer than 4 edges. Therefore, it can be removed (we represented this by crossing it out). The same happens in instances 13, 24 and 30. An instance can also be discarded if its K-diameter is above d (considering all edges that have been set to operate and the ones that were not yet used as pivots). These reduction and pruning operations cannot be applied in CLR context. The main idea underlying `FACT` -that of conditioning according to the two possibles states of an edge- can be extended to consider several edges at the same time. For example, in the recursive variance-reduction Monte Carlo method RVR [69], the configuration space is split according to combination of the states of the individual edges of a (conveniently chosen) path. The resulting algorithm for the DCR can be found in [70]. Other ways of conditioning can be devised. The basic idea is that of splitting the problem into several subproblems that are recursively invoked. This splitting must be done according to some set of events that result in a partition i.e. that are at the same time disjoint and complete.

5.2.3   Other procedures

In [40] an algorithm is proposed for exact computation of two-terminal CLR; it proceeds by exploring paths in depth-first-search and generating disjoint "branches" whose probabilities are totaled to compute $R(G, \{s, t\})$. This method (named `AHMAD` after his author) is generalized in [71] to all-terminal CLR with a version that has space complexity (the amount of memory required for execution) linear in the size of the problem.
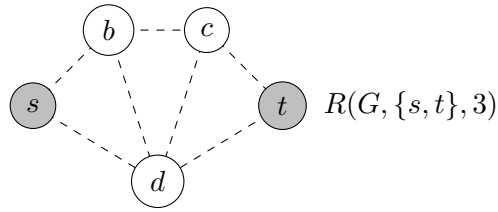
Figure 5.2: An instance of the 3-2-DCR; s and t must be connected by paths no longer than 3 edges.
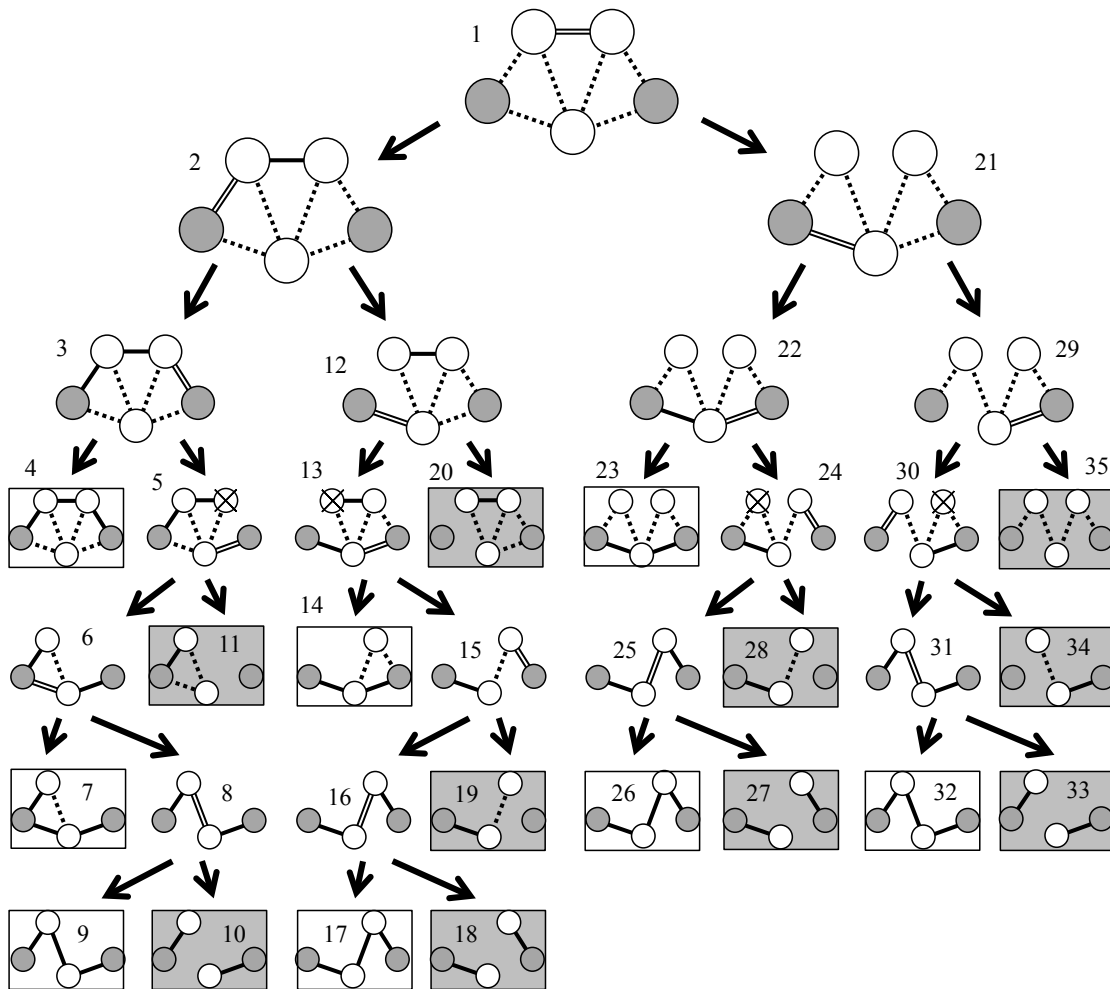
Figure 5.3: An example of how the FACT algorithm proceeds ($d = 3$).

The so-called branches consist of spanning trees of operational edges, accompanied by some failing edges, necessary to ensure disjointness respect to previously found branches. In [72] the algorithm is generalized for the K-terminal CLR (general set $K \in V$); in this case the branches are based on K-spanning Steiner trees. Figure 5.4 illustrates how the algorithm proceeds, on the same network that in Figure 5.3, to compute the CLR. Observe that in this case the CLR coincides with the d-DCR for any $d \geq 4$. The network is explored from source to terminal in the order of the edge labels. Six branches are found. Bold lines represent edges that must operate in the branch, while lines "banned" represent edges that must fail in it. It turns out that

$$
\begin{aligned}
R(G, \{s,t\}) = & r_1 r_2 r_3 + r_1 r_2 (1 - r_3) r_4 r_5 + r_1 (1 - r_2) r_6 r_4 r_3 + \\
& r_1 (1 - r_2)(1 - r_3) r_6 r_5 + (1 - r_1) r_7 r_6 r_2 r_3 + (1 - r_1) r_7 (1 - r_6) r_5
\end{aligned}
$$

where terms correspond to branches and are written in the order they are generated.

Although exponential in time, these methods perform well for networks up to certain sizes [5] and have an interesting feature: they accumulate the probability that the network operates by finding pathsets, starting by more straightforward ones (simpler "branches") and exploring more complex ones towards the end. If the algorithm is run during a certain time t, it will accumulate the probability of several disjoint branches found, thus giving a lower bound for the network reliability; then its corresponding set of pathsets could be used to feed a Monte Carlo algorithm like the ones in Chapter 7. If several depth-first-search explorations are done in parallel, branches with higher disjointness level could be built more quickly; as we explain in Chapter 7 this can result in better performance for the simulations.

These algorithms can be adapted for use in DCR context. For two-terminal d-DCR, the algorithm is basically the same; the only difference is that any branch corresponding to a path from the source node longer than d must be discarded. In Figure 5.5, the same network as that of Figure 5.4 is used but the parameter d is set to three. This results in just four branches; in this case, we have that

$$
R(G, \{s,t\}, 3) = r_1 (1 - r_2) r_6 r_5 + (1 - r_1) r_7 r_4 r_3 + (1 - r_1) r_7 (1 - r_4) r_5.
$$

For all-terminal or K-terminal d-DCR the exploration of the space of pathsets becomes more complex. This is due to the fact that, while in CLR context all minpaths are trees, the same does not hold in DCR context. Further research is necessary to devise smart strategies for exploring the pathset space and build the corresponding branches.
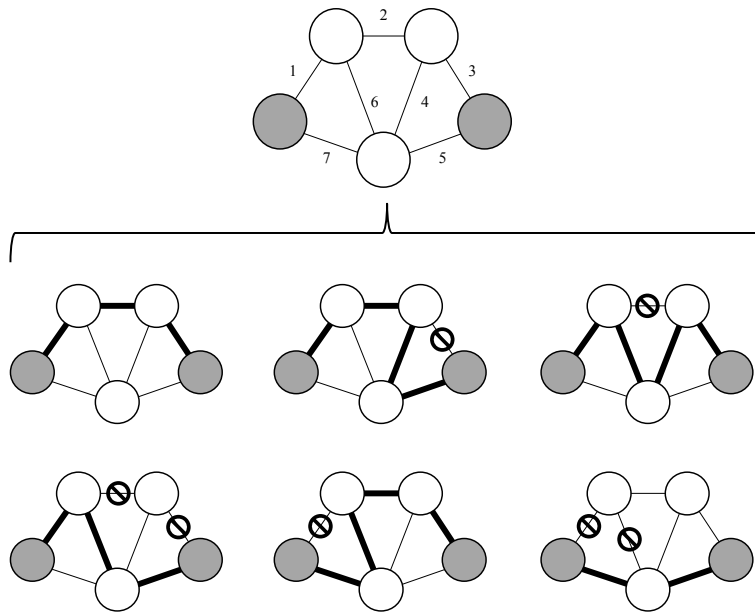
Figure 5.4: An example of how the `AHMAD` algorithm proceeds for CLR ($d \geq 4$).
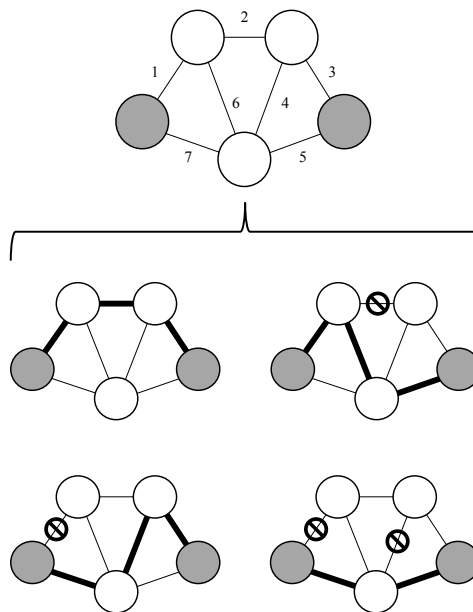
Figure 5.5: An example of how the `AHMAD` algorithm proceeds for DCR ($d = 3$).

Chapter 6

Exact polynomial methods for a family of topologies and asymptotic analysis

In this chapter we illustrate the exact computation of the classical (CLR) and diameter-constrained (DCR) reliabilities in the context of certain particular topologies. We start by a topology that is simple yet not trivial, namely the "ladder" graph. This will allow us to see an example of how the existence of diameter constraints increases the complexity of the problem when compared to the CLR. We begin by developing an algorithm for computing the two-terminal CLR for the ladder with any length ℓ; then we extend it to compute the two-terminal DCR. We generalize these algorithms to a family that we call "multi-ladders". Then we develop a similar discussion for another family that we call "the Spanish fans". We characterize a broader family of topologies for which computing the two-terminal DCR can be done in polynomial time. Finally, in Section 6.4 we introduce the problem of computing the asymptotic value of the DCR, in graphs that grow infinitely following given probabilistic rules, and give some basic results.

6.1   The ladder network

Let us work with a family of networks L_ℓ whose topologies, shown in Fig. 6.1, are defined by a parameter $\ell \in \mathcal{Z} : \ell \geq 1$ (ℓ is the number of "steps" of the ladder).

The ladder has a height of ℓ steps. We set as terminal nodes $K = \{u_0, v_\ell\}$ and so we want to compute the probability that the nodes u_0 and v_ℓ are connected (with and without length constraints). We keep the hypothesis that nodes are perfect and edges fail independently with know probabilities (that can have different values). We denote as $\alpha_1, \ldots, \alpha_\ell$ the reliability of edges $a_1, \ldots, a_\ell$; as $\beta_1, \ldots, \beta_\ell$ the reliability of edges $b_1, \ldots, b_\ell$; and as $\gamma_1, \ldots, \gamma_{\ell+1}$ the reliability of edges $c_1, \ldots, c_{\ell+1}$. Finally, $\bar{\alpha}_i, \bar{\beta}_i, \bar{\gamma}_i$ denote respectively $1 - \alpha_i, 1 - \beta_i, 1 - \gamma_i$ (the edge unreliabilities).
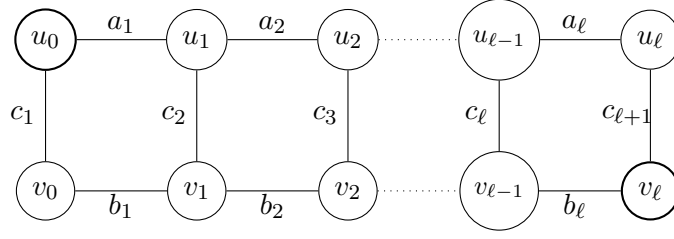
Figure 6.1: The ladder network

6.1.1 Computing the CLR of the ladder

Now we develop an algorithm for computing $R_{u_0,v_\ell}(L_\ell)$. To do so we introduce the following definitions.

- We denote as $u \underset{i}{\rightsquigarrow} v$ the event where there is a path that connects nodes u and v using only edges whose labels have subindices between 1 and i;

- $\mathcal{A}_i = (u_0 \underset{i}{\rightsquigarrow} u_i) \wedge \neg(u_0 \underset{i}{\rightsquigarrow} v_i)$;

- $\mathcal{B}_i = (u_0 \underset{i}{\rightsquigarrow} v_i) \wedge \neg(u_0 \underset{i}{\rightsquigarrow} u_i)$;

- $\mathcal{C}_i = (u_0 \underset{i}{\rightsquigarrow} u_i) \wedge (u_0 \underset{i}{\rightsquigarrow} v_i)$.

For example, the network configuration depicted in Fig. 6.2 belongs to the events $\mathcal{C}_1$, $\mathcal{B}_2$ and $\mathcal{C}_3$ but not to the events $\mathcal{A}_1$, $\mathcal{B}_1$, $\mathcal{A}_2$, $\mathcal{C}_2$, $\mathcal{A}_3$ nor $\mathcal{B}_3$.
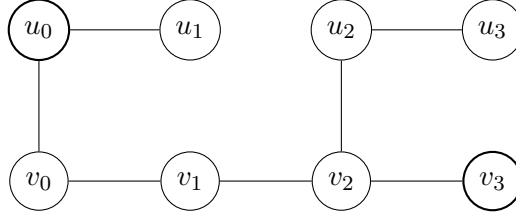
Figure 6.2: Example of a configuration of the ladder L_3

Next, for ease of notation, we denote as x and $\bar{x}$ the events in which an edge labeled x is operating or failing respectively, and as xz the intersection of two such events x and z. Observe that, for $i = 1, \ldots, \ell$, it holds that

$$
\begin{aligned}
\mathcal{A}_i = \mathcal{A}_{i-1} \wedge & \left(a_i\bar{b}_i\bar{c}_i \vee a_ib_i\bar{c}_i \vee a_i\bar{b}_ic_i\right) \vee \\
& \mathcal{B}_{i-1} \wedge a_i\bar{b}_ic_i \vee \\
& \mathcal{C}_{i-1} \wedge \left(a_i\bar{b}_i\bar{c}_i \vee a_i\bar{b}_ic_i\right)
\end{aligned}
\tag{6.1}
$$

$$\begin{aligned}
\mathcal{B}_i = \mathcal{A}_{i-1} \wedge \bar{a}_i b_i c_i \vee \\
\mathcal{B}_{i-1} \wedge (\bar{a}_i b_i \bar{c}_i \vee a_i b_i \bar{c}_i \vee \bar{a}_i b_i c_i) \vee \\
\mathcal{C}_{i-1} \wedge (\bar{a}_i b_i \bar{c}_i \vee \bar{a}_i b_i c_i)
\end{aligned} \tag{6.2}$$

$$\begin{aligned}
\mathcal{C}_i = \mathcal{A}_{i-1} \wedge a_i b_i c_i \vee \\
\mathcal{B}_{i-1} \wedge a_i b_i c_i \vee \\
\mathcal{C}_{i-1} \wedge (a_i b_i \bar{c}_i \vee a_i b_i c_i) .
\end{aligned} \tag{6.3}$$

The unions and intersections of events involving the edges a_i, b_i and c_i were written in canonical form, to make evident each of the eight possible conjoint states yielded by the individual states of the edges, and how they contribute to $\mathcal{A}_i$, $\mathcal{B}_i$ and $\mathcal{C}_i$. Moreover, all unions involve disjoint events and all intersections involve independent events, therefore holding the following identities for the probabilities $\Pr(\mathcal{A}_i)$, $\Pr(\mathcal{B}_i)$ and $\Pr(\mathcal{C}_i)$:

$$\begin{aligned}
\Pr(\mathcal{A}_i) = \Pr(\mathcal{A}_{i-1}) \left(\alpha_i \bar{\beta}_i \bar{\gamma}_i + \alpha_i \beta_i \bar{\gamma}_i + \alpha_i \bar{\beta}_i \gamma_i \right) + \\
\Pr(\mathcal{B}_{i-1}) \alpha_i \bar{\beta}_i \gamma_i + \\
\Pr(\mathcal{C}_{i-1}) \left(\alpha_i \bar{\beta}_i \bar{\gamma}_i + \alpha_i \bar{\beta}_i \gamma_i \right)
\end{aligned} \tag{6.4}$$

$$\begin{aligned}
\Pr(\mathcal{B}_i) = \Pr(\mathcal{A}_{i-1}) \left(\bar{\alpha}_i \beta_i \gamma_i \right) + \\
\Pr(\mathcal{B}_{i-1}) \left(\bar{\alpha}_i \beta_i \bar{\gamma}_i + \alpha_i \beta_i \bar{\gamma}_i + \bar{\alpha}_i \beta_i \gamma_i \right) + \\
\Pr(\mathcal{C}_{i-1}) \left(\bar{\alpha}_i \beta_i \bar{\gamma}_i + \bar{\alpha}_i \beta_i \gamma_i \right)
\end{aligned} \tag{6.5}$$

$$\begin{aligned}
\Pr(\mathcal{C}_i) = \Pr(\mathcal{A}_{i-1}) \left(\alpha_i \beta_i \gamma_i \right) + \\
\Pr(\mathcal{B}_{i-1}) \left(\alpha_i \beta_i \gamma_i \right) + \\
\Pr(\mathcal{C}_{i-1}) \left(\alpha_i \beta_i \bar{\gamma}_i + \alpha_i \beta_i \gamma_i \right) .
\end{aligned} \tag{6.6}$$

Note that computing the probabilities with subindices i can be recursively done invoking events with subindices $i-1$ and edge probabilities with indices i. The base cases are given by $\Pr(\mathcal{A}_0) = 1$, $\Pr(\mathcal{B}_0) = 0$ and $\Pr(\mathcal{C}_0) = 0$. So we can build an algorithm that is linear in ℓ (shown in Fig. 6.3), to compute $R_{u_0,v_\ell}(L_\ell)$. The expression in the `return` statement is necessary to consider the events where u_ℓ is reached thanks to the last "step" $c_{\ell+1}$ of the ladder.

6.1.2 Computing the DCR of the ladder

Now we show how the previous method can be extended from computing the two-terminal d-DCR, denoted as $R_{u_0,v_\ell}(L_\ell, d)$. First, we need to introduce the following Lemma.

Procedure CLR-Ladder$(\ell, \vec{\alpha}, \vec{\beta}, \vec{\gamma})$

1: $a \leftarrow 1; b \leftarrow 0; c \leftarrow 0$
2: **for all** $i \in 1, \ldots, \ell$ **do**
3: $\quad a' \leftarrow a \left(\alpha_i \bar{\beta}_i \bar{\gamma}_i + \alpha_i \beta_i \bar{\gamma}_i + \alpha_i \bar{\beta}_i \gamma_i \right) + b \alpha_i \bar{\beta}_i \gamma_i + c \left(\alpha_i \bar{\beta}_i \bar{\gamma}_i + \alpha_i \bar{\beta}_i \gamma_i \right)$
4: $\quad b' \leftarrow a \left(\bar{\alpha}_i \beta_i \gamma_i \right) + b \left(\bar{\alpha}_i \beta_i \bar{\gamma}_i + \alpha_i \beta_i \bar{\gamma}_i + \bar{\alpha}_i \beta_i \gamma_i \right) + c \left(\bar{\alpha}_i \beta_i \gamma_i + \bar{\alpha}_i \beta_i \gamma_i \right)$
5: $\quad c' \leftarrow a \left(\alpha_i \bar{\beta}_i \bar{\gamma}_i + \alpha_i \beta_i \bar{\gamma}_i + \alpha_i \bar{\beta}_i \gamma_i \right) + b \left(\alpha_i \bar{\beta}_i \gamma_i \right) + c \left(\alpha_i \bar{\beta}_i \bar{\gamma}_i + \alpha_i \bar{\beta}_i \gamma_i \right)$
6: $\quad a \leftarrow a'; b \leftarrow b'; c \leftarrow c'$
7: **end for**
8: **return** $\ b + c + a \gamma_{\ell+1}$

Figure 6.3: Algorithm for computing the s, t-CLR of a ladder graph

Lemma 6.1. *Suppose that $u_0 \underset{i}{\leadsto} u_i$ and $u_0 \underset{i}{\leadsto} v_i$. The shortest paths that connect u_0 to u_i and u_0 to v_i $(i = 1, \ldots, \ell)$, using only edges whose labels have subindices up to i, have lengths that differ exactly in one.*

Proof. Let p_1 and p_2 be examples of the shortest paths that connect u_0 to u_i and v_i respectively (using only edges with subindices up to i). Both will have the general structure of Fig. 6.4 where, walking from u_0 to u_i and v_i, they meet a certain number of times, before they reach u_i and v_i. Let w be the last common node. The part p'_1 of p_1 going from u_0 to w has the same length that the part p'_2 of p_2 going from u_0 to w; otherwise, one of p_1 or p_2 could be made shorter by replacing p'_1 by p'_2 or vice versa. The parts of p_1 and p_2 going from w to u_i and v_i must be as in Fig. 6.4, where the two possible cases $w = u_j$ and $w = v_j$ with $j \in \{0, i\}$ are shown. Regardless what the case is, clearly the difference in length is one; otherwise w would not be the last common node. $\square$
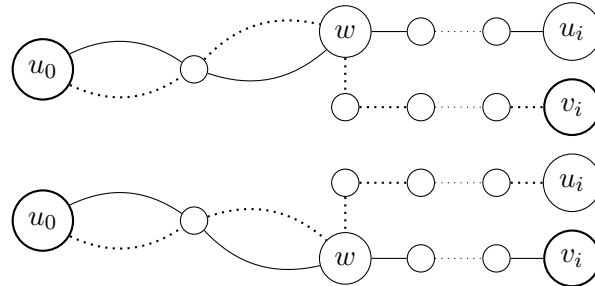
Figure 6.4: Lemma 6.1

In the case of the DCR, for each $i = 1, \ldots, \ell$, we have to deal with more events than in the case of the CLR ($\mathcal{A}_i$, $\mathcal{B}_i$ and $\mathcal{C}_i$). Now the events get defined by the pairs of distances from u_0 to u_i and v_i. Let us denote by $\mathcal{E}_i^{m,n}$ the event where $dist(u_0, u_i) = m$ and $dist(u_0, v_i) = n$ when only considering operational edges with subindices up to i. In light of Lemma 6.1 and the specific topology of the ladder, no event other than $\mathcal{E}_i^{\infty,\infty}$,

$\mathcal{E}_i^{m,\infty}$, $\mathcal{E}_i^{m+1,m}$, $\mathcal{E}_i^{m,m+1}$, $\mathcal{E}_i^{\infty,m}$ can be feasible, being $i = 0, \ldots, \ell$ and $m = i, \ldots, 2i + 1$. As we did for the CLR, we can express the events for i using only events with subindices $i - 1$:

$$\mathcal{E}_i^{m,\infty} = \mathcal{E}_{i-1}^{m-1,\infty} a_i \overline{b_i c_i} \vee \mathcal{E}_{i-1}^{m-1,m} a_i \bar{b}_i \vee \mathcal{E}_{i-1}^{\infty,m-2} a_i \bar{b}_i c_i \vee \mathcal{E}_{i-1}^{m-1,m-2} a_i \bar{b}_i \qquad (6.7)$$

$$\mathcal{E}_i^{m+1,m} = \mathcal{E}_{i-1}^{\infty,m-1} a_i b_i c_i \vee \mathcal{E}_{i-1}^{m,m-1} a_i b_i \qquad (6.8)$$

$$\mathcal{E}_i^{m,m+1} = \mathcal{E}_{i-1}^{m-1,\infty} a_i b_i c_i \vee \mathcal{E}_{i-1}^{m-1,m} a_i b_i \qquad (6.9)$$

$$\mathcal{E}_i^{\infty,m} = \mathcal{E}_{i-1}^{\infty,m-1} b_i \overline{a_i c_i} \vee \mathcal{E}_{i-1}^{m,m-1} \bar{a}_i b_i \vee \mathcal{E}_{i-1}^{m-2,\infty} \bar{a}_i b_i c_i \vee \mathcal{E}_{i-1}^{m-2,m-1} \bar{a}_i b_i. \qquad (6.10)$$

Similar considerations to those for the CLR about disjointness of unions and independency of intersections apply to the events involved in these equations. The only base case ($i = 0$) with non-zero probability is $\Pr(\mathcal{E}_0^{0,\infty}) = 1$. Now we can build an algorithm (shown in Fig. 6.5) to compute $R_{u_0,v_\ell}(L_\ell, d)$. It starts by considering the trivial cases where the threshold distance d between u_0 and v_ℓ is too low to be reachable (the lowest possible one is $\ell + 1$). Similarly, if d is above $2\ell - 1$, then the operating configurations are exactly the same as for the CLR, since given the topology of the network, no configuration exists where the distance between u_0 and v_ℓ is above $2\ell - 1$. Next, the algorithm defines a square array, initialized with zeros, large enough to store the probability of all the events $\mathcal{E}$. For ease of notation we assume that any reference involving out-of-range subindices returns zero. The constant t is set to $2\ell + 3$ to represent the array index for ∞. The probability of the base case $\mathcal{E}_0^{0,\infty}$ is set to one. Then the algorithm proceeds by sequentially processing the edges, one "ladder step" at a time. For each step, only those values of m that can accumulate probability are considered; they range from i to $2i$. Once the array is computed, the algorithm builds a vector $\vec{p}$ that stores the probability that the distance between u_0 and v_ℓ is an integer m, for every index m ranging from $\ell + 1$ to $2\ell + 1$. Care is taken to consider the impact of the edge $c_{\ell+1}$, similarly as we did for the CLR-Ladder algorithm. Finally, the algorithm returns the cumulative probability that the distance is any between $\ell + 1$ and the parameter d.

The execution time is dominated by the nested iteration, and thus the algorithm has a complexity in time that is

$$O(\sum_{i=1}^{\ell} \sum_{m=i}^{2i}) = O(\sum_{i=1}^{\ell} \ell(i+1)) = O((\ell+1)\ell/2 + \ell) = O(\ell^2)$$

that is, quadratic with respect to ℓ. We implemented the algorithm of Figure 6.5 in C++ and tested on an Intel Core2 Duo T5450 CPU machine with 2 GB RAM. Table 6.1 shows the values of R_{u_0,v_ℓ} obtained for ℓ between 1 and 20, the parameter d between 2 and 41 (which are the minimal and maximal finite distances for $\ell = 1$ and $\ell = 20$ respectively), and all edge reliabilities equal to 0.3. Table 6.2 shows the values of R_{u_0,v_ℓ} obtained for

Procedure DCR-Ladder$(\ell, d, \vec{\alpha}, \vec{\beta}, \vec{\gamma})$

1: **if** $d \leq \ell$ **then**
2:     **return** 0
3: **else if** $d \geq 2\ell - 1$ **then**
4:     **return** CLR-Ladder$(\ell, \vec{\alpha}, \vec{\beta}, \vec{\gamma})$
5: **end if**
6: $t \leftarrow 2\ell + 3$
7: $e(\cdot, \cdot) \leftarrow$ array $(t + 1, t + 1)$ initialized with 0's
8: $e(0, t) \leftarrow 1$
9: $\vec{x_1}, \vec{x_2}, \vec{x_3}, \vec{x_4} \leftarrow$ vectors with subindices $1, \ldots, 2\ell + 1$
10: **for all** $i \in 1, \ldots, \ell$ **do**
11:     **for all** $m \in i, \ldots, 2i + 1$ **do**
12:         $x_1(m) \leftarrow e(m-1,t)\alpha_i\overline{\beta_i\gamma_i} + e(m-1,m)\alpha_i\bar{\beta}_i + e(t,m-2)\alpha_i\bar{\beta}_i\gamma_i + e(m-1,m-2)\alpha_i\bar{\beta}_i$
13:         $x_2(m) \leftarrow e(t,m-1)\beta_i\overline{\alpha_i\gamma_i} + e(m,m-1)\bar{\alpha}_i\beta_i + e(m-2,t)\bar{\alpha}_i\beta_i\gamma_i + e(m-2,m-1)\bar{\alpha}_i\beta_i$
14:         $x_3(m) \leftarrow e(t,m-1)\alpha_i\beta_i\gamma_i + e(m,m-1)\alpha_i\beta_i$
15:         $x_4(m) \leftarrow e(m-1,t)\alpha_i\beta_i\gamma_i + e(m-1,m)\alpha_i\beta_i$
16:     **end for**
17:     **for all** $m \in i, \ldots, 2i$ **do**
18:         $e(m,t) \leftarrow x_1(m)$
19:         $e(t,m) \leftarrow x_2(m)$
20:         $e(m+1,m) \leftarrow x_3(m)$
21:         $e(m,m+1) \leftarrow x_4(m)$
22:     **end for**
23:     $e(i-1,t) \leftarrow 0; e(t,i-1) \leftarrow 0; e(i,i-1) \leftarrow 0; e(i-1,i) \leftarrow 0$
24: **end for**
25: $\vec{p} \leftarrow$ vector with subindices $\ell + 1, \ldots, 2\ell + 1$
26: **for all** $m \in \ell + 1, \ldots, 2\ell + 1$ **do**
27:     $p(m) \leftarrow \left(\sum_{j=m-1,m+1,t} e(j,m)\right) + e(t,m)$
28:     $p(m) \leftarrow p(m) + (e(m-1,m-2) + e(m-1,t))\gamma_{\ell+1}$
29: **end for**
30: **return** $\sum_{m=\ell+1,\ldots,d} p(m)$

Figure 6.5: Algorithm for computing the s, t-DCR of a ladder graph

ℓ between 1 and 40, d between 2 and 81 (again, covering all possible finite distances between u_0 and v_ℓ), this time with edge reliabilities equal to 0.9. Figure 6.7 shows (left) the elapsed time for computing L_1 to L_{150} and (right) the square root of these times, making clear their quadratic evolution. Finally, four tests were run for the network L_{20}, and their results are shown in Figure 6.8. The four charts show the probabilities that the random variable "distance between u_0 and v_{20}" has the values that range from 21 to 41. (These are the possible finite values it can have for L_{20}). The difference between 1 and the cumulative probability of each chart is the probability that both nodes are disconnected. This kind of probability distribution functions are used in [67] to compute the two-terminal DCR of networks with junction points by means of the convolution operator.

6.2 The Spanish fan network

Let us see another example based on a topology that we call the "Spanish fan". This family of networks F_ℓ is shown is Fig. 6.6, parameterized by $\ell \in \mathbb{Z} : \ell > 1$. It can be seen as a ladder in which one of the sides was collapsed into a single node (v). Again, we set as terminal two nodes $K = \{u_0, u_\ell\}$ and we want to compute the probability that the nodes u_0 and u_ℓ are connected by a path with length up to a certain integer d.
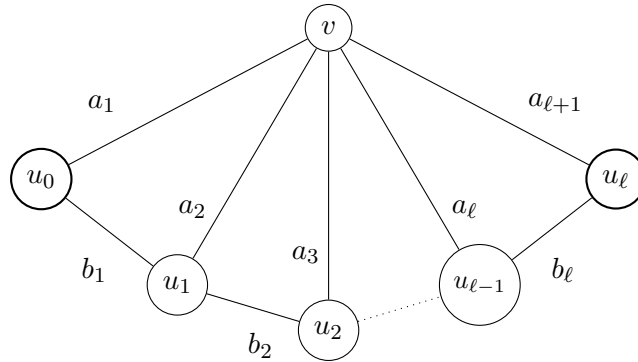
Figure 6.6: The Spanish fan network

It is easy to see that, in any configuration, when only considering edges with subindices up to i, the following statements hold true:

- the distance between u_0 and u_i can be any of the range $\min(2, i), \ldots, i + 1$ or infinite;

- the distance between u_0 and v can be any of the range $1, \ldots, \ell + 1$ or infinite.

Now, analogously to Section 6.1.1, we can define $\mathcal{E}_i^{m,n}$ as the event where $dist(u_0, v) = m$ and $dist(u_0, u_i) = n$ when only considering operational edges with subindices up to i. For a given value of $i \in 1, \ldots, \ell$, the only elementary events that can occur with probabilities other than zero, and at the end contribute to $R_{u_0, u_\ell}(F_\ell, d)$, are:

- $\mathcal{E}_i^{\infty,i}$

- $\mathcal{E}_i^{m,\infty} : m \in 1,\ldots,i$

- $\mathcal{E}_i^{m,n} : n \in 1,\ldots,i \wedge m \in 1,\ldots,n$

- $\mathcal{E}_i^{m,i+1} : m \in 1,\ldots,i$

So, as i grows, the number of such feasible elementary events has order $O(i^2)$. It is interesting to note that, although being a simpler topology, these events outnumber those of the ladder network; since by virtue of Lemma 6.1, the number of events of the ladder grows with order $O(i)$. Again, we can express the events for a given value of i just using events defined for $i-1$, as follows:

$$\mathcal{E}_i^{\infty,i} = \mathcal{E}_{i-1}^{\infty,i-1}\bar{a}_i b_i$$

$$\mathcal{E}_i^{m,\infty} = \left(\cup_{j=m}^{\infty}\mathcal{E}_{i-1}^{m,j}\bar{b}_i\right) \vee \mathcal{E}_{i-1}^{m,\infty}\bar{a}_i b_i \vee \mathcal{E}_{i-1}^{\infty,m-1}a_i\bar{b}_i \quad (m \in 1,\ldots,i)$$

$$\mathcal{E}_i^{m,n} = \mathcal{E}_{i-1}^{m,n-1}b_i \vee \mathcal{E}_{i-1}^{\infty,n-1}(m=n)a_i b_i \quad (n \in 1,\ldots,i; m \in 1,\ldots,n)$$

$$\mathcal{E}_i^{m,i+1} = \mathcal{E}_{i-1}^{m,i}b_i \quad (m \in 1,\ldots,i)$$

Again, the probabilities with subindices i can be computed just recurring to events that just involve the subindex $i-1$. This time the base case is given by $\mathcal{E}_0^{\infty,0} = 1$. Similarly as we did for the ladder, we can build an algorithm that computes the probability of the $O(i^2)$ relevant events with subindices i, using the correspondent probabilities for $i-1$. This results in a dynamic programming algorithm with a number of operations that is $O(i^3)$, therefore with time complexity $O(i^3)$; once again, polynomial in i.

6.3 A general family of topologies with polynomial complexity

The examples we have introduced in Sections 6.1 and 6.2 exemplify the use of dynamic programming techniques for computing the DCR of particular topologies. Next we introduce a more general family of topologies that we call $\mathcal{G}_h$. Although being an arbitrary family of topologies, it serves for illustrating how the existence of a certain structure in the network can lead to efficient (in the sense of polynomial) procedures for computing the DCR; as well as how the level of structural decomposition allows to obtain lower degrees in the polynomial. We show that the two-terminal DCR can be computed for any member of $\mathcal{G}_h$ in a time that is polynomial in the number of nodes. We give an algorithm that behaves according to this property. Besides illustration purposes, our main interest in this algorithm lies on having a tool to create non-trivial graphs for which the two-terminal DCR can be exactly computed; this is interesting, for example, for testing bounds and/or approximation methods.

	1	2	3	5	8	11	15	20
2	1.72E-01							
3	1.72E-01	7.56E-02						
4	1.72E-01	7.56E-02	2.99E-02					
5	1.72E-01	7.68E-02	2.99E-02					
6	1.72E-01	7.68E-02	3.14E-02	3.98E-03				
7	1.72E-01	7.68E-02	3.14E-02	3.98E-03				
8	1.72E-01	7.68E-02	3.14E-02	4.74E-03				
9	1.72E-01	7.68E-02	3.14E-02	4.74E-03	1.60E-04			
10	1.72E-01	7.68E-02	3.14E-02	4.75E-03	1.60E-04			
11	1.72E-01	7.68E-02	3.14E-02	4.75E-03	2.51E-04			
12	1.72E-01	7.68E-02	3.14E-02	4.75E-03	2.51E-04	5.73E-06		
13	1.72E-01	7.68E-02	3.14E-02	4.75E-03	2.58E-04	5.73E-06		
14	1.72E-01	7.68E-02	3.14E-02	4.75E-03	2.58E-04	1.24E-05		
15	1.72E-01	7.68E-02	3.14E-02	4.75E-03	2.58E-04	1.24E-05		
16	1.72E-01	7.68E-02	3.14E-02	4.75E-03	2.58E-04	1.37E-05	6.16E-08	
17	1.72E-01	7.68E-02	3.14E-02	4.75E-03	2.58E-04	1.37E-05	6.16E-08	
18	1.72E-01	7.68E-02	3.14E-02	4.75E-03	2.58E-04	1.37E-05	2.02E-07	
19	1.72E-01	7.68E-02	3.14E-02	4.75E-03	2.58E-04	1.37E-05	2.02E-07	
20	1.72E-01	7.68E-02	3.14E-02	4.75E-03	2.58E-04	1.37E-05	2.65E-07	
25	1.72E-01	7.68E-02	3.14E-02	4.75E-03	2.58E-04	1.37E-05	2.74E-07	1.78E-09
30	1.72E-01	7.68E-02	3.14E-02	4.75E-03	2.58E-04	1.37E-05	2.74E-07	2.05E-09
41	1.72E-01	7.68E-02	3.14E-02	4.75E-03	2.58E-04	1.37E-05	2.74E-07	2.05E-09

Table 6.1: $R_{u_0,v_\ell}(L_\ell, d)$ for ℓ between 1 and 20, d between 2 and 41, with all edge reliabilities equal to 0.3

	1	3	6	10	15	20	30	40
2	9.64E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
3	9.64E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
4	9.64E-01	9.30E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
5	9.64E-01	9.30E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
6	9.64E-01	9.48E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
7	9.64E-01	9.48E-01	8.36E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
8	9.64E-01	9.48E-01	8.36E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
9	9.64E-01	9.48E-01	9.14E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
10	9.64E-01	9.48E-01	9.14E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
11	9.64E-01	9.48E-01	9.15E-01	6.86E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
12	9.64E-01	9.48E-01	9.15E-01	6.86E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
13	9.64E-01	9.48E-01	9.15E-01	8.66E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
14	9.64E-01	9.48E-01	9.15E-01	8.66E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
15	9.64E-01	9.48E-01	9.15E-01	8.73E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
16	9.64E-01	9.48E-01	9.15E-01	8.73E-01	5.07E-01	0.00E+00	0.00E+00	0.00E+00
17	9.64E-01	9.48E-01	9.15E-01	8.73E-01	5.07E-01	0.00E+00	0.00E+00	0.00E+00
18	9.64E-01	9.48E-01	9.15E-01	8.73E-01	7.93E-01	0.00E+00	0.00E+00	0.00E+00
19	9.64E-01	9.48E-01	9.15E-01	8.73E-01	7.93E-01	0.00E+00	0.00E+00	0.00E+00
20	9.64E-01	9.48E-01	9.15E-01	8.73E-01	8.22E-01	0.00E+00	0.00E+00	0.00E+00
22	9.64E-01	9.48E-01	9.15E-01	8.73E-01	8.23E-01	3.60E-01	0.00E+00	0.00E+00
24	9.64E-01	9.48E-01	9.15E-01	8.73E-01	8.23E-01	7.05E-01	0.00E+00	0.00E+00
26	9.64E-01	9.48E-01	9.15E-01	8.73E-01	8.23E-01	7.71E-01	0.00E+00	0.00E+00
28	9.64E-01	9.48E-01	9.15E-01	8.73E-01	8.23E-01	7.75E-01	0.00E+00	0.00E+00
30	9.64E-01	9.48E-01	9.15E-01	8.73E-01	8.23E-01	7.75E-01	0.00E+00	0.00E+00
32	9.64E-01	9.48E-01	9.15E-01	8.73E-01	8.23E-01	7.75E-01	1.67E-01	0.00E+00
34	9.64E-01	9.48E-01	9.15E-01	8.73E-01	8.23E-01	7.75E-01	5.04E-01	0.00E+00
36	9.64E-01	9.48E-01	9.15E-01	8.73E-01	8.23E-01	7.75E-01	6.59E-01	0.00E+00
38	9.64E-01	9.48E-01	9.15E-01	8.73E-01	8.23E-01	7.75E-01	6.86E-01	0.00E+00
40	9.64E-01	9.48E-01	9.15E-01	8.73E-01	8.23E-01	7.75E-01	6.89E-01	0.00E+00
50	9.64E-01	9.48E-01	9.15E-01	8.73E-01	8.23E-01	7.75E-01	6.89E-01	6.11E-01
81	9.64E-01	9.48E-01	9.15E-01	8.73E-01	8.23E-01	7.75E-01	6.89E-01	6.12E-01

Table 6.2: $R_{u_0,v_\ell}(L_\ell, d)$ for ℓ between 1 and 40, d between 2 and 81, with all edge reliabilities equal to 0.9
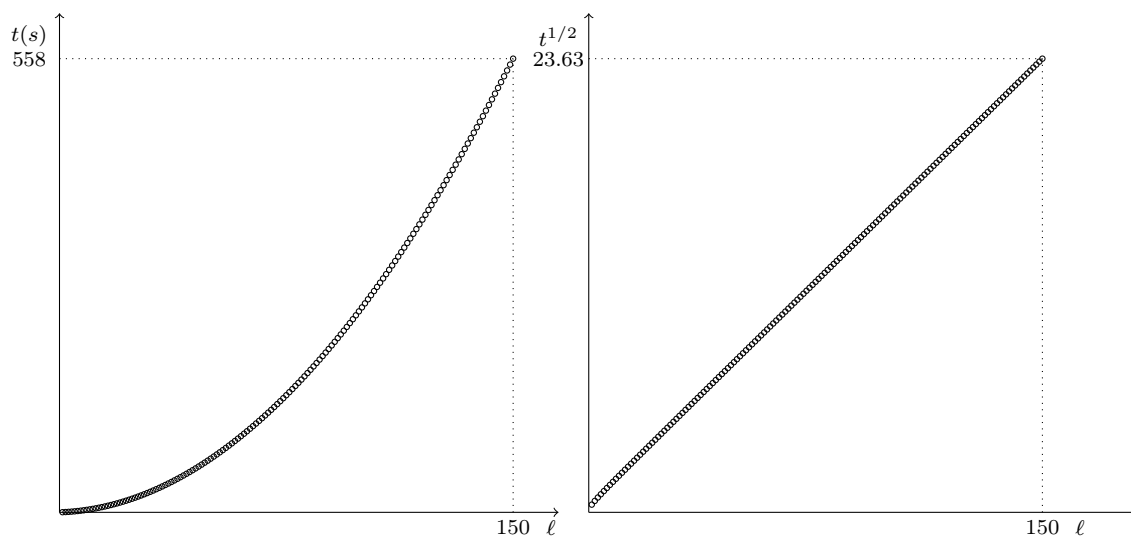
Figure 6.7: Elapsed time, and its square root, for computing L_1 to L_{150} using `DCR-Ladder`
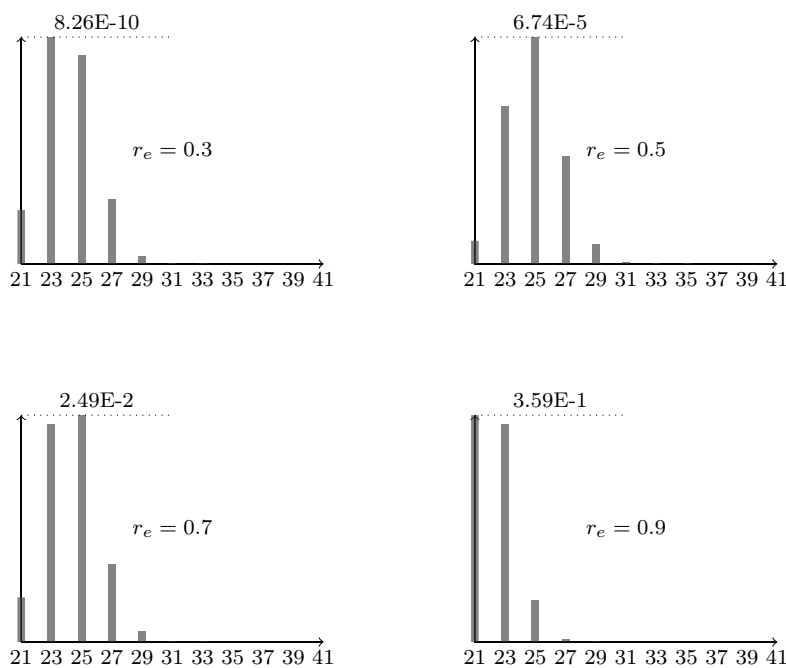
Figure 6.8: Distributions of probability for the distance between u_0 and v_{20} in L_{20}

Definition 21. *Given a positive integer h, we call $\mathcal{G}_h$ the set of all undirected graphs $G = (V, E)$ such that:*

- *ℓ is any positive integer or zero;*

73

- V has a partition $\{s\}, V_0, V_1, \ldots, V_\ell, \{t\}$ with $|V_i| \leq h$ for $i = 0, \ldots, \ell$;

- E has a partition $E_0, \ldots, E_{\ell-1}, S_0, \ldots, S_\ell, W_0, \ldots, W_\ell, T$;

- $E_i \subseteq \{\{x, y\} : x \in V_i \wedge y \in V_{i+1}\}$ for $i = 0, \ldots, \ell - 1$;

- $S_i \subseteq \{\{s, x\} : x \in V_i\}$ for $i = 0, \ldots, \ell$;

- $W_i \subseteq \{\{x, y\} : x, y \in V_i\}$ for $i = 0, \ldots, \ell$;

- $T \subseteq \{\{t, x\} : x \in V_\ell\}$.

Figure 6.9 depicts the structure of a generic element of $\mathcal{G}_h$. Gray bars represent sets of edges that link pairs of nodes belonging to different components of V. The minimum possible distance between s and t is 2, thanks to any path $s - x - t$ where $x \in V_\ell$. But, in contrast to the examples of the ladder and the Spanish fan networks, there may be $s \rightsquigarrow t$ paths whose sequence of visited subsets V_i goes back and forward several times. For instance, Figure 6.10 shows one path whose sequence of V subindices goes downwards twice. This prevents us from the possibility of just keeping track of the distances from s to each node in V_i considering only edges with subindices up to i, as we did in the previous cases. More information has to be tracked as the algorithm proceeds by moving forward in i, namely the distance between each pair of nodes in V_i, considering only edges with subindices up to i. Next we describe an algorithm for computing the DCR of any network in $\mathcal{G}_h$ and prove that it takes a time that is polynomial in the number of nodes. To do so, we first need to prove the following lemmas.
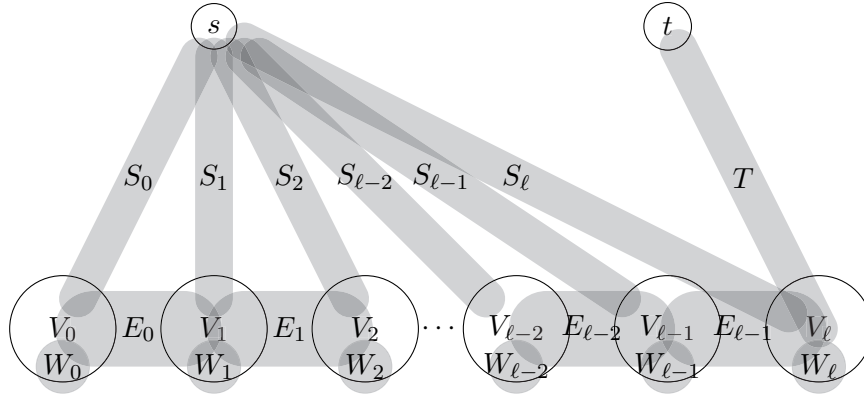
Figure 6.9: The family of graphs $\mathcal{G}_h$

Lemma 6.2. *Let $G = (V, E)$ be an undirected graph; u, v, t, w any nodes of V with $u \neq v$ and $t \neq w$; $d_{u,v}$ the distance between u and v in G. Then, the distance between t and w in the graph $G' = (V, E \cup \{u, v\})$ (denoted $d'_{t,w}$) is*

$$d'_{t,w} = \min\{d_{t,w}, 1 + d_{t,u} + d_{v,w}, 1 + d_{t,v} + d_{u,w}\}$$
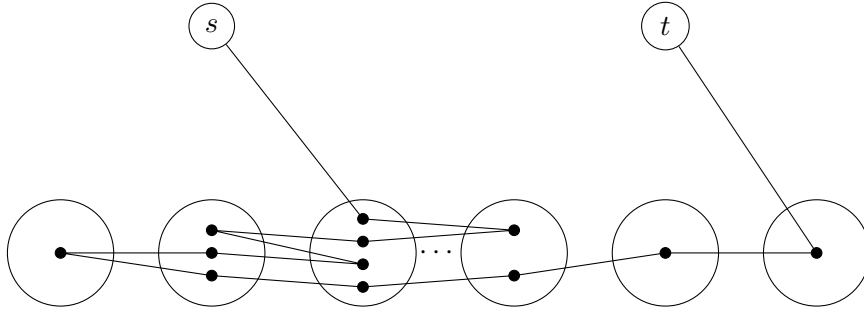
Figure 6.10: Example of an s, t-path moving back and forward in the sequence of the V_i

Proof. For each path that connects t and w in G', exactly one of the three possible statements hold true for the corresponding trail from t to w: (i) it does not uses the edge $\{u, v\}$; (ii) it uses the edge $\{u, v\}$ and visits u before v; (iii) it uses the edge $\{u, v\}$ and visits v before u. Let d_0, d_1 and d_2 be the length of the shortest paths $t \rightsquigarrow w$ in G' for which (i), (ii) and (iii) respectively hold (∞ if there is no such path). We have that

- $d_0 = d_{t,w}$;

- $d_1 = d'_{t,u} + 1 + d'_{v,w} = d_{t,u} + 1 + d_{v,w}$;

- $d_2 = d'_{t,v} + 1 + d'_{u,w} = d_{t,v} + 1 + d_{u,w}$.

Then the shortest path between t and w in G' will have a distance equal to $\min\{d_0, d_1, d_2\}$ wich completes the proof. $\square$

Lemma 6.3. *Let $G = (V, E)$ be an undirected graph; $H \subseteq V$; $d_{u,v}$ is the distance between u and v in G for every $u, v \in H$; and $E' \subseteq \{\{u, v\} : u, v \in H\}$ is a set of edges whose nodes belong to H (but not necessarily to E). Then, the set of all distances $d'_{u,v} : u, v \in H$ between nodes of H in $G' = (V, E \cup E')$ can be computed knowing only $d_{u,v} : u, v \in H$, with an amount of elementary operations that is $O(|H|^4)$.*

Proof. We prove it by giving an algorithm that returns the desired distances with $O(|H|^4)$ elementary operations. Consider the following algorithm.

Procedure New-Distances(H, d, E')
1: **for all** $\{u, v\} \in E'$ **do**
2:    **for all** $\{x, y\} \in \{\{t, w\} : t, w \in H\}$ **do**
3:       $d'_{x,y} \leftarrow \min\{d_{x,y}, 1 + d_{x,u} + d_{v,y}, 1 + d_{x,v} + d_{u,y}\}$
4:    **end for**
5:    $d \leftarrow d'$
6: **end for**
7: **return** d

The interior loop has at most $|H|(|H| - 1)/2$ iterations. Each one performs four additions, a minimum-of-three comparison and an assignment, i.e. a constant number

of elementary operations. By virtue of Lemma 6.2, after each interior iteration, we have the set of distances d for the new graph obtained after adding one edge of E'. After repeating this (external loop) for each one of the (at most) $|H|(|H|-1)/2$ edges of E', the returned d contains the desired distances, and the number of elementary operations is at most $|H|^2(|H|-1)^2/4$ times a constant number, i.e. it is $O(|H|^4)$. $\qquad\square$

Now we are able to prove the main theorem of this section.

Theorem 6.1. *When restricted to a family of graphs $\mathcal{G}_h$, the DCR can be computed in a time that is a polynomial in the number of nodes. The order degree depends only on h.*

Proof. We start the proof by presenting an algorithm that satisfies the thesis in Fig. 6.11. The algorithm receives the parameters ℓ, the edge reliabilities $\vec{r}$, the maximum allowed K-diameter $d_{\max}$, and the partitions that build the sets of nodes and edges of the graph. We employ "triads" (x,y,z) that mean "the distance between nodes x and y is z". Given a certain i, a set of triads can be seen as the event where the distances (avoiding edges that belong to components labeled above i) are those specified by the triads. Each set $\mathcal{D}_i$ will record sets of triads (x,y,z), i.e. $\mathcal{D}_i$ is a set of sets of triads; we use $\mathcal{D}_i$ to store the mentioned events. We maintain also maps μ_i, indexed by the elements of $\mathcal{D}_i$, that store the probability of each event given by a set of triads. The base case is given by the triad $(s,s,0)$; the probability p_0 that the single-element set of triads $\{(s,s,0)\}$ holds true is 1. The algorithm then performs an outer loop (lines 8 to 30) to sequentially consider i from 0 to ℓ. In each of its iterations, a new set F of edges belonging to components labeled with subindex i is considered. For each i, an intermediate loop (lines 14 to 29) iterates through all the feasible sets of triads that were previously recorded in $\mathcal{D}_i$. The inner loop (lines 15 to 28) matches every such set of triads with every subset of edges of F (to consider every possible combination of operational and failing edges in F). The probability of this combination of operational and failing edges is computed and assigned to ρ. The procedure `New-Distances` of Lemma 6.3 is invoked to compute the distances between nodes in $V_i \cup V_{i+1} \cup \{s\}$ considering edges that belong to components whose labels have subindices up to i. It returns a new set of triads d', whose probability is accumulated in the map μ_{i+1} for use in the next outer iteration.

After leaving the nested iterations, $\mu_{\ell+1}$ records the probability of a number of sets of triads, including all the possible scenarios for distances between s and t. The probability for those that respect the diameter constraint parameter $d_{\max}$ are totaled and returned at the end as $R_{s,t}(G, d_{\max})$.

Let us analyze the computational complexity of this algorithm. Loop (8-30) performs $\ell + 1 = O(\ell)$ iterations. Each iteration of loop (8-30) is dominated by loop (14-29), that performs $|\mathcal{D}_i|$ iterations. At its time, each iteration of loop (14-29) performs $2^{|F'|} \le 2^{h+h^2+h(h-1)/2} \le 4^{h^2}$ iterations. The latter iterations are dominated by procedure `New-Distances` that executes in time $O((2h+1)^4) = O(h^4)$. So, the total time has order

$$\text{order} = \sum_{i=0}^{\ell} \left(|\mathcal{D}_i| 4^{h^2} h^4 \right) = f(h) \sum_{i=0}^{\ell} |\mathcal{D}_i| \tag{6.11}$$

where $f(h)$ is a constant that depends on the parameter h (that grows extremely fast with h). Regarding $|\mathcal{D}_i|$, note that each element of $\mathcal{D}_i$ specifies distances for pairs of at most $h + h + 1$ nodes, that is, $(2h + 1)(2h + 1 - 1)/2 = (2h + 1)h$ pairs. Each pair can have a distance between 2 and $h + h\ell = h(\ell + 1)$ (given by a path whose trail goes back and forward h times from V_0 to V_i). So, $|\mathcal{D}_i|$ is bounded by $(h(i + 1))^{(2h+1)h}$. Replacing in Eq. (6.11) we have that the order of execution time is

$$\text{order} = O\left(f(h)\sum_{i=0}^{\ell}(h(i + 1))^{(2h+1)h}\right) = O\left(g(h)\ell^{(2h+1)h}\right) \tag{6.12}$$

where, again, $g(h)$ is a constant that depends on the parameter h (growing even faster than f with h). This completes the proof.

$\square$

Observe that ladder network belongs to the family $\mathcal{G}_3$, as illustrated in Figure 6.12 with $\ell - 1$ components for a ladder with ℓ steps. By means of a simple technique that we illustrate below for a generalization of the ladders (in Figure 6.13), it can be reduced to $\mathcal{G}_2$. Then the execution time of Eq. (6.12) is $O(\ell^5)$, while the one of the algorithm suggested in Section 6.1 was ℓ^2. This happens because in the algorithm `DCR-Ladder` we took advantage of additional knowledge about the structure of the network, namely that there cannot be "back and forward" paths in a ladder. In other words, the general algorithm considers eventually that there can be edges linking any pair of nodes in each V_i or between V_i and V_{i+1}, but the possibilities allowed by the ladder network are much more restrictive than that. An intermediate case would be that of a family that we could name "multi-ladders", shown in Figure 6.13. There, back-and-forward paths are feasible, but the number of feasible edges in each V_i or between V_i and V_{i+1} is bounded by h instead of h^2. In this figure we illustrate a reduction technique that allows to set h as the number of nodes existing in each step of the multi-ladder (the "step size"). A multi-ladder G with ℓ steps and step size h, is converted into a member G' of $\mathcal{G}_h$. This is done by adding a new node t', linked exclusively to the destination node t of the multi-ladder, by an edge with reliability equal to one (a perfect link). Observe that the random variable "distance s-t'" is equal to the random variable "distance s-t" plus one. Hence, we have that $R(G, \{s, t\}, d) = R(G', \{s, t'\}, d+1)$, so we can invoke the algorithm `DCR-Polynomial` and compute $R(G, \{s, t\}, d)$ with a complexity determined by the size of each step h.

6.4 On the asymptotic d-K-DCR of random graphs

We now introduce the problem of computing the asymptotic value of the DCR, in graphs that grow infinitely, following given probabilistic rules. The basic question is that of determining if the DCR converges (and to what value) when a given network grows by adding nodes and edges according to precise probabilistic rules. In this section we derive some direct results from Random Graph Theory, for Gilbert's model of random graphs.

Procedure DCR-Polynomial $(\ell, \vec{r}, d_{max}, V_0, \ldots, V_\ell, S_0, \ldots, S_\ell, W_0, \ldots, W_\ell, E_0, \ldots, E_\ell, T)$

1: **for all** $i \in 0, \ldots, \ell + 1$ **do**
2:     initialize $\mathcal{D}_i \leftarrow \emptyset$
3:     initialize $\mu_i(\cdot)$ as an empty map of reals
4: **end for**
5: $d \leftarrow \{(s, s, 0)\}$
6: $\mathcal{D}_0 \leftarrow \{d\}$
7: $\mu_0(d) \leftarrow 1$
8: **for all** $i \in 0, \ldots, \ell$ **do**
9:     **if** $i < \ell$ **then**
10:         $F = S_i \cup W_i \cup E_i$
11:     **else**
12:         $F = S_i \cup W_i \cup T$
13:     **end if**
14:     **for all** $d \in \mathcal{D}_i$ **do**
15:         **for all** $F' \subseteq F$ **do**
16:             **if** $i < \ell$ **then**
17:                 $d' \leftarrow$ New-Distances$(V_i \cup V_{i+1} \cup \{s\}, d, F)$
18:             **else**
19:                 $d' \leftarrow$ New-Distances$(V_i \cup \{s, t\}, d, F)$
20:             **end if**
21:             $\mathcal{D}_{i+1} \leftarrow \mathcal{D}_{i+1} \cup \{d'\}$
22:             $\rho \leftarrow \prod_{x \in F'} r_x \prod_{x \in F \setminus F'} (1 - r_x)$
23:             **if** μ_{i+1} has an element with index d' **then**
24:                 $\mu_{i+1}(d') \leftarrow \mu_i(d') + \mu_{i+1}(d) \cdot \rho$
25:             **else**
26:                 $\mu_{i+1}(d') \leftarrow \mu_i(d) \cdot \rho$
27:             **end if**
28:         **end for**
29:     **end for**
30: **end for**
31: $r \leftarrow 0$
32: **for all** $d \in \mathcal{D}_{\ell+1}$ **do**
33:     **if** d has a triad (s, t, δ) with $\delta \leq d_{max}$ **then**
34:         $r \leftarrow r + \mu_{\ell+1}(d)$
35:     **end if**
36: **end for**
37: **return** r

Figure 6.11: Pseudocode for computing the DCR in $\mathcal{G}_h$ graphs
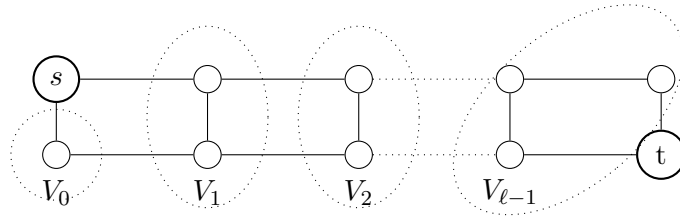
Figure 6.12: The ladder toplogy as a member of $\mathcal{G}_3$
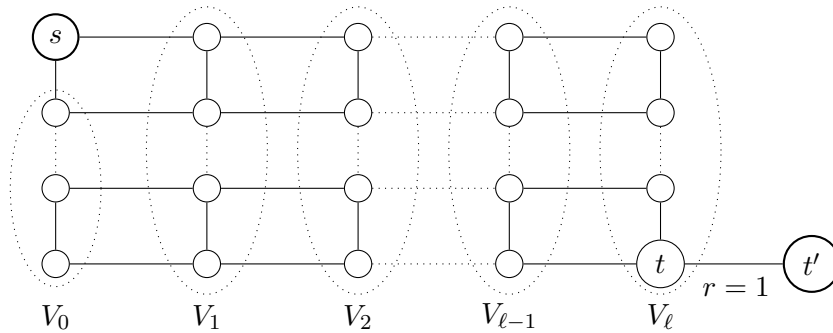
Figure 6.13: The multi-ladder family of networks

The theory of Random Graphs studies the asymptotic behaviour (in the number of nodes or edges) of the properties of graphs that grow following certain rules. Random graphs were first defined in [73, 74]; good reference books on the subject are [75, 76]. In the model proposed by Gilbert [74], denoted $G(n, p)$, the graph has n nodes and each possible edge between them occurs with a probability p. Erdos-Renyi [73] suggested a closely related model $G(n, m)$ in which a graph is chosen uniformly at random from the set of all graphs that have n nodes and m edges. Models introduced later tried to mimic properties and growth patterns of networks present in diverse disciplines and contexts like Biology, Sociology, website link referencing, academic publication referencing, etc. Among these the most studied ones are the Albert-Barabasi model [77] of preferential attachment for generating random scale-free networks, and the Watts-Strogatz model [78] that produces graphs with small-world properties.

The diameter of a graph is directly related to the all-terminal DCR. There are several known results for the diameter of random graphs (see e.g. [79, 80] and in particular the chapter 5.2 of [81]). Next we illustrate how some results on the diameter of random graphs can be applied to determine the asymptotic behaviour of the 2-DCR.

Consider a Gilbert's graph $G(n, p) = \langle V, E \rangle$. It has n nodes and is built with the following procedure: start with no edges, then for each pair of nodes u, v, draw an uniform real number s from $[0, 1]$ and add an edge (u, v) if and only if $s \geq p$, being $p \in [0, 1]$. It is clear that the expected value of $m = |E|$ is $n(n-1)p/2$. One can ask how different properties of $G(n, p)$ behave when $n \to \infty$. Let us first study the asymptotic

behaviour of the subproblem 2-$\{s,t\}$-DCR. The probability that the distance between both terminals s, t is above two is

$$\Pr(d(s,t) > 2) =$$
$$\Pr((s,t) \notin E \wedge (\forall w \in V \setminus \{s,t\}) \neg ((s,w) \in E \wedge (w,t) \in E)) =$$
$$(1-p)(1-p^2)^{n-2}$$

which tends to zero when $n \to \infty$ for any $p > 0$. So we have that

$$\lim_{n \to \infty} R(G(n,p), \{s,t\}, 2) = 1.$$

Moreover, the 2-K-DCR tends to one whatever the set K is. This result comes immediately from the well know fact ([82, 83] and, as a particular case of a more general result, [84]) that the probability that the diameter of a Gilbert's random graph $G(n,p)$ is two tends to one as $n \to \infty$ (provided that $p > 0$). Hence, the following lemma holds.

Lemma 6.4. *Given any real p and integer d such that $0 < p \leq 1$ and $d \geq 2$, regardless how the terminal set K evolves as $|V|$ grows, the d-K-DCR tends asymptotically to one, that is,*

$$\lim_{n \to \infty} R(G(n,p), K, d) = 1 \qquad \text{for } p > 0, d \geq 2.$$

Proof. Since the probability that the graph has diameter two tends to one, it then follows that the probability that the graph is d-connected for any $d \geq 2$ tends also to one. It follows that the same holds for the less restrictive condition that the graph is d-K-connected. $\qquad \square$

A more general result can be drawn from the following property, first proved in [85, 86]:

Property 2. *The probability that a Gilbert's graph $G(n,p)$ has diameter d tends to one when $n \to \infty$ if $(pn)^{(d-1)}/n \to 0$ and $(pn)^d/n \to \infty$.*

It immediately follows from Property 2 the following lemma.

Lemma 6.5. *Given any real p and integer d such that $(pn)^{(d-1)}/n \to 0$ and $(pn)^d/n \to \infty$, and any integer $d' \geq d$, regardless how the terminal set K evolves as $|V|$ grows, the d'-K-DCR tends asymptotically to one, that is,*

$$\lim_{n \to \infty} R(G(n,p), K, d') = 1.$$

The conditions in the hypothesis of Lemma 6.5, applied to the case $d = 2$, yield that $pn/n \to 0$ and $p^2 n \to \infty$, that is, $p\sqrt{n} \to \infty$. Therefore, observe that the 2-K-DCR keeps tending to one, even if the edge reliability degrades when n grows, provided that $n^{-1/2}$ is an infinitessimal of higher order than $p(n)$. For any integer $d \geq 2$, if we set $p(n) = n^\alpha$ with $\alpha < 0$, then the conditions of Lemma 6.5 are

$$n^{\alpha(d-1)}n^{d-2} \to 0 \quad \text{and} \quad n^{\alpha d}n^{d-1} \to \infty$$

and so

$$\alpha < -\frac{d-2}{d-1} \quad \text{and} \quad \alpha > -\frac{d-1}{d}.$$

Therefore, for any integer $d' \geq d$, the d'-K-DCR tends to one when $\alpha \in (-\frac{d-1}{d}, -\frac{d-2}{d-1})$. Observe that if $\alpha < \alpha' < 0$ then $\Pr(G(n, n^\alpha) \text{is connected}) < \Pr(G(n, n^{\alpha'}) \text{is connected})$, so the right edge of the interval can be set to zero. Therefore, the following lemma holds.

Lemma 6.6. *For any $d \geq 1$, if the edge reliabilities evolve as $p(n) > n^\alpha$ ($\alpha < 0$), then the d-K-DCR tends to one when $\alpha > -\frac{d-1}{d}$.*

Further work regarding the asymptotic d-K-DCR includes investigating its behavior with other models of random graphs as those of [77, 78].

Part IV

Monte Carlo analysis of DCR metrics

Chapter 7

About the application of Monte Carlo to the analysis of reliability

7.1 Motivation

We have seen that in the general case, as well as for several subproblems, both the CLR and the DCR problems belong to the $\mathcal{N}$P-hard computational complexity class of problems. This means that, even when using the best-performing known algorithms, computing $R(G, K)$ or $R(G, K, d)$ implies a number of operation that grows exponentially with the number of nodes n. This, in practice, limits the size of the networks for which the exact computation is feasible to several dozens of nodes (see e.g. Section 11.4 in [87]). This has motivated intense research, both on developing bounds and simulation techniques, to estimate the network reliability with precision goals set by each specific application and context. In this part of the thesis we focus on the application of Monte Carlo methods for computing the DCR, by means of a particular family of methods previously applied to the CLR. In Section 7.2 we resume the basic concepts behind Monte Carlo applied to the CLR. In Chapter 8 we adapt the mentioned family of methods to the DCR context and present several numerical results. In Chapter 9 we generalize this to estimate a network performability measure that we define in terms of the K-diameter.

The contents of this chapter were published in the journal Computer Communications; see [14].

7.2 Monte Carlo methods for the CLR

The estimation of any reliability measure R by the standard (also referred to as "crude" or "naive") Monte Carlo method consists of the following steps:

- draw N independent samples $X^{(1)}, \ldots, X^{(N)}$ of network configurations;

- compute the binary state $\Phi(X)$ for each sampled X;

- estimate R through the unbiased estimator $\widehat{R_N} = \frac{1}{N} \sum_{i=1}^{N} \Phi(X^{(i)})$.

7. ABOUT THE APPLICATION OF MONTE CARLO TO THE ANALYSIS OF RELIABILITY

The precision of the estimator will depend on the sample size N and on the probability that the network is in an operational state. Intuitively, a high precision means that the estimations obtained after completing several simulations will closely fluctuate around the real value R of the measure to estimate. Formally, the standard way to measure the estimator precision is by its variance. Since $\Phi(X)$ is a Bernoulli random variable, its variance is $\sigma_\Phi^2 = R(1 - R)$, and then the variance of the estimator $\widehat{R_N}$ is

$$\text{Var}(\widehat{R_N}) = R(1 - R)/N. \tag{7.1}$$

The Central Limit Theorem can be used to construct a confidence interval for $\widehat{R_N}$. So, the following items are strongly related; given any two of them, the third is perfectly determined:

- the variance $\text{Var}(\widehat{R_N})$;

- an interval $I = [R - \epsilon, R + \epsilon]$;

- the confidence level for the interval I (the probability that $\widehat{R_N}$ falls within the interval).

For instance, suppose we want to run a simulation experiment so that there is a 95% probability that the outcoming $\widehat{R_N}$ falls within an interval $[R - \epsilon, R + \epsilon]$. Then the Central Limit Theorem states that, for a sample size N sufficiently large, the variance must converge to $(\epsilon/1.96)^2$. Now, how can we build such a Monte Carlo experiment? Eq. (7.1) yields that

$$N = \frac{R(1 - R)}{(\epsilon/1.96)^2} \tag{7.2}$$

Hence, observe that the needed sample size N grows proportionally to the square of the confidence interval width ϵ. An important remark is that, as Eq.(7.2) shows, the sample size does depend on the exact value R that we want to determine. Since, in general, we do not know it[1], all we can do is compute an estimator for $\text{Var}(\widehat{R_N})$, i.e. an estimator for the variance of the reliability estimator. Un unbiased estimator $\widehat{\text{Var}}(\widehat{R_N})$ typically used for $\text{Var}(\widehat{R_N})$ is the following:

$$\widehat{\text{Var}}(\widehat{R_N}) = \widehat{R_N}(1 - \widehat{R_N})N/(N - 1).$$

7.2.1 The rare event issue

As discussed in Section 1.1, edge reliabilities in real networks are very high. Therefore, sampling a network configuration that corresponds to the state "the network fails" is a rare event. This means that a huge number of samples must be drawn in order to obtain a few cases where the network fails. Consider the ratio `relerr` between the confidence

---

[1]One could know the exact value of R for a certain network and want to test a certain simulation method; but in the general case, R is not known and can only be estimated.

interval width and the unreliability $1 - R$ of the network, known as the *relative error*. From Eq. (7.2) we have that

$$\texttt{relerr} = \frac{\epsilon}{1 - R} = z\sqrt{\frac{R}{N(1 - R)}}$$

where z is the Gaussian z-value that corresponds to the attained confidence level. Observe that, once fixed a certain sample size N and confidence level given by z, the relative error unboundedly grows as the network reliability R tends to one. In other words, when working with highly reliable networks, huge sample sizes may be needed in order to obtain an estimator for R that has an acceptable relative error. In [5] an example is given where the network has 14 nodes, 25 edges, mean degree 3.57 and uniform reliabilities equal to 0.99 resulting in a network unreliability $1 - R \approx 4.18 \times 10^{-8}$. In this example, just to have a couple of significant figures in $1 - \widehat{R_N}$, the sample size must exceed 1.6×10^{11}. This would be several orders of magnitude worse for real links, for which unreliability orders of -4, -5, -6 are much more realistic that only -2 (that of the example). To cope with this problem, several variance-reduction techniques have been proposed for estimating $R(G, K)$. Surveys can be found in [11, 87, 88]. More recent works include [89, 90, 91, 92, 93]. These methods allow to significantly reduce the sample size yet attaining the same confidence interval/levels than when using a crude Monte Carlo simulation. It is important to note that the crude Monte Carlo method is such that nothing is said about the structure function Φ; the only desirable property to expect from it is that it is easily computable for a given configuration X. This is something to bear in mind when working with more complex methods since many of them rely on certain properties of the measure R (e.g. coherence). Another typical issue that must be studied for each simulation methods in the rare event context is the *robustness* of the method. This relates to the degree up to which an estimated confidence interval matches the theoretical one when $R \to 1$. They will be in general some mismatch, because the variance of the estimator of R must be estimated itself, as we saw above. The gains offered by variance-reduction techniques, in terms of needing smaller sample sizes, have a counterpart with regards to the higher execution times that they may involve. This is due to the nature of their sampling procedures, that are in general more complex than the simple Bernoulli's experiments needed to sample a random configuration in crude Monte Carlo. From the user's point of view, what matters is the balance between precision and execution time. The *relative efficiency* is a measure that allows to compare the efficiency of several simulation techniques, taking into account both the ratios between spent times and variances; we define (and apply it) in Section 8.4.

The Monte Carlo techniques used for estimating CLR measures can in general be adapted to the DCR context, provided that convenient mechanisms for sampling and computing Φ in diameter-sensitive contexts are devised. We do it next for a family of methods previously introduced for CLR estimation.

Chapter 8

Monte Carlo estimation of the DCR conditioned by pathsets and cutsets

In this chapter we focus on a family of Monte Carlo sampling plans described in [94], based on employing *a priori* knowledge about pathsets and cutsets (see Section 8.2.1). Van Slyke and Frank [95] and Kumamoto, Tanaka and Inoue [96] suggested such sampling plans, that operate by restricting the sampling space, thanks to the previous knowledge of a set of pathsets and cutsets. Fishman includes in [94] these previous works under a general framework of conditioned Monte Carlo sampling plans and in [97] compares them to other sampling plans for $K = \{s, t\}$, evidencing the advantages of this approach. This chapter proceeds as follows. Section 8.1 introduces the needed definitions and notation. Section 8.2 presents the general framework and discusses the consequences that switching to the DCR context has, regarding the topological components employed for computing the bounds (pathsets and cutsets) and each step of the Monte Carlo simulation. Section 8.3 briefly describes each sampling plan comparing their requirements and limitations. Section 8.4 presents numerical results obtained after running tests on sparse and dense topologies. It also illustrates how the existence (and its value) of a diameter constraint and the size of the set of terminal nodes affect the attained performance. The tests also illustrate how the performance improvements over the crude Monte Carlo method tend to be higher in the context of the DCR when compared to the CLR. Finally Section 8.5 summarizes main conclusions.

8.1 Definitions and notation

For ease of reference we resume here the relevant notation for this chapter. As usual an instance of the CLR is defined by an undirected graph $G = (V, E, \vec{r})$ with $n = |V|, m = |E|, E = \{e_1...e_m\}$, a set of terminal nodes $K \subseteq V$ with $k = |K|$ and known probabilities of operation $\vec{r} = (r_e)_{e \in E}$ for each edge. Recall that edge states are assumed to be independent of one another. The addition of a diameter constraint d defines an instance of the DCR which we denote by d-DCR. For each $e \in E$, let X_e be a random binary variable whose value is 1 if e operates and 0 otherwise. We call the *network configuration*

or *state vector* any m-tuple $X = (X_1...X_m) \in \{0,1\}^m$ encoding the states of all edges. Let $\mathcal{X}$ be the set of the 2^m possible configurations and denote by $\pi(x)$ the probability that the random configuration is x, that is, $\pi(x) = \Pr(X = x)$. We denote as Φ the structure function that maps each configuration $x \in \mathcal{X}$ into 1 or 0 according to the fact that the network operates or fails when the components are in the states encoded in x. As a consequence, $\Phi(X)$ is a random network state and $R(G, K, d) = \Pr(\Phi(X) = 1)$. For ease of reference, we next list the variables and notations that appear in places other than just where they are defined.

- R: network reliability (CLR or DCR given by context);

- P, C: sets of d-pathsets and d-cutsets *a priori* known (see Section 8.2.1);

- L, U: the sets of those configurations where at least one of P operates (L) and at least one of C fails (U);

- Γ: the set of those configurations not in L nor in U;

- R_L, R_U: lower and upper bound for R, given by L and U (see Section 8.2);

- $\widehat{R_N}, \widehat{\mathrm{Var}(R_N)}$: unbiased estimators for R and its variance obtained using any simulation method;

- N: simulation sample size (number of iterations);

- M, N: size of the lowest-cardinality d-pathset and d-cutset that exist according to G, d and K (see Section 8.3.2);

- Ω: set of edges that belong to at least one element of P or C.

8.2 Conditioned Monte Carlo simulation

The d-DCR for a graph G with terminal set K can be computed as $R = R(G, K, d) = \sum_{x \in \mathcal{X}} \Phi(x)\pi(x)$. By sampling N configurations $\{X^{(1)}...X^{(N)}\}$ according to the probability distribution given by $\pi(x)$ (crude Monte Carlo sampling plan) we can build an unbiased estimator $\sum_{i=1}^{N} \Phi(X^{(i)})/N$ for R whose variance is $R(1-R)/N$. Suppose we partition $\mathcal{X}$ into three pairwise disjoint sets U (set of configurations known to fail i.e. for which $\Phi = 0$), L (set of configurations known to be operational i.e. $\Phi = 1$) and Γ (set of configurations for which we do not know the state of the network), as shown in Figure 8.1. Then, defining $R_L = \pi(L) = \sum_{x \in L} \pi(x)$ and $R_U = 1 - \pi(U) = 1 - \sum_{x \in U} \pi(x)$ we have that $R_L \leq R \leq R_U$ (R_L and R_U are lower and upper bounds for the DCR) and that

$$R = R_L + \Pr(X \in \Gamma \wedge \Phi(X) = 1)$$
$$= R_L + (R_U - R_L)\Pr(\Phi(X) = 1 \mid X \in \Gamma).$$
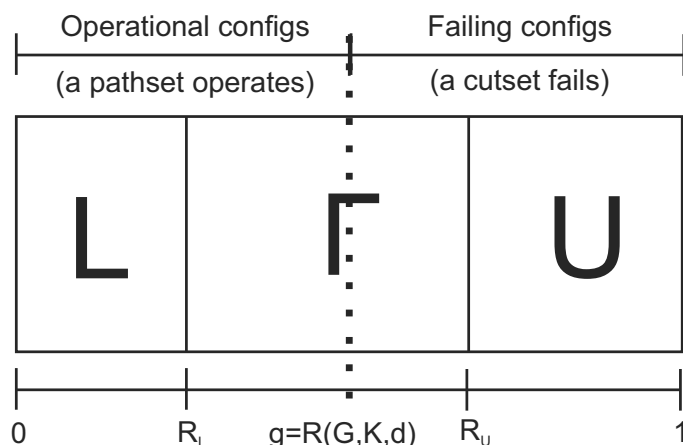
Figure 8.1: Monte Carlo sampling conditioned by pathsets and cutsets

So, if R_L, R_U are known, and if we can sample the component states from the conditional probabilities $\Pr(X_e = 1 \mid X \in \Gamma)$, one can draw N configurations from Γ according to that conditional probability and build an unbiased estimator $\widehat{R_N} = R_L + (R_U - R_L) \sum_{i=1}^{N} \Phi(X^{(i)})/N$ for R whose variance can be shown to be $(R_U - R)(R - R_L)/N$ and thus unbiasedly estimated as $\widehat{\mathrm{Var}(\widehat{R_N})} = (R_U - \widehat{R_N})(\widehat{R_N} - R_L)/(N-1)$, achieving a variance reduction against the crude sampling plan if $R_L > 0$ or $R_U < 1$. This is presented in [94] together with bounds on the amount of samples needed to achieve desired confidence level and interval width goals for the absolute and relative errors. The methods here discussed differ in the way the sets U and L are defined and the way they implement a configuration sampling coherent with $\Pr(. \mid X \in \Gamma)$. Unless otherwise specified we will assume that the edge states are independent. A generic pseudocode for these simulations is shown in Figure 8.2. There are four basic steps for which we next discuss the differences introduced when working in DCR context respect to CLR.

8.2.1 Gathering information about the topology of the network that is useful for bounding the configurations space

The following discussion applies to the steps "Precomputing" and "Bounding" of Figure 8.2. The most common topological structures used in the literature for computing, bounding and estimating network reliability are the concepts of *pathset* and *cutset*. Recall that a pathset is any set of edges such that if they all operate, the network operates, regardless of the state of edges not in the pathset. Similarly, a cutset is any set of edges such that if they all fail, the network fails as well, regardless the state of edges not in the cutset. These definitions are also valid for the DCR context, where then notation d-pathset and d-cutset is used to make explicit that there is a diameter parameter d. A pathset is said to fail when at least one of its edges fails (otherwise the pathset is said to be operational); whereas a cutset is said to fail when all of its edges fail (otherwise it is said

8. MONTE CARLO ESTIMATION OF THE DCR CONDITIONED BY PATHSETS AND CUTSETS

Procedure Simulate(G, K, d, π, N)

1: Precomputing: $Prop \leftarrow$ (properties of the graph, subsets, auxiliary tables, etc.)
2: Bounding: $R_L, R_U \leftarrow$ compute-bounds$(G, K, d, \pi, Prop)$
3: $Z \leftarrow 0$
4: **for all** $iter \in \{1 \dots N\}$ **do**
5:     Sampling: $x \leftarrow$ sample-a-configuration$(\pi, Prop)$
6:     Checking: if connected(x, K, d) then $Z \leftarrow Z + 1$
7: **end for**
8: $\hat{R} \leftarrow R_L + (R_U - R_L)Z/N;$
   $\hat{\sigma}^2 \leftarrow (R_U - R_L)^2(1 - Z/N)Z/(N(N-1))$
9: **return** $\hat{R}, \hat{\sigma}^2$

Figure 8.2: Generic conditioned Monte Carlo simulation pseudocode

to be operational). Both pathsets and cutsets are said to be minimal if there is no edge whose suppression still yields a pathset or a cutset, respectively. Some sampling plans discussed here require determining the number of edges in the pathsets and cutsets with smallest cardinality. Following [94] we denote these cardinalities as M and N respectively. In the CLR context, determining M takes $O(n^{2/3}m)$ time and determining N takes $O(m)$ time for $k \in \{2, n\}$ and $O(kmn)$ in general. Other plans take advantage of knowing as many edge-disjoint pathsets and cutsets as possible. Building N disjoint pathsets and M disjoint cutsets[1] (useful for the plans of Section 8.3.3) takes $O(Mm)$ and $O(m)$ times respectively. See [94] for the above results. All these tasks become more complex when working with diameter constraints. Golovach and Thilikos [9] summarize the complexity of the decision problems asking for the existence of edge-disjoint pathsets and cutsets with a given cardinality κ for a given constraint d. These problems are NP-hard when one or both of κ and d are considered as inputs and belong to the FTP complexity class[2] when both are parameters. Note that a certain edge set that is a pathset under diameter d is not necessarily a pathset under diameter $d' < d$; whilst all pathsets under diameter d' are also pathsets under diameter d. The opposite happens regarding cutsets. In other words, for the same topology, as the required diameter decreases there are fewer pathsets and more cutsets, which shifts the [lower bound, upper bound] intervals computed through pathsets and cutsets suggested in Section 8.3.3. This is intuitive since the DCR decreases monotonously when d decreases (the constraint being more demanding). We will assume that M, N and/or a set of pathsets and cutsets are given *a priori* though finding them can be a difficult task (particularly for the DCR as seen). It is useful to treat these precomputing tasks as problems separated from the simulation itself; once solved for a given topology and set K they can be used multiple times for estimating reliabilities

---

[1]I.e. as many pathsets as the smallest cardinality of a cutset, and as many cutsets as the smallest cardinality of a pathset, as suggested in [94] for bounding the simulation relative error.

[2]The class of complexity of the parameterized problems taking time $O(f(p)g(i))$ where f is any function on the parameters p and g is any polynomial on the inputs i.

with different values of $\vec{r} = (r_e)_{e \in E}$, for instance to perform a sensitivity analysis, or for optimizing purposes.

8.2.2 Sampling configurations

As discussed in Section 8.3, in this step a configuration must be randomly chosen, from the subspace yielded after dropping off the configurations for which the network state is known thanks to the bounding method applied. The selection must be made with probabilities proportional to the ones in the original configuration space. As we will see, regarding the algorithms applied in this step in the sampling plans here discussed, no differences arise when moving from the CLR to the DCR context.

8.2.3 Determining the network state for a sampled configuration

After sampling a configuration x it must be determined whether it corresponds to an operational or to a failed network state (by computing $\Phi(x)$). This can be done in $O(m)$ time for the CLR running a breadth-first-search (BFS) starting at any node of K. Under DCR the time required is in general $O(km)$ (BFS with depth bounded by d starting at each node of K). The values of d and k determine whether this step takes more or less time in DCR than in CLR. With low values of d each BFS might execute faster in DCR than in CLR since it reaches a depth of at most d. On the contrary high values of k will force to run several BFS in DCR instead of one in CLR thus increasing the checking time in DCR. The effect of both parameters is made evident in Section 8.4, where we show their impact on the relative efficiency (RE) of the methods against crude Monte Carlo attained in CLR and DCR.

8.3 Description of the methods

8.3.1 SIM1: Crude Monte Carlo

As seen above, when applying crude Monte Carlo, the sets U and L are empty (no information about failing or operational configurations is used when sampling) and thus $R_L = 0$, $R_U = 1$ and $\sigma_N^2 = R(1 - R)/N$. The steps "Precomputing" and "Bounding" are trivial (nothing to compute). The step "Sampling" is $O(m)$ since it involves running m Bernoulli trials, one for each edge e with success probability $\Pr(X_e = 1)$ if failures are independent, or conditioned to the already sampled edges to compute $\Pr(X_e = 1 \mid$ "state of already sampled edges") if there are dependencies. The step "Checking" is identical for all methods and was discussed in Section 8.2.3.

8.3.2 SIM2: Bounds based on minimal cardinality pathsets and cutsets

In [95] Van Slyke and Frank suggested the following way to build U and L. Let M be the cardinality of the pathset with the lowest cardinality (for a given instance G, K, d) and let

N be the cardinality of the cutset with the lowest cardinality. It is clear that any configuration with fewer than M operational edges leads to a failing network. At the same time, any configuration with more than $m-N$ operational edges leads to an operational system. So we can define U as the set of all configurations having more than $m-N$ operational edges and L as the set of all configurations having fewer than M operational edges. The step "Precomputing" implies determining M and N (refer to Section 8.2.1) which can be more difficult for DCR than CLR; in large instances it can be easier to bound M and N by certain integers $M' \leq M$ and $N' \geq N$ though getting potentially less tight bounds. Determining R_L involves adding the occurrence probabilities of all configurations with fewer than M operational edges. Note that when working with uniform edge reliabilities p ($r_e = p$ for all edge e) any configuration with j operational states has an occurrence probability equal to $p^j(1-p)^{m-j}$. Then, defining $F_i(m,p) = \sum_{j=0}^{i} \binom{m}{j} p^j(1-p)^{m-j}$ we have that $R_L = F_{M-1}(m,p)$ and $R_U = 1 - F_{m-N}(m,p)$ computable in $O(M)$ and $O(m-C+1)$ times respectively. When probabilities are not uniform, algorithms known in the literature as of the "k-out-of-n" type must be used [98, 99, 100]. Computing a table with all values for F can be seen as part of "Precomputing". Then, "Sampling" implies that a random configuration must be drawn with at least M but fewer than $m-N$ operational edges according to their probabilities of occurrence. To do so with uniform probabilities involves two steps. First, one determines the number Q of edges by choosing an uniformly distributed random cutpoint U in the interval $(F_{M-1}(m,p), 1-F_{m-N}(m,p))$ and then applying the ordinary inversion method on a table of cumulative $F_i(m,p)$ values, in order to obtain a value for i and assign it to Q. Second, a configuration with such number of operational edges is randomly built by generating an uniformly distributed random integer in $[0, \binom{m}{Q})$ and "translating it" into a unique set of Q operational edges through a bijective function (in [94] the k-canonical representation of integers of Kruskal-Katona is suggested). Both can be executed in $O(m)$ time provided that tables with the values of F and precomputed needed combinations exist. Note that the absence or presence (and value) of a diameter constraint d has no explicit impact on the algorithm used for sampling edges, though it has an indirect impact through the values of M and N, since the minimum-cardinality pathsets and cutsets depend on the constraint d.

8.3.3 SIM3, SIM4, SIM5: Bounds based on precomputed pathsets and cutsets

Kumamoto, Tanaka and Inoue [96] suggested a method that takes advantage of more information about the topology of the problem instance than just knowing M and C, namely knowing a set of I pathsets $P = P_1...P_I$ and a set of J cutsets $C = C_1...C_J$; this method is presented under a more comprehensive framework in [94]. The set U (as defined in Section 8.3 together with L) will be exactly the set of all configurations where at least one of $C_1...C_J$ fails (thus making the network fail). The set L will be exactly the set of all configurations where at least one of $P_1...P_I$ is operational (thus guaranteeing that the network is operational). So the set Γ from which samples must be drawn contains all configurations for which we do not have evidence of their state if only considering the information given by P and C.

Let us first address the general case where the elements of P are not required to be pairwise disjoint nor are the elements of C; let us call this plan SIM3. "Precomputing" in this context involves generating the sets P and C which (not having to be edge-disjoint) can be easily done through DFS algorithms for P (both for CLR and DCR) and drop-and-test algorithms for C (observe that we are not constrained to generate all pathsets or cutsets; this is just a matter of trade-off between precomputing effort and bounds tightness). "Bounding" (computing R_L and R_U) can be exponentially complex if the elements in P and/or C are numerous and highly overlap (though trivial if they are disjoint, by factoring probabilities). Regarding "Sampling" let Ω be the set of all edges in $P_1...P_I$ and $C_1...C_J$. Sampling edges in $E \setminus \Omega$ involves just $m - |\Omega|$ Bernoulli trials. Edges in Ω can be sequentially sampled in $O(|\Omega|(\sum_{h=1}^{I} |P_h| + \sum_{h=1}^{J} |C_h|)$. In [94] formulae are presented for sampling the state of the edges in Ω. Although they are presented in a context where general (not necessarily disjoint) pathsets and cutsets are used, they are indeed only valid for disjoint pathsets and cutsets. In Section 8.6 we concisely derive valid formulae both for non-disjoint and disjoint contexts. A variant presented in [94] that we call SIM4 implies precomputing the probability of the occurrence of each of the $2^{|\Omega|}$ possible sub-configurations that exist when only considering the edges of Ω, in $O(2^{|\Omega|})$ time. Then sampling involves just choosing any of them through a random cutpoint access on a table accumulating the precomputed probabilities (thus in $O(m)$ time). This variant is feasible only for limited-sized sets Ω and thus limited-sized problem instances (because the bounds loosen as m grows with a fixed $|\Omega|$). A third variant that we call SIM5 makes use only of pairwise disjoint pathsets $P_1...P_I$ and cutsets $C_1...C_J$. Disjointness allows an easy factorized computation of R_L and R_U in $O(\sum_{h=1}^{I} |P_h|)$ and $O(\sum_{h=1}^{J} |C_h|)$ steps respectively, and sampling of all edges in $O(m)$; the algorithm, presented in [94] with mistyped formulae as noted in [101], is included together with its proof in 8.6. It is important to note that the previous discussions about SIM3, SIM4 and SIM5 are valid for non-uniform probabilities. Again, note that for SIM3, SIM4 and SIM5 the absence or presence (and its value) of a diameter constraint d has no explicit impact on the sequential computation of the probabilities for sampling the edges of Ω. Yet there is an indirect impact when building the sets P and C since their elements must be then d-pathsets and d-cutsets for a given value of d. Once in the sampling phase their elements intrinsically carry the constraint within their topology but the sampling algorithm remains identical (and independent of d) in CLR and DCR.

8.3.4   Comparison of sampling plans

Table 8.1 compares the requirements on uniformity of probabilities, previous-to-simulation knowledge needed and times order for precomputing, bounding and sampling. The notation $\exp(P, C)$ means "potentially exponential in the edges of P and C". As the tests show later, there are significant differences in time even in cases with identical order.

Table 8.1: Comparison of computational orders, by step and sampling plan, for the DCR.

.	SIM1	SIM2	SIM3	SIM4	SIM5
$r_{e \in E}$	Any	Uniform	Any	Any	Any
Pre-knowledge	-	min.sizes M, N	P, C general	P, C general	P, C disjoint
Precomputing	-	m (for table F)	-	$2^{|\Omega|}$	-
Bounding	-	$M + m - N$	exp(P,C)	exp(P,C)	$\sum |P_i| + \sum |C_i|$
Sampling	m	m	$\sum |P_i| + \sum |C_i|$	m	m

8.4 Experimental results

8.4.1 Test instances

In order to study the performance of the methods presented in the previous sections, we provide computational results obtained in four different tests. Tests I and II are based on the "dodecahedron" network, a common topology in network reliability literature, shown in Figure 8.3. Test I compares the performance of all methods with different d values for a terminal set $K = \{s, t\}$. Test II compares SIM5 and crude Monte Carlo when specifying a larger set K and diameter d. Test III is based on the countrywide transport network topology of ANTEL, the largest telecommunications provider in Uruguay, shown in Figure 8.4. It also illustrates the effect of specifying different d values as well as two different cases of link reliabilities. In these cases a comparison against CLR is also done. Test IV aims to illustrate the behaviour of the methods when applied to dense networks. It is based on the networks $G_{15,3}$, $G_{15,4}$ and $G_{15,5}$ (we denote as $G_{n,h}$ the uniform graph with n nodes labeled $0 \cdots n - 1$ and where the set of neighbors of each node i is exactly $\cup_{j=-h...h, j \neq i} \{i +_n j\}$ being $+_n$ the addition modulo n). The diameter is set to three and the terminals are $K = \{0, 8\}$. The performance is compared for different link reliabilities.

In particular it is interesting to analyze how the RE of each variance reduction methods change when moving from the CLR to the DCR context; this has much to do with the following reason. Let τ_I^N and τ_{II}^N be respectively the times consumed to generate a sample configuration under two sampling plans I and II (e.g. crude Monte Carlo and a variance reduction plan) and let τ^Φ be the time consumed for evaluating the corresponding Φ. Then the RE among both plans is given by the ratios among the times and the variances obtained, σ_I^2 and σ_{II}^2; that is RE $= (\tau_I^N + \tau^\Phi)/(\tau_{II}^N + \tau^\Phi) \times \sigma_I^2/\sigma_{II}^2$. We see that a growth in τ^Φ softens the effect of the difference between τ_I^N and τ_{II}^N, increasing the impact on the RE of the difference between σ_I^2 and σ_{II}^2.

All methods were implemented in `C++` and the tests were run on an Intel Core2 Duo T5450 machine with 2 GB of RAM; reported times are expressed in seconds.

8.4.2 Test I

In this test we set $K = \{1, 9\}$, $r_e = 0.95$ for all $e \in E$, and we estimated $R(G, K, d)$ running all the discussed methods with a sample size $N = 2^{18}$ and diameter constraints
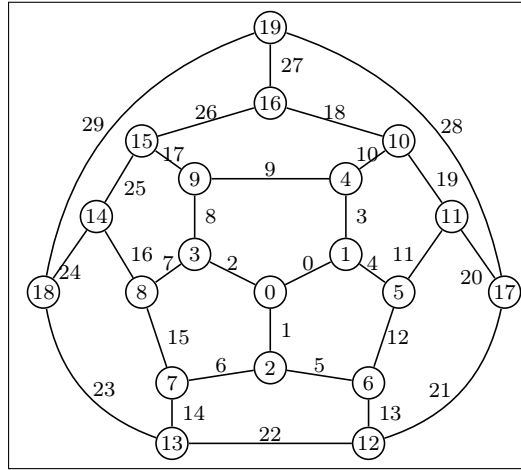
Figure 8.3: dodecahedron network (for test cases I and II)

$d = 2, 3, 4, 5, 6, 7$ plus CLR ($d = \infty$). The bounds (M, N) for SIM2 are $(2, 1)$ for $d = 2$, $(2, 2)$ for $d = 3, 4$ and $(2, 3)$ for $d = 5, 6, 7, \infty$. The disjoint sets P contain

- for $d = 2$: $\{3, 9\}$;

- for $d = 3, 4, 5$: $\{3, 9\}, \{8, 2, 0\}$;

- for $d = 6, 7, \infty$: $\{3, 9\}, \{8, 2, 0\}, \{17, 26, 18, 19, 11, 4\}$,

while the disjoint sets C contain

- for $d = 2$: $\{3\}, \{9\}$;

- for $d = 3, 4$: $\{0, 3\}, \{8, 9\}$;

- for $d = 5, 6, 7, \infty$: $\{0, 3, 4\}, \{8, 9, 17\}$.

For SIM3 and SIM4 larger (non-disjoint) sets P, C are used for most values of d as follows. Sets P contain

- for $d = 5$: $\{3, 9\}, \{9, 10, 19, 11, 4\}, \{8, 2, 0\}$;

- for $d = 6$: $\{3, 9\}, \{8, 2, 0\}, \{17, 26, 18, 19, 11, 4\}, \{17, 26, 18, 10, 3\}, \{8, 7, 15, 6, 1, 0\}$;

- for $d = 7, \infty$: $\{3, 9\}, \{8, 2, 0\}, \{17, 26, 18, 19, 11, 4\}, \{17, 26, 18, 10, 3\}$, $\{8, 7, 15, 6, 1, 0\}, \{17, 25, 16, 7, 2, 0\}$,

while the sets C contain

- for $d = 3, 4$: $\{0, 3\}, \{8, 9\}, \{2, 9\}, \{3, 8\}$;

97
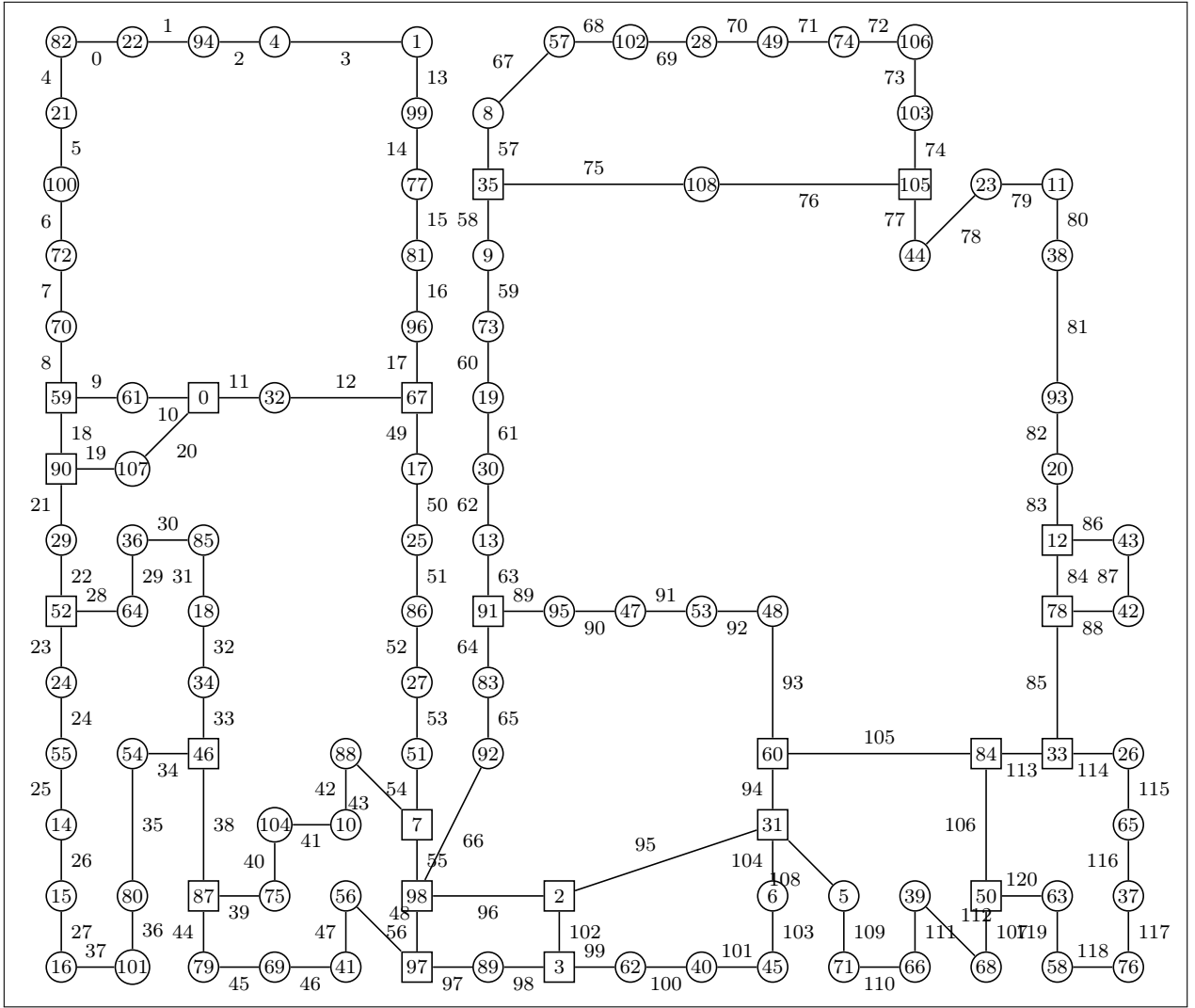
Figure 8.4: Transport network topology of ANTEL (Uruguayan national telecommunications provider; for test case III)

- for $d = 5$: $\{0, 3, 4\}, \{8, 9, 17\}, \{3, 8, 19\}, \{3, 8, 9\}$;

- for $d = 6, 7, \infty$: $\{0, 3, 4\}, \{8, 9, 17\}, \{8, 9, 25, 26\}, \{0, 3, 11, 12\}$.

Table 8.2 presents the results. Simulation times are reported as t_{sim}. $\text{RE}_{sim(z)}$ stands for the relative efficiency of SIM(z) when compared to SIM1. First, note that the estimated reliability grows when d goes from 2 to 7 and then ∞ as expected due to the progressive loosening of the diameter constraint. Second, observe that the variance is zero for SIM3,4,5 and $d = 2, 3, 4$ due to the fact that the set Γ is empty in those cases (the used pathsets and cutsets easily determined by visual inspection capture all

98
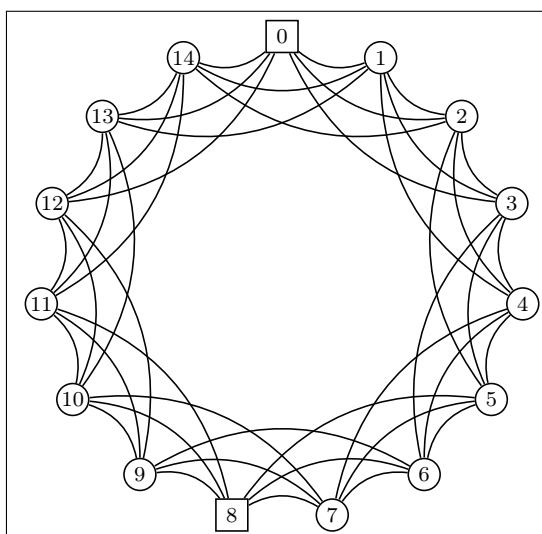
Figure 8.5: Dense graph $G_{15,3}$ (one of the familiy of networks for test case IV)

information about R). Third, note that execution times tend to grow when d grows, due to the deeper BFS performed when checking each sample. Fourth, with regard to relative efficiency RE, SIM2 outperforms crude Monte Carlo for all $d \geq 5$ and SIM3, 4, 5 for all d. SIM4 has by far the largest RE ratios though one has to take into account the limitations already discussed in Section 8.3.3 (e.g. in this case building the precomputed tables took a time equivalent to that consumed by the simulation itself and would exponentially grow for larger topologies). SIM3 does not require precomputing the table but it does require a significant effort to find a minimal-size polynomial to compute the probabilities for sampling edges in Ω; and with highly overlapping sets P, C (as is for $d = 5, 6, 7$) the computation time growths outweigh the gains in bound tightness delivering a poorer performance when compared to SIM5. Fifth, in this case ($k = 2$) the time needed for sample checking is equal or lower for DCR than for CLR (one single BFS in both cases, potentially deeper for CLR) and thus the RE in CLR outperforms those in DCR; this changes in Tests II and III as we see next.

8.4.3 Test II

In this test we set $K = \{7, 9, 11, 12, 18, 19\}$, $r_e = 0.95$ for all $e \in E$, and we estimated $R(G, K, 9)$ and $R(G, K)$ comparing crude Monte Carlo with SIM5. The following two disjoint pathsets are employed:

- $\{19, 10, 9, 17, 26, 27, 29, 23, 14, 22\}$,

- $\{28, 20, 11, 4, 0, 2, 7, 8, 16, 24, 21, 13, 5, 6, 15\}$;

as well as the following five disjoint cutsets:

- $\{27, 28, 29\}$,

99

Table 8.2: Numerical results for test case I

Diameter (d)	2	3	4	5	6	7	∞
$\hat{R}$	0.902943	0.986343	0.986263	0.997276	0.999355	0.999577	0.999702
$\widehat{\mathrm{Var}}(\hat{R})_{sim1}$	3.34E-07	5.14E-08	5.17E-08	1.04E-08	2.46E-09	1.61E-09	1.13E-09
t_{sim1}	3.29	3.62	3.62	3.64	3.65	3.67	2.68
$\widehat{\mathrm{Var}}(\hat{R})_{sim2}$	2.54E-07	2.30E-08	2.30E-08	2.04E-09	4.49E-10	3.18E-10	2.10E-10
t_{sim2}	10.12	10.33	10.19	10.36	10.39	10.45	9.41
RE_{sim2}	0.43	0.78	0.80	1.78	1.92	1.78	1.54
$\widehat{\mathrm{Var}}(\hat{R})_{sim3}$	0.00E+00	0.00E+00	0.00E+00	4.83E-11	2.20E-12	6.96E-13	1.71E-13
t_{sim3}	11.47	17.39	17.26	34.37	51.84	59.92	46.74
RE_{sim3}	∞	∞	∞	22.67	78.71	141.89	381.08
$\widehat{\mathrm{Var}}(\hat{R})_{sim4}$	0.00E+00	0.00E+00	0.00E+00	4.83E-11	2.20E-12	6.96E-13	1.71E-13
t_{sim4}	2.92	5.21	4.25	6.66	7.80	8.06	6.12
RE_{sim4}	∞	∞	∞	117.07	523.03	1054.48	2908.92
$\widehat{\mathrm{Var}}(\hat{R})_{sim5}$	0.00E+00	0.00E+00	0.00E+00	1.10E-10	4.65E-12	2.57E-12	5.40E-13
t_{sim5}	2.89	4.56	4.73	7.77	8.57	8.69	5.80
RE_{sim5}	∞	∞	∞	44.06	225.00	265.34	971.93
Bounds							
$R_L(\mathrm{SIM2})$	0.214639	0.553542	0.553542	0.812179	0.812179	0.812179	0.812179
$R_U(\mathrm{SIM2})$	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
$R_L(\mathrm{SIM3,4})$	0.902500	0.986094	0.986094	0.991612	0.997850	0.998447	0.998447
$R_U(\mathrm{SIM3,4})$	0.902500	0.990381	0.990381	0.999513	0.999738	0.999738	0.999738
$R_L(\mathrm{SIM5})$	0.902500	0.986094	0.986094	0.986094	0.996316	0.996316	0.996316
$R_U(\mathrm{SIM5})$	0.902500	0.995006	0.995006	0.999750	0.999750	0.999750	0.999750

Table 8.3: Numerical results for test case II

.	$\hat{R}$	$\widehat{\text{Var}(\hat{R})}$	t_{sim}	R_L	R_U	RE
SIM1 (CLR)	0.999138	3.49E-09	4.63	0.000000	1.000000	
SIM5 (CLR)	0.999130	2.00E-10	6.26	0.784639	0.999375	12.89
SIM1 ($d = 9$)	0.999081	3.72E-09	19.44	0.000000	1.000000	
SIM5 ($d = 9$)	0.999105	2.05E-10	21.26	0.784639	0.999375	16.59

- $\{8, 9, 17\}$,

- $\{11, 19, 20\}$,

- $\{6, 14, 15\}$,

- $\{13, 21, 22\}$.

Table 8.3 presents the results. As expected the estimated reliability is slightly higher for CLR than for DCR (operational configurations with diameters higher than nine exist although they represent a small fraction of the total). In this test, the values of k and d are higher than in Test I. This causes the checking time for DCR (which is the same for SIM1 and SIM5) to gain relevance in the total time, and thus the ratio of SIM5 time respect to SIM1 is significantly lower in DCR (21.26/19.44) than that for CLR (6.26/4.63). With similar variance ratios (0.205/3.72 vs 0.200/3.49), then the RE is higher for DCR than for CLR (16.59 vs 12.89). The lower reliability bound R_L is less close to R than R_U due to the fact that the employed pathsets are fewer (and larger) than the cutsets; disjointness is often a very tough requirement, particularly for sparse topologies. When relaxed by allowing to share some edges many more pathsets can be taken into account getting tighter lower bounds though at the expense of more costly computation of probabilities when building samples.

8.4.4 Test III

The network for this case has 121 edges and 109 nodes among which 22 are terminal (squares in Figure 8.4). SIM1 and SIM5 were run with $N = 2^{20}$ samples, uniform edge reliabilities $p = 0.99$ and $p = 0.999$ and diameters $d = 25, 30, 35$ plus ∞ (CLR). Set P consists of a single pathset P_1 with the following edges: 18, 21, 22, 19, 20, 11, 12, 49, 50, 51, 52, 53, 54, 43, 42, 41, 40, 39, 38, 55, 56, 66, 65, 64, 63, 62, 61, 60, 59, 58, 75, 76, 96, 102, 95, 94, 105, 106, 113, 85 and 84. Set C consists of fifteen disjoint cutsets, easily generated by grouping the edges incident to fifteen terminal nodes, namely nodes 59, 0, 67, 52, 46, 7, 97, 3, 91, 35, 105, 31, 50, 33 and 12. The results are shown in Table 8.4. First observe that the constraint $d = 25$ is very demanding for this topology and set K (e.g. the distance between terminals 52 and 105 is 24) hence the reliability drop obtained for $d = 25$ when compared to the other values of d. This happens particularly

Table 8.4: Numerical results for test case III

.	$\hat{R}$	$\widehat{\mathrm{Var}}(\hat{R})_{sim1}$	$\widehat{\mathrm{Var}}(\hat{R})_{sim5}$	t_{sim1}	t_{sim5}	RE
$p = 0.99$						
CLR	0.990734	8.65E-09	2.90E-09	95.77	108.94	2.62
$d = 35$	0.984802	1.41E-08	4.67E-09	1596.03	1583.41	3.05
$d = 30$	0.953564	4.21E-08	1.29E-08	1547.34	1469.22	3.44
$d = 25$	0.788949	1.59E-07	2.55E-08	1335.24	817.93	10.17
$p = 0.999$						
CLR	0.999916	7.64E-11	3.20E-12	95.49	108.64	21.00
$d = 35$	0.999866	1.27E-10	5.13E-12	1628.81	1641.85	24.62
$d = 30$	0.998572	1.38E-09	5.28E-11	1613.50	1598.75	26.46
$d = 25$	0.977157	2.13E-08	3.78E-10	1590.04	945.27	94.84

for $p = 0.99$ because with lower edge reliabilities many failed configurations are sampled when working with d values close to 24. Second, observe that the RE (SIM5 vs SIM1) is higher for $d = 25, 30, 35$ than for CLR. One reason is that the time spent on checking d-connectedness is much higher than that spent on checking simple connectedness (as a consequence of high values of k and d) which dilutes the lower sampling time implied by a simpler computation of samples for SIM1. Another reason (particularly visible for $d = 25$ but also for $d = 30$) is that execution times are indeed lower for SIM5 than for SIM1. This is due to the fact that forcing P_1 to fail (in order to sample within Γ) often leads to sampling failed configurations, quickly identified as such when being checked, thus cutting down average iteration times in SIM5.

8.4.5   Test IV

The three networks $G_{15,3}$, $G_{15,4}$ and $G_{15,5}$ used in this case have 15 nodes and 45, 60 and 75 edges respectively and allow to illustrate the performance attained as the network becomes denser. Figure 8.5 shows the fifteen nodes labeled $0 \cdots 15$, and the edges for the case $G_{15,3}$. The methods SIM1 and SIM5 were run with $N = 2^{18}$ samples, uniform probabilities $p = 0.40$, $p = 0.60$, $p = 0.80$ and $p = 0.95$ and diameter $d = 3$. Two terminals were specified ($K = \{0, 8\}$), shown as squares in Figure 8.5.

Figure 8.6 shows[1] the 3-pathsets and 3-cutsets employed in each case. The test results are shown in Table 8.5. As expected note that the estimated reliability grows when the network gets denser and when the link reliability is higher. Note that in two cases ($G_{15,5}$, $p = 0.80$ and $p = 0.95$) the variance estimated by both methods was zero since no failing configurations were sampled after the 2^{18} trials due to their rarity. In one case ($G_{15,4}$, $p = 0.95$) the same applied but only for SIM1. In the remaining cases the ratio RE could be computed and always favoured SIM5. Note that RE tends to grow

---

[1]In what follows we denote as (x, y) the edge that links nodes x and y and as $(x - y - \cdots - z)$ the set of edges of the path that traverses in order the nodes $x, y, \cdots, z$.

Table 8.5: Numerical results for test case IV

.	$\hat{R}$	$\widehat{\mathrm{Var}(\hat{R})}_{sim1}$	$\widehat{\mathrm{Var}(\hat{R})}_{sim5}$	t_{sim1}	t_{sim5}	RE
$p = 0.40$						
$G_{15,3}$	0.412745	9.25E-7	2.19E-7	7.41	7.85	3.99
$G_{15,4}$	0.817238	5.71E-7	1.39E-7	9.52	10.89	3.60
$G_{15,5}$	0.955233	1.65E-7	2.46E-8	11.53	13.29	5.81
$p = 0.60$						
$G_{15,3}$	0.832648	5.30E-7	7.22E-8	20.03	20.14	7.30
$G_{15,4}$	0.991864	3.03E-8	1.81E-9	22.70	28.21	13.47
$G_{15,5}$	0.999596	1.64E-9	1.54E-11	26.40	33.53	83.85
$p = 0.80$						
$G_{15,3}$	0.991225	3.35E-8	5.87E-10	20.17	21.75	52.92
$G_{15,4}$	0.999984	5.82E-11	5.31E-14	19.66	27.78	775.67
$G_{15,5}$	1.000000	0.00	0.00	22.29	32.29	n/a
$p = 0.95$						
$G_{15,3}$	0.999989	1.46E-11	1.84E-15	17.78	20.95	6734.15
$G_{15,4}$	1.000000	0.00	4.12E-24	15.23	24.96	n/a
$G_{15,5}$	1.000000	0.00	0.00	17.24	28.86	n/a

when the link reliabilities become higher as well as when the network gets dense, showing that SIM5 becomes more valuable as the network failures get rarer. It is interesting to analyze how the execution time depends on the number of edges and the link reliability. Denser networks always resulted in higher times; and the times for SIM1 were always lower than those for SIM5. Note that for a given method and network, the execution time was not a monotonous function of the link reliability but reached a maximal value for an "intermediate" reliability. This is related to the fact that evaluating Φ is faster with very low or very high reliabilities rather than with intermediate ones, because the BFS quickly aborts or reaches the destination in those cases.

8.5 Conclusions

In this chapter we have shown how Monte Carlo sampling plans conditioned by d-pathsets and d-cutsets can be applied to estimate the d-DCR. We briefly presented methods which make use of previously-known topological information and discussed the consequences of working in the DCR context. Then we illustrated them with four tests. Three of them are based on sparse networks (a case usually employed in the literature and a real network topology) while the fourth is based on three dense networks. As for the CLR in [94], these tests show that the methods based on precomputing pathsets and cutsets have the best relative efficiencies when compared to crude Monte Carlo. When the d-pathsets and d-cutsets involve a limited amount of edges, precomputing tables (SIM4) can be the best option (Test II), particularly if a large number of samples are to be generated due to

Table 8.6: 3-pathsets and 3-cutsets for test case IV

Test	3-pathsets	3-cutsets
$G_{15,3}$	$(0-3-6-8)$ $(0-2-5-8)$ $(0-12-9-8)$ $(0-13-10-8)$ $(0-14-11-8)$	$\{(0,2),(0,13),(0,14),(0,3),(0,12)\}$ $\{(8,6),(8,9),(8,10),(8,5),(8,11)\}$
$G_{15,4}$	$(0-4-8)$ $(0-12-8)$ $(0-11-8)$ $(0-13-9-8)$ $(0-14-10-8)$ $(0-3-7-8)$ $(0-2-6-8)$ $(0-1-5-8)$	$\{(0,1),(0,2),(0,3),(0,4),(0,14),(0,13),(0,12),(0,11)\}$ $\{(8,7),(8,6),(8,5),(8,4),(8,9),(8,10),(8,11),(8,12)\}$
$G_{15,5}$	$(0-3-8)$ $(0-4-8)$ $(0-5-8)$ $(0-1-6-8)$ $(0-2-7-8)$ $(0-13-8)$ $(0-12-8)$ $(0-11-8)$ $(0-10-8)$ $(0-14-9-8)$	$\{(0,1),(0,2),(0,3),(0,4),(0,5),(0,14),(0,13),(0,12),(0,11),(0,10)\}$ $\{(8,7),(8,6),(8,5),(8,4),(8,3),(8,9),(8,10),(8,11),(8,12),(8,13)\}$

high significance goals for the estimators. Otherwise, disjoint d-pathsets and d-cutsets (SIM5) are recommended due to lower sampling times, although it is often the case that allowing a small number of overlapping edges significantly tightens the bounds (leading to variance reductions) without adding major complexity to the sampling computations; further research has to be conducted regarding this trade-off. We have also discussed and illustrated the important fact that two features present in most real cases (large sets K and diameters d) yield higher relative efficiencies for DCR than for CLR (Tests II and III) which makes the research on high variance reduction methods for DCR even more appealing than for CLR. This work shall be pursued by adapting other Monte Carlo variance-reduction methods (see the references in Section 7.1) from the CLR to the DCR context and comparing their performance on a common set of test cases.

8.6 Appendix: Sequential sampling for pathset/cutset based bounding

Let P, C and Ω be defined as in Section 8.3.3. Let the edges of Ω be arbitrarily ordered with $h = |\Omega|$, $(z_1, ..., z_h) \in \{0, 1\}^h$ an h-tuple encoding the sampled states for the edges of Ω and $\rho_1, ..., \rho_h$ their (unconditioned) probabilities of operation. Suppose that the sampling procedure has achieved a certain point where $s < h$ edges from Ω have been sampled. Then, in order to sample a value for z_{s+1} (corresponding to an edge that we will denote e) we need to compute the probability p that the $(s + 1)$-th edge of Ω is operational provided that the configuration under construction belongs to Γ, and the state of the first s edges has already been sampled. We denote as E_Γ the event where the configuration belongs to Γ and as E_s the event where the state of the first s edges received certain specific sampled values. In the context of this sampling method, belonging to Γ means that the event $E_{ncf} =$"no cutset of C fails" must be met while at the same time the event $E_{omp} =$"one or more paths of P are operational" must not. It is easy to see that $E_{omp} \to E_{ncf}$. For convenience we will denote the probability of event A conditioned to event B both as $\Pr(A|B)$ and $\Pr|_B(A)$. Using this notation,

$$p = \Pr(z_{s+1} = 1 \mid E_\Gamma \wedge E_s) = \frac{\Pr(z_{s+1} = 1 \wedge E_\Gamma \wedge E_s)}{\Pr(E_\Gamma \wedge E_s)}$$

$$= \frac{\Pr|_{E_s}(z_{s+1} = 1 \wedge E_\Gamma) \Pr(E_s)}{Pr|_{E_s}(E_\Gamma) \Pr(E_s)}$$

$$= \Pr|_{E_s}(z_{s+1} = 1) \frac{\Pr|_{E_s}(E_\Gamma \mid z_{s+1} = 1)}{\Pr|_{E_s}(E_\Gamma)}$$

Since the failures are independent we have that $\Pr|_{E_s}(z_{s+1} = 1) = \Pr(z_{s+1} = 1) = \rho_{s+1}$; and since $E_{omp} \to E_{ncf}$ then

$$p = \rho_{s+1} \frac{\Pr|_{E_s}(E_{ncf} \mid z_{s+1} = 1) - \Pr|_{E_s}(E_{omp} \mid z_{s+1} = 1)}{\Pr|_{E_s}(E_{ncf}) - \Pr|_{E_s}(E_{omp})} \tag{8.1}$$

$$= \rho_{s+1} \frac{\Pr(E_{ncf} \mid E_s \wedge z_{s+1} = 1) - \Pr(E_{omp} \mid E_s \wedge z_{s+1} = 1)}{\Pr(E_{ncf} \wedge E_s) - \Pr(E_{omp} \wedge E_s)} \tag{8.2}$$

Let $\omega(\rho_1, ..., \rho_n)$ and $\lambda(\rho_1, ..., \rho_n)$ be the polynomials that respectively compute $p(E_{ncf})$ and $p(E_{omp})$. Then Eq. 8.1 translates into

$$p = \rho_{r+1} \frac{\omega(z_1, ..., z_s, 1, \rho_{s+2}, ..., \rho_h) - \lambda(z_1, ..., z_s, 1, \rho_{s+2}, ..., \rho_h)}{\omega(z_1, ..., z_s, \rho_{s+1}, ..., \rho_h) - \lambda(z_1, ..., z_s, \rho_{s+1}, ..., \rho_h)} \tag{8.3}$$

Finding the polynomials ω and λ can be a quite complex task if the pathsets and cutsets are numerous and highly overlap. Only if they are pairwise disjoint they can be easily computed as

$$\omega(\rho_1, ..., \rho_h) = \prod_{j=1}^{J} \left(1 - \prod_{i \in C_j} \rho_{\Omega(i)} \right) \tag{8.4}$$

$$\lambda(\rho_1, ..., \rho_h) = 1 - \prod_{j=1}^{I} \left(1 - \prod_{i \in P_j} \rho_{\Omega(i)} \right) \tag{8.5}$$

where $\Omega(i)$ denotes the index of edge i in the ordering used for the edges of Ω. In this case the probability for sampling each edge of Ω can be computed in $O(|\Omega|)$ time as next section shows.

8.6.1 Sequential sampling for SIM5

Let $\lambda_j = \prod_{i \in P_j} r_i$ for $j = 1, ..., I$ be the probability that pathset P_j is operational and $\omega_j = \prod_{i \in C_j} q_i$ for $j = 1, ..., J$ the probability that cutset C_j fails (with $q_i = 1 - r_i$). Then let us denote $\lambda = 1 - \prod_{j=1}^{I}(1 - \lambda_j)$ the probability of the event E_{omp} and $\omega = \prod_{j=1}^{J}(1 - \omega_j)$ the probability of the event E_{ncf}. For convenience let us also define $\lambda_0 = \omega_0 = 0$ and the following pointers j_i (pathset to which edge $\Omega(i)$ belongs) and k_i (cutset to which edge $\Omega(i)$ belongs):

$$j_i = \begin{cases} j & \text{if } e_{\Omega(i)} \in P_j \quad j = 1...I \\ 0 & \text{if } e_{\Omega(i)} \notin \bigcup_{k=1...I} P_k, \end{cases}$$

$$k_i = \begin{cases} j & \text{if } e_{\Omega(i)} \in C_j \quad j = 1...J \\ 0 & \text{if } e_{\Omega(i)} \notin \bigcup_{k=1...J} C_k. \end{cases}$$

From Eq. 8.1 the failure probability to sample the first edge of Ω will be

$$q = \frac{\Pr(z_1 = 0 \wedge E_{ncf}) - \Pr(z_1 = 0 \wedge E_{omp})}{\omega - \lambda} \tag{8.6}$$

The event $z_1 = 0 \wedge E_{ncf}$ will be met whenever the edge $e_{\Omega(1)}$ fails and all cutsets C (among which one could eventually include this edge) have at least one operating edge, so

$$\Pr(z_1 = 0 \wedge E_{ncf}) = q_{\Omega(1)} \prod_{j=1, j \neq k_1}^{J} (1 - \omega_j)(1 - \prod_{i \in C_{k_1} \setminus \{h\}} q_{\Omega(1)})$$

and thanks to the definition of ω_0 this yields the compact form

$$\Pr(z_1 = 0 \wedge E_{ncf}) = q_{\Omega(1)} \frac{\omega}{1 - \omega_{k_1}} \left(1 - \frac{\omega_{k_1}}{q_{\Omega(1)}}\right) \tag{8.7}$$

At the same time, the event $z_1 = 0 \wedge E_{omp}$ will be met whenever the edge $e_{\Omega(1)}$ fails and not every pathset in P fails (note that an eventual pathset including this edge will be failed), so

$$\Pr(z_1 = 0 \wedge E_{omp}) = q_{\Omega(1)} \left(1 - \prod_{j=1, j \neq j_1}^{I} (1 - \lambda_j)\right)$$

that, thanks to the convenient definition of λ_0 yields

$$\Pr(z_1 = 0 \wedge E_{omp}) = q_{\Omega(1)} \frac{\lambda - \lambda_{j_1}}{1 - \lambda_{j_1}} \tag{8.8}$$

Once the probability of failure for edge $e_{\Omega(1)}$ has been computed through Eq. 8.7,8.8 and 8.6, z_1 is sampled and its resulting state must be taken into account when computing the probability of failure for sampling the remaining edges of Ω. First of all, the probability ω_{k_1} that the cutset that includes $e_{\Omega(1)}$ fails (provided it exists) will be zero if the edge operates; but will exclude the factor $(1 - p_{\Omega(1)})$ from its definition if it fails. Similarly, the probability λ_{j_1} that the pathset that includes that edge operates (provided it exists) will be zero if it fails; but will exclude the factor $p_{\Omega(1)}$ from its definition if it operates, so before sampling next edges in Ω both must be adjusted in the following way (apostrophes stand for new values to take before addressing the next edge):

$$\omega'_{k_1} = \frac{(1 - z_1)\omega_{k_1}}{q_h} \qquad \lambda'_{j_1} = \frac{\lambda_{j_1} z_1}{p_{\Omega(1)}}$$

Finally ω and λ must also be updated to reflect the changes in ω_{k_1} and λ_{j_1}:

$$\omega' = \frac{\omega}{1 - \omega_{k_1}}(1 - \omega'_{k_1}) \qquad \lambda' = 1 - \frac{1 - \lambda}{1 - \lambda_{j_1}}(1 - \lambda'_{j_1})$$

By repeating the preceding steps for $z_2, ..., z_h$ the remaining edges of Ω get sampled. This is summarized in the pseudocode of Figure 8.6.

Procedure SampleOmega$(E, \Omega, \vec{r}, I, J, \lambda, \omega, \{\lambda_i\}, \{\omega_i\}, \{j_i, k_i\})$
Returns: Set of edges in Ω sampled as operating

1: Sample the state of each edge e in $E \setminus \Omega$ as Binomial(r_e)
2: $S \leftarrow \emptyset$
3: **for all** $h = 1, ..., |\Omega|$ **do**
4:      $\alpha \leftarrow \frac{\omega}{1-\omega_{k_h}}\left(1 - \frac{\omega_{k_h}}{q_{\Omega(h)}}\right)$
5:      $\beta \leftarrow \frac{\lambda - \lambda_{j_h}}{1 - \lambda_{j_h}}$
6:      $q \leftarrow q_{\Omega(h)} \frac{\alpha - \beta}{\omega - \lambda}$
7:      Sample z_h as Binomial$(1-q)$
8:      **if** $z_h = 1$ **then**
9:          $S \leftarrow S \cup \{e_{\Omega(h)}\}$
10:      **end if**
11:      $\omega'_{k_h} \leftarrow \frac{(1-x_h)\omega_{k_h}}{q_{\Omega(h)}}$
12:      $\lambda'_{j_h} \leftarrow \frac{\lambda_{j_h} x_h}{p_{\Omega(h)}}$
13:      $\omega' \leftarrow \frac{\omega}{1-\omega_{k_h}}(1 - \omega'_{k_h})$
14:      $\lambda' \leftarrow 1 - \frac{1-\lambda}{1-\lambda_{j_h}}(1 - \lambda'_{j_h})$
15:      $\omega_{k_h} \leftarrow \omega'_{k_h}; \lambda_{j_h} \leftarrow \lambda'_{j_h}; \omega \leftarrow \omega'; \lambda \leftarrow \lambda'$
16: **end for**
17: **return** S

Figure 8.6: Pseudocode for sampling edges in SIM5

Chapter 9

Monte Carlo analysis of K-diameter-dependent performability metrics

9.1 Introduction

The DCR, as a generalization of the CLR, is conceived to model situations where limits exist in the acceptable distance between any pair of distinguished nodes. This allows e.g. to model limits in delay times or in the number of hops undergone by data packets in a communication network. This model keeps the binary nature, not only of each component's state (nodes and edges can operate or fail), but also the binary nature of the network state itself (the network operates or fails). Solving a DCR instance implies finding a *reliability* measure, that is, the probability that the network is in one of the two possible states - namely, the operational one.

In several contexts there is need to employ metrics that are defined over a larger number of network states. These are *performability* measures, a quantification of the performance of a network with regard to a certain specific goal. Think for example of quantifying the user perception of the performance of a certain network (e.g. scoring it between 0 and 10 in a survey). This score might be strongly correlated to the average of a discrete, more-than-binary metric that partitions the network configuration into several classes. For example, in a two-terminal context, the time needed to transfer packets from source to terminal can define the different classes, and the relative frequency with which each one occurs ultimately determines the overall user's experience. This performability measures are then a generalization of reliability measures, where the underlying network states are not necessarily binary. See [19, 102, 1, 20, 21].

There are several contexts where the classes can be defined in terms of the distance between terminals in the random network configuration. The mentioned two-terminal network is an example. Another example is that of voice-over-IP and video streaming applications, where the perceived quality is affected by latency (the time spent by packets traveling from the source to the destination) [103], which is in turn determined by the

number of links traversed by data packets. The quality deteriorates as hig hop-distance states become more probable in the network. In web applications that heavily rely on dynamic page technologies, e.g. Asynchronous JavaScript and XML (Ajax) [104], the quality perceived by the end user is influenced by responsiveness, where latency determines at a large extent the delay between the user actions and their effect on his/her browser. Quality deteriorates as high hop-distance states occur more frequently in the network. This accounts for the dominating model of web application deployment nowadays, where mirrors of the same application are placed as close as possible to the physical location of the users. In general packet switching networks, longer paths involve more overall RAM memory being used throughout the network, resulting in higher infrastructure costs. Consider also other possible contexts where costs are born each time a link is traversed e.g. vehicle or packet routing, where costs can relate to time spent, tolls, fuel, etc. In all these examples, performability measures can be conceived that are defined by the distance between terminals.

As we saw in Chapter 7, many Monte Carlo methods have been proposed for efficient estimation of the CLR and DCR. Now, in this chapter we introduce a Monte Carlo method to estimate a particular family of performability metrics, namely those defined as the expected value of a random variable that depends only on the K-diameter of the partial graph induced by the operational links. In real communication networks, link reliabilities are normally very high and thus, when applying simulation methods, sampling a network configuration with distant or disconnected pairs of terminals can be a rare event. We have seen that in the context of Monte Carlo simulations for network reliability analysis, once fixed a certain confidence interval goal, the needed sample size unboundedly grows as link reliabilities become higher, and thus it is convenient to apply variance-reduction techniques. The Monte Carlo method hereby proposed generalizes the methods developed for DCR in Chapter 7. Observe that the DCR metric is the expected value of a binary variable whose value is one if and only if the K-diameter is up to a certain parameter d. The metric defined in this chapter allows to further split the K-diameter into more than two tranches, assigning different values to the random variable according to each one; hence the generalization.

The edge sets (d-pathsets and d-cutsets) used for restricting the sampling space were considered *a priori* known in Chapter 7 and within the literature therein referenced, due to a number of reasons. On one hand, there is a clear split between the preliminary phase of determining the sets and the simulation itself, and the main efforts have been devoted to improving the latter for achieving sampling plans with good tradeoffs between the variance reduction and the sampling effort. On the other hand, the instances typically used for testing were of limited size, allowing for edge sets selection through simple visual inspection. In our case there are several reasons why to introduce heuristics for automated generation of the edge sets. First, the method requires in general a much higher number of edge sets than previous works, as shown in Section 9.4. Second, when working with distance-constraints, determining the maximum number of disjoint paths and cuts, and specific ones, are in general $\mathcal{N}P$-hard tasks; see [9]. Third, as shown in Test 2, the chosen sets can significantly affect the performance attained by the simulation

method. Consequently, we hereby introduce and compare a family of heuristics. We present numerical evidence of the significant efficiency gains attained by chaining these heuristics to the proposed simulation method when compared to a crude Monte Carlo simulation.

This chapter is organized as follows. Section 9.2 includes definitions, notation and model formalization. Section 9.3 describes the crude Monte Carlo method and the estimators it gives for the metric under study as well as for its variance. Section 9.4 describes the suggested Monte Carlo method and shows the variance reductions achieved relative to the crude one. Section 9.5 describes the heuristics for selecting the d-pathsets and d-cutsets that will be applied in the simulation. Section 9.6 presents three numerical examples, based on mesh-like networks. It compares several variations of the heuristic, the relative efficiency of the proposed Monte Carlo method vs. the crude one and how it is influenced by link reliabilities. Finally, conclusions and further work are summarized in Section 9.7.

The contents of this chapter were published in [15], accepted for publication in [18] and submitted after revision to [17].

9.2 Definitions and notation

We present here the definitions and notation needed for the rest of the chapter; most of it was already introduced in previous chapters, but we include it here too for ease of reference. As usual, the network is modeled by an undirected labeled probabilistic graph $G = (V, E, \vec{r})$ with $n = |V|$, $m = |E|$, $E = \{e_1 \ldots e_m\}$ and $\vec{r} = \{r_1 \ldots r_m\}$, being r_i the probability of operation of the link represented by e_i. The set of terminal nodes is a certain $K \subseteq V$. The following definitions and notation are also employed:

- X_i is a binary random variable that indicates the event "e_i operates". It has a Bernoulli distribution with parameter r_i. We denote $X = (X_1 \ldots X_m)$;

- $x = (x_1 \ldots x_m) \in \{0, 1\}^m$ is a *network configuration*. $\mathcal{X}$ is the set of the 2^m possible network configurations. Each one encodes a partial graph of G, denoted G_x, where edge e_i occurs if and only if $x_i = 1$;

- $\pi(x) = \Pr(X = x)$ (probability that the random network configuration is x);

- $d_G(s, t)$ is the distance between nodes s and t in the graph G. The set K is *d-connected* in G if and only if $\forall s, t \in K$, $d_G(s, t) \leq d$. The K-diameter of G is $\max\{d_G(s, t) : s, t \in K\}$;

- $\Delta : \mathcal{X} \to \{0, ..., n, \infty\}$ is the function defined as $\Delta(x) = \max\{d_{G_x}(s, t) : s, t \in K\}$.

Let $\Phi : \mathcal{X} \to \mathbb{R}$ be a *structure function*, interpreted as follows. The real number $\Phi(x)$ is a quantitative measure of the performance of the network, when its operational links are those that correspond to the edges of G_x. Our final goal is to estimate $\nu = \mathrm{E}(\Phi(X))$. We are concerned with structure functions whose value is determined by the maximal

9. MONTE CARLO ANALYSIS OF K-DIAMETER-DEPENDENT PERFORMABILITY METRICS

Procedure Phi$(x, K, I, \phi_0 \ldots \phi_r)$

1: Compute $\Delta(x)$ through breadth-first-search starting at every terminal
2: find $i : \Delta(x) \in I_i$
3: **return** ϕ_i

Figure 9.1: Pseudocode to compute $\Phi : \mathcal{X} \to \mathbb{R}$.

K-distance $\Delta(x)$. Figure 9.2 illustrates the model and notations used. Without losing generality, we can partition $\mathbb{Z}^*$ into several subintervals $I = \{I_0 \cup I_1 \cup \cdots \cup I_r\}$, having $r + 1$ components, such that $\Phi(x) = \Phi(y) \iff \Delta(x), \Delta(y) \in I_i$ for all $i = 0 \ldots r$. Each interval can be seen as a 'quality level' given by ranges of the K-diameter. This partition is defined by r integers $0 < d_0 < \cdots < d_{r-1}$, with $I_0 = (0, d_0]$, $I_i = (d_{i-1}, d_i]$ for $i \in [1, r-1]$ and $I_r = (d_{r-1}, \infty]$. Let us now partition $\mathcal{X}$ in $r + 1$ components (that we call *regions*) $\mathcal{X} = \mathcal{X}_0 \cup \cdots \cup \mathcal{X}_r$, where $\mathcal{X}_i = \{x \in \mathcal{X} \mid \Delta(x) \in I_i\}$ for $i = 0 \ldots r$. Let $p_i = \Pr(x \in \mathcal{X}_i)$ and $\phi_i = \Phi(x)$ for any $x \in \mathcal{X}_i$. Figure 9.1 shows how $\Phi(x)$ is computed once K, I and $\phi_0, \ldots, \phi_r$ are given.

$\mathcal{X}$	p_0	p_1		p_{r-1}	p_r
I (intervals for Δ)	$(0, d_0]$	$(d_0, d_1]$		$(d_{r-2}, d_{r-1}]$	$(d_{r-1}, \infty]$
regions	$\mathcal{X}_0$	$\mathcal{X}_1$		$\mathcal{X}_{r-1}$	$\mathcal{X}_r$
Φ	Φ_0	Φ_1		Φ_{r-1}	Φ_r
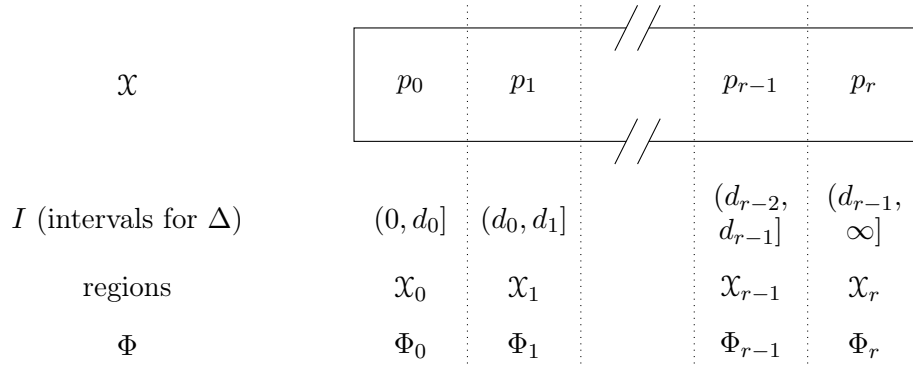
Figure 9.2: Partitions of the network configuration space

9.3 Crude Monte Carlo method

A crude Monte Carlo simulation estimates $\nu = \mathrm{E}(\Phi(X))$ by independently sampling N network configurations $x^{(1)}, \ldots, x^{(N)}$, computing Φ for each one, and building an estimator

$$\widehat{\Phi_N} = \frac{1}{N} \sum_{i=1}^{N} \Phi(x^{(i)})$$

whose variance is

$$\mathrm{Var}(\widehat{\Phi_N}) = \frac{N}{N^2}\mathrm{Var}(\Phi) = \frac{1}{N}(\mathrm{E}(\Phi^2) - \mathrm{E}^2(\Phi)) = \frac{1}{N}(\sum_{i=1}^{N} \phi_i^2 p_i - \nu^2).$$

Sampling each $x^{(i)}$ involves m Bernoulli trials for determining the state of each edge. To compute our performability measure, we first need to determine the K-diameter of $G_{x^{(i)}}$. This can be done by applying a breadth-first-search (BFS) algorithm starting at every node of K. Then $\Phi(x^{(i)}) = \phi_j$, being I_j the interval to which the K-diameter belongs. So each iteration takes a time that is $O(\max\{m, |K|^2\})$.

9.4 The proposed Monte Carlo method

Suppose that certain topological knowledge about $G = (V, E)$ is available in the form of certain edge sets. This are sets of d_i-*pathsets* and d_i-*cutsets* for given values of $i = 0 \ldots r-1$. Recall that, if P is any subset of E and $G' = (V, P)$ is the partial graph of G yielded by P, then

- P is a *d-pathset* if and only if K is d-connected in G';

- a d-pathset is said to *operate* when all of its edges operate.

Similarly, recall that if C is any subset of E and $G' = (V, E \setminus C)$ is the partial graph of G yielded by $E \setminus C$, then

- C is a *d-cutset* if and only if K is not d-connected in G';

- a d-cutset is said to *fail* when all of its edges fail.

Suppose that a certain network configuration x is such that a given 5-pathset operates while a given 2-cutset fails. It follows that the maximal distance between the nodes of K must be any of $\{3, 4, 5\}$ in G_x. If one of the intervals I_j is such that $\{3, 5\} \subseteq I_j$, then the region to which x belongs is known, as well as $\Phi(x) = \phi_j$. This example illustrates how Φ can be immediately known when certain combinations of events occur. This is the core idea behind the proposed method. As we see in Section 9.4.1 it allows for reducing the Monte Carlo sampling space compared to crude Monte Carlo. Let us now formalize this idea. First, we define some sets of d-pathsets and d-cutsets, assumed to be a priori available:

- $\mathcal{P}_0, \ldots, \mathcal{P}_{r-1}$: r sets such that each $\mathcal{P}_i$ is a set of some d_i-pathsets;

- $\mathcal{P}_r = \emptyset$ (for convenience of notation);

- $\mathcal{C}_0 = \emptyset$ (for convenience of notation);

- $\mathcal{C}_1, \ldots, \mathcal{C}_r$: r sets such that each $\mathcal{C}_i$ is a set of some d_{i-1}-cutsets.

9. MONTE CARLO ANALYSIS OF K-DIAMETER-DEPENDENT PERFORMABILITY METRICS

In the previous definitions, the word 'some' means that every set can contain zero or more elements. At least one of the sets must be non-empty, otherwise the method coincides with crude Monte Carlo. Now, we define the following random events in terms of the defined edge sets:

- $T_i = $ (some element of $\mathcal{P}_i$ operates) $\quad (i = 0 \ldots r - 1)$;

- $K_j = $ (some element of $\mathcal{C}_j$ fails) $\quad (j = 1 \ldots r)$;

- $Z_0 = T_0$;

- $Z_h = T_h \wedge K_h \quad (h = 1 \ldots r - 1)$;

- $Z_r = K_r$.

Observe that given a network configuration x, it holds that:

- $(Z_0 \to (\Delta(x) \in (0, d_0]))$;

- $(Z_i \to (\Delta(x) \in (d_{i-1}, d_i]))$ for $i \in [1 \ldots r - 1]$;

- $(Z_r \to (\Delta(x) \in (d_{r-1}, \infty)))$.

In other words, each event $Z_{h=0\ldots r}$ determines a precise value ϕ_h. Let us denote as $\mathcal{Z}_i$ the set of configurations for which Z_i holds $(i = 0 \ldots r)$. Then each $\mathcal{Z}_i \subseteq \mathcal{X}_i$. Let also $\mathcal{Z} = \cup_{i=0}^r \mathcal{Z}_i$, $z = \Pr(Z) = \Pr(x \in \mathcal{Z})$ and $z_i = \Pr(Z_i) = \Pr(x \in \mathcal{Z}_i)$ for $i = 0 \ldots r$. We can write

$$\nu = \left(\sum_{i=0}^r \sum_{x \in \mathcal{Z}_i} \Pr(x) \Phi(x) \right) + \sum_{x \in \mathcal{X} \backslash \mathcal{Z}} \Pr(x) \Phi(x). \tag{9.1}$$

Calling ν_Z the first term, we have that $\nu_Z = \sum_{i=0}^r \phi_i z_i$. This is immediate to compute provided that we know the values of $z_0, \ldots, z_r$; in Section 9.4.2 we comment on the complexity of computing the latter. Our method estimates the second term of Eq. (9.1) by sampling N network configurations $x^{(1)}, \ldots, x^{(N)}$ within $\mathcal{X} \backslash \mathcal{Z}$ (with a probability distribution that respects the relative probabilities of the network configurations in $\mathcal{X} \backslash \mathcal{Z}$), computing Φ for each one, and building an unbiased estimator $\widetilde{\Phi_N}$ as follows:

$$\widetilde{\Phi_N} = \nu_Z + \frac{1 - z}{N} \left(\sum_{i=1}^N \Phi(x^{(i)}) \right)$$

where the correction factor $1 - z$ is due to the fact that $x^{(1)}, \ldots, x^{(N)}$ are conditioned to $\mathfrak{X} \backslash \mathcal{Z}$. The variance is then

$$
\begin{aligned}
\mathrm{Var}(\widetilde{\Phi_N}) &= \frac{N(1-z)^2}{N^2} \mathrm{Var}\,(\Phi) = \frac{(1-z)^2}{N} \left(\mathrm{E}(\Phi^2) - \mathrm{E}^2(\Phi) \right) \\
&= \frac{(1-z)^2}{N} \left(\sum_{i=0}^{r} \frac{\phi_i^2(p_i - z_i)}{1 - z} - \left(\sum_{i=0}^{r} \frac{\phi_i(p_i - z_i)}{1 - z} \right)^2 \right) \\
&= \frac{1}{N} \left((1 - z) \sum_{i=0}^{r} \Phi_i^2(p_i - z_i) - (\nu - \nu_Z)^2 \right).
\end{aligned}
$$

9.4.1 Variance reduction

The variances obtained through the crude and the proposed Monte Carlo methods are next compared. For simplicity it is done for $N = 1$.

$$
\begin{aligned}
\mathrm{Var}(\widehat{\Phi_1}) - \mathrm{Var}(\widetilde{\Phi_1}) &= \sum_{i=0}^{r} \phi_i^2 p_i + (1 - z) \sum_{i=0}^{r} \phi_i^2(p_i - z_i) - 2\nu\phi + \phi^2 \\
&= z \sum_{i=0}^{r} \phi_i^2 p_i - z \sum_{i=0}^{r} \phi_i^2 z_i + \sum_{i=0}^{r} \phi_i^2 z_i - 2(\sum_{i=0}^{r} \phi_i p_i)(\sum_{i=0}^{r} \phi_i z_i) + (\sum_{i=0}^{r} \phi_i z_i)^2 \\
&= \sum_{i=0}^{r}\sum_{j=0}^{r} \phi_i^2 p_i z_j - \sum_{i=0}^{r}\sum_{j=0}^{r} \phi_i^2 z_i z_j + \sum_{i=0}^{r}\sum_{j=0}^{r} \phi_j^2 p_i z_j - 2\sum_{i=0}^{r}\sum_{j=0}^{r} \phi_i\phi_j p_i z_j + \sum_{i=0}^{r}\sum_{j=0}^{r} \phi_i\phi_j z_i z_j \\
&= \sum_{i=0}^{r}\sum_{j=0}^{r} (\phi_i - \phi_j)^2 p_i z_j + \sum_{i=0}^{r}\sum_{j=0}^{r} (\phi_i\phi_j - \phi_i^2) z_i z_j \\
&= \sum_{i=0}^{r}\sum_{j=0}^{r} (\phi_i - \phi_j)^2 (p_i - z_i) z_j + \sum_{i=0}^{r}\sum_{j=0}^{r} \left((\phi_i - \phi_j)^2 + \phi_i\phi_j - \phi_i^2 \right) z_i z_j \\
&= \sum_{i=0}^{r}\sum_{j=0}^{r} (\phi_i - \phi_j)^2 (p_i - z_i) z_j + \sum_{i=0}^{r}\sum_{j=0}^{r} (\phi_j(\phi_j - \phi_i)) z_i z_j \\
&= \sum_{i=0}^{r}\sum_{j=0}^{r} (\phi_i - \phi_j)^2 (p_i - z_i) z_j + \sum_{i=0}^{r-1}\sum_{j=i+1}^{r} (\phi_j - \phi_i)^2 z_i z_j.
\end{aligned}
$$

The following Lemma characterizes the variance reduction in terms of the regions $\mathfrak{X}_i$ and their respective subsets $\mathcal{Z}_i$.

Lemma 9.1. *It holds that* $\mathrm{Var}(\widetilde{\Phi_1}) \leq \mathrm{Var}(\widehat{\Phi_1})$ *and the inequality is strict if and only if there is a pair* (i, j) *such that* $\phi_i \neq \phi_j$, $p_i > 0$ *and* $z_j > 0$.

Proof. Observe that both summations in the final expression involve just non-negative terms, hence the difference of variances is always non-negative. Regarding strict inequality, assume that two regions $\mathfrak{X}_i$ and $\mathfrak{X}_j$ exist such that $\phi_i \neq \phi_j$, $p_i > 0$ and $z_j > 0$

(therefore $p_j > 0$). Then, if $p_i > z_i$, the first summation will have a strictly positive term given by the subindices i, j. If $p_i = z_i$, then $z_i > 0$ and therefore the second summation will have a strictly positive term given by the subindices i, j. This proves the strict inequality. Conversely, assume that $\text{Var}(\widetilde{\Phi}_1) < \text{Var}(\widehat{\Phi}_1)$. Then there must exist i, j such that at least one of the corresponding terms in the first and second summation is strictly positive. If the term for the first summation is strictly positive, then $\phi_i \neq \phi_j$, $p_i > z_i$ and $z_j > 0$. Since $(p_i > z_i \rightarrow p_i > 0)$ the existence of the pair (i, j) is proved. If the term for the second summation is strictly positive, then $\phi_i \neq \phi_j$, $z_i > 0$ and $z_j > 0$. Again, $p_i > 0$ and the existence of the pair (i, j) is proved. $\qquad \square$

9.4.2 Sampling within $\mathcal{X} \setminus \mathcal{Z}$

We can sample network configurations within $\mathcal{X} \setminus \mathcal{Z}$ with procedures analogous to those explained in Section 8.3.3. Let $\Omega = (\bigcup_{i=1}^r \bigcup_{p \in \mathcal{P}_i} p) \cup (\bigcup_{j=1}^r \bigcup_{c \in \mathcal{C}_j} c)$ be the set of all edges occurring in at least one d-pathset or d-cutset that build $\mathcal{Z}$. Note that the event Z is independent from the state of all edges that do not belong to Ω. Then the state of the edges of $E \setminus \Omega$ can be easily sampled with $m - |\Omega|$ Bernoulli trials. A general sequential procedure to sample the state of the edges of Ω, conditioning to be in $\mathcal{X} \setminus \mathcal{Z}$, is the following. Let $(o_1 \ldots o_{|\Omega|}) \in \{0, 1\}^{|\Omega|}$ an $|\Omega|$-tuple encoding the sampled states of Ω, for a certain ordering of its edges. Assume that $o_1 \ldots o_s$ have already been sampled (being $s < |\Omega|$) and let E_s be the event 'the first s edges of Ω have states $o_1 \ldots o_s$ respectively'. Then the probability that the $(s+1)$-th edge of Ω is operational conditioned to $\mathcal{X} \setminus \mathcal{Z}$ is:

$$\Pr(o_{s+1} = 1 \mid \bar{Z} \wedge E_s) = \Pr(o_{s+1} = 1) \frac{\Pr(\bar{Z} \mid E_s \wedge (o_{s+1} = 1))}{\Pr(\bar{Z} \mid E_s)}$$

where $\Pr(o_{s+1} = 1)$ is the reliability of the $(s+1)$-th edge of Ω. The function that given the reliabilities $\rho_1, \ldots, \rho_{|\Omega|}$ of the edges of Ω returns $\Pr(\bar{Z})$ is a polynomial $\mathbb{P}(\rho_1, \ldots, \rho_{|\Omega|})$. So, computing $\Pr(\bar{Z} \mid E_s)$ involves replacing $\rho_1, \ldots, \rho_s$ by 0 or 1 according to the states sampled for the first s edges and then evaluating $\mathbb{P}$. Observe that $\Pr(\bar{Z}) = 1 - \Pr(Z) = 1 - \Pr(\bigwedge_{i=0}^r Z_i) = 1 - \sum_{i=0}^r \Pr(T_i \wedge K_i)$ (recall the pairwise independence of all Z_i). To compute each $\Pr(T_i \wedge K_i)$, if every edge involved occurs just in one of $\mathcal{P}_i$ and $\mathcal{C}_i$, then it is possible to get simple factorized expressions for $\mathbb{P}$. This is due to the fact that each $\mathcal{P}_i$ and $\mathcal{C}_i$ operates/fails independently. In such case, building and evaluating $\mathbb{P}$ can be done in time $O(|\Omega|)$. This results in an efficient way to compute $z_0, \ldots, z_r$ (as shown in lines 3-8 of Figure 9.3) as well as to perform the sequential sampling. When there is high overlapping between the edges of the $\mathcal{P}_i$ and $\mathcal{C}_i$, building and evaluating $\mathbb{P}$ can be very complex, since the independence is lost.

Again, as seen in Section 8.3.3, for limited cardinalities of Ω an alternative approach can be used, consisting in precomputing the probability of the occurrence of the $2^{|\Omega|}$ possible sub-configurations that the edges of Ω induce, in $O(2^{|\Omega|})$ time. Then, sampling their states just involves choosing a sub-configuration at random, through a cut-point access on a table that accumulates the precomputed probabilities (in $O(|\Omega|)$ time). This

is the fastest way to sample the states for the edges of Ω and compute $z_0, \ldots, z_r$, but at the expense of an exponential-in-$|\Omega|$ effort for precomputing the table, that can limit its applicability on large networks. Details and algorithms to implement these sampling methods can be found in the Appendix of [14].

9.4.3 The simulation algorithm

Figure 9.3 presents the pseudocode of the simulation algorithm we used in the tests section; it uses the cut-point table method for sampling the edges of Ω. The procedure receives the graph G, terminals K, intervals I, structure function values for each interval $\phi_0 \ldots \phi_r$, the r sets of d-pathsets and d-cutsets, and the sample size N. Lines 1-2 build a table T that accumulates the probability that every subset of edges of Ω are operational. This requires that a certain ordering is assumed in the powerset $\mathcal{P}(\Omega)$. Lines 3 to 9 compute z_i for $i = 0 \ldots r$ through factorized probabilites, z and ν_Z. Lines 11-18 execute the main loop N times. The state of the edges of Ω is sampled by uniformly drawing a random real in $[0, 1]$ and accessing through it the table T. The state of each remaining edge e is sampled by independent Bernoulli's trials whit parameter r_e. The procedure Phi is used to accumulate ν and σ. Finally, lines 19-21 complete the computation and return the estimates for $\widetilde{\Phi_N}$ and $\text{Var}(\widetilde{\Phi_N})$.

9.5 A heuristic for d-pathset and d-cutset generation

In this section we introduce a heuristic that generates d_i-pathsets and d_{i-1}-cutsets for every region $\mathcal{X}_i$. We develop an algorithm for the two-terminal problem that can be easily generalized to the problem for general sets K. For every i, we build a set of d_i-pathsets and d_{i-1}-cutsets, such that no two elements share edges. This set should ideally be the one that maximizes the probability z_i that one d_i-pathset operates and one d_{i-1}-cutset fails at the same time, to attain the largest variance reduction that is possible. The basic idea consists of a greedy randomized generation of paths with lengths in $(d_{i-1}, d_i]$, followed by a greedy randomized generation of d_{i-1}-cutsets, without using any edge included in the former paths. The cycle is repeated several times and the combination of sets that yield the higher probability is finally chosen.

9.5.1 Paths generation

The algorithm shown in Fig. 9.4 receives the graph G, source and destination nodes (s, t) and the minimum and maximum distances ℓ_1, ℓ_2 that define the region. It returns a random path whose length is in the interval $[\ell_1, \ell_2]$ or the null ($\perp$) element if no such path could be found. The algorithm proceeds with a greedy selection of nodes (`currNode`) that are added to the path under construction `newPath`. It starts by selecting s and ends after reaching t or achieving a point where it is impossible to complete a path with the required constraints. In each iteration, the shortest path (`shP`) between the current node and t is computed. If there is no such path, or its length exceeds the difference between ℓ_2 and the length of the already built part, the algorithm returns the null element $\perp$.

Procedure Simulate$(G, K, I, \phi_0 \ldots \phi_r, \mathcal{P}_0 \ldots \mathcal{P}_r, \mathcal{C}_0 \ldots \mathcal{C}_r, N)$

1: $\Omega \leftarrow (\bigcup_{i=1}^r \bigcup_{p \in \mathcal{P}_i} p) \cup (\bigcup_{j=1}^r \bigcup_{c \in \mathcal{C}_j} c)$
2: compute a cut-point table T mapping $\mathcal{P}(\Omega) \rightarrow [0,1]$
3: $z_0 \leftarrow 1 - \prod_{p \in \mathcal{P}_0}(1 - \prod_{e \in p} r_e)$
4: **for all** $i = 1 \ldots r-1$ **do**
5: $\quad z_i \leftarrow (1 - \prod_{p \in \mathcal{P}_i}(1 - \prod_{e \in p} r_e))(1 - \prod_{c \in \mathcal{C}_i}(1 - \prod_{e \in c}(1 - r_e)))$
6: **end for**
7: $z_r \leftarrow 1 - \prod_{c \in \mathcal{C}_r}(1 - \prod_{e \in c}(1 - r_e))$
8: $z \leftarrow \sum_{i=0}^r z_i$
9: $\nu_Z \leftarrow \sum_{i=0}^r z_i \phi_i$
10: $\nu \leftarrow 0; \sigma \leftarrow 0$
11: **for all** sample $= 1 \ldots N$ **do**
12: $\quad$ rnd $\leftarrow$ pick a random real uniformly in $[0,1]$
13: $\quad w \leftarrow$ the Ω configuration $w = \max\{w : T(w) \leq \text{rnd}\}$
14: $\quad y \leftarrow$ sample states of $E \setminus \Omega$ with $|E| - \Omega$ Bernoulli's trials
15: $\quad x \leftarrow$ configuration built from w and y
16: $\quad \nu \leftarrow \nu + \text{Phi}(x, K, I, \phi_0 \ldots \phi_r)$
17: $\quad \sigma \leftarrow \sigma + \text{Phi}(x, K, I, \phi_0 \ldots \phi_r)^2$
18: **end for**
19: $\nu \leftarrow \nu_Z + \nu(1-z)/N$
20: $\sigma \leftarrow \sigma(1-z)/N^2 - (\nu - \nu_Z)^2/N$
21: **return** ν, σ

Figure 9.3: Simulation procedure using disjoint edge sets and cut-point table.

Procedure generatePath$(G, s, t, \ell_1, \ell_2)$

1: $\ell \leftarrow 0$; currNode $\leftarrow s$; newPath $\leftarrow \emptyset$
2: **while** currNode $\neq t$ **do**
3: shP $\leftarrow$ shortestPath$(G, \text{currNode}, t, \text{newPath})$
4: dist $\leftarrow$ length(shP)
5: **if** $(\text{shP} = \perp) \vee (\ell + \text{dist} > \ell_2)$ **then**
6:   **return** $\perp$
7: **else**
8:   **if** (currNode has only one neighbor in G) $\vee$ (rand() $< (\ell + \text{dist} - \ell_1)/(\ell_2 - \ell_1)$)
     then
9:    next $\leftarrow$ neighbor of currNode in shP
10: **else**
11:    next $\leftarrow$ any neighbor of currNode in $G \setminus \text{newPath}$
12: **end if**
13: remove the edge (currNode,next) from G
14: append the edge (currNode,next) to newPath
15: currNode $\leftarrow$ next
16: $\ell \leftarrow \ell + 1$
17: **end if**
18: **end while**
19: **return** newPath

Figure 9.4: Heuristic algorithm for generating paths with lengths in $[\ell_1, \ell_2]$.

Otherwise, the next edge can be either the first one of `shP`, or another one different from it. When there are more than one feasible edges, the first one of `shP` is chosen with a probability proportional to the ratio between the length of `shP` and the maximum number of edges that can be added to the path under construction without violating the $[\ell_1, \ell_2]$ constraint. The function `rand()` in the algorithm returns a random uniform real number in [0,1). Therefore the algorithm tends to stick to a shortest path when `newPath` has reached a length that leaves few chances to divert. On the contrary, when there is still room for many more edges than the length of a shortest path, the algorithm will tend to choose, at random, other directions to extend `newPath`. In line 11, care is taken to not repeat a node of `shP`, which would result in having cycles within the `newPath`. Lines 13-16 remove the chosen edge from G for future iterations, append it to `newPath` and update its length ℓ and `currNode`. Randomness is therefore introduced in the decision of wheter to make a step in the direction of a shortest path or not, as well as in the selection of the next node, in case it was decided not to follow the shortest path.

9.5.2 d-Cutsets generation

The algorithm for creating an ℓ-cutset given a certain integer ℓ is shown in Fig. 9.5. It receives the graph G, the source and terminal nodes (s, t), the integer ℓ and a set of edges H. It starts by building a first-in-first-out queue $\vec{d}$ with all edges of $G \setminus H$ inserted in increasing order of distance to s. Randomness is introduced by shaking these distances prior to insertion. For example, in our tests we swapped each pair of values in $\vec{d}$ with a probability inversely proportional to their difference. The queue is later used to add each edge to the cutset under construction `newCut` in this shaken order; the idea behind this is that dropping edges in the vecinity of s is a good strategy to find low cardinality ℓ-cutsets for s, t. The set H will be used when invoking this algorithm to avoid using edges already used for other ℓ-pathsets or ℓ-cutsets found prior to the one under construction. The `while` loop proceeds adding edges to `newCut` and dropping them from G until one of the following occurs: i) the distance between s and t is greater than ℓ (so we have an ℓ-cutset); or ii) the queue is empty (so no ℓ-cutset exists, which happens if and only if there was a path whose length is not above ℓ built exclusively with edges of $G \setminus H$). After the `while` loop, `newCut` is an ℓ-cutset not necessarily minimal (i.e. some edges can be dropped from it and still be an ℓ-cutset). The `for` loop builds an ℓ-cutset `minCut` that is minimal in this sense (although, in general, it will not be a minimum-cardinality ℓ-cutset).

9.5.3 The main heuristic

Our main algorithm iterates until a given amount of time is spent. In each iteration a disjoint edge set for a certain region $\mathcal{X}_i$ is generated. Each iteration begins with the generation of one d_i-pathset and one d_{i-1}-cutset. It then continues adding more sets, following a certain sequence that defines a "version" of the algorithm. Each iteration may not exceed a certain parameter time MAX_TIME; if it does then it is discarded and a new iteration is run. For each generated sets P and C of d_i-pathsets and d_{i-1}-cutsets, the probability of the event z_i that they define is computed; the P,C pair with the highest $\Pr(z_i)$ is recorded as the algorithm proceeds and returned after timing out. The algorithm shown in Fig. 9.6 corresponds to the version where the sequence pathset-cutset-pathset-cutset is followed in each iteration; we will denote it as PCPC. It illustrates how to invoke the procedures `generatePath` and `generateCutset`. In Section 9.6.2 we compare the results obtained with seven different versions. The algorithm receives the graph G, source and destination nodes (s, t) and the range of distances allowed for the zone $[\ell_1, \ell_2]$. It returns the pair (P, C) whose probability was the highest among all the pairs generated. The pair is not necessarily built by two d-pathsets and two d-cutsets. Note the way that `generatePath` and `generateCutset` are invoked in lines 18 and 26. Passing $G \setminus P$ and $P \cup C$ allows respectively to obtain disjoint d-pathsets and d-cutsets respect to those so far generated in the current iteration. If a subprocedure is invoked MAX_TIMES times without succeeding to return a d-pathset or d-cutset, a new iteration is started. Similar algorithms PP, PPP, ... and CC, CCC, ... are used for the border regions $\mathcal{X}_0$ and $\mathcal{X}_r$, generating only d-pathsets or only d-cutsets.

Procedure generateCutset(G, s, t, ℓ, H)

1: $\vec{d} \leftarrow$ vector of distances between s and every node in $G \setminus H$
2: randomAlter$(\vec{d})$
3: queue $\leftarrow$ all edges of $G \backslash H$ inserted in increasing order of their values in $\bar{d}$
4: newCut $\leftarrow \emptyset$; flag $\leftarrow$ true
5: **while** flag **do**
6:     shP $\leftarrow$ shortestPath$(G, s, t, \emptyset)$
7:     **if** (shP $= \perp$) $\vee$ (length(shP) $> \ell$) **then**
8:        flag $\leftarrow$ false
9:     **else if** isEmpty(queue) **then**
10:        **return** $\perp$
11:     **else**
12:        newEdge $\leftarrow$ pop(queue)
13:        remove newEdge from G
14:        add newEdge to newCut
15:     **end if**
16: **end while**
17: minCut $\leftarrow \emptyset$
18: **for all** $e \in$ newCut **do**
19:    add e to G
20:    shP $\leftarrow$ shortestPath$(G, s, t, \emptyset)$
21:    **if** (shP $\neq \perp$) $\wedge$ (length(shP) $\leq \ell$) **then**
22:       remove e from G
23:       add e to minCut
24:    **end if**
25: **end for**
26: **return** minCut

Figure 9.5: Heuristic algorithm for generating an ℓ-cutset.

Procedure PCPC$(G, s, t, \ell_1, \ell_2)$

1: bestP $\leftarrow \perp$; bestC $\leftarrow \perp$; highestProb $\leftarrow 0$
2: **while** ellapsed_time $<$ MAX_TIME **do**
3:     P $\leftarrow \emptyset$; C $\leftarrow \emptyset$
4:     **repeat**
5:       p $\leftarrow$ generatePath(G, s, t, l_1, l_2)
6:     **until** (p $\neq \perp$) or MAX_TRIES attempts were done
7:     **if** (p $= \perp$) **then** continue //*aborts current "while" iteration*
8:     P $\leftarrow$ P $\cup$ {p}
9:     **repeat**
10:       c $\leftarrow$ generateCutset$(G, s, t, l_1 - 1, \text{P})$
11:     **until** (c $\neq \perp$) or MAX_TRIES attempts were done
12:     **if** (c $= \perp$) **then** continue //*aborts current "while" iteration*
13:     C $\leftarrow$ C $\cup$ {c}
14:     **if** Pr(P,C) $>$ highestProb **then**
15:       highestProb $\leftarrow$ Pr(P,C); bestP $\leftarrow$ P; bestC $\leftarrow$ C
16:     **end if**
17:     **repeat**
18:       p $\leftarrow$ generatePath$(G \setminus \text{P}, s, t, l_1, l_2)$
19:     **until** (p $\neq \perp$) or MAX_TRIES attempts were done
20:     **if** (p $= \perp$) **then** continue //*aborts current "while" iteration*
21:     P $\leftarrow$ P $\cup$ {p}
22:     **if** Pr(P,C) $>$ highestProb **then**
23:       highestProb $\leftarrow$ Pr(P,C); bestP $\leftarrow$ P; bestC $\leftarrow$ C
24:     **end if**
25:     **repeat**
26:       c $\leftarrow$ generateCutset$(G, s, t, l_1 - 1, \text{P} \cup \text{C})$
27:     **until** (c $\neq \perp$) or MAX_TRIES attempts were done
28:     **if** (c $= \perp$) **then** continue //*aborts current "while" iteration*
29:     C $\leftarrow$ C $\cup$ {c}
30:     **if** Pr(P,C) $>$ highestProb **then**
31:       highestProb $\leftarrow$ Pr(P,C); bestP $\leftarrow$ P; bestC $\leftarrow$ C
32:     **end if**
33: **end while**
34: **return** bestP, bestC, highestProb

Figure 9.6: Pseudocode for the main heuristic; version PCPC.

Test	Goal	Size(n,m)	How the edge sets were chosen
1	To illustrate the gains in RE of our simulation method	(22,31)	Visual inspection
2	To compare the heuristics	(64,112) (225,420)	PC,PCP,PCC,PCPP, PCPC,PCCP,PCCC
3	To see the combined effect of chaining the heuristics and the simulation	(60,110)	PPPP $(\mathcal{Z}_0)$ PCCP $(\mathcal{Z}_1)$ CCCC $(\mathcal{Z}_2)$

Table 9.1: Goal, sizes and heuristics for the numerical tests.

9.6   Numerical examples

This section provides numerical examples based on mesh-like topologies. Tests 1 and 3 are inspired on the following situation. There is a contract between a communication network provider and a customer who needs to periodically exchange data between two sites s and t. They agree on a scale of fines, to be paid by the provider, according to the number of hops that each packet undergoes. The aim is to estimate the expected value of the fines that will be paid during the contract lifetime. To do so, simulations based on crude Monte Carlo and the proposed method (the version shown in Figure 9.6) were run and their results compared. Test 2 compares different versions of the heuristics. The methods were implemented in C++ and the tests were run on an Intel Core2 Duo T5450 machine with 2 GB of RAM, executing 10^7 iterations (sample size). Table 9.1 shows the goals, sizes and heuristics applied.

9.6.1   Test I - ANTEL's transport network

Test case I is based on the countrywide transport network topology of ANTEL, the largest telecommunications provider in Uruguay, shown in Figure 9.7. Nodes represent the sites whose interfaces perform routing activity, thus adding significant latency. Links represent the existing paths between these sites. Three scenarios are considered, corresponding to interface failure probability values of 0.10, 0.05 and 0.01 respectively. The test illustrates the effect that the rarity of edge failures has on the attained efficiency relative to crude Monte Carlo. Assume that nodes 4 and 14 (shown as squares) are to be connected in a context where low latencies are desirable. Table 9.2 lists the scale of fines payed according to the hop distance between both nodes. Three scales are used, each one corresponding to a certain value of link reliability in the network model (0.90, 0.95 and 0.99). The simulations will estimate the expected value of the average fine as well as its variance. The fine scales were proportionaly adjusted so that the expected values of the fines to pay were rather similar, by running short simulations. Note how the fines per region must quickly increase to yield the same expected value of fines when link reliabilities become higher, due to the fact that network configurations with high distances, or disconnected, become rarer events. In other words, this means that the provider can agree on paying

Δ	region	$r_e = 0.9$	$r_e = 0.95$	$r_e = 0.99$
up to 5	$\mathfrak{X}_0$	0	0	0
6 to 7	$\mathfrak{X}_1$	5	30	1,000
above 7	$\mathfrak{X}_2$	10	60	2,000
disconnected	$\mathfrak{X}_3$	20	120	4,000

Table 9.2: Test I: fines per region.

higher fines per region still facing the same fine expected value, because of improved link reliabilities. Table 9.3 shows the d-pathsets and d-cutsets employed, using the edge labels of Figure 9.7.
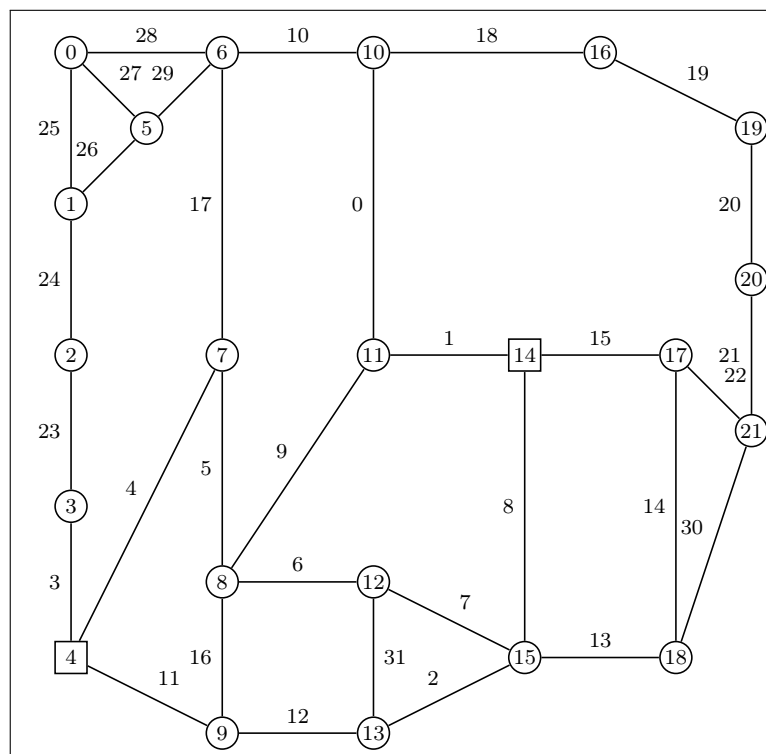
Figure 9.7: Test 2: reduced transport network topology of ANTEL, Uruguay's national telecommunications provider

Table 9.4 shows $\widehat{\Phi_N}$ and $\widetilde{\Phi_N}$ (the estimators of ν returned by the crude and proposed methods respectively); the estimates obtained for their variances; and the total times (in seconds). As above mentioned the scale of fines was set up so that the expected values of the fines to pay were approximately the same (they ranged from 0.341855 to 0.361926). Times spent are essentially the same across the three reliability scenarios, with the crude method taking a time approximately 11% lower than the proposed method. Observe the

Region	$\mathcal{P}_i$	$\mathcal{C}_i$
$\mathfrak{X}_0$	$\mathcal{P}_0 = \{\{4, 5, 9, 1\}, \{11, 12, 2, 8\}\}$	$\mathcal{C}_0 = \emptyset$
$\mathfrak{X}_1$	$\mathcal{P}_1 = \{\{11, 12, 2, 13, 14, 15\}\}$	$\mathcal{C}_1 = \{\{1, 8\}\}$
$\mathfrak{X}_2$	$\mathcal{P}_2 = \{\{4, 17, 10, 18, 19,$ $20, 21, 22, 15\}\}$	$\mathcal{C}_2 = \{\{1, 8, 13\}\}$
$\mathfrak{X}_3$	$\mathcal{P}_3 = \emptyset$	$\mathcal{C}_3 = \{\{3, 4, 11\}, \{1, 8, 15\}\}$

Table 9.3: Test I: d-pathsets and d-cutsets.

.	Crude	Proposed	ratio
$r_e = 0.9$			
Φ	0.341855	0.341856	-
σ^2	4.538235×10^{-7}	3.328820×10^{-8}	13.63
t(s)	290.9	323.7	0.8987
$r_e = 0.95$			
Φ	0.361926	0.361632	-
σ^2	2.467931×10^{-6}	5.451282×10^{-8}	45.27
t(s)	285.8	321.3	0.8896
$r_e = 0.99$			
Φ	0.338000	0.334667	-
σ^2	6.018858×10^{-5}	6.401462×10^{-8}	940.2
t(s)	280.4	321.7	0.8715

Table 9.4: Test I: numerical results.

significant reductions achieved in the variance by the proposed method (13.63, 45.27 and 940.2 for r_e equal to 0.90, 0.95 and 0.99 respectively). An efficiency comparison can be reported via the *relative efficiency* RE, which is a standard ratio employed in simulation literature (see e.g. [94], [12]), defined as follows:

$$\text{RE} = \frac{\text{Var}(\widehat{\Phi_N})}{\text{Var}(\widetilde{\Phi_N})} \times \frac{\text{time}_{\text{crude}}}{\text{time}_{\text{proposed}}}$$

The RE expresses the attained variance reduction adjusted by the spent-time ratio; in this case, its values are 12.25, 40.27 and 819.4 respectively for $r_e = 0.90$, $r_e = 0.95$ and $r_e = 0.99$.

9.6.2 Test II - Square grids

This test illustrates the behaviour of seven versions of the heuristic algorithm. One version (that we call PC) always returns one d-pathset and one d-cutset. Two versions (PCP, PCC) can return one extra edge set. Finally, four versions (PCPP, PCPC, PCCP, PCCC) return two, three or four edge sets whose nature corresponds to each letter (P=pathset, C=cutset). Two network topologies were employed: square grids with 8×8

Instance	Size	s	t	ℓ_1	ℓ_2	$\Pr(z_i)$
Grid1	8×8	(2,2)	(4,4)	6	10	1.895E-4
Grid2	8×8	(2,2)	(5,5)	8	12	1.974E-4
Grid3	15×15	(4,7)	(11,7)	10	20	2.529E-6
Grid4	15×15	(4,4)	(11,11)	18	24	9.575E-9

Table 9.5: Test II: grid-based instances for heuristic tests.

and 15×15 nodes respectively. Table 9.5 shows the characteristics of the four instances of the problem that were run for each version of the algorithm. Nodes s and t are specified by their "x, y coordinates" in the grid (numbered from zero). The reliability of each edge was randomly set, according to a triangular distribution (0.985, 0.99, 0.995). The parameters MAX_TIME and MAX_TRIES were set to 40 seconds and 5 tries. Last column of the table shows the highest probability found among those returned by each version of the algorithm.

Table 9.6 shows the four tests results; within each one, the algorithms are sorted by descending order of the probability that each one returned. Column labeled %best reports the ratio between the returned probability and the highest one for that particular test. Column #edges reports the total number of edges involved in the d-pathsets and d-cutsets of the returned solution. Columns '2 sets', '3 sets' and '4 sets' report the number of solutions generated that had two, three and four components. Finally column best(P,C) reports how many d-pathsets and d-cutsets there are in the solution returned by each algorithm. First, note that in all cases there is a significant difference between the best and worst returned solutions (their ratio ranging from 1,22 to 2,94). Second, three of the best solutions had four components and the remaining one had three. These results suggest that it might be worth to spend time looking for solutions with many components, rather than striving to get the best "one d-pathset and one d-cutset" possible solution, in topologies alike. Third, there is no clear winner, although PCCC, PCPC and PCCP seem to have the best overall results.

9.6.3   Test III - Randomized extension of Arpanet

This test illustrates the combined effect of the heuristic algorithm for edge sets generation and the variance-reduction technique. The network, shown in Fig 9.8, has 60 nodes and 110 edges. It was generated by growing the original Arpanet (20 nodes and 25 edges) with 40 new nodes, sequentially linked to the previous ones according to the random network model of [77]. As in Test 1, three instances were run, each one with all edges set to the same reliability. In this case the nodes s, t are represented as squares. Table 9.7 lists the scale of fines, split in three zones; again they were adjusted so that the expected value was similar for the different edge reliabilities. The heuristic PCCP was applied for $\mathcal{X}_1$ and it returned a solution built by one 12-pathset and two 5-cutsets. Heuristics PPPP and CCCC where applied in the regions $\mathcal{X}_0$ and $\mathcal{X}_2$, returning respectively three d-pathsets and two d-cutsets. The parameter MAX_TIME was set again to 40 seconds.

	%best	#edges	2 sets	3 sets	4 sets	best (P,C)
Grid1						
PCCP	1.000	22	1,055	726	687	2,2
PCPC	0.998	24	671	637	255	2,2
PCCC	0.911	18	497	336	71	1,3
PCC	0.911	14	1,086	729	0	1,2
PCP	0.509	18	3,186	3,052	0	2,1
PCPP	0.501	18	2,635	2,518	943	2,1
PC	0.465	10	4,135	0	0	1,1
Grid2						
PCPC	1.000	28	417	401	152	2,2
PCCP	1.000	28	604	392	377	2,2
PCCC	0.898	20	463	299	266	1,3
PCC	0.898	16	605	387	0	1,2
PCP	0.502	22	2,012	1,907	0	2,1
PCPP	0.501	24	1,709	1,623	621	2,1
PC	0.458	12	2,439	0	0	1,1
Grid3						
PCCC	1.000	26	82	55	43	1,3
PCPC	0.772	40	79	61	30	2,2
PCCP	0.769	42	105	73	54	2,2
PCC	0.667	23	110	74	0	1,2
PCPP	0.394	56	286	196	59	3,1
PCP	0.386	37	310	204	0	2,1
PC	0.340	18	380	0	0	1,1
Grid4						
PCP	1.000	50	169	105	0	2,1
PCPP	0.996	52	180	98	64	2,1
PCPC	0.996	52	56	27	13	2,1
PCCP	0.988	62	31	21	12	2,2
PC	0.821	28	202	0	0	1,1
PCC	0.821	28	37	17	0	1,1
PCCC	0.821	28	28	17	12	1,1

Table 9.6: Test II: ranking of results for the grid-based instances.

Δ	region	$r_e = 0.90$	$r_e = 0.95$	$r_e = 0.99$
up to 5	$\mathcal{X}_0$	0	0	0
6 to 12	$\mathcal{X}_1$	10	150	85,000
above 12	$\mathcal{X}_2$	1,000	15,000	8,500,000

Table 9.7: Test III: fines per region.

.	Crude	Proposed	ratio
$r_e = 0.90$			
Φ	0.928540	0.928373	-
σ^2	9.135802×10^{-5}	2.591886×10^{-6}	35.25
t(s)	606.4	975.2	0.6218
$r_e = 0.95$			
Φ	0.933135	0.915608	-
σ^2	1.366002×10^{-3}	6.188060×10^{-6}	220.75
t(s)	586.2	950.4	0.6168
$r_e = 0.99$			
Φ	0.935000	0.928744	-
σ^2	7.232224×10^{-1}	3.217148×10^{-5}	22,480.23
t(s)	513.1	914.4	0.5612

Table 9.8: Test III: numerical results.

The results of the test are summarized in Table 9.8; each time reported for the proposed method includes the 120 seconds spent generating the d-pathsets and d-cutsets. Note the significant relative efficiencies, in particular for rarer failures, obtained in this test: 21.92, 136.15 and 12,615.39 respectively for r_e equal to 0.90, 0.95 and 0.99.

9.7 Conclusions and future work

The proposed simulation method showed its capability to achieve significant variance reductions when applied on mesh-like networks. The necessary and sufficient conditions under which there is a reduction in the variance of the estimated parameter, with respect to the crude method, were shown in Lemma 9.1. The tests also showed that the proposed heuristic was able to generate sets that exploited the mentioned variance-reduction potential at significant levels. Even considering the extra time required for running the heuristic and for sampling with the proposed plan, the efficiency gains are noteworthy, specially when the links become more reliable.

The heuristics hereby introduced, yet resulting in important efficiency gains when chained to the simulation in these simple versions, can be improved in several ways. The algorithm could adapt the amount of effort spent in searching higher cardinality sets of d-pathsets and d-cutsets according to statistics on the sets so far found or on the
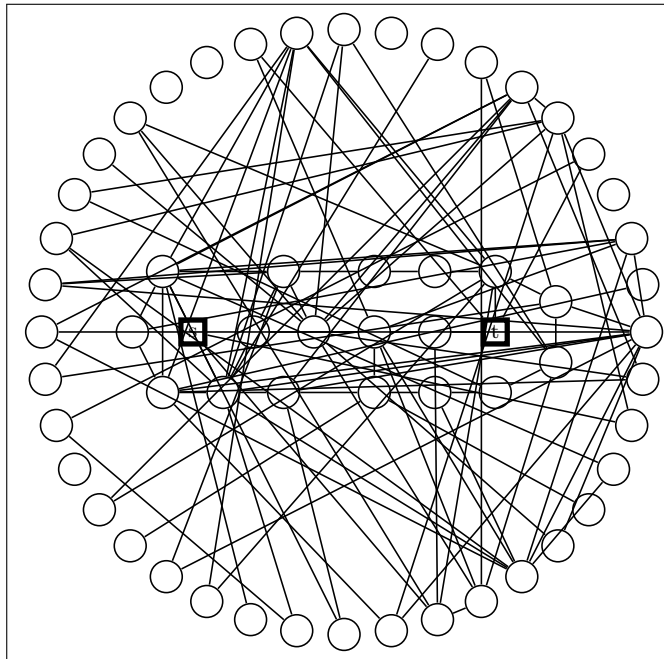
Figure 9.8: Network for Test 3 - Random extension of Arpanet

connectivity level of the terminal set. It could also alter the relative effort devoted to the generation of different sequences of d-pathsets and d-cutsets, reacting to the results so far obtained during execution. The reliability of each edge should be taken into account when choosing which one to add to the d-pathset or d-cutset under construction, particularly in networks where the reliabilities significantly differ. The time spent generating the sets MAX_TIME could also be initially set according to the sample size and the number of edges and regions (all of which determine at a large extent the time spent by the simulation). Moreover, it could be adjusted during the algorithm execution in light of the number and quality of the so far found sets.

Chapter 10

An interpolation technique for the edge-reliability problem

In this chapter we deal with the Edge-Reliability Problem: given a connected simple graph $G = (V, E)$ with perfect nodes and links failing independently with equal probability $1 - p$, find the probability $R_G(p)$ that the network remains connected. The *edge reliability* $R_G(p)$ is a polynomial in p with degree $m = |E|$, and the exact computation of $R_G(p)$ is in the class of $\mathcal{N}$P-hard problems. However, the related literature is vast, and offers bounds and estimation techniques, as well as exponential algorithms to exactly find that polynomial.

We propose a novel interpolation-based technique that exploits the structure of the Hilbert space $L^2[0, 1]$, and combines the efficiency of Newton interpolation with the simplicity of Monte Carlo reliability estimation in selected abscissas in $[0, 1]$. The aim is to guide the polynomial search respecting algebraic properties of the target polynomial coefficients. We illustrate the effectiveness of the algorithm in the lights of a naive graph with limited size, and discuss several hints for future work.

This chapter was accepted for publication in the 5th International Workshop on Reliable Networks Design and Modeling; see [16].

10.1   Introduction

Consider a connected simple graph $G = (V, E)$ with node-set $V = \{v_1, \ldots, v_n\}$ and links $E = \{e_1, \ldots, e_m\}$. The network has perfect nodes, but links operate independently, with identical probability p (or fail with probability $1 - p$). The link states are represented by a binary vector $(x_1, x_2, \ldots, x_m)$, and the network state is expressed by the structural indicator random variable $\Phi : \{0, 1\}^m \to \{0, 1\}$, defined by $\Phi(x_1, \ldots, x_m) = 1$ whenever the resulting subgraph with links $E' = \{e_i : x_i = 1, i = 1, \ldots, m\}$ is connected. Our objective is to find the *all-terminal reliability*, defined by the expected value $R_G(p) = E(\Phi) = P(\Phi(x_1, \ldots, x_m) = 1)$. In words, it is the connectedness probability of a random graph with perfect nodes and failure of its links.

10. AN INTERPOLATION TECHNIQUE FOR THE EDGE-RELIABILITY PROBLEM

A cut-set is a set of links C such that $\Phi(E/C) = 0$ (the subgraph $(V, E/C)$ is disconnected), and a min-cut is a minimum cardinality cut-set; we will denote c the minimum cardinality of a cut-set, and n_c to the number of such min-cuts. Similarly, a path-set is a set of links P such that $\Phi(P) = 1$ (the subgraph (V, P) is connected), and a min-path is a path-set with minimum cardinality; we will denote l to that cardinality, and n_l is the number of those min-paths.

In the all-terminal case, min-paths are spanning trees, so $l = n - 1$, and the number of spanning trees is the *complexity of a graph* or *tree-number*, introduced by Kirchhoff from his pioneer work in electrical networks: $n_l = \kappa(G)$, and $\kappa(G)$ is exactly a minor of the Laplacian matrix $L_G = \Delta_G - A_G$, being $\Delta_G = diag[deg(v_1), \ldots, deg(v_n)]$ and A_G is the adjacency matrix of G [26].

By means of Ford-Fulkerson's algorithm, the number c can be found in polytime with the size of the graph. Provan and Ball found in polytime the number n_c of min-cuts [32]. As a consequence, the four numbers l, n_l, c and n_c can be found in polytime, a distinguished characteristic for the all-terminal reliability problem (the computation of n_c is $\mathcal{N}$P-hard in the corresponding two-terminal connectedness problem, and consequently in the general K-terminal problem [105]).

There are several ways to express the all-terminal reliability; we will work with the F-form, defined as follows. Let $F_i = |\{E' \subseteq E : |E'| = i, \Phi(E/E') = 1\}|$ the number of subgraphs that remain connected after the deletion of exactly i links. The outcome of one of those graphs has probability $p^{m-i}(1-p)^i$, and as a consequence, the all-terminal reliability is the following polynomial:

$$R_G(p) = \sum_{i=0}^{m} F_i p^{m-i}(1-p)^i. \tag{10.1}$$

It is clear that $R_G(p)$ is a polynomial with integer coefficients, and $0 \leq F_i \leq \binom{m}{i}$. From prior observations, we know that $F_c = \binom{m}{c} - n_c$ and $F_{m-n+1} = n_l$, and both can be found in polytime. Moreover, by its definition we get that $F_i = 0$ whenever $i > m - n + 1$, whereas on the other hand $F_i = \binom{m}{i}$ if $i < c$. Re-writing Equation (10.1) we get that:

$$R_G(p) = \sum_{i=0}^{c-1} \binom{m}{i} p^{m-i}(1-p)^i + (\binom{m}{c} - n_c)p^{m-c}(1-p)^c$$

$$+ \sum_{i=c+1}^{m-n} F_i p^{m-i}(1-p)^i + n_l p^{n-1}(1-p)^{m-n+1}. \tag{10.2}$$

However, the general computation of the remaining $m - n - c$ coefficients of the F-vector (or equivalently, the reliability polynomial) is an $\mathcal{N}$P-hard problem [32]. As a consequence, the analysis is focused essentially in the design of exponentially exact algorithms, polynomial designs for special sub-classes of graphs, reliability estimation for a fixed operation p, or bounds.

Satyanarayana et. al. proved that the coefficients of the polynomial in the indeterminate variable p has integer coefficients that alternate in signs, from dominating set

theory [58]. The reliability polynomial is a special evaluation of the Tutte polynomial, and therefore it satisfies the deletion-contraction formula $R_G(p) = (1-p)R_{G-e}(p) + pR_{G*e}(p)$, where $G - e$ represents the deletion of a chosen link $e \in E$, and $G * e$ is the contraction operation (where the terminal nodes from e are now the same node, and multi-links are deleted). This formula leads to a exponentially time algorithm to recursively reduce the size of the graph and find $R_V(G)$ exactly. This idea has been exploited in many ways, by previous deletion of irrelevant links and series-parallel equivalences, that preserve the reliability measure. A related area is the pointwise estimation of $R_G(p)$, where the effort is primarily focused on the design of unbiased and time-efficient estimators with reduced variance, and the main mathematical machinery is Monte Carlo-based simulation methods [12, 106, 69].

Another alternative is the design of bounds, where the issue is to find $\underline{R_G(p)}$ and $\overline{R_G(p)}$ such that $\underline{R_G(p)} \leq R_G(p) \leq \overline{R_G(p)}$, uniformly over the compact interval $p \in [0, 1]$. Birnbaum et. al [107] noticed that the fraction of operational subgraphs with i edges over all subgraphs with i edges is monotonically increasing with i, and as a consequence $F_{i-1} \leq F_i \frac{i}{m-i+1}$ for all $i \in \{0, \dots, m\}$. Kruskal [108] and Katona [109] independently found a better bound (which is indeed the best for hereditary families of sets, without exploiting coherency) inspired in counting simplices of a complex: $F_{i-1} \leq F_i^{(i-1)/i}$. Alternative bounding techniques exploit coherency information from the network model. Brown et. al. for instance inspired in the underlying combinatorics of Stanley theorems on Hilbert spaces [110]. Other more recent fast bounding techniques rely on scanning weak cutsets, and trade accuracy for computational cost [111]. Exact computations are usually based on reliability-preserving transformations (such as the deletion-contraction formula), and try to speed-up the process [112].

In this chapter we address the computation of the reliability polynomial via an alternative interpolation-based technique, which, in the best case, returns the exact polynomial efficiently (in polytime with the size of the graph). The basic ingredients are Crude Monte Carlo (CMC) estimation for the reliability $R_G(p_i)$ over some finite set $\{p_i\} \in [0, 1]$, Newton interpolation, and least square error to find a feasible reliability polynomial from the interpolation.

This chapter is structured as follows. Section 10.2 presents CMC and Newton interpolation, which are the building blocks of our interpolation-based methodology. It also includes an analysis of variance in order to get an insight of the resulting error. Section 10.3 exploits the structure of the Hilbert space $L^2[0, 1]$ to find the nearest polynomial with integer coefficients in a least square integral sense. The full algorithm, its adjustment as a discussion of possible improvements is covered in Section 10.4. An illustrative numerical example is shown in Section 10.5. A discussion of trends and future work is included in Section 10.6, whereas Section 10.7 summarizes the main conclusions.

10.2 Crude Monte Carlo and Newton Interpolation

Suppose we are given a fixed probability of operation p, and we want to find the connectedness probability $\gamma = R_G(p) = P(\Phi = 1)$.

10. AN INTERPOLATION TECHNIQUE FOR THE EDGE-RELIABILITY PROBLEM

Crude Monte Carlo (CMC) simulates a sample $\Phi_1, \ldots, \Phi_N$ of N independent identically distributed variables respecting the structural law Φ, and proposes the estimator $\overline{\Phi_n}$ for $P(\Phi = 1)$:

$$\overline{\Phi_N} = \frac{1}{N} \sum_{i=1}^{N} \Phi_i. \tag{10.3}$$

As corollary from Kolmogorov's Strong Law, the estimator $\overline{\Phi_n}$ is consistent for γ. Moreover, $E(\overline{\Phi_N}) = R_G(p) = \gamma$ for all $N \geq 1$, so it is also unbiased. Assume N is sufficiently large to apply the Central Limit Theorem. Therefore, a confidence interval at level α for $R_G(p)$ is centered at $\overline{\Phi_n}$ and its radius is approximately equal to:

$$\delta = \frac{\sqrt{\overline{\Phi_N}(1 - \overline{\Phi_N})} t_{\alpha/2}(N-1)}{\sqrt{N}} \tag{10.4}$$

where $P(T > t_{\alpha/2}(n-1)) = \frac{\alpha}{2}$ for a t-Student random variable T with $n-1$ degrees. Observe that if we could choose the probability p such that $R_G(p) \approx 1/2$, we avoid corner effects coming from a *rare event simulation* [12]. In case $\overline{\Phi_N} \approx 1/2$, the radius is roughly:

$$\delta \approx \frac{z_{\alpha/2}}{2\sqrt{N}}, \tag{10.5}$$

where $P(> z_{\alpha/2}) = \frac{\alpha}{2}$ for a normal standard variable Z. Equation (10.5) states that the global error is inversely proportional to the square root of the sample size N.

Now, suppose we choose $m - n - c$ probabilities of operation $\{p_1, \ldots, p_{m-n-c}\}$ such that $R_G(p_i) \approx 1/2$ (or at least, $\min\{1 - R_G(p_i), R_G(p_i)\}_{i=1,\ldots,m-n-c} > \delta$ for some $\delta > 0$ large enough. A primitive bipartition in the set $[0,1]$ makes this assumption feasible. The idea is to obtain the unknowns $F_{c+1}, \ldots, F_{m-n}$ by means of Equation (10.2) and the resulting CMC estimations $y_i = \overline{\Phi_N}(p_i)$, by polynomial interpolation. We will first show that our problem is a classical interpolation. The second sum from Equation (10.2) can be re-written in a general form:

$$f(p) = \sum_{i=c+1}^{m-n} F_i p^{m-i} (1-p)^i$$

$$= p^m \left(\frac{1-p}{p} \right)^{c+1} \sum_{i=0}^{m-n-c-1} F_{c+1+i} \left(\frac{1-p}{p} \right)^i \tag{10.6}$$

If we change the variable $x = (1-p)p^{-1}$ and define $g(p) = f(p)p^{-m+c+1}(1-p)^{-c-1}$, the polynomial g is in its general form:

$$g(x) = \sum_{i=0}^{m-n-c-1} F_{c+1+i} x^i, \tag{10.7}$$

and the coefficients are exactly our unknowns. Finally note that via simple calculations we can get the estimations $\{f^*(x_1), f^*(x_2), \ldots, f^*(x_{m-n-c})\}$ starting from the set $\{y_i = \overline{\phi_N}(p_i)\}_{i=1,\ldots,m-n-c}$.

There exist several ways to find the unique interpolant. Vandermonde's method is conceptually simple, but an ill-conditioned linear system must be faced. Lagrange polynomials are easy to write, but hard to handle. We will see that Newton interpolation method is simultaneously simple and easy to evaluate. Last but not least, an error analysis is also feasible using Newton interpolation. The reader can find a thorough exposition of interpolation theory in the book [113]. By the previous reasons, we will focus on the Newton interpolation method. This method iteratively constructs the final interpolation including in each iteration a new point $(x_i, g(x_i))$. In fact, if g_i interpolates the first i points, then the new polynomial $g_{i+1}(x) = g_i(x) + c_{i+1} \prod_{j \leq i}(x - x_j)$ respects $g_{i+1}(x_k) = g_i(x_k)$ for all $k \leq i$, and c_{i+1} is found choosing $g_{i+1}(x_{i+1}) = y_{i+1}$.

Moreover, if g is the polynomial interpolation of a certain function h in the points $\{(x_i, h(x_i))\}_{i=1,\ldots,m-n-c}$ by means of Newton method, the global error can be found *exactly* for a given $x \in [x_1, x_{m-n-c}]$ [114]:

$$h(x) - g(x) = [x_1, x_2, \ldots, x_{m-n-c}, x](h) \prod_{i=1}^{m-n-c}(x - x_i), \qquad (10.8)$$

where the operator $[x_1, x_2, \ldots, x_n](f)$ is recursively defined by $[x_1]f = f(x_1)$ and

$$[x_1, \ldots, x_k]f = \frac{[x_2, \ldots, x_k](f) - [x_1, \ldots, x_{k-1}](f)}{x_k - x_1}. \qquad (10.9)$$

Therefore, we can find the order of a uniform bound for the worst error $\{|h(x) - g(x)|\}_{x \in [0,1]}$ in terms of N if we further neglect the rounding errors. Indeed, by (10.9), the propagation error can be duplicated in the worst case in each step, so the new radius is of order $\delta' = 2^{m-n-c}\delta$. If we choose the abscissas such that $1 > p_i > 1/2$, then $x_i = (1 - p_i)p_i^{-1} \in (0,1)$, and the product $\prod_{i=1}^{m-n-c}(x - x_i)$ is $O(1)$. As a consequence, we get that $\delta' = O(2^m/\sqrt{N})$, which suggests the sampling size N should increase exponentially with respect to the size of the graph. This highlights again the hardness of the problem we are addressing. However, we will see the success of the interpolation technique for graphs with limited size.

10.3 The reliability polynomial as a projection

The construction of Section 10.2 gives a polynomial interpolator $R^*(p)$ with real coefficients. However, the correct polynomial $R_G(p)$ has integer coefficients. What is more: they alternate in sign in its general form, and respect certain bounds. Let $R'(p) = \sum_{i=0}^{m} a_i p^i$ be the closest polynomial to $R^*(p)$ with respect to the metric from $L^2[0, 1]$:

$$R'(p) = \arg \min_{r(p)=\sum_{i=0}^{m} a_i p^i, \, a_i \in \mathbb{Z}} \int_0^1 (R^*(p) - r(p))^2 dp. \qquad (10.10)$$

It is not true that a_i are roundings of the coefficients from $R^*(p)$. The correct solution can be found using an orthonormal base in the Hilbert space $L^2[0, 1]$, for instance, the

normalized *shifted Legendre* polynomials $\hat{P}_i(x) = P(2x - 1)$, being $P_i(x)$ the Legendre polynomials, defined recursively by $P_0(x) = 1$, $P_1(x) = x$ and $(i + 1)P_{i+1}(x) = (2i + 1)xP_i(x) - iP_{i-1}(x)$.

If $< u, v >= \int_0^1 u(x)\overline{v(x)}dx$ denotes the inner product in $L^2[0, 1]$, then $< \hat{P}_i, \hat{P}_j >= \delta_{i,j}/(2j+1)$, being $\delta_{i,j}$ the δ-Kronecker function. The set $\{\hat{P}_i\}_{i \geq 0}$ is in fact an orthonormal basis for $L^2[0, 1]$[1]. If we use the orthonormal basis: $R_G(p) = \sum_{i=0}^m b_i\sqrt{2i + 1}\hat{P}_i(p)$ and $R'(p) = \sum_{i=0}^m \hat{b}_i\sqrt{2i + 1}\hat{P}_i(p)$, then $\hat{b}_i$ is the rounding of b_i. In fact:

$$\|R' - R_G\|_2^2 = \int_0^1 |R' - R_G|^2 dp$$
$$= \int_0^1 \left(\sum_{i=0}^m \sqrt{2i + 1}(b_i - \hat{b}_i)\hat{P}_i(p) \right)^2 dp$$
$$= \sum_{i=0}^m (b_i - \hat{b}_i)^2,$$

and the last sum is minimized exactly when the integer $\hat{b}_i$ is the rounding of b_i for all $i = 0, \ldots, m$.

10.4 Main Algorithm

In this section we present the main algorithm called *RelInterpol*, that combines the observations from previous sections and puts them all together in a coherent order.

Algorithm 1: $R_G = RelInterpol(G, x, N)$

1: $(c, n_c, l, n_l) \leftarrow Find(G)$
2: $y \leftarrow MonteCarlo(x, N)$
3: $g \leftarrow Newton(x, y)$
4: $R^* \leftarrow Build(g, c, n_c, l, n_l)$
5: $R' \leftarrow Hilbert(R^*)$
6: **while not** $HaltingTest(R') = 1$ **do**
7:    $Increment(N)$
8:    **go to Line 2**
9: **end while**
10: **return** $R_G = R'$

In Line 1, the four quantities c, n_c, l, n_l defined in Section 10.1 are found in polytime for the input graph G. Given the set of abscissas x in $[0, 1]$, a Crude Monte Carlo technique returns the vector y with the corresponding estimations for the all-terminal reliability, in Line 2. Newton's interpolation technique is applied in Line 3 regarding the

---
[1]Legendre polynomials can be obtained applying Gram-Schmidt orthogonalization process to the independent set of polynomials $\{x^i\}_{i \geq 0}$, and are dense in $L^2[-1, 1]$ as corollary of Stone-Weierstrass theorem.

abscissas x and corresponding estimations y, using Equation (10.7) to obtain the auxiliary polynomial g. The all-terminal reliability estimation R^* is built in Line 4. Function $Build$ just takes the coefficients of g and the four elementary quantities of G, to combine them using Equation (10.2). In Line 5, Function $Hilbert$ is called. Recall that R^* may have real coefficients. Therefore, Function $Hilbert$ returns an integer-valued polynomial R', whose coefficients are chosen in order to minimize the L^2 distance between R^* and R'. This is possible by the projection explained in Section 10.3. Function $HaltingTest$ is introduced in Line 6. This boolean function can either be true whenever the coefficients do not move largely, or check whether the algebraic properties shared by all reliability polynomials hold. Finally, if $HaltingTest$ is not true, the sampling size in Monte Carlo is increased in Line 7 in order to minimize the estimation error for each point in a new interpolation, and we return to Line 2. Otherwise, $R_G = R'$ is the output, and respects all previous requirements.

It is worth to notice that we can not certify whether the output polynomial is the correct one or not. However, in the lights of the uniform bound $O(2^m/\sqrt{N})$ for the error in $[0, 1]$, a large enough sampling size N is guarantee of success at a level $1 - \alpha$ (unless numerical errors dominate).

10.5   Numerical example

In this section we show a pictorial example to illustrate the effectiveness of our interpolation-based algorithm $RelInterpol$ with the small-size sample graph shown in Figure 10.1.
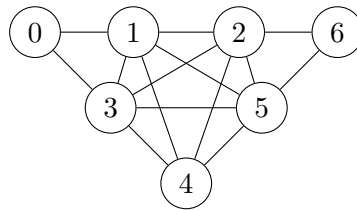

Figure 10.1: Sample network with $n = 7$ nodes and $m = 14$ edges.

Thanks to pioneer works from Kirchhoff, Ball and Provan, we know that the size of the mincut is $c = 2$, the number of feasible mincuts is $n_c = 2$, the size of a minpath is precisely $l = n-1 = 6$ and the tree-number equals $n_l = \kappa_K = 720$. Given that the sample graph has exactly $m = 14$ edges and $n = 7$ nodes, we need exactly $m - n - c = 5$ point-wise estimations for the reliability polynomial. Moreover, in this small-size instance an alternative exact algorithm can be implemented, to get the correct reliability polynomial:

$$R_G(p) = 720p^6 - 4136p^7 + 10741p^8 - 16356p^9 +$$
$$+ 15894p^{10} - 10056p^{11} + 4034p^{12} - 936p^{13} + 96p^{14} \qquad (10.11)$$

If we let $\epsilon = 1 - p$, the anti-reliability polynomial is defined by $Q_G(\epsilon) = 1 - R_G(1 - \epsilon)$. Replacing $p = 1 - \epsilon$ from Expression (10.11) we get a closed expression for the anti-reliability of our sample-graph:

$$U_G(\epsilon) = 2\epsilon^2 + 8\epsilon^5 - 8\epsilon^6 - 13\epsilon^8 - 64\epsilon^9$$
$$+ 78\epsilon^{10} + 288\epsilon^{11} - 602\epsilon^{12} + 408\epsilon^{13} - 96\epsilon^{14}. \tag{10.12}$$

It is worth to notice that the whole information for the reliability polynomial is stored by $g(x)$ from Expression (10.7). Therefore, we performed Newton's interpolation technique for $g(x)$ at five points in $[0, 1]$. More specifically, we considered the five abscissas in the high (and more interesting) zone $p_k = \frac{5+2k}{14}$, where $k \in \{0, 1, 2, 3, 4\}$, provided the reliability is not rare for that cases. Consider different sampling-sizes $N_i \in \{10^i, i = 5, 6, 7, 8\}$ and coefficients F_j, for $j \in \{6, \ldots, 14\}$. Table 10.1 reveals the estimated coefficients P_j^* for each sampling size N_i, versus the exact coefficients P_j of our sample graph.

Table 10.1: Performance for different sampling sizes.

Sample size (10^i)	P_6	P_7	P_8	P_9	P_{10}
5	720	-4020	9648	-12054	6674
6	720	-4047	9895	-12963	8477
7	720	-4113	10528	-15523	14110
8	720	-4125	10633	-15926	14957
Exact:	720	-4136	10741	-16356	15894

Sample size (10^i)	P_{11}	P_{12}	P_{13}	P_{14}
5	1595	-4669	2629	-523
6	-494	-3249	2103	-441
7	-7791	2328	-230	-28
8	-8851	3118	-554	29
Exact:	-10056	4034	-936	96

It can be appreciated that the relative error for the coefficients is consistently reduced with respect to the sampling size, and the estimation is more accurate in dominant coefficients. Remarkably, the dominant coefficient could always be guessed, even with low sample sizes. This fact highlights that our interpolation technique sounds promising for rare event analysis, where failures occur with very small probability.

In order to assess numerical effects, we computed the exact values of the reliability polynomial, and then we applied Newton's interpolation technique to them. The resulting polynomial identical to the exact one. As a consequence, we neglected numerical effects, since the error coming from other sources (mainly crude Monte Carlo estimation) dominate.

It is worth to notice that this methodology can be extended to other coherent network models, such as K-terminal reliability (connection between K special nodes). A

slight difference is that counting the number of mincuts or pathsets are both $\mathbb{N}$P-hard problems under K-terminal scenarios (so we have additional unknowns), leading to minor modifications. The number of mincuts in the K-terminal case can be guessed with this approach, regarding the stability of the coefficients when the sample size is increased.

10.6   Trends and Future Work

There are several issues that deserve further research in the design of efficient interpolation techniques to guess an all-terminal edge reliability polynomial. In this section we will discuss promising enhancements.

   The whole methodology rests in the accuracy of point-wise estimates for $R_G(p)$ using Crude Monte Carlo. There exist more efficient point-wise estimation techniques, that combine transformations that preserve the reliability measure and define a recursive estimation with lower variance than Crude Monte Carlo. Recursive Variance Reduction [69] is a remarkable example, where the term *efficiency* is defined by the variance-computational effort ratio.

   Apparently, there is a trade-off in the selection of interpolation points (precisely, the abscissas). On one hand, Newton's interpolation technique suggests to select regular steps in the points from the compact set $[0, 1]$ to reduce the interpolation error in $[0, 1]$. On the other hand, the operation near the borders of $[0, 1]$ makes the reliability (or unreliability) a rare event. This problem imposes a bridge between classical interpolation theory and stochastic analysis of rare events.

   The halting function could check algebraic properties shared by all reliability polynomials in the all-terminal model. Specifically, an ambitious design should check whether all the following requirements are met:

1. $F_i = 0$, $\forall i \leq m - n$; $F_i = \binom{m}{i}, \forall i < c$

2. $F_c = \binom{m}{c} - n_c$, $F_{m-n+1} = \kappa(G)$

3. $0 \leq F_i \leq \binom{m}{i}$, $\forall i : c + 1 \leq i \leq m - n$,

4. Kruskal-Katona inequalities hold: $F_{i+1} \leq (F_i)^{(i-1)/i}$

5. R' alternates signs when written in its general form.

6. Its real non-trivial roots are in the set $(1, 2]$.

   Other aspect for future research is a fast computation of the coefficients F_i, exploiting different orthonormal sets for $L^2[0, 1]$. The key point here is to observe that if $R_G(p) = \sum_{i=0}^{m} \alpha_i P_i(x)$, where $\{P_i(x)\}_{i=1,...,m}$ is an orthonormal set, then the coefficients can be expressed in terms of an integral, i.e.: $\alpha_i = <R_G, P_i> = \int_0^1 R_G(p)P_i(p)dp$. The approximation of an integral is a highly studied problem, and there are valuable numerical methods, including Monte Carlo for integrals (which in this case, it would be used for both point-wise estimation of R_G and its integration). Let us briefly explore one possibility.

Observe that $R_G(p) = \sum_{i=0}^{m} F_i p^{m-i}(1-p)^i$, where the polynomial set $S = \{p^{m-i}(1-p)^i\}$ is linearly independent but orthonormal. By means of Gram-Schmidt orthonormalization process to S, another orthonormal set S' can be found. However, an explicit determination for S' waits for a special determinant to be analytically solved. More specifically, the Gram matrix has the dot product $< p^i(1-p)^{m-i}, p^j(1-p)^{m-j} > = \frac{1}{2m+1}\binom{2m}{i+j}$ in its entry (i, j), and its determinant is needed to find the orthonormal set S' of polynomials. Observe that the Gram matrix of size k is $D_k = (< v_{i-1}, v_{j-1} >)_{1 \le i,j \le k}$ and its determinant for each k is:

$$|D_k| = (2m+1)^{-k} \left| \left(\binom{2m}{i+j-2} \right) \right|, \tag{10.13}$$

the matrix whose entry (i, j) is exactly $|\binom{2m}{i+j-2}|$ is a special one, called *Hankel matrix*, where all the entries of an anti-diagonal are identical (precisely, those entries (i, j) such that the sum $i + j$ is constant). To the best of our knowledge, the latter determinant has not been calculated analytically. A thorough revision of determinants is included in [115].

A complementary approach to be explored is to select $|x| > m - n - c$ abscissas and choose subsets of size $m - n - c$, in order to define a class C of $\binom{|x|}{m-n-c}$ polynomials. The Hausdorff distance between the compact set C and the class of polynomials with integer-valued and bounded coefficients (for a fixed upper-bound) gives an insight of the problem complexity and hints to guess the correct reliability polynomial.

Finally, bounds and exact results related with the coefficients of the reliability polynomial should be added in the last verification.

10.7 Conclusions

In this chapter we introduced a novel interpolation algorithm to estimate the all-terminal reliability polynomial. The main engine is a Crude Monte Carlo combined with Newton's method for polynomial interpolation. The algorithm exploits the Hilbert metric induced in $L^2[0, 1]$ to return a polynomial with integral coefficients that respects elementary properties shared by all reliability polynomials.

The order of the uniform error bound for the resulting and real polynomial is $O(2^m/\sqrt{N})$, being m graph size (i.e. its number of edges) and N the sampling size in Crude Monte Carlo. This result highlights the hardness of the polynomial computation.

A pictorial example with graphs of small size shows the algorithm consistency. The dominant coefficient was guessed with small sample sizes, which is promising for the study of rare events, where the dominant coefficient plays a central role. Moreover, once the estimation is achieved, a network sensibility analysis with respect to the elementary operational probability p can be easily performed, with vanishing error.

We encourage the scientific community to perform an in-depth analysis of this novel technique, which can be improved in several steps, and mixes statistical, algebraic and computational challenges. Indeed, the selection of the abscissas is in the world of classi-

cal interpolation theory, whereas more clever estimations than Crude Monte Carlo (for instance, Recursive Variance Reduction [69]) should be taken into account.

Part V

Conclusions

Chapter 11

Conclusions and perspectives

In this thesis we have contributed several results regarding the diameter-constrained network reliability (DCR) problem. We presented an introduction about several types of network properties that can be encompassed under the broad concept of *network dependability*. Then, we focused specifically on the role played by the evaluation of reliability metrics and in particular the DCR. Our contributions refer to the following topics:

- the computational complexity of particular cases of the DCR;

- topologies for which computing the DCR has polynomial complexity;

- asymptotic behavior of the DCR metric when the network grows following certain probabilistic rules;

- Monte Carlo simulation methods applied to estimating the DCR;

- a performability metric that generalizes the DCR and its estimation by a Monte Carlo method;

- the use of interpolation techniques combined with Monte Carlo for the CLR and DCR.

Next we summarize the main conclusions for each topic.

11.1   Computational complexity

The DCR problem generalizes the CLR problem, by constraining the operational network configurations to be such that there is un upper bound d for the distance between terminals. Several problems result in the same or increased computational complexity when adding such a constraint, for most values of the distance parameter d (see e.g. [9]). We have seen that the same applies when switching from CLR to DCR. It was known that the K-CLR was an $\mathcal{N}$P-hard problem in general, as well as for the particular case

where $d \geq 3$ and $|K| \geq 2$ is fixed. We proved that the case 2-K-DCR with any fixed $|K|$ can be solved employing dynamic programming techniques in a time that is linear in the number of nodes, although the algorithm that we propose is strongly exponential in $|K|$.

The classification of the computational complexity remains as an open problem for the d-K-DCR for all $d \geq 2$ whenever $|K|$ is a free input. As we commented in Section 4.4, this problem is at least as complex than that of counting the number of partial graphs of the network that have a given diameter. Hence, proving that the latter is $\mathcal{N}$P-hard would yield that the d-K-DCR is $\mathcal{N}$P-hard too. We are currently focusing our efforts on proving this. An extension of this complexity classification for the DCR can be done for particular topologies; mainly looking for topologies and combinations of $|K|$ and d for which polynomial algorithms can be devised.

Figure 11.1 reproduces the classes of the subproblems that arise for different combinations of $|K|$ and d and what is currently known about their complexity.

Figure 11.1: Complexity of DCR subproblems for any d and fixed $|K|$

11.2 Exact and asymptotic analysis of DCR metrics

It is known that both the CLR and the DCR are in general $\mathcal{N}$P-hard complex problems, yet for many particular classes of topologies they can be solved in polynomial time respect to the number of nodes. We have illustrated the use of dynamic programming techniques to compute the DCR for two simple families of networks. We have also characterized a broad family of topologies for which the two-terminal DCR can be computed in polynomial time. The graphs must be such that the node set can be partitioned in subsets whose cardinality is bounded by a certain parameter h. The partition defines what the feasible edges are. The time order is polynomial in the cardinality of the partition (and

thus in the total number of nodes) yet exponential in the parameter h. This polynomial algorithm allows to compute the exact DCR of large graphs that belong to these family. This can be useful e.g. for computing bounds (following methods based on identifying partial graphs with particular topologies), or for having exact values against which to compare the outcomes of estimation and bounding methods.

When networks grow following certain patterns, it is natural to ask how its reliability properties evolve. We have introduced the question of analyzing the asymptotic behavior of CLR and DCR metrics, in networks that grow according to models extracted from Random Graph Theory. In Chapter 6.4 we derive some results for the Gilbert's mode, according to which each pair of nodes has the same constant probability p of being linked by an edge. We proved that the K-DCR of such a network tends to one, no matter how small (yet strictly positive) is p. Moreover, we showed that a certain progressive degradation of the link reliability p is admissible, yet resulting in the network reliability tending to one. This work can be extended by studying the asymptotic behavior for other random models like the ones of Albert-Barabasi and Watts-Strogatz. Possible applications of this analysis could be devised in contexts where networks reach large sizes following predictable random patterns, like the Internet itself, marketing campaigns, knowledge sharing networks, etc.

11.3 Monte Carlo estimation of DCR

The high computational complexity of the CLR has motivated intense research on bounding and estimation procedures. Among the latter, several Monte Carlo methods have been conceived. In real communication networks, the reliability is usually very close to one, and thus sampling a network in failed state is a rare event. If the crude Monte Carlo method is used, this forces to employ large sample sizes to reach statistically significative results. Moreover, the sample sizes unboundedly grow as the network reliability is closer to one. Therefore, Monte Carlo methods with variance-reduction techniques have been developed for the CLR. In Chapter 8 we have adapted a family of such methods to the DCR context, showing how the transition from CLR to DCR context affects each step of the simulation. We studied their relative efficiency vs. crude Monte Carlo. This family is based on conditioning the sampling space based on pre-computed pathsets and cutsets. The tests run evidenced the following:

- the methods based on precomputing pathsets and cutsets have better relative efficiencies, as it was the case for the CLR;

- when the number of edges involved in the pathsets and cutsets ($|\Omega|$) allows for pre-computing a table with all the configurations and their elemental probabilities of occurrence, the forthcoming simulation has the best performance. This is useful in particular if the same probabilistic graph is to be tested several times, with different values for d;

- high values of $|K|$ and d result in the conditioned method having even better

relative efficiencies for DCR than for CLR, due to the higher times required for testing d-K-connectivity;

The disjointness condition imposed to the pathsets and cutsets in the best-performing variation of the proposed methods, can be relaxed if a certain level of overlapping is allowed, allowing for further reduction of the sampling space and variance reduction. Yet the size of the polynomials that must be evaluated when sampling edges in Ω grows exponentially with the number of edges overlapping, negatively affecting the sampling time. This trade-off has to be studied aiming to improve the overall performance. Other Monte Carlo methods developed for CLR will be adapted and tested for DCR on a common set of benchmarking cases in order to compare their relative performances. We are working also on hybrid methods that combine simulation techniques and polynomial interpolation to estimate the reliability polynomial (or its most relevant coefficients in the context of rare failures).

11.4 K-diameter-dependent performability metrics

Besides reliability metrics, that focus on connectedness of the network, performability metrics have been introduced, better reflecting concerns like e.g. overall performance and quality perceived. These metrics are often defined as a weighted average of the probability that the network is in certain states if these are discrete, or a weighted integral of the density of probability for the state function in general. Many methods that are applied for computing and estimating performability metrics find their roots in previous work developed for reliability metrics. In Chapter 9 we have adapted such a Monte Carlo method, for estimating any metric defined in terms of the K-diameter of the network. We assume that the possible diameters are partitioned in a number of ranges and each range is given a weight for building the metric. Performability metrics that consider the K-diameter can be useful in contexts where latencies affect the user experience, like P2P networks, VoIP or web applications. We showed that significant variance reductions relative to crude Monte Carlo are achieved by running tests on mesh-like networks. We introduced heuristics that generate pathsets and cutsets to be used for conditioning the sampling spaces. Further work to be done include:

- the heuristics, yet having resulted in important relative efficiency gains when chained to the simulation, can be improved in several ways. They should take into account, reacting to the ongoing results, aspects like the effort put in each range of diameters and possible sequence of pathset/cutset generation, individual edge reliabilities, time spent generating the sets and others;

- tests could be devised to study the correlation between this family of metrics and the quality perceived by users, e.g. in audio or video streaming;

- other performability metrics derived from the K-diameter can be conceived, e.g. defined in terms of the variance of the K-diameter rather than on its static values.

11.5   Interpolation techniques for reliability metrics

A novel interpolation algorithm was introduced in Chapter 10 to estimate reliability polynomials. Although we developed it for the all-terminal CLR, it can also be applied to the DCR and general sets K. The algorithm is based on Monte Carlo simulation for estimating the network reliability for different edge reliabilities followed by a Newton's polynomial interpolation. The Hilbert metric induced in $L^2[0,1]$ is used to project and return a polynomial with integral coefficients that satisfies elementary properties shared by all reliability polynomials. The pictorial example shows that the first (highest degree) coefficients are estimated with more precision than the remaining. This is a promising result since the dominant coefficients are the most relevant in the context of rare failures. An important advantage of knowing the reliability polynomial is that network sensitivity analysis can be easily done respect to the edge reliability p, with vanishing error. We are extending this line of research by analyzing the algorithm behavior on networks with real sizes and topologies.

Bibliography

[1] C. J. Colbourn, "Reliability issues in telecommunications network planning," in *Telecommunications network planning, chapter 9*, pp. 135–146, Kluwer Academic Publishers, 1999. xv, xvi, 3, 4, 109

[2] T. Ohno, *Toyota Production System: Beyond Large-Scale Production.* Productivity Press, 1988. xvii, 5

[3] G. Tennant, *Six Sigma: SPC and TQM in manufacturing and services.* Gower Publishing, Ltd., 2001. xvii, 5

[4] C. J. Colbourn, *The Combinatorics of Network Reliability.* New York, NY, USA: Oxford University Press, Inc., 1987. xviii, 7

[5] G. Rubino, *Network reliability evaluation. In State-of-the-art in performance modeling and simulation*, ch. 11. Gordon and Breach Books, 1996. xviii, 7, 60, 87

[6] F. T. Boesch, A. Satyanarayana, and C. L. Suffel, "A survey of some network reliability analysis and synthesis results," *Netw.*, vol. 54, pp. 99–107, Sept. 2009. xviii, 7

[7] L. Petingi and J. Rodriguez, "Reliability of networks with delay constraints," in *Congressus Numerantium*, vol. 152, pp. 117–123, 2001. xviii, xxi, 7, 18

[8] L. Petingi, "A diameter-constrained network reliability model to determine the probability that a communication network meets delay constraints," *WSEAS Transactions on Communications (WTOC)*, vol. 7, pp. 574–583, June 2008. xviii, 7, 35

[9] P. A. Golovach and D. M. Thilikos, "Paths of bounded length and their cuts: Parameterized complexity and algorithms.," in *IWPEC'09*, pp. 210–221, 2009. xxiii, 29, 92, 110, 143

[10] H. Cancela and L. Petingi, "Reliability of communication networks with delay constraints: computational complexity and complete topologies," *International Journal of Mathematics and Mathematical Sciences*, vol. 2004, pp. 1551–1562, 2004. xxiv, 29, 50, 144

[11] M. Ball, C. Colbourn, and J. Provan, "Network Reliability," tech. rep., University of Maryland TR 92-74, June 1992. xxiv, 16, 28, 30, 50, 54, 87, 144

[12] G. Rubino and B. Tuffin, *Rare event simulation using Monte Carlo methods*. Wiley, 2009. xxv, 125, 133, 134

[13] E. Canale, H. Cancela, F. Robledo, G. Rubino, and P. Sartor, "On computing the 2-diameter-constrained K-reliability of networks," *International Transactions in Operational Research*, vol. 20, no. 1, pp. 49–58, 2013. xxvii, 8, 9, 31

[14] H. Cancela, F. Robledo, G. Rubino, and P. Sartor, "Monte Carlo estimation of diameter-constrained network reliability conditioned by pathsets and cutsets," *Computer Communications*, vol. 36, no. 6, pp. 611 – 620, 2013. xxvii, 9, 85, 117

[15] H. Cancela, F. Robledo, G. Rubino, and P. Sartor, "A Monte Carlo sampling plan for estimating diameter-dependent network parameters," in *Proceedings of the 4th International Workshop on Reliable Networks Design and Modeling*, pp. 129–134, 2012. xxvii, 9, 111

[16] F. Robledo, P. G. Romero, and P. Sartor, "A novel interpolation technique to address the Edge-Reliability problem," in *5th International Workshop on Reliable Networks Design and Modeling (RNDM'13)*, (Almaty, Kazakhstan), pp. 77–82, Sept. 2013. xxvii, 9, 131

[17] H. Cancela, F. Robledo, G. Rubino, and P. Sartor, "Efficient estimation of distance-dependent metrics in edge-failing networks," *To appear in ITOR (International Transactions in Operational Research)*, 2013. xxvii, 9, 111

[18] F. Robledo and P. Sartor, "A simulation method for network performability estimation using heuristically computed pathsets and cutsets," *To appear in IJMH (International Journal of Metaheuristics)*, 2013. xxvii, 9, 111

[19] J. Meyer, "On evaluating the performability of degradable computing systems," *Computers, IEEE Transactions on*, vol. C-29, no. 8, pp. 720–731, 1980. 4, 109

[20] B. Haverkort, R. Marie, G. Rubino, and K. Trivedi, *Performability Modelling: Techniques and Tools*. Chichester, England: John Wiley & Sons, 2001. 4, 109

[21] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, pp. 11–33, Jan. 2004. 5, 16, 109

[22] K. Godel, *On formally undecidable propositions of Principia Mathematica and related systems*. Dover, 1992. 26

[23] A. Turing, "On computable numbers with an application to the Entscheidungsproblem," *Proceeding of the London Mathematical Society*, 1936. 26

[24] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, (New York, NY, USA), pp. 151–158, ACM, 1971. 27

[25] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations* (R. E. Miller and J. W. Thatcher, eds.), pp. 85–103, Plenum Press, 1972. 27, 29

[26] N. Biggs, *Algebraic Graph Theory*. Cambridge Mathematical Library, Cambridge University Press, 1993. 28, 132

[27] E. Moore, "The shortest path through a maze," *Ann. Computational Lab. Harvard University*, vol. 30, pp. 285–292, 1959. 29

[28] S. Even and R. Tarjan, "Network flow testing and graph connectivity," *SIAM J. Computing*, vol. 4, pp. 507–518, 1975. 29

[29] L. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton University Press, 1962. 29

[30] M. O. Ball and J. S. Provan, "Calculating bounds on reachability and connectedness in stochastic networks," *Networks*, vol. 13, no. 2, pp. 253–278, 1983. 29

[31] G. Kirchoff, "Über die auflösung der gleichungen, auf welche man bei der untersuchung der linearen verteilung galvanischer ströme geführt wird," *Ann. Phys. Chem.*, vol. 72, pp. 497–508, 1847. 29

[32] M. O. Ball and J. S. Provan, "The complexity of counting cuts and of computing the probability that a graph is connected," *SIAM J. Computing*, vol. 12, pp. 777–788, 1983. 29, 132

[33] R. Bixby, "The minimum number of edges and vertices in a graph with edge-connectivity n and m n-bonds," *Networks*, vol. 5, pp. 259–298, 1975. 29

[34] L. Valiant, "The complexity of enumeration and reliability problems," *SIAM Journal on Computing*, vol. 8, no. 3, pp. 410–421, 1979. 29

[35] S. Provan, "The complexity of reliability computations in planar and acyclic graphs," *SIAM J. Comput.*, vol. 15, pp. 694–702, Aug. 1986. 29

[36] S. Rai, M. Veeraraghavan, and K. S. Trivedi, "A survey of efficient reliability computation using disjoint products approach," *Networks*, vol. 25, no. 3, pp. 147–163, 1995. 53

[37] L. Fratta and U. Montanari, "A boolean algebra method for computing the terminal reliability in a communication network," *Circuit Theory, IEEE Transactions on*, vol. 20, pp. 203 – 211, may 1973. 54

[38] J. Abraham, "An improved algorithm for network reliability," *Reliability, IEEE Transactions on*, vol. R-28, pp. 58 –61, april 1979. 54

[39] M. Locks, "Recursive disjoint products: A review of three algorithms," *Reliability, IEEE Transactions on*, vol. R-31, pp. 33 –35, april 1982. 54

[40] S. H. Ahmad, "A simple technique for computing network reliability," *IEEE Transactions on Reliability*, vol. R-31, pp. 41 –44, april 1982. 54, 58

[41] S. H. Ahmad and A. T. M. Jamil, "A modified technique for computing network reliability," *IEEE Transactions on Reliability*, vol. R-36, pp. 554 –556, dec. 1987. 54

[42] U. Niessen and W. Schneeweiss, "A practical comparison of several algorithms for reliability calculations," *Reliability engineering and system safety*, vol. 31, no. 1, pp. 309 –319, 1991. 54

[43] R. Marie and G. Rubino, "Direct approaches to the 2-terminal reliability problem," in *The third international symposium on computer and information sciences*, (Cesme, Izmir, Turkey), pp. 740–747, Ege University, E.Gelembe, E.Orhun and E.Basar, 1988. 54

[44] A. Granov, L. Kleinrock, and M. Gerla, "A new algorithm for symbolic reliability analysis of computer communication networks," in *Pacific Telecommunications Conference, Honolulu, HI*, pp. 1A11–1A19, Jan 1980. 54

[45] M. Veeraraghavan and K. Trivedi, "An improved algorithm for symbolic reliability analysis," *Reliability, IEEE Transactions on*, vol. 40, no. 3, pp. 347–358, 1991. 54

[46] T. Luo and K. Trivedi, "An improved algorithm for coherent-system reliability," *Reliability, IEEE Transactions on*, vol. 47, pp. 73 –78, mar 1998. 54

[47] A. Rauzy, "Binary decision diagrams for reliability studies," in *Handbook of Performability Engineering* (K. Misra, ed.), pp. 381–396, Springer London, 2008. 54

[48] A. Shooman and A. Kershenbaum, "Methods for communication-network reliability analysis: probabilistic graph reduction," in *Reliability and Maintainability Symposium, 1992. Proceedings., Annual*, pp. 441 –448, jan 1992. 54

[49] J. Hagstrom, "Redundancy, substitutes and complements in system reliability," tech. rep., College of Business Administration, University of Illinois, 1990. 54

[50] M. G. C. Resende, "A program for reliability evaluation of undirected networks via polygon-to-chain reductions," *IEEE Transactions on Reliability*, vol. 35, pp. 24 –29, april 1986. 54

[51] R. K. Wood, "A factoring algorithm using polygon-to-chain reductions for computing K-terminal network reliability," *Networks*, vol. 15, no. 2, pp. 173–190, 1985. 54, 55

[52] A. Rosenthal, "Computing the reliability of complex networks," *SIAM Journal on Applied Mathematics*, vol. 32, no. 2, pp. 384–393, 1977. 54

[53] A. Rosenthal and D. Frisque, "Transformations for simplifying network reliability calculations," *Networks*, vol. 7, no. 2, pp. 97–111, 1977. 54

[54] J. N. Hagstrom, "Using the decomposition-tree of a network in reliability computation," *IEEE Transactions on Reliability*, vol. R-32, pp. 71 –78, april 1983. 54

[55] F. Moskowitz, "The analysis of redundancy networks," *AIEE Transactions on Communication Electronics*, vol. 39, pp. 627–632, 1958. 55

[56] H. Mine, "Reliability of physical systems," *IRE Transactions on Circuit Theory*, vol. 6, pp. 138–151, 1959. 55

[57] A. Satyanarayana and A. Prabhakar, "New topological formula and rapid algorithm for reliability analysis of complex networks," *Reliability, IEEE Transactions on*, vol. R-27, pp. 82 –100, june 1978. 55

[58] A. Satyanarayana and M. K. Chang, "Network reliability and the factoring theorem," *Networks*, vol. 13, no. 1, pp. 107–120, 1983. 55, 133

[59] A. Agrawal and R. Barlow, "A survey of network reliability and domination theory," *Operations Research*, vol. 32, pp. 478–492, 1984. 55

[60] R. Barlow and S. Iyer, "Computational complexity of coherent systems and the reliability polynomial," *Probability in the Engineering and Informational Sciences*, vol. 2, pp. 461–469, 9 1988. 55

[61] A. Huseby, "A unified theory of domination and signed domination with applications to exact reliability domination. statistical research report," tech. rep., Institute of Mathematics, University of Oslo, 1984. 55

[62] C. A.A. and C. Z.A., "A unified domination approach for reliability analysis of networks with arbitrary logic in vertices.," *Microelectronics and Reliability*, vol. 37, no. 4, pp. 696–696, 1997. 55

[63] R. K. Wood, "Factoring algorithms for computing K-terminal network reliability," *IEEE Transactions on Reliability*, vol. 35, pp. 269 –278, aug. 1986. 55

[64] H. Cancela and L. Petingi, "Diameter constrained network reliability: exact evaluation by factorization and bounds," in *ICIL'2001 (International Conference on Industrial Logistics)*, pp. 359–366, 2001. 56, 57

[65] H. Cancela, M. El Khadiri, and L. Petingi, "An exact method to evaluate the source-to-terminal diameter constrained reliability of a communication network," in *Proceedings of International Conference on Industrial Logistics (Logistic and Sustainability*, pp. 365–372, 2010. 56

[66] H. Cancela, M. El Khadiri, and L. Petingi, "Polynomial-time topological reductions that preserve the diameter constrained reliability of a communication network," *IEEE Transactions on Reliability*, vol. 60, pp. 845–851, dec 2011. 56

[67] D. Migov, "Computing diameter constrained reliability of a network with junction points," *Automation and Remote Control*, vol. 72, pp. 1415–1419, 2011. 10.1134/S0005117911070095. 57, 69

[68] H. Cancela and L. Petingi, "On the characterization of the domination of a diameter-constrained network reliability model," *Discrete Appl. Math.*, vol. 154, pp. 1885–1896, August 2006. 57

[69] H. Cancela and M. E. Khadiri, "The recursive variance-reduction simulation algorithm for network reliability evaluation," *IEEE Transactions on Reliability*, vol. 52, no. 2, pp. 207–212, 2003. 58, 133, 139, 140

[70] M. Bertinat, "Study of recursive variance reduction methods for diameter-constrained network reliability (original in Spanish)," Master's thesis, Engineering School, University of the Republic, Uruguay, 2011. 58

[71] H. Cancela, G. Rubino, and M. Urquhart, "An algorithm to compute the all-terminal reliability measure," *Journal of the Indian Operational Research Society (OPSEARCH)*, vol. 38, no. 6, pp. 567–579, 2001. 58

[72] H. Cancela, M. Urquhart, and G. Rubino, "Network reliability evaluation by the Ahmad method," in *XXIII Latin-american Conference on Informatics - CLEI 97*, vol. 1, pp. 327–336, November 1997. 60

[73] P. Erdos and A. Rényi, "On random graphs," *Publicationes Mathematicae*, vol. 6, pp. 290–297, 1959. 79

[74] E. N. Gilbert, "Random graphs," *Annals of Mathematical Statistics*, vol. 30, pp. 1141–1144, 1959. 79

[75] B. Bollobás, *Random Graphs*. Cambridge University Press, 2001. 79

[76] N. Alon and J. Spencer, *The Probabilistic Method (2nd. ed)*. Wiley-Interscience, 2000. 79

[77] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, vol. 74, pp. 47–97, Jan 2002. 79, 81, 126

[78] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, June 1998. 79, 81

[79] B. Bollobás, "The diameter of random graphs," *Transactions of the American Mathematical Society*, vol. 267, no. 1, pp. pp. 41–52, 1981. 79

[80] B. Bollobás and O. Riordan, "The diameter of a scale-free random graph," *Combinatorica*, vol. 24, pp. 5–34, Jan. 2004. 79

[81] F. Chung and L. Lu, *Complex Graphs and Networks*, vol. 107. American Mathematical Society, 2006. 79

[82] J. Moon and L. Moser, "Almost all (0,1) matrices are primitive," *Studia Sci. Math. Hungar*, vol. 1, pp. 153–156, 1966. 80

[83] Y. Burtin, "Assymptotic estimates of the diameter, independence and domination numbers of a random graph," *Soviet Mat. Doklady*, vol. 14, pp. 497–501, 1973. 80

[84] M. Horváthová, "On (k,l)-radius of random graphs," *Acta Mathematica Universitatis Comenianae*, vol. 75, no. 2, pp. 149–152, 2006. 80

[85] J. D. Burtin, "Extremal metric characteristics of a random graph i. asymptotic estimates," *Teor. Verojatnost. i Primenen*, vol. 19, pp. 740–754, 1974. 80

[86] J. D. Burtin, "Extremal metric characteristics of a random graph ii. limit distributions," *Teor. Verojatnost. i Primenen*, vol. 20, pp. 82–99, 1975. 80

[87] H. Cancela, M. El Khadiri, and G. Rubino, *Rare event analysis by Monte Carlo techniques in static models. In Rare Event Simulation using Monte Carlo Methods, G. Rubino, B. Tuffin, Eds.*, ch. 7, pp. 145–170. Chichester: John Wiley & Sons, 2009. 85, 87

[88] I. B. Gertsbakh and Y. Shpungin, *Models of Network Reliability: Analysis, Combinatorics, and Monte Carlo.* Boca Raton, FL, USA: CRC Press, Inc., 1st ed., 2009. 87

[89] H. Cancela, P. L'Ecuyer, M. Lee, G. Rubino, and B. Tuffin, *Analysis and improvements of path-based methods for Monte Carlo reliability evaluation of static models. In Simulation Methods for Reliability and Availability of Complex Systems, S. M. J. Faulin, A. A. Juan, and E. Ramirez-Marquez, Eds.*, pp. 65 – 84. Berlin, Germany: Springer-Verlag, 2009. 87

[90] H. Cancela, P. L'Ecuyer, G. Rubino, and B. Tuffin, "Combination of conditional Monte Carlo and approximate zero-variance importance sampling for network reliability estimation," in *Proceedings of the 2010 Winter Simulation Conference (WSC)*, pp. 1263 –1274, dec. 2010. 87

[91] P. L'Ecuyer, G. Rubino, S. Saggadi, and B. Tuffin, "Approximate zero-variance importance sampling for static network reliability estimation," *IEEE Transactions on Reliability*, vol. 60, pp. 590–604, 2011. 87

[92] R. Zenklusen and M. Laumanns, "High-confidence estimation of small s-t reliabilities in directed acyclic networks," *Networks*, vol. 57, no. 4, pp. 376–388, 2011. 87

[93] Z. I. Botev, P. L'Ecuyer, G. Rubino, R. Simard, and B. Tuffin, "Static network reliability estimation via generalized splitting," *INFORMS Journal on Computing*, vol. 25, no. 1, pp. 56–71, 2013. 87

[94] G. S. Fishman, "A Monte Carlo sampling plan for estimating network reliability," *Operations Research*, vol. 34, pp. 581–594, july-august 1986. 89, 91, 92, 94, 95, 103, 125

[95] R. Van Slyke and H. Frank, "Network reliability analysis: Part i," *Networks*, vol. 1, no. 3, pp. 279–290, 1971. 89, 93

[96] H. Kumamoto, K. Tanaka, and K. Inoue, "Efficient evaluation of system reliability by Monte Carlo method," *IEEE Transactions on Reliability*, vol. R-26, pp. 311 –315, dec. 1977. 89, 94

[97] G. S. Fishman, "A comparison of four Monte Carlo methods for estimating the probability of s-t connectedness," *IEEE Transactions on Reliability*, vol. 35, pp. 145 –155, june 1986. 89

[98] R. Barlow and K. Heidtmann, "Computing k-out-of-n system reliability," *IEEE Transactions on Reliability*, vol. 33, pp. 58 – 61, oct 1984. 94

[99] L. Belfore, "An o(n.(log[2](n))[2]) algorithm for computing the reliability of k-out-of-n:g and k-to-l-out-of-n:g systems," *IEEE Transactions on Reliability*, vol. 44, no. 1, pp. 132 – 136, 1995. 94

[100] G. Arulmozhi, "Direct method for reliability computation of k-out-of-n: G systems," *Applied Mathematics and Computation*, vol. 143, no. 2-3, pp. 421 – 429, 2003. 94

[101] E. Manzi, M. Labbe, G. Latouche, and F. Maffioli, "Fishman's sampling plan for computing network reliability," *IEEE Transactions on Reliability*, vol. 50, pp. 41 –46, mar 2001. 95

[102] M. Veeraraghavan and K. S. Trivedi, "A combinatorial algorithm for performance and reliability analysis using multistate models," *IEEE Trans. Comput.*, vol. 43, pp. 229–234, Feb. 1994. 109

[103] Z. Wang and J. Crowcroft, "Quality of service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 1228–1234, 1996. 109

[104] J. J. Garrett, *Ajax: A New Approach to Web Applications.* *http://www.adaptivepath.com/publications/essays/archives/000385.php.* Last accessed July 8th 2013, Feb. 2005. 110

[105] M. O. Ball, C. J. Colbourn, and J. S. Provan, "Network Reliability," in *Network Models* (M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, eds.), vol. 7 of *Handbooks in Operations Research and Management Science*, ch. 11, pp. 673–762, The Netherlands: Elsevier Science, 1995. 132

[106] H. Cancela, M. E. Khadiri, and G. Rubino, "A new simulation method based on the rvr principle for the rare event network reliability problem," *Annals OR*, vol. 196, no. 1, pp. 111–136, 2012. 133

[107] Z. Birnbaum, J. Esary, S. Saunders, B. S. R. L. M. R. Laboratory, W. U. S. L. O. S. RESEARCH., and U. S. O. of Naval Research, *Multi-component Systems and Structures and Their Reliability.* Defense Technical Information Center, 1960. 133

[108] J. B. Kruskal, "The number of simplices in a complex," *Mathematical Optimization Techniques, University of California Press*, pp. 251–278, 1963. 133

[109] G. Katona, *A theorem of finite sets.* Budapest: Akadémia Kiadó, 1968. 133

[110] J. I. Brown, C. J. Colbourn, and J. S. Devitt, "Network transformations and bounding network reliability," *Networks*, vol. 23, no. 1, pp. 1–17, 1993. 133

[111] A.-R. Sharafat and O. Marouzi, "All-terminal network reliability using recursive truncation algorithm," *Reliability, IEEE Transactions on*, vol. 58, no. 2, pp. 338–347, 2009. 133

[112] A. S. Rodionov, O. K. Rodionova, and H. Choo, "Using meaning of coefficients of the reliability polynomial for their faster calculation," in *ICCSA (2)*, pp. 557–571, 2009. 133

[113] G. Phillips, *Interpolation and Approximation by Polynomials.* CMS Books in Mathematics, Springer, 2003. 135

[114] E. Isaacson and H. Keller, *Analysis of numerical methods.* Dover Books on Mathematics Series, Dover Publications, Incorporated, 1994. 135

[115] R. Vein and P. Dale, *Determinants and Their Applications in Mathematical Physics.* No. v. 134 in Determinants and their applications in mathematical physics, Springer, 1998. 140