Defense against epidemics in networks

Paul Beaujean

▶ **To cite this version:**

Paul Beaujean. Defense against epidemics in networks. Cryptography and Security [cs.CR]. Université Paris sciences et lettres, 2019. English. ⟨NNT : 2019PSLED063⟩. ⟨tel-03222096⟩

HAL Id: tel-03222096

https://theses.hal.science/tel-03222096v1

Submitted on 10 May 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à Université Paris-Dauphine

Défense contre les épidémies dans les réseaux

Soutenue par
Paul Beaujean
Le 16 décembre 2019

École doctorale n°543
École doctorale de Dauphine

Spécialité
Informatique

Composition du jury :

Cristina Bazgan Professeur des universités LAMSADE, Université Paris-Dauphine	*Directrice de thèse*
Cédric Bentz Maître de conférences HDR CEDRIC, CNAM	*Rapporteur*
David Coudert Directeur de recherche INRIA I3S, Université Nice Sophia Antipolis	*Président du jury*
Stéphane Gaubert Directeur de recherche INRIA CMAP, École Polytechnique	*Rapporteur*
Éric Gourdin Ingénieur de recherche Orange Labs Châtillon	*Co-encadrant*
Frédéric Roupin Professeur des universités LIPN, Université Paris 13	*Examinateur*

Dauphine | PSL UNIVERSITÉ PARIS

To my grandparents.

Acknowledgements

In retrospect, I am not sure I remember when I started toying with the idea of pursuing a PhD in computer science. My earliest memory related to the discipline probably goes back to the difficult period of my *classes préparatoires* years as I was about to abandon my original interest in organic chemistry. What felt so different about this subject, I could not put it in words at the time. Looking back, I can only imagine that my younger mind was struggling with the concept that molecules are essentially combinatorial objects.

During a chemistry class, students were invited to experiment with model kits which depicted atoms as colorful spheres which could be connected with tubular links. A powerful image comes to mind of my classmates helping me assemble a replica of a carbon fullerene molecule, the so-called C_{60}. I had probably learned about it in a popular science magazine and was quite taken with the idea that the space of configurations constrained by physical laws would allow for the existence of such a beautiful and symmetrical object. In other words, the fullerene model looked like a football ball.

Probably disappointed to hear about the most likely career prospects in organic chemistry and eager to learn about a new field that some of my friends were studying at the time, I haphazardly decided to apply to an engineering school in which computer science would be the primary subject. At the time, my younger self had the very vague and mistaken idea that computer science was the study of programming languages. Instead, what I discovered at ENSIIE was an intriguing mix of two complementary topics: one focused on logic, types, and programming languages, and the other focused on algorithms and optimization. My lecturers at the time, Prof. Catherine Dubois, Prof. Xavier Urbain, Prof. Sourour Elloumi, Prof. Alain Billonnet, and many others opened the doors for me of a field that seemed to have a dizzying amount of depth.

It took me a long time to cast away my original impression of computer science, but Prof. Liljana Trajković, who kindly allowed me to spend an internship in her Communications Networks Laboratory at Simon Fraser University, encouraged me to read research papers and to question them, even as I had a very limited understanding of networking at the time. Slowly, the underlying abstraction that was growing inside my mind managed to take shape. What is it that makes some combinatorial structures “good”? Something that lurked

behind as I read about the power-law of the Internet topology, as I discovered with surprise that the fullerene graph had been studied by mathematicians, in essence I was discovering that the mathematics of networks indeed existed.

After an awe-inspiring experience with the topology of supercomputers during an internship with Matthieu Hautreux at CEA, I started to come to terms with the idea that networks themselves could be the subject of optimization, instead of the usual paths, tours, and cliques. Fortunately, the MPRO Master's degree program allowed me to become more familiar with the tools necessary to achieve my goal: make networks "better". It was via its curriculum as well as through my cultivated, and now ingrained, habit of reading research papers that I slowly assimilated the strictness required by the field of computational complexity. I would like in particular to thank Prof. Xavier Urbain who taught an introductory course at ENSIIE on the theory of computing which had a profound impact on my mathematical inclinations. With this newfound obsession for efficiency, two lecturers in the MPRO program Cédric Bentz and Frédéric Roupin, who I am honored to have in my committee, showed me that even hard computational problems could be studied under the perspective of efficient algorithms.

I cannot thank Éric Gourdin enough for offering me an internship at Orange Labs. A memorable part of the MPRO program consisted in regular visits from research engineers and other professionals who used optimization, or more generally methods from operations research, to solve real life problems. In one way, I have a strong personal preference for abstraction and purity, but at the same time I observe that my interest in mathematical tools and frameworks seems to originate from a desire to have some kind of tangible impact on the real world. It may be that computer science itself is afflicted with this contradictory impulse. The several speakers who presented their work at the MPRO industry sessions never failed to describe this ambivalence, even if in hindsight I realize I might have projected it onto the stories they told.

I discovered at Orange Labs a large group of scientists doing the kind of networking research I had read about a few years before. Above anything else, the people I met at Orange made me feel welcome, and encouraged me in my own research, especially on days when doubt and feelings of insecurity were waiting next to scribbled pages of failed attempts and disappointing leads. I would like to thank in particular Nabil and Adam who helped turn the inexperienced intern I was into a PhD student. For their continued support and caring advice I want to thank Alain, Philippe, Christian, Nancy, Yannick, Raluca, Amal, and Bruno. However, the members of the Traffic and Resource Management research group who shared the most of my struggles were my fellow PhD students, interns, and apprentice. I am very grateful to having

been able to have such amazing conversations with Yassine, Thibaut L., Guanglei, Mathieu, Léonce, Felipe, Claudio, Nicaise, Ahlam, Wesley, Vincent, Iaad, Paul, and Thibaut C. but also Wuyang, Thomas, Zaid, An, Yasmine, Valentin, Abdelhakim, and last but not least Bobby.

My time at Orange was composed of a multitude of meaningful experiences. From learning about the intricacies of a large multinational corporation from the perspective of a research department to interacting with networking professionals and promoting the use of optimization during a research fair, all of this while mentoring a younger student and seeing him transform into a full-fledged developer. I thank Éric again for this opportunity which at the same time grounded my research into a real community, not only connected by technical interests, but also wove it into a network of trust and friendship.

Unlike many PhD students, the first time I met my PhD advisor Cristina Bazgan was after having decided on my research topic. The CIFRE process started as I was an intern at Orange Labs which led to the unusual, in the French higher education system, situation where a student with research funding is looking for an advisor. I have always associated the origin of this PhD thesis as a combination of research papers. My attraction to semidefinite optimization came from the famous paper of Goemans and Williamson on the maximum cut problem [GW95] which Frédéric Roupin presented during his lecture. I associate the notion of trying to close the approximation gap with the clear depiction of the multiflow-multicut gap that was the focus of Cédric Bentz's lecture and which I would later remember via the paper by Feige et al. on approximating the minimum vertex separator problem [FHL08]. It is then not surprising that my first encounter with Cristina was through a paper she had co-authored with her PhD student Morgan Chopin on the subject of maximizing the spread of influence in networks [Baz+14].

I cannot count the number of times Cristina has bent the rules in my favor, nor the hours she has spent listening to my misguided ideas, incomplete proofs, and other overly ambitious projects. She has always supported me in my research and has been an extremely forgiving advisor in accepting the very modest speed of my progress. She has also given me a clear window onto the arcane world of the French higher education system and been an excellent colleague to discuss research news with. The only regret I have in the time I have been advised by Cristina is that I did not manage to match my initial goal of proving a lower bound on the maximum spectral subgraph problem, which, I believe, would have been the best way for me to repay her kindness.

As a holder of a CIFRE contract, I have spent a lot less time at my home university than my fellow PhD students have. This relatively short period was a way for me to discover through my peers the wider lens of computer

science research. The LAMSADE laboratory was for me a place to learn about academic research as it was being done at the moment. I am glad for the moments I have shared with many of its members, be it during the annual laboratory conference, the weekly PhD talks I managed to attend, and the innumerable cups of coffee that were shared on the 6^{th} floor. I would like to thank in particular Marie-Jo, Tristan, Denis, Laurent, Eun Jung, Michael, Rida, Ridha, Vangelis, Florian, Sonia, Alexis, Daniel, Florian, and Olivier for having been supportive of my research and to some of them for having introduced me to very interesting research problems. Among my fellow PhD students, I extend my warm thanks to Thomas, Pierre, George, Satya, Sami, Fabien, Ioannis, Justin, Ian, Meriem, Khalil, Mehdi, Anne, and Anaëlle for all the conversations we shared and I hope that our paths will cross again in the future.

I would like to extend special thanks to the members of my PhD committee for accepting to review my work. In particular, I am extremely grateful to my examiners Cédric Bentz and Stéphane Gaubert for the impressive depth of their reviews. I would also like to thank David Coudert for having shown interest in my research during the AERES evaluation of the LAMSADE laboratory and for the interesting conversation that we had in front of my poster. I thank Frédéric Roupin for having ignited my love of spectral methods by introducing me to semidefinite programming.

Finally, none of this would have been possible without the help of my family, the moral support of my friends, and the extreme resilience of my girlfriend, who never stopped being there for me while understanding how much time I needed to spend by myself. A tragic coincidence makes me relate the PhD degree with the death of a grandparent. As a child, my maternal grandmother passed away which contributed to my mother giving up on finishing her PhD thesis. Both of my paternal grandparents passed away during my PhD, unfortunately before I was able to show them the fruits of my research. For this reason, I would like to dedicate this manuscript to my grandparents. Thanks to two of my high school friends who have just become or will very soon become fathers, I will be able to associate the PhD experience with happier circumstances as well.

List of symbols

Symbol	Description
t	time or time step
$S, S(t)$	set or number of susceptible individuals in SIS models
$I, I(t)$	set or number of infected individuals in SIS models
β	birth rate of the SIS models
δ	death rate of the SIS models
ε	a small positive constant
$G = (V, E)$	an undirected graph
$H = (V, E')$	a partial subgraph of G
V	node set of G
E	edge set of G
$\Gamma(v)$	set of nodes adjacent to node v
$\deg(v)$	degree or number of nodes adjacent to node v
$\bar{d}(G)$	average degree of nodes in G
$\Delta(G)$	maximum degree in G
$\nu(G)$	matching number of G
A	adjacency matrix
D	degree matrix
L	Laplacian matrix
$\mathrm{Spec}(M)$	spectrum or set of eigenvalues of M
$\lambda_i(M)$	the i^{th} eigenvalue of matrix M
$\lambda_{\max}(H)$	λ_1, the largest eigenvalue of Hermitian matrix H
$\lambda_{\min}(H)$	λ_n, the smallest eigenvalue of Hermitian matrix H
e_i	the eigenvector associated with λ_i
$\mathbf{1}$	the all-ones vector
$\mathbf{0}$	the zero vector

Symbol	Description
x	a scalar variable
$\mathrm{Ber}(p)$	a Bernoulli distribution of mean p
$\boldsymbol{x} \sim D$	$\boldsymbol{x}$ is a random variable drawn from distribution D
$\boldsymbol{x}$	a scalar random variable
$G(n, p)$	the Erdös-Rényi-Gilbert random graph model
$\mathbf{G}$	a random undirected graph
M	a matrix
$M^\intercal$	the transpose of M
M^*	the conjugate transpose of M
I	the identity matrix
E_{ij}	an element of the standard basis of $M_{n,n}$
$\boldsymbol{M}$	a matrix-valued random variable
$A \succeq 0$	A is positive semi-definite
$A \succ 0$	A is positive definite
$A \succeq B$	$A - B$ is positive semi-definite
$A \odot B$	a different multiplication operator for positive definite matrices
f	some function
f_σ	the spectral mapping of f
$\operatorname{tr} M$	the trace of M
$\mathbb{E}$	the expectation operator
$\mathrm{Var}(\boldsymbol{x})$	the variance of $\boldsymbol{x}$
$v(\boldsymbol{X})$	the matrix variance of $\boldsymbol{X}$

Symbol	Description
$\mathbb{N}, \mathbb{Z}, \mathbb{R}$	natural numbers, integers, real numbers
$\mathbb{B}$	Boolean values
$M_{k,l}$	k by l matrices
$\mathbb{S}_n$	symmetric matrices
$\mathbb{S}_n^+$	positive semi-definite matrices
$\mathbb{S}_n^{++}$	positive definite matrices
P	decision problems solvable in polynomial time
NP	decision problems verifiable in polynomial time
PSPACE	decision problems solvable with polynomial memory space
EXP	decision problems solvable in exponential time
BPP	decision problems solvable in polynomial time with probability at least $2/3$
PCP	family of decision problems verifiable from a bounded number of queries with some probability
PO	optimization problems solvable in polynomial time
FPTAS	optimization problems solvable to any desired precision in polynomial time
APX	optimization problems solvable with a constant factor approximation in polynomial time

Contents

Acknowledgements 3

List of symbols 9

Contents 13

Résumé de la thèse en français 19

Introduction 29

1 Epidemics 33
- 1.1 A brief history of epidemic models . . . 34
- 1.2 The SIS model . . . 35
  - 1.2.1 Definition . . . 35
  - 1.2.2 Solving the differential equation . . . 36
  - 1.2.3 Fast growth . . . 38
  - 1.2.4 Slow extinction . . . 38
  - 1.2.5 Fast extinction . . . 39
  - 1.2.6 Convergence modes of the SIS model . . . 40
- 1.3 The networked SIS model . . . 42
  - 1.3.1 Definition . . . 42
  - 1.3.2 Eventual convergence . . . 43
  - 1.3.3 Asymptotic stability . . . 43
  - 1.3.4 Exponential convergence . . . 49

2 Algorithms 55
- 2.1 Problems and solutions . . . 57
  - 2.1.1 Decision problems . . . 57
  - 2.1.2 Optimization problems . . . 58
  - 2.1.3 Reductions . . . 60
  - 2.1.4 Verifiers and proofs . . . 62
- 2.2 Complexity and efficiency . . . 63
  - 2.2.1 **P**: deciding correctly in polynomial time . . . 64
  - 2.2.2 **NP**: efficient verification with short proofs . . . 66

2.2.3 **BPP**: probabilistic correctness . . . 68
2.3 Approximation . . . 70
2.3.1 Worst case *a priori* guarantees . . . 71
2.3.2 Probabilistic feasibility . . . 74
2.3.3 Relaxation and rounding . . . 75
2.4 Critical thresholds in hardness of approximation . . . 77
2.4.1 A varied landscape . . . 78
2.4.2 The unique coverage problem . . . 79
2.4.3 A randomized $O(\log m)$-approximation algorithm . . . 81
2.4.4 Hardness of approximating unique coverage . . . 84

3 Random graphs **89**
3.1 The concentration of measure phenomenon . . . 90
3.2 Concentration inequalities . . . 91
3.2.1 The classical Chernoff bound . . . 91
3.2.2 The matrix Chernoff bound . . . 94
3.3 An interlude of spectral graph theory . . . 98
3.3.1 The adjacency matrix . . . 98
3.3.2 The Laplacian matrix . . . 99
3.3.3 Algebraic connectivity . . . 100
3.4 The $G(n, p)$ random graph model . . . 102
3.4.1 Properties of random graphs . . . 103
3.5 The connectivity threshold . . . 104
3.5.1 The Laplacian matrix of a random graph . . . 105
3.5.2 Shifting the spectrum of a Laplacian matrix . . . 106
3.5.3 Bounds on every summand . . . 107
3.5.4 Smallest eigenvalue of the expected matrix . . . 108
3.5.5 Applying the matrix Chernoff bound . . . 108

4 Intermission **113**
4.1 The advent of controllable networks . . . 113
4.1.1 Systems as abstraction . . . 113
4.1.2 The case of network resources . . . 114
4.1.3 Network security . . . 115
4.2 Epidemic models for network security . . . 116
4.2.1 Modeling computer viruses and propagating threats . . . 116
4.2.2 Learning model parameters . . . 117
4.3 Reactive countermeasures to threats . . . 117
4.3.1 Deploying security appliances . . . 118
4.3.2 Topology reconfiguration . . . 118

4.4 A novel network security framework . . . 119
4.4.1 Finding a temporary topology reconfiguration . . . 119
4.4.2 Contributions . . . 120

5 Approximating the maximum spectral subgraph 123
5.1 Introduction . . . 123
5.1.1 Context . . . 123
5.1.2 Related work . . . 124
5.2 Preliminaries . . . 126
5.2.1 Computational complexity . . . 126
5.3 Relaxation and matrix randomized rounding . . . 127
5.3.1 The case of star graphs . . . 129
5.3.2 Erdös-Rényi stars . . . 131
5.3.3 Without the degree constraint . . . 132
5.4 Spectral subgraphs in general graphs . . . 133
5.4.1 Following the matrix Bernstein bound . . . 134
5.4.2 Proof of Theorem B . . . 135
5.5 Maximum matching . . . 137
5.6 Independent rounding: an intrinsic $\Omega(\log n)$ barrier . . . 139
5.6.1 Cliques and the $G(n, p)$ random graph model . . . 139
5.6.2 Proof of Theorem D . . . 140
5.7 Conclusion and perspectives . . . 142

6 Experimental design for randomized approximation algorithms 147
6.1 Introduction . . . 147
6.2 The scientific method in experimental algorithmics . . . 148
6.3 Algorithms as experimental subjects . . . 149
6.3.1 Performance metrics . . . 150
6.3.2 Datasets . . . 150
6.3.3 Random samples and randomized experiments . . . 151
6.4 Factors of influence . . . 152
6.4.1 Problem instances . . . 152
6.4.2 Implementation of algorithms . . . 154
6.5 Towards systematic parameter setting . . . 156
6.5.1 Gaussian process regression . . . 157
6.5.2 Libraries . . . 158
6.6 Issues in experimental evaluation of algorithms . . . 159
6.6.1 Qualitative differences . . . 159
6.6.2 Practical implementation details . . . 160

6.7 Experiments 162
6.7.1 Pilot experiments 162
6.7.2 Experimental setup 169
6.8 Pre-registration 171

Conclusion **175**

Bibliography **179**

Résumé de la thèse en français

L'époque contemporaine est marquée par l'intensification des moyens de communication, en particulier à travers l'émergence et la multiplication des réseaux informatiques. De même que les sociétés humaines se sont transformées à mesure que leurs réseaux de routes terrestres, de navigation, et de transport aérien se sont développés, les informations qui transitent dans les réseaux informatiques sont devenues de plus en plus complexe.

L'événement déterminant dans cette complexification des informations est sans doute le transfert de programmes auto-répliquants qui se propagent à travers l'ensemble d'un réseau informatique. Des logiciels malveillants conçus sur ce modèle peuvent non seulement compromettre l'intégrité d'un système informatique mais aussi se propager à d'autres systèmes qui y sont connectés, à la manière d'une épidémie.

Cette analogie est à l'origine d'un effort de recherche dans le domaine de la sécurité des réseaux informatiques pour se prémunir des logiciels malveillants en appliquant des stratégies similaires à celles déjà utilisées dans l'épidémiologie humaine, comme la mise en quarantaine ou la distribution de vaccins.

Dans le même temps, les réseaux informatiques sont devenus de plus en plus flexibles. Si les premiers réseaux de télécommunication étaient semblables à des routes ou des fleuves dont le tracé perdure pendant des années ou des siècles, les réseaux actuels sont maintenant reconfigurables à la volée de manière à satisfaire un besoin de connectivité particulier : une ligne d'urgence pour une équipe de pompiers, une ligne de messagerie chiffrée entre journalistes d'investigation, ou encore un réseau de capteurs météorologiques.

Le paradigme de gestion des réseaux à l'origine de cette nouvelle flexibilité est nommé *software defined networking* ou réseau défini par logiciel en français. Les réseaux sont ainsi commandés par des programmes qui peuvent réagir à différents événements comme l'accroissement du trafic, le changement de l'état des équipements réseaux, ou encore la connexion de nouveaux éléments au réseau. Il est donc possible de concevoir, quasiment en temps réel, la topologie d'un réseau pour parvenir à un but précis.

Bien que de nombreux travaux aient exploré la possibilité de "vacciner" les éléments d'un réseau pour contrer la propagation d'une épidémie de logiciels malveillants, la question de reconfiguration du réseau comme outil de défense

contre les épidémies n'a pas été explorée en profondeur.

Dans cette thèse, nous étudions la possibilité de calculer une reconfiguration pour n'importe quel réseau et n'importe quelle épidémie donnés qui entraînerait l'extinction de l'épidémie en un temps limité. En faisant appel à des résultats en épidémiologie mathématique, en algorithmique, et en probabilités, nous proposons un algorithme d'approximation aléatoire pour lequel nous montrons qu'il est efficace et qu'il garantit des solutions de bonne qualité.

Pour contextualiser l'utilisation de cet algorithme, nous commençons par rappeler des résultats connus en épidémiologie qui décrivent l'évolution d'un processus propagatif à travers un graphe non-orienté. Ensuite nous nous plaçons dans le cadre de la théorie de la complexité pour définir précisément le sens d'algorithme efficace et d'approximation dans le pire des cas. Enfin, nous adaptons des résultats récents en probabilités qui décrivent le comportement des valeurs propres extrêmes de matrices aléatoires.

Modèles épidémiologiques

De la même manière que les physiciens ont souhaité représenter l'écoulement d'un fluide à travers le formalisme des équations différentielles et en invoquant l'abstraction de la particule de fluide, les épidémiologues se sont focalisés sur une simplification des maladies en un système dans lequel les individus d'une population peuvent basculer d'un état à un autre parmi un nombre fixe d'états. Certains de ces états représentent la contagion où un individu malade infecte une fraction du reste de la population.

La simplification la plus grande étant la propagation d'une maladie à deux états : S pour "sain" et I pour "infecté". Motivés par des études similaires dans le domaine de l'écologie et en particulier les modèles de proies et prédateurs dont les populations sont interdépendantes, les épidémiologues développèrent dans la première moitié du XXème siècle des modèles de plus en plus précis capturant les différents stades d'une infection dont certaines phases peuvent être plus contagieuses que d'autres.

Toutefois, cet effort de modélisation se heurta à la réalité d'un changement profond des modes de communication. En effet, si les épidémies se propageaient majoritairement à une échelle locale, géographique, jusqu'au milieu du XXème siècle, l'intensification des déplacements des humains et des marchandises avec l'avènement du tourisme de masse et du commerce mondialisé a rendu caduc l'hypothèse implicite dans la majorité des modèles existants qu'un individu peut entrer en contact avec d'autres de manière équiprobable.

En réalité, la façon dont les individus d'une population humaine sont en

contact les uns avec les autres se rapproche davantage de la vision topologique de la théorie des graphes que de celle géométrique de la géographie. Le début du XXIème siècle a ainsi vu le développement de nouveaux modèles épidémiologiques qui ne présupposent aucun réseau social particulier entre les individus d'une population. Ces modèles sont donc génériques et applicables à des situations précises dans lesquels les liens de relations entre individus sont au moins aussi cruciaux que les caractéristiques de l'agent contagieux dans l'étude de la dynamique de l'épidémie.

Pour illustrer cette axe de recherche, nous nous arrêtons sur un résultat de Prakash et al. [Pra+11] qui décrit les dynamiques possibles d'un modèle épidémiologique à états qui peut se propager entre les nœuds d'un graphe s'ils sont connectés par une arête.

Ce théorème détermine que la trajectoire de l'épidémie est décidée par la comparaison entre deux quantités, une dépendant des caractéristiques intrinsèques de la maladie, l'autre dépendant des propriétés du graphe représentant le réseau social par lequel la maladie se propage. De manière imagée, on peut se représenter l'agent infectieux comme ayant une vitesse minimale de survie d'un côté et le réseau comme ayant une limitation de vitesse de l'autre. Quand la limite est plus basse que la vitesse minimale de la maladie, l'épidémie s'arrête d'elle-même. Dans le cas contraire, elle continuera.

Le problème qui se pose donc est de savoir si l'on peut modifier le réseau de manière à abaisser la limite juste en dessous de la vitesse minimale de survie de l'épidémie de manière à forcer son extinction naturelle.

Modèles de calcul

Les programmes informatiques sont des artefacts complexes qui dépendent de nombreux détails techniques tels que les langages dans lesquels ils ont été écrits, les composants matériels des ordinateurs qui les exécutent, mais aussi de tout un écosystème d'autres programmes dont le système d'exploitation et les environnements d'exécution font partie.

Pour pouvoir réfléchir à un programme informatique sans s'encombrer de ces détails, il est possible de faire appel à des modèles mathématiques qui offrent des abstractions utiles et dont on espère qu'elles se rapprochent de la réalité.

L'algorithmique est le domaine de l'étude des algorithmes au sein de ces modèles de calcul. Cette discipline théorique est à l'origine de nombreuses découvertes importantes qui sont liées à des changements profonds dans l'histoire des fondements des mathématiques, de la logique et de l'informatique.

Une de ces découvertes est l'étude formelle de la complexité en termes de problèmes liés les uns aux autres ainsi qu'aux méthodes de calcul, si elles existent, qui tentent d'en donner des solutions génériques.

Depuis le milieu du XXème siècle, des dizaines de milliers de problèmes ont été étudiés et mis en relation les uns avec les autres. Au fur et à mesure des avancées de la discipline, une même observation s'est peu à peu imposée à la communauté scientifique : certains problèmes sont faciles à résoudre, c'est-à-dire qu'il est possible de trouver un algorithme rapide (dont le nombre d'étapes de calcul ne croît que polynomialement avec les données du problème) pour calculer une réponse correcte. Pour un plus grand nombre de problèmes, les études successives n'ont réussi qu'à prouver qu'il était facile de vérifier une réponse présentée. De manière évidente, les problèmes faciles à résoudre sont faciles à vérifier.

Toutefois il existe certains problèmes dont on sait que si on avait un algorithme efficace pour les résoudre, alors on pourrait résoudre efficacement tous les problèmes faciles à vérifier. Depuis les années 1970, aucun de ces problèmes difficiles, dans le sens de "au moins aussi difficile que la classe des problèmes faciles à vérifier", n'a été résolu efficacement. En effet, malgré les efforts de la communauté scientifique en algorithmique, ces problèmes difficiles paraissent résister et invitent à la conjecture suivante : il existe une séparation entre problèmes faciles à résoudre et problèmes difficiles.

Du point de vue de la conception de programmes informatiques, il s'agit donc de comprendre à quel point il est possible de résoudre un problème donné et ensuite de concevoir un algorithme qui s'approche le plus possible de ces limites théoriques. Si l'on donne du poids à la conjecture de séparation de la complexité des problèmes, on admet qu'il n'est pas possible de résoudre efficacement certains d'entre eux.

Parmi les problèmes étudiés par les théoriciens, on trouve des problèmes dont le but est de trouver un élément, meilleur selon un critère, parmi un ensemble. Ce formalisme rejoint une tradition mathématique plus ancienne, l'optimisation qui s'intéresse aux objets mathématiques extrêmes.

Ainsi, si on établit qu'un problème d'optimisation est difficile, la conjecture nous invite à ne pas chercher d'algorithme pour le résoudre efficacement. En effet, si la conjecture de séparation de la complexité est vraie, il n'en existe pas. Toutefois, il est légitime d'aller explorer les limites de ce qu'il est possible de calculer efficacement. Dans le cadre de l'optimisation, il est intéressant par exemple de chercher à calculer efficacement une solution de bonne qualité au lieu d'une solution qui serait la meilleure possible. Cette tâche, potentiellement moins difficile, pourrait admettre un algorithme efficace.

Une autre manière de contourner la difficulté d'un problème peut être

d'élargir la définition d'un algorithme de résolution. Par exemple, au lieu de s'intéresser à des algorithmes qui donnent à chaque exécution une réponse correcte, on peut se contenter d'algorithmes qui donnent souvent une réponse correcte, en imposant au moins une certaine fréquence de réussite.

Un résultat de Van Mieghem et al. [Van+11] nous donne la preuve que le problème de reconfiguration est difficile dans le sens de la théorie de la complexité.

Graphes aléatoires

La reconfiguration d'un réseau est un processus complexe. Dans notre contexte de défense contre les épidémies, il s'agit d'abaisser le rayon spectral qui est un paramètre global d'un graphe particulier. De plus, cette quantité est non-linéaire car elle dépend de toutes les arêtes du graphe à la fois. En représentant le fait que chaque arête du réseau peut être potentiellement désactivée par une variable booléenne (vrai/faux), le rayon spectral se retrouve être un polynôme en toutes les variables des arêtes du graphe.

Comment faire pour construire un sous-graphe partiel, c'est-à-dire désactiver des arêtes, de manière à abaisser le rayon spectral ? Pour des raisons géométriques, une variante de ce problème peut être résolue efficacement. En effet si on considère des variables dont les valeurs sont comprises entre 0 et 1, la question s'apparente à un problème d'optimisation dit convexe de type conique dans lequel le cône en question est un ensemble de matrices symétriques, objets mathématiques qui représentent des transformations linéaires d'espaces géométriques. Le problème devient de concevoir une matrice symétrique dont le rayon spectral est borné.

En résolvant cette généralisation du problème combinatoire qui nous intéresse en problème géométrique, on obtient des arêtes partiellement désactivées, c'est-à-dire un graphe pondéré par des valeurs comprises entre 0 et 1. Il ne faut qu'un pas pour interpréter ces valeurs comme des fréquences, voire des probabilités.

Ainsi on peut faire l'usage de tirages au sort pour déterminer si chaque arête doit être conservée ou désactivée. En effectuant ces tirages, on espère obtenir un sous-graphe dont le rayon spectral sera similaire à celui du graphe pondéré.

Pour cela, nous nous inspirons de résultats récents en théorie des probabilités qui décrivent à quel point un graphe aléatoire (obtenu par tirage) est proche du graphe moyen à partir duquel il a été généré. Ici, nous souhaitons savoir si ces deux graphes sont proches en terme de rayon spectral.

Ces considérations multiples nous fournissent des ingrédients pour con-

cevoir un algorithme efficace en deux étapes : d'abord résoudre une version généralisée du problème et ensuite tirer au sort des sous-graphes candidats à partir de la solution précédente.

De la concentration de la mesure à l'algorithme

En théorie des probabilités, la concentration de la mesure désigne un phénomène commun qui se présente pour des objets aléatoires constitués d'un grand nombre de variables aléatoires indépendantes. Plus la dimension de l'objet augmente, plus les fluctuations aléatoires semblent diminuer pour ne conserver qu'un mince halo autour d'un objet moyen, souvent facile à caractériser. L'ajout de variables indépendantes rend particulièrement improbable de grandes déviations par rapport à une tendance typique. C'est ce principe étudié tout au long du XXème siècle que nous nous proposons de mettre à profit pour concevoir un algorithme aléatoire.

La contribution théorique principale de cette thèse est la preuve de la concentration de la mesure pour les sous-graphes générés à partir d'une solution au problème généralisé. En particulier, nous prouvons que l'écart entre le graphe moyen et ses sous-graphes du point de vue du rayon spectral est, dans le pire des cas, un facteur logarithmique de la taille de la population. Toutefois nous n'obtenons ce résultat que dans le cas où la limite de vitesse est au moins le logarithme de la taille de la population.

Nous pouvons dériver de ce phénomène de concentration une méthode de calcul simple d'un sous-graphe dans le cas où la limite est au moins logarithmique.

1. calculer le graphe moyen par la résolution du problème géométrique
2. diminuer d'un facteur logarithmique les probabilités de conserver chaque arête
3. tirer au sort chaque arête selon les probabilités calculées précédemment
4. vérifier le rayon spectral du sous-graphe obtenu

La première étape de calcul peut être réalisée de manière efficace en utilisant des résultats classiques dans le domaine de l'optimisation. La deuxième étape ne consiste qu'en une liste de divisions à effectuer. La troisième étape peut être réalisée efficacement par l'utilisation de résultats classiques dans le domaine des générateurs de nombres aléatoires. Enfin la quatrième et dernière étape ne

nécessite que d'une primitive d'algèbre linéaire pour obtenir la plus grande valeur propre d'une matrice associée au graphe.

Une analyse de cette algorithme nous montre que les sous-graphes obtenus par ce calcul sont dans le pire des cas plus petit d'un facteur logarithmique qu'une solution optimale. L'utilisation de l'aléatoire nous contraint aussi à ne pouvoir retourner une solution correcte que dans la très grande majorité des cas. En pratique cela correspond à recommencer les étapes 3 et 4 jusqu'à ce qu'un sous-graphe correct soit produit, la probabilité d'échec décroissant rapidement avec le nombre d'essais.

Le cas restant de la limite de vitesse basse (moins qu'un logarithme) peut être résolu par un algorithme existant que nous ne détaillerons pas ici et qui produit une solution d'une taille plus petite dans le pire des cas d'un facteur polylogarithmique. En conclusion, en combinant ces deux algorithmes nous obtenons un algorithme efficace qui garantit au pire un facteur polylogarithmique de perte par rapport à une solution optimale.

L'algorithme que nous proposons est donc simple, utilise de méthodes de calcul préexistantes, et possède des garanties théoriques valables pour n'importe quelles données d'entrée. Qu'en est-il néanmoins de son utilité pratique ?

De l'algorithme au programme

L'algorithme tel que décrit dans la théorie de la complexité et compris à travers les modèles de calcul est une simplification importante de la réalité de l'exécution d'un programme.

En particulier, l'algorithme que nous avons proposé, bien que simple en théorie, ne peut pas être directement utilisé tel quel en pratique. Par exemple, le facteur logarithmique de correction du à un cas extrême de concentration de la mesure n'est pas nécessaire pour tous les graphes donnés en entrée. Il est donc raisonnable d'étudier l'impact de modifications pratiques de cette algorithme pour l'adapter à un cas d'usage réaliste.

De la même manière, les étapes 1 et 4 font appel à des routines de calcul préexistantes mais dont le comportement est paramétrable par de nombreuses options comme la précision voulue des résultats ou encore le nombre maximal d'itérations de leurs sous-routines. Des tests préliminaires montrent par exemple que selon l'homogénéité du nombre de voisins par nœud dans le graphe, la résolution de l'étape 1 peut se faire de manière plus ou moins grossière, la perte en précision n'ayant qu'un impact minime au vu des gains de temps de calcul conséquents obtenus.

Notre contribution n'étant qu'un exemple simple d'un algorithme aléatoire

d'optimisation à garantie de performance, nous proposons une méthode expérimentale appropriée à l'étude d'un ensemble de paramètres et de variantes algorithmiques comme facteurs de performances en termes de temps de calcul, de qualité de solution, et de gestion de l'aléatoire. Cette méthode se veut adaptée à l'étude expérimentale des algorithmes d'optimisations et des métaheuristiques, bien que nous la spécialisions à la prise en compte des garanties de performance a priori.

Enfin, nous proposons des perspectives de recherche futures qui portent sur des problèmes connexes mettant en lien les propriétés spectrales (liées à l'algèbre linéaire) des graphes. Nous espérons ainsi que l'approche théorique ainsi que la méthode expérimentale proposés puissent être applicables de manière plus générale.

Introduction

The study of defending against epidemics is a foundational theme of mathematical epidemiology. In this thesis, we consider an idealized version of this problem from the perspective of graph theory and attempt to design an effective solution in the form of an efficient algorithm.

Our main motivation comes from the flexibility provided by software-defined networking, a set of technological advances which allows for a fast and centralized control of a network, to the point of being able to reconfigure its topology dynamically.

The combinatorial optimization problem we consider is related to spectral graph theory, the branch of mathematics which studies the relationship between graphs and their associated matrices. In particular, these matrices can be studied from the point of view of their spectrum, that is the set of their eigenvalues, which in turns provides new perspectives on diverse properties of graphs.

To facilitate the exposition of our work, we provide in Chapter 1 a short overview of classical results in mathematical epidemiology as well as a detailed review of newer findings on the dynamics of epidemics on graphs. In Chapter 2, we describe the theoretical framework commonly used in the theory of algorithms and computational complexity which is the setting in which we have searched for an efficient algorithm. Because the algorithm we have proposed relies heavily on the theory of high-dimensional random variables, we present in Chapter 3 classical results related to the concentration of measure phenomenon, which illustrates that a large number of random variables generally remain close to their average value. It is notable that this property remains true in the case of random matrices, which we illustrate with a detailed presentation of the connectivity threshold in random graphs.

A reader familiar with all three domains and who would be aware of the existence of critical phenomena that are commonly encountered in dynamical systems, in computational complexity, and in high-dimensional probability could assuredly focus their attention on Chapter 4 which describes the rationale for this work. We give a short overview of modern practices in networking and explain their relevance to network security. This allows us to introduce the maximum spectral subgraph problem which is a core routine of a reactive

network security framework we propose.

In Chapter 5, we use mathematical tools that were introduced in the first three chapters to design an efficient randomized approximation algorithm for the maximum spectral subgraph problem and provide guarantees on the quality of the solutions it returns. To complete the analysis of our algorithm, we exhibit a family of graphs for which the algorithm provides solutions that match the guarantee, establishing the tightness of our analysis. However, we note that this does not rule out the existence of better algorithms. Chapter 5 is based on a paper written in collaboration with Cristina Bazgan and Éric Gourdin [BBG18] presented at the 12$^{\text{th}}$ Internatial Conference on Combinatorial Optimization and Applications as well as its extended version [BBG20] which was accepted for publication in the Journal of Combinatorial Optimization.

The framework described in Chapter 2 provides a theoretical definition of efficiency. However, practical implementations of algorithms, even theoretically inefficient algorithms, can be fast in practice. In Chapter 6 we propose an experimental setup in which the practical efficiency of our algorithm could be assessed fairly. This experimental setup is a response to several issues in the application of the scientific method in experimental algorithmics. As a whole, this chapter is a pre-registration of experiments to come, which is a way to combat publication bias as well as a tool to restore the balance between methodology and results. Chapter 6 is based on a paper written in collaboration with Cristina Bazgan and Éric Gourdin [BBG] in preparation to be submitted to the International Symposium on Experimental Algorithms.

1 Epidemics

Introduction

A whydunit, a colloquial term for "why done it?" or "why has it been done?", is a detective story in which the focus is on the motives for committing a crime. In this thesis, the main motive is to leverage the effect of the contact graph on the dynamics of an epidemic. Indeed, the theme of this work is the study of propagating processes and how the knowledge of their dynamics can be used to control them indirectly [NPP16]. Although the term epidemiology brings to mind disciplines such as medicine, public health, or organizations such as the World Health Organization, we will exclusively consider the mathematical aspects of epidemiology whose roots can be traced back to statistical physics and to the study of differential equations or their discrete-time counterparts, difference equations.

This chapter starts with a short section 1.1 on the history of the applied mathematics used to model and understand epidemics together with some bibliographical references which provide the background for the mathematical tools that we employ. In section 1.2, we place ourselves at the beginning of the history of mathematical epidemiology and discuss the so-called SIS model which has been used to represent the evolution of a infectious disease from which individuals can recover. However, recovered individuals in the SIS model can contract the disease again, that is they do not acquire immunity. As the SIS model is described by an ordinary differential equation, it suffices to apply elementary calculus to obtain precise information on its dynamics. On the other hand, section 1.3 is concerned with contemporary developments of mathematical epidemiology. We keep a connection with the previous section by considering the networked SIS model which includes specific interaction patterns between individuals represented by a contact graph. The networked SIS model is then described as a discrete-time Markov chain which we analyze via an autonomous non-linear discrete dynamical system. This allows us to utilize results from the theory of dynamical systems to characterize the dynamics of this more complex epidemic model. In a last section, we summarize the results presented in this chapter and refer the reader to recent developments in the analysis of networked epidemic models.

In this chapter, we will work towards establishing the following result: in a network, the behavior of epidemics undergoes a phase transition between fast extinction and slow extinction. This phase transition occurs at a level determined jointly by the network topology and by the epidemic itself. For this purpose we will start by considering a simple epidemic model and gradually work towards more realistic models which highlight the importance of the underlying graph.

1.1 A brief history of epidemic models

At the beginning of 20th century, many burgeoning scientific fields embraced the traditional approaches and methodology of physics. In particular the life sciences strove to expand beyond qualitative theories that had been the norm until then. The main idea was that, by allowing for simplifying assumptions, one could model the behaviour of biological processes to the same degree of precision achieved in physics with e.g. fluid dynamics or electromagnetism. This can be considered as one of the first uses of applied mathematics outside of its traditional roots in engineering. Indeed in the field of ecology, the Lotka-Volterra equations have been used to model the evolving populations of two species in a predator-prey relationship, see for example Goell et al. [GMM71]. As an example of this underlying movement in the history of science, Lotka in 1910 originally published his equations [Lot10] in the context of the study of multiple autocatalytic chemical reactions i.e. system of chemical reactions in which some reactions produce helpers (catalysts) for other reactions. In the early 1920s, Lotka realized that the same differential equations could model the dynamics of natural populations of predators and preys as he focused his interests to mathematical biology, work then expanded upon by Volterra in [Vol28]. Mathematically speaking, the differential equations they studied were similar to the work done by Verhulst in the middle of the 19th on the logistic map which modeled population growth.

From the late 1920s to the 1950s, these models were extended to include intermediate states which could represent the different steps occuring during an infection: an individual could be healthy, incubating, contagious, immune, dead, or any number of conditions which would match the medical knowledge at the time. The so-called compartmental models pioneered by Kermack and McKendrick [KM27] can then be considered one of the first epidemic models, see for example the survey by Brauer that describes historical developments of these models [Bra17]. In section 1.2, we study the SIS model which is one of the most simple compartmental models and shares many similarities with the

previous Lotka-Volterra models.

So far, all models used differential equations as underlying mathematical objects which limited their applicability to small populations or infrequent interaction between individuals. Furthermore, while differential equations accurately matched medical records of known epidemics, the early phases of an epidemic were not as precisely described. Indeed, consider an epidemic outbreak which starts with a small number of infected individuals who go on to infect an entire population. This case was seen to be analogous to the Galton-Watson process, a stochastic process proposed by Galton in the late 19^{th} century to study the extinction or persistence of family names in England [WG75]. The proof of convergence of the Galton-Watson process was completed in the 1930s and spurred interest in generalization of this process called branching processes. It is much later in the 1980s that branching processes were considered as potential epidemic models, see for example the book by Diekmann and Heesterbeek [DH00].

Finally, in the early 2000s, concomitant with the advent of the Internet, epidemic models were generalized to include potentially varied patterns of interaction betweeen individuals. These developments relied on the work done in diverse branches of probability theory, statistical physics, as well as in random graph theory which describe critical phenomena [Sla+08] in a probabilistic way.

1.2 The SIS model

The SIS model is a classical example of a compartmental model which is very similar to the Lotka-Volterra model. However, despite its simplicity, this model contains the characteristic behavior of epidemics as critical phenomena: the outcomes of this system is entirely determined by the change of a parameter through a threshold. The prerequisites for this section are elementary knowledge in differential equations and familiarity with the logistic equation.

1.2.1 Definition

We now describe a simple model for a spreading epidemic in which individuals can be in one of two possible states: “S” for susceptible to be infected and “I” for infected. In this model, healthy individuals can become infected and subsequently recover to a healthy state in which they might catch the disease again in the following way. We denote respectively by S and I the number of individuals in each class, with $S + I = N$ where N is the total population.

Because we are interested in the evolution of such a system, we consider the aforementioned quantities as being functions of time $S(t), I(t)$ while N is considered to be constant. The SIS model is described by a system of differential equations with parameters β the rate of infection, and δ the rate of recovery. These parameters are also referred to as the birth rate β and the death rate δ, assuming the point of view of the epidemic itself.

$$\frac{dS}{dt} = -\beta SI + \delta I$$
$$\frac{dI}{dt} = +\beta SI - \delta I$$

together with an initial condition $I(0) = k$ for some $1 \leq k \leq N$.

This system of equations can be easily interpreted: the rate of infection β represents the fraction of sane individuals that would be contaminated by an infected individual while the rate of recovery δ which represents the fraction of the infected population that will recover. To obtain the number of newly infected individuals, we rely on the *full-mixing assumption* i.e. every individual is in contact with the entire population. This means that each infected individual generates βS new infections, for a total of βSI extra infections. On the other hand, the number of recovering individuals is simply δI. This gives us the following pictorial representation of the SIS model.

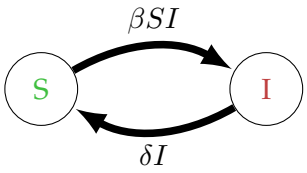

Figure 1.1: SIS model with parameters β and δ

We now wish to analyze the dynamics of this model to understand the importance of its parameters β and δ.

1.2.2 Solving the differential equation

We start by noticing that, since our system is *closed* i.e. the total population does not change $S + I = N$, we can limit ourselves to analyzing only one of the two compartments of the model. Replacing S with $N - I$ in the system of differential equations, we obtain a single differential equation describing the

dynamics of the infected population over time:

$$\frac{dI}{dt} = \beta(N - I)I - \delta I$$

which we rearrange into

$$\begin{aligned}
\frac{dI}{dt} &= \beta(N - I)I - \delta I \\
&= (\beta N - \delta)I - \beta I^2 \\
&= I((\beta N - \delta) - \beta I) \\
\frac{dI}{dt} &= (\beta N - \delta)I\left(1 - \frac{\beta}{\beta N - \delta}I\right).
\end{aligned}$$

In this form we immediately recognize a logistic differential equation:

$$\frac{df(x)}{dx} = af(x)\left(1 - \frac{f(x)}{b}\right) \tag{1.1}$$

with parameters $a = \beta N - \delta$ and $b = N - \delta/\beta$. The logistic differential equation belongs to the class of Bernoulli first-order ordinary differential equations which admit closed-form solutions.

The closed-form solution of the logistic differential equation is a logistic function depending on the initial value of the function $f_0 = f(0)$ and the two logistic parameters a and b:

$$f(x) = \frac{b}{1 + \left(\frac{b - f_0}{f_0}\right)e^{-ax}}. \tag{1.2}$$

We now return to our application domain with some appropriate renaming:

$f(x)$	$\Longleftrightarrow$ $I(t)$	infected population over time,
f_0	$\Longleftrightarrow$ I_0	initial infected population,
b	$\Longleftrightarrow$ $\hat{I}_\infty$	final endemic infected population,
a	$\Longleftrightarrow$ r	growth rate of the epidemic.

While the first two quantities are part of the description of the SIS model, we interpret the parameters of the logistic differential equation a and b in light of their role in the logistic function 1.2.

The growth rate of the epidemic r is the most important property of the SIS model. By definition we have $r = \beta N - \delta$ and since all three quantities are

positive i.e. $N \in \mathbb{N}_+, a, b \in \mathbb{R}_+$ then the growth rate is a real number which may be negative or positive i.e. $r \in \mathbb{R}$. To understand the importance of this quantity, let us consider the three possible cases:

$$r > 0 \iff N > \delta/\beta \tag{1.3}$$

$$r = 0 \iff N = \delta/\beta \tag{1.4}$$

$$r < 0 \iff N < \delta/\beta \tag{1.5}$$

First, recall that $r = \hat{I}_\infty \beta$ which means that r and $\hat{I}_\infty$ have the same sign. We will first consider the case 1.3 where the epidemic has a positive growth rate to discuss the limiting behavior of the infected population.

1.2.3 Fast growth

Assuming $r > 0$, we introduce the positive quantity $c = (\hat{I}_\infty - I_0)/I_0$ which lets us write $I(t)$ as a function of positive quantities:

$$I(t) = \frac{\hat{I}_\infty}{1 + ce^{-rt}}. \tag{1.6}$$

As the exponential quantity goes to 0 as time goes to infinity, we have that:

$$\lim_{t \to \infty} I(t) = \hat{I}_\infty \tag{1.7}$$

which justifies our symbol for the final endemic infected population.

1.2.4 Slow extinction

A peculiar situation is the case of the epidemic having a growth rate exactly equal to 0. One way to handle this case is to argue that this case is "unnatural" because real-life epidemics are unlikely to have their intrinsic properties β and δ such that $N = \delta/\beta$. The more rigorous way to handle this case is by simply returning to our initial differential equation for $I(t)$ when $r = 0$ that is $\beta N = \delta$.

In the case where $r = 0$ the dynamics of the infected population are given by:

$$\frac{dI(t)}{dt} = \beta N I - \beta I^2 - \delta I \tag{1.8}$$

Grouping the first and last expressions together we have:

$$\frac{dI(t)}{dt} = (\beta N - \delta)I - \beta I^2 \tag{1.9}$$

Which means that the infected population follows a much simpler differential equation:

$$\frac{dI(t)}{dt} = -\beta I^2. \tag{1.10}$$

We recognize a separable ordinary differential equation whose exact solution is:

$$I(t) = \frac{I_0}{1 + I_0\,\beta t}. \tag{1.11}$$

In this case of zero growth rate of the epidemic, we see that the limiting behavior of the number of infected individuals is also to decrease and converge to 0 albeit at a much slower pace than the case $r < 0$ which we will cover in the next section. Nevertheless, even with zero growth rate, the epidemic becomes extinct:

$$\lim_{t\to\infty} I(t) = 0. \tag{1.12}$$

1.2.5 Fast extinction

If we assume instead that $r < 0$ then it follows that $\hat{I}_\infty < 0$ and $c < 0$. In order to write $I(t)$ as a function of positive quantities we introduce opposites of all our negative quantities $\rho = -r$, $\iota = -\hat{I}_\infty$ and $\gamma = -c$ which lets us write the following:

$$I(t) = \frac{-\iota}{1 - \gamma e^{\rho t}}$$

which we rearrange into:

$$I(t) = \frac{\iota}{\gamma e^{\rho t} - 1}. \tag{1.13}$$

Since this exponential quantity goes to infinity, we have that:

$$\lim_{t\to\infty} I(t) = 0 \tag{1.14}$$

and we introduce another symbol $\check{I}_\infty = 0$.

The number of infected individuals over time 1.13 can also be read as a formula for how much time is needed to reach a small infected population. Indeed if we denote that small population by ε, we can compute that the epidemic dies out in time $\Omega(\log(\varepsilon^{-1}))$.

1.2.6 Convergence modes of the SIS model

The analysis above allows us to completely characterize the dynamics of the SIS model using a simple threshold condition.

Theorem 1. *(The three modes of SIS)*
The dynamics of a SIS model with parameters β and δ on a total population of size N admit three different convergence patterns. The first mode of convergence is for the infection to grow exponentially fast towards a stable endemic population, the second mode is a slow extinction of the epidemic, while the third mode is an exponentially fast extinction.

$$\lim_{t\to\infty} I(t) = \begin{cases} \hat{I}_\infty = N - \delta/\beta & \text{if } N > \delta/\beta \\ \check{I}_\infty = 0 & \text{if } N = \delta/\beta \\ \check{I}_\infty = 0 & \text{if } N < \delta/\beta. \end{cases}$$

To illustrate this theorem we draw in Figure 1.2 some possible scenarios of an infection represented by a SIS model. We fix $N = 1000$ and $I_0 = 200$ and vary the δ and β parameters as mentioned in Table 1.1 to depict the three possible modes of convergence identified in Theorem 1. Scenarios A, B, and C respectively correspond to cases $r > 0$, $r = 0$, and $r < 0$ following the presentation of the above theorem.

Scenario	β	δ	N		δ/β
A	0.001	0.3	1000	>	300
B	0.0001	0.1	1000	=	1000
C	0.0001	0.3	1000	<	3000

Table 1.1: Different parameters for a SIS model with $N = 1000$ and $I_0 = 200$

In this section we have explored the dynamics of the SIS model, a simple epidemic model which already offers some interesting behaviour. In this model an epidemic either goes endemic or goes extinct. Assuming a fixed population, the dynamics of the epidemic are entirely controlled by its growth rate. Whenever the growth rate of the epidemic is positive, the number of infected individuals grows exponentially fast towards some fixed fraction of the population. Otherwise, the epidemic dies out and the number of infected individuals goes to 0 as time goes on. However, the rate of extinction can be either slow, when the growth rate of the epidemic is zero, or fast, when the epidemic has a negative growth rate.

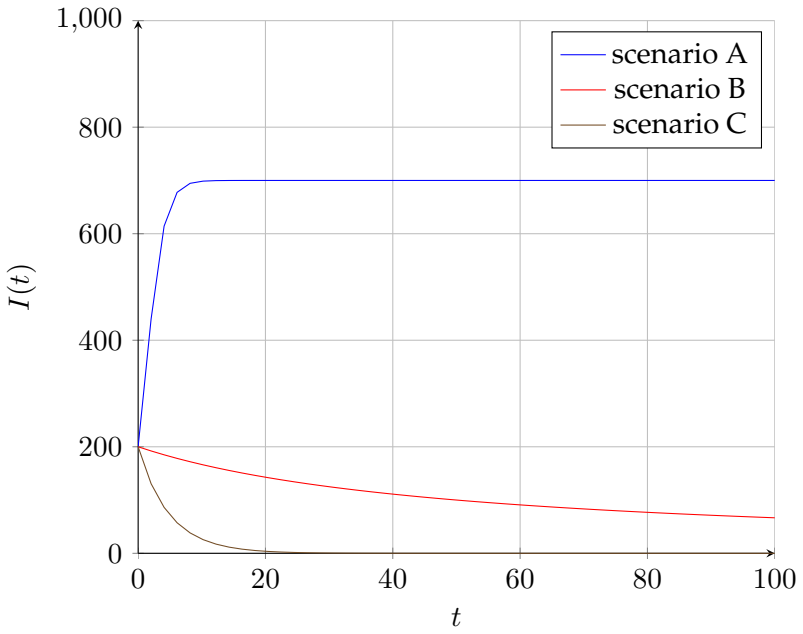

Figure 1.2: Evolution of the infected population in 3 scenarios

The dynamics of the SIS model can directly be interpreted as a recipe to eradicate any epidemic. Assuming that some of the key parameters of an epidemic: total population N, rate of infection β, or rate of recovery δ can be altered by some policy, achieving $N < \delta/\beta$ can be seen as a direct way to eradicate a given epidemic.

However, the applicability of the SIS model is limited by several shortcomings. First, the differential equations that describe the evolution of an epidemic rely heavily on the *full-mixing assumption* which does not hold in many systems and particularly never holds in systems with a very large number of individuals. Furthermore, the epidemic is characterized in the SIS model as a purely deterministic process evolving over time. However in many systems individuals interact on a discrete basis and many real-life factors which contribute to an infected individual infecting a sane individual amount to a higher chance of infection per interaction.

In the next section we will try to address some of these shortcomings by considering the networked SIS model which is composed of two mathematical objects: a graph representing the network of relationships between individuals, and a Markov chain which describes an infection as a probabilistic process. The

undirected graph addresses the variability of interaction between individuals while the Markov chain gives weight to the element of chance in the spread of an epidemic, especially at the beginning of an outbreak.

1.3 The networked SIS model

We now describe a generalization of the SIS model in which the full-mixing assumption is replaced by an explicit mathematical object: an undirected graph $G = (V, E)$ in which every node $v \in V$ plays the role of a particular individual while an edge $uv \in E$ correspond to a possible interaction between individuals u and v. The assumption in the SIS model that the total population N is constant becomes that V is fixed, and as such we let $|V| = n$.

The main difference with the SIS model comes from the fact that the sane population S was simply a non-negative number representing its headcount while in the networked SIS model the sane population is a subset of nodes $S \subseteq V$. This networked model is more fine-grained and describes the state of every individual over time and uses a Markov chain to represent the infection. Recall that in the SIS model the differential equations represented the fact that an infected individual would spread the infection to a β fraction of the healthy individuals. In this model we interpret the previous statement as an expectation by modeling the infection as having probability β to spread between an infected node and a healthy node. Indeed, we can represent the full-mixing assumption by considering a complete graph i.e. every node being connected to every other node, and counting how many new infections happen on average.

This section summarizes the breakthrough results of Chakrabarti et al. [Cha+08] and Ganesh et al. [GMT05] which formalized the preliminary results given by Wang et al. [Wan+03]. To simplify exposition, we adopt the formalism of Chakrabarti et al. which is useful to understand generalizations to other networked compartmental models such as work by Prakash et al. [Pra+11]. The prerequisites for this section include elementary graph theory knowledge, basic probability theory and simple properties of Markov chain, as well as familiarity with classical results in dynamical systems.

1.3.1 Definition

The networked SIS model is a Markov chain represented by the two following transition rules. Denoting by β the probability of infection and by δ the

probability of recovery we have for any given node $v \in V$:

$$v : \begin{cases} S \to I & \text{with probability } \beta\,|I \cap \Gamma(v)|, \\ I \to S & \text{with probability } \delta. \end{cases} \tag{1.15}$$

where $\Gamma(v) = \{u \in V \mid uv \in E\}$ is the set of nodes adjacent to v.

The first transition rule $S \to I$ describes that the infection process happens independently between each infected-healthy pair of nodes. We have allowed a slight abuse of notation by writing a probability whose value is potentially greater than one. The second transition rule $I \to S$ is identical in spirit to the δI term found in the differential equations of the SIS model.

1.3.2 Eventual convergence

The states of this Markov chain are all possible partitions of V in S and I. There are 2^n possible partitions of the node set. As a result, the state space of the Markov chain is also 2^n. By definition of the first transition rule, the state where all individuals are healthy $S = V$ is an absorbing state since in that case $I = \varnothing$ and as a result $|I \cap \Gamma(v)| = |I| = 0$. Furthermore, notice that every other state has a non-zero probability of transitioning to the $S = V$ state. Taken together, this means that the Markov chain we consider has a unique absorbing state. This means that the networked SIS model is an absorbing chain and admits the following limiting behavior:

$$\lim_{t \to \infty} I(t) = \varnothing. \tag{1.16}$$

This limiting behavior is at odds with our understanding of the classical SIS model which had a limit where the epidemic became endemic. This shows that parallels between the two models are not always possible. However, we will show that the dynamics of this Markov chain obey several modes of convergence that are similar to the SIS model but also follow convergence rates that resemble those of the classical SIS model.

1.3.3 Asymptotic stability

Recall that in the SIS model the condition $N < \delta/\beta$ was equivalent to an exponentially fast extinction. We will now see whether such a behavior is preserved in the networked SIS model.

In order to simplify the analysis of the networked SIS model we will translate statements about the Markov chain into properties of an equivalent discrete-time dynamical system which describes the probabilities of nodes of being sick at each time step.

We start by describing in Table 1.2 specific events which will help us understand the dynamics of the networked SIS model.

Event	Probability	Description
$R_{v,t}$	$r_{v,t}$	node v resists infection by its neighbors at time t
$X_{v,t}$	$x_{v,t}$	node v is sick at time step t
$B_{u,v}$	β	node v is infected by sick node u (birth)
D_v	δ	node v recovers (death)

Table 1.2: Events and their probabilities in a networked SIS model

According to the definition of the model, event $R_{v,t}$ happens when all the neighbors of v are either healthy or infected but each fails to infect v. Formally speaking:

$$R_{v,t} = \bigcap_{u \in \Gamma(v)} \overline{X}_{u,t-1} \cup (X_{u,t-1} \cap \overline{B}_{u,v}) \tag{1.17}$$

This immediately gives us an expression for the probability of such event:

$$r_{v,t} = \prod_{u \in \Gamma(v)} [(1 - x_{u,t-1}) + x_{u,t-1}(1 - \beta)].$$

As expected, the probability of resisting infection from one's neighbors is a function of the probabilities for neighbors to be infected:

$$r_{v,t} = \prod_{u \in \Gamma(v)} (1 - \beta\, x_{u,t-1}) \tag{1.18}$$

Furthermore, event $X_{v,t}$, that is a node v being healthy at a given time step t, is the consequence of two possible cases:

- either v was healthy at time $t-1$ and resists infection from its neighbors at time t
- or v was sick at time step $t-1$ but recovered at time t

which translates into:

$$\overline{X}_{v,t} = (\overline{X}_{v,t-1} \cap R_{v,t}) \cup (X_{v,t-1} \cap D_v)$$

or in terms of probability:

$$1 - x_{v,t} = (1 - x_{v,t-1}) r_{v,t} + x_{v,t-1} \cdot \delta$$

giving us an expression of $x_{v,t}$:

$$x_{v,t} = 1 - r_{v,t} + x_{v,t-1}r_{v,t} - \delta x_{v,t-1}. \tag{1.19}$$

Substituting $r_{v,t}$ for its value we obtain a formula for $x_{v,t}$ as a function of several $x_{u,t-1}$, that is:

$$x_{v,t} = 1 - \delta x_{v,t-1} - (1 - x_{v,t-1}) \prod_{u \in \Gamma(v)} (1 - \beta x_{u,t-1}). \tag{1.20}$$

This means that the vector x for a given time step is the result of a non-linear function of x of the previous time step. We can then remove the time index and rewrite this relationship as a non-linear autonomous dynamical equation, i.e. a discrete time analogue of non-linear partial differential equations.

Just like the Markov chain we started with, this representation embodies the fact that the immediate future of a networked SIS model is only determined by its current state, not by a longer history of the system. Thus, we represent the dynamics of the networked SIS model as the following diffeomorphism:

$$\begin{aligned} f &: [0,1]^n \to [0,1]^n \\ (f_v)_{v \in V} &: x_v \mapsto 1 - \delta x_v - (1 - x_v) \prod_{u \in \Gamma(v)} (1 - \beta x_u) \end{aligned}$$

where repeated applications of f correspond to time steps.

One way to study a given autonomous dynamical system is to characterize the fixed points of its associated diffeomorphism. For this we invoke a classical theorem which classifies fixed points that act like attractors. Similarly to the indirect Lyapunov method, this theorem offers a way to analyze the properties of a fixed point via the linearization of the dynamical system at this point. More formally, we have the following:

Theorem 2. *(Stability conditions [Hol12])*

Let x^* be a fixed point of a discrete autonomous dynamical system represented by a diffeomorphism $f : E \to E$, that is $f(x^*) = x^*$, and let Df denote the Jacobian matrix of f. The eigenvalues λ_i of $Df(x^*)$ determine the stability of x^* in the following way:

$$x^* \text{ is } \begin{cases} \textit{asymptotically stable} & \textit{if } \max_i |\lambda_i(Df(x^*))| < 1 \\ \textit{unstable} & \textit{if } \max_i |\lambda_i(Df(x^*))| > 1 \end{cases}$$

We will not give the precise definitions of stability and asymptotic stability which can be found in textbooks, e.g. the book by Hirsh and Smale [HDS74], and instead focus on the meaning of these concepts from the theory of dynamical systems. A fixed point is said to be stable if the points in a neighborhood around it remain in that neighborhood by application of the diffeomorphism. Furthermore, an asymptotically stable fixed point corresponds to the popular image of an attractor point, as points in its neighborhood eventually converge to the point after repeated applications of the diffeomorphism. On the contrary, a fixed point that is unstable can possibly have points in its neighborhood that quickly exit that neighborhood.

The trajectory of the networked SIS model is thus related to the fixed points of this diffeomorphism. As mentioned above, the state where all nodes are healthy is an absorbing state of the Markov chain and it is easy to see that it corresponds to $x = \mathbf{0}$ i.e. every node has probability 0 of being infected. Unsurprisingly, this zero vector is a fixed point of f:

$$f(\mathbf{0}) = \mathbf{0} \tag{1.21}$$

First, let us compute the Jacobian matrix of f at $\mathbf{0}$ which is a square matrix containing partial derivatives of each component function f_j with respect to each node infection probability $x_i, \forall i \in V$. We distinguish three cases when calculating a partial derivative of the following form:

$$\frac{\partial f_j}{\partial x_i} = \frac{\partial}{\partial x_i}\left(1 - \delta x_j - (1 - x_j)\prod_{\nu \in \Gamma(j)}(1 - \beta x_\nu)\right)$$

The first case when $i = j$ is straightforward:

$$\frac{\partial f_i}{\partial x_i} = 0 - \delta - 0 + \prod_{\nu \in \Gamma(i)}(1 - \beta x_\nu)$$

which when evaluated at $\mathbf{0}$ yields:

$$\frac{\partial f_i}{\partial x_i}\Big|_{x=\mathbf{0}} = 1 - \delta \tag{1.22}$$

The second case when $i \neq j$ reveals the adjacency structure of the diffeomorphism. To study this case we return to the formula for the probability of resisting an epidemic at a given time step $r_{v,t}$.

Here we consider $r_{v,t+1}$ as a function of x_t and write r_j as a function of x. Assuming that $i \in \Gamma(j)$ we have:

$$r_j = (1 - \beta x_i) \prod_{\substack{\nu \in \Gamma(j) \\ \nu \neq i}} (1 - \beta x_\nu)$$

so its partial derivative is simply:

$$\frac{\partial r_j}{\partial x_i} = -\beta \prod_{\substack{\nu \in \Gamma(j) \\ \nu \neq i}} (1 - \beta x_\nu)$$

which we encounter in the partial derivative of f_j:

$$\frac{\partial f_j}{\partial x_i} = 0 - 0 - \frac{\partial r_j}{\partial x_i} + x_j \frac{\partial r_j}{\partial x_i}.$$

It is easy to see that the fourth term of this expression will be cancelled when we evaluate it at $\mathbf{0}$, leaving only:

$$\left.\frac{\partial f_j}{\partial x_i}\right|_{x=\mathbf{0}} = \beta \text{ if } i \in \Gamma(j). \tag{1.23}$$

Finally, the third case when $i \neq j$ and $i \notin \Gamma(j)$ can be easily derived from the second as the partial derivative of s_j simply becomes 0. This gives us:

$$\left.\frac{\partial f_j}{\partial x_i}\right|_{x=\mathbf{0}} = 0 \text{ if } i \notin \Gamma(j). \tag{1.24}$$

To summarize, the Jacobian of f at $\mathbf{0}$ is a square matrix given by the following:

$$(Df(\mathbf{0}))_{ij} = \begin{cases} \beta A_{ij} & \text{if } i \neq j \\ 1 - \delta & \text{if } i = j \end{cases}$$

where A_{ij} is the (i, j) cell of the adjacency matrix of our graph G. In matrix notation, this can be written as:

$$Df(\mathbf{0}) = \beta A + (1 - \delta) I. \tag{1.25}$$

Since G is an undirected graph, its adjacency matrix A is a symmetric matrix whose eigenvalues are real. We can diagonalize A which gives us an expression for the eigenvalues of $Df(\mathbf{0})$ needed to apply Theorem 2 as follows:

$$Df(\mathbf{0}) = \beta P^\intercal \Lambda P + (1 - \delta) I$$
$$Df(\mathbf{0}) = P^\intercal (\beta \Lambda + (1 - \delta) I) P$$

which implies that:

$$\lambda_i(Df(\mathbf{0})) = \beta\lambda_i(A) + 1 - \delta. \tag{1.26}$$

With this characterization of the eigenvalues of $Df(\mathbf{0})$ as being directly related to the eigenvalues of the adjacency matrix A, we can leverage the structure of A to understand the fixed point $\mathbf{0}$. The adjacency matrix of our input graph is, without loss of generality, an irreducible non-negative matrix as the graph G is connected.

Theorem 3. *(Perron-Frobenius for irreducible non-negative matrices) [Per07]*
Let M be a non-negative irreducible matrix. The following statements hold:

- $\lambda_1(M) > 0$,
- $\lambda_1(M) = \max_i |\lambda_i(M)|$,
- $\lambda_1(M)$ *is simple,*
- *the eigenspace associated with* $\lambda_1(M)$ *has dimension one.*

Applying Theorem 3 on A allows us to express the spectral radius of $Df(\mathbf{0})$ as a function of the largest eigenvalue of A, that is:

$$\max_i \lambda_i(Df(\mathbf{0})) = \beta\lambda_1(A) + 1 - \delta.$$

Thus, a sufficient condition for $\mathbf{0}$ to be asymptotically stable is:

$$\beta\lambda_{\max}(A) + 1 - \delta < 1$$

or in a more familiar form:

$$\lambda_{\max}(A) < \delta/\beta. \tag{1.27}$$

is a sufficient condition for an epidemic, which is described by a networked SIS model with parameters β and δ, to be guaranteed to disappear.

We leave out the case $\lambda_{\max} = \delta/\beta$ open as it is not covered by our stability theorem. Furthermore, one could argue that this exact equality does not correspond to the physical reality of epidemics.

1.3.4 Exponential convergence

Now, we work towards quantifying the speed of convergence of a networked SIS model in the case where $\lambda_{\max}(A) < \delta/\beta$. We start by deriving a simple lower bound for the probability of resisting an infection. For some time step t, we denote by C_{uv} the event in which a node v contracts the infection at time step t from being exposed to a node u which was sick at time step $t-1$. Recall that R_v represents is the event where a node v has resisted an infection from its neighbors. In other words, resisting an infection from all neighbors is equivalent to not contracting it from any of them:

$$R_v = \bigcap_{u \in \Gamma(v)} \overline{C_{uv}}$$

Using De Morgan's law we can rewrite this event as:

$$R_v = \overline{\bigcup_{u \in \Gamma(v)} C_{uv}}$$

and apply the union bound on $\overline{S_v}$:

$$\Pr(\overline{R_v}) \leq \sum_{u \in \Gamma(v)} \Pr(C_{uv})$$

which gives us:

$$1 - r_v \leq \sum_{u \in \Gamma(v)} \beta x_u$$

a lower bound on the probability of node v resisting infection by its sick neighbors:

$$r_v \geq 1 - \beta \sum_{u \in \Gamma(v)} x_u. \tag{1.28}$$

We will now proceed to derive a lower bound for the probability of a node v to be healthy $1 - f(x_v)$ after one time step. Reinterpreting formula 1.20 in our diffeomorphism formalism, we have that:

$$1 - f(x_v) = \delta x_v + (1 - x_v) r_v$$

using the lower bound 1.28 obtained earlier we have:

$$1 - f(x_v) \geq \delta x_v + (1 - x_v) \left(1 - \sum_{u \in \Gamma(v)} \beta x_u \right)$$

which is trivially at least:

$$1 - f(x_v) \geq \delta x_v + 1 - x_v - \sum_{u \in \Gamma(v)} \beta x_u$$

or with some rearraging:

$$f(x_v) \leq \beta \left(\sum_{u \in \Gamma(v)} x_u \right) + (1 - \delta) x_v \tag{1.29}$$

which in matrix form resembles an expression we have seen earlier:

$$f(x) \leq (\beta A + (1 - \delta) I) x \tag{1.30}$$

indeed $\beta A + (1-\delta)I$ is $Df(\mathbf{0})$ the Jacobian matrix of f evaluated at $\mathbf{0}$. We write that, given x_0 the initial state of the system, and $x_k = f^k(x_0)$ the state of the system after k applications of f:

$$\begin{aligned} x_k &\leq Df(\mathbf{0}) f^{k-1}(x_0) \\ &\leq Df(\mathbf{0})^2 f^{k-2}(x_0) \\ x_k &\leq Df(\mathbf{0})^k x_0 \end{aligned}$$

Using our knowledge of the eigendecompostion of $Df(\mathbf{0})$ and denoting by λ_i, e_i its pairs of eigenvalues and associated eigenvectors, we have:

$$\begin{aligned} x_k &\leq \sum_i \lambda_i^k \left(e_i e_i^\intercal \right) x_0 \\ x_k &\leq \lambda_{\max}^k \, c \end{aligned}$$

where $\lambda_{\max}$ is the largest eigenvalue of the Jacobian matrix and $c = (e_{\max} e_{\max}^\intercal) x_0$ is a constant vector.

Because the largest eigenvalue of a networked SIS model satisfying $\lambda_{\max}(A) < \delta/\beta$ implies that $\lambda_{\max}(Df(\mathbf{0})) < 1$ we know that the largest eigenvalue of our Jacobian matrix $\lambda_{\max} = e^{-\alpha}$ for some $\alpha > 0$. Thus, we have that a vector x_k or sickness probabilities obtained after k time steps decreases exponentially:

$$x_k \leq e^{-\alpha k} c. \tag{1.31}$$

which is not far from showing that the all-healthy state is an exponentially stable fixed point of a networked SIS model whenever $\lambda_{\max}(A) < \delta/\beta$.

Following the methodology in Chakrabarti et al. [Cha+08], we have thus proved a topology-aware and discrete-time analogue of the convergence patterns of the SIS model which describes the case when the epidemic disappears quickly.

Summary

In this chapter, we have studied epidemic models through two particular examples: the SIS model and its generalization to arbitrary interaction patterns, the networked SIS model. The main observation is that both models display critical behavior based on small number of parameters.

In the case of the SIS model, solving the differential equation gave us quantitative trajectories for the size of the infected population $I(t)$ which critically depend on the total population size N and on the epidemic parameters: the birth rate β and the death rate δ. Indeed we have:

$$I(t) : \begin{cases} O(1/(1+e^{-t})) \text{ convergence to } N - \delta/\beta & \text{if } N > \delta/\beta \\ O(1/t) \text{ convergence to } 0 & \text{if } N = \delta/\beta \\ O(e^{-t}) \text{ convergence to } 0 & \text{if } N < \delta/\beta. \end{cases}$$

In the case of the networked SIS model, analyzing the discrete autonomous system gave us qualitative information on the stability of the graph analogue to $I = 0$ which is the state in which all nodes are in S or equivalently if we denote by x_t a vector representing the probability for each node to be infected at a given time step then $x_t = \mathbf{0}$.

$$\mathbf{0} : \begin{cases} \text{is unstable} & \text{if } \lambda_{\max}(A) > \delta/\beta \\ \text{is asymptotically stable} & \text{if } \lambda_{\max}(A) < \delta/\beta \end{cases}$$

and furthermore we established a quantitative trajectory towards the extinction of the epidemic:

$$x_t : O(e^{-t}) \text{ convergence to } \mathbf{0} \text{ if } \lambda_{\max}(A) < \delta/\beta.$$

This leads us to state that the networked SIS model critically depends on both the largest eigenvalue of the adjacency matrix of the graph and also on the epidemic parameters of the SIS model.

We mention that the analysis of the networked SIS model can be completed [GMT05] to show that in the case $\lambda_{\max}(A) > \delta/\beta$, the expected time until x converges to $\mathbf{0}$ is exponential in the number of nodes of the graph which is an adequate analogue to the trajectory of the original SIS model.

From the point of view of the defense against epidemics, these critical thresholds should be seen as opportunities to guarantee that a given epidemic will disappear in a very short amount of time. However, a reader might wonder if such simple thresholds only apply to the simple SIS model. Thankfully, this is not the case, and it is remarkable that the two kinds of analysis we have

detailed in this chapter generalize to a very wide family of compartmental models. More complex compartmental models include: the SIR model where nodes Recover completely by gaining immunity, the SIRS model where nodes recover temporarily, the SEIR model where nodes become Exposed as they incubate the disease, and many more. Furthermore, the networked generalization of such models display the same critical behavior as the networked SIS model with theorems of the form:

Theorem 4 (Critical behavior of networked epidemic models [Pra+11])**.** *Given a networked epidemic model on a graph G with adjacency matrix A, epidemic parameters $\pi = (\pi_1, \ldots, \pi_k) \in \mathbb{R}^k$ and assuming that at least one node is infected $\exists\, v \in V, v \in I$, then there is a function $\epsilon : \mathbb{R}^k \to \mathbb{R}$ such that:*

$$\lambda_{\max}(A) < \epsilon(\pi) \implies I(t) \text{ converges to } \mathbf{0} \text{ exponentially fast},$$
$$\lambda_{\max}(A) > \epsilon(\pi) \implies I(t) \text{ converges to } \mathbf{0} \text{ exponentially slowly}.$$

where in the case of the networked SIS model $\pi = (\beta, \delta)$, and $\epsilon(\beta, \delta) = \delta/\beta$.

The question now becomes: "How do we achieve $\lambda_{\max}(A) < \epsilon(\pi)$?"

2 Algorithms

Introduction

A howdunit, for "how has it been done?", is a type of detective story in which the focus is the mystery of how the crime actually took place. In this thesis, the methodology we follow essentially corresponds to common practice in the theory of algorithms. The analysis of algorithms is a major branch of theoretical computer science which can be seen as a *positive* theory of computational problems via the study of constructive procedures that are designed to solve them.

In essence, the theory of algorithms is concerned with designing algorithms and in particular obtaining theorems that establish performance guarantees for a given algorithm with regards to various resources. For example, a common goal would be to try to design an algorithm for which we can prove that it only uses a bounded amount of memory and also that it completes its computation correctly in a bounded amount of time. However, many variations can be considered such as allowing for other types of computational resources or obtaining more precise guarantees. Other examples include: studying algorithms that work with input data that is revealed over time, obtaining guarantees for algorithms which return solutions that might not be entirely correct, or even analyzing algorithms that use randomness to compute solutions to problems. Despite this diversity, it is often the kind of computational task which determines the desirable criteria to look for in algorithms.

A long-lasting practice in the theory of algorithms pioneered in the 1960s by Edmonds [Edm65] is to consider performance in a conservative manner, the so-called worst-case analysis. This corresponds to trying to determine the performance of an algorithm faced with the worst possible input data it can encounter. We follow this strategy in our thesis for two main reasons: the first reason is to simplify analysis as it is generally simplifies the use of mathematical tools to represent a given algorithm and its possible inputs, the second reason is that worst-case guarantees always apply even if they may not reflect the properties of a given algorithm in more specific settings.

The theory of algorithms is intimately related to the theory of computational complexity whose main goal is to establish impossibility theorems. An impos-

sibility theorem states that for a given computational problem there can be no algorithm whose performance exceeds a certain level while consuming a certain amount of resources.

In this sense, the existence of an algorithm to solve a given problem can be seen as an "upper bound" while an impossibility theorem would be a corresponding "lower bound". The shared goal of the two disciplines is thus to close the gap between the current best algorithm and the current best impossibility theorem. A computational problem is considered to be well understood when the best algorithm matches the lower bound. For example, the computational task of sorting a list of integers can be done in time $O(n \log n)$ and a corresponding theorem states that no algorithm can achieve a better worst-case time complexity.

Beyond the resources they require, algorithms can be categorized in broad classes following the mathematics they employ. In the history of science, algorithms have been developed in both discrete and continuous settings but we underline that the theoretical computer science community has focused on combinatorial problems from its beginnings during the early 20$^{\text{th}}$ century. On the other hand, the numerical analysis community has developed methods to solve computational problems together with a greatly different analytical methodology, often focusing on more precise issues such as speed of convergence, robustness to errors, or access to specific linear algebraic operations.

In this chapter we give an overview of the methodology used in analysis of algorithms and describe its relationship with methods originating from the continuous optimization literature. We also detail the way randomness is handled in the analysis of algorithms framework and what constitutes the efficient use of random sources.

We split our review of the theory in six parts: first we give a quick introduction to basic concepts used to describe computational problems followed by a brief tour of computational complexity with a note on the use of randomness. This gives us the appropriate vocabulary to introduce approximation algorithms as well as randomized approximation. Turning to the mathematical objects we consider in our problems, we recall elementary results of spectral graph theory. Some spectral properties can be interpreted as optimization problems which we describe in a section on semidefinite optimization.

In this chapter we will work towards establishing the existence of a phase transition in complexity when solving certain optimization problems. By assuming P$\neq$NP we will determine that the unique coverage problem can be approximated in polynomial time within a ratio of $O(\log n)$ and at the same time cannot be approximated in polynomial time to any better accuracy.

2.1 Problems and solutions

In this section we review the basic theoretical tools used by computer scientists to assess the difficulty of solving problems and conversely to define what constitutes efficient computation, see also the comprehensive resources by Garey and Johnson [GJ] and by Arora and Barak [AB09]. We start by reminding the reader of basic concepts from the theory of computation and optimization, mostly convex optimization [BV04], [BN01], in order to introduce notions on randomization [MR10] and then approximation algorithms [WS11] and on the various guarantees that can be obtained on the output of algorithms. We conclude by showcasing the so-called relaxation and rounding framework using the concepts that we previously described.

2.1.1 Decision problems

A major theme of theoretical computer science is the study of computational problems. The simplest class of computational problems is the class of decision problems, that is problems for which the solution is either yes or no. However, it is crucial to understand that the notion of computational problem corresponds not to a particular question but to a general question which could be asked in a wide variety of contexts. The input data that specifies the context of a general question is called an *instance* of the problem.

We give an example of a decision problem, graph connectivity, which is concerned with detecting a simple property in undirected graphs. We say that a graph $G = (V, E)$ is connected when for any pair $s \neq t$ of nodes in V, there exists a $s - t$ path, a sequence $(su, uv, vw, \ldots, xy, yz, zt)$ of consecutive edges sharing a node, which starts at the source node s and ends at the target node t.

GRAPH CONNECTIVITY	
Instance:	An undirected graph $G = (V, E)$.
Question:	Is G connected?

Formally speaking, a computable problem P is equivalent to a language L defined by the set of binary strings recognized by a Turing machine. All the instances of the problem for which the answer is YES are exactly the binary strings that result in an accepting state for the corresponding Turing machine. However, it is more practical to work on problems by separating instances from questions, as it leads to studying relationships between problems. Many problems can be created by slight modifications of a given question, or by changing the set of instances, either expanding or narrowing it. For example,

one could consider a variant of graph connectivity where instead of asking for simple connectivity, one would want to know if in a graph there exists at least two edge-disjoint paths which connect any pair of nodes, a problem called 2-edge connectivity. Another variant could be to consider the generalization to directed graphs, a problem known as directed connectivity.

2.1.2 Optimization problems

Optimization is a category of computational problems that is concerned with the construction of specific mathematical objects. Optimization problems are composed of more elements than decision problems but they share in common the notion of an instance set I which represents the different ways a general task can be specialized to a given context.

First, an optimization problem must include a domain or *feasible region*. This region is defined by a function $F : I \to S$ which specifies the set of mathematical objects which can be returned as solutions given a specific instance. Second, an optimization problem must define an *objective* which consists of a goal and a metric. The goal is simply a $\min$ or $\max$ operator while the metric, often called the objective function, $m : S \to \mathbb{R}$ is used to evaluate feasible solutions. An optimal solution x^*, often denoted by a star subscript, is an element of the feasible region which optimizes the metric, and the value of this solution is called the optimal value m^*, or sometimes the optimal solution value.

OPTIMIZATION PROBLEM O	
Instance:	$i \in I$.
Goal:	$\text{opt} \in \{\min, \max\}$.
Metric:	$m : S \to \mathbb{R}$.
Domain:	$F : I \to S$.

Or in a compact format:

$$O : I \to S \times \mathbb{R}$$
$$O : i \mapsto \text{opt}_{x \in F(i)}\, m(x).$$

When reading the definition of an optimization problem it is possible to form a full sentence by replacing each section (Instance, Goal, Metric, Domain) by the following expression: "Given $i \in I$, we want to opt the metric m over $F(i)$".

Note that the domain $F(i)$ of a problem for a particular instance i might be empty and for that reason finding an optimal solution of a problem implicitly requires to solve a decision problem called the infeasibility problem of $F(i)$ (often improperly referred to as the feasibility problem) before attempting to construct a solution.

INFEASIBILITY	
Instance:	A domain $F(i)$.
Question:	Is $F(i)$ empty?

We give an example of an optimization problem which amounts to computing the second smallest eigenvalue of a symmetric matrix for which the all-ones vector $\mathbf{1}$ is the eigenvector associated with its smallest eigenvalue. While this problem might come to the reader as oddly specific, it will play an important role in later chapters. The problem comes from the variational characterization of eigenvalues.

Theorem 5. *Courant-Fischer-Weyl [LL01]*

Let $M \in \mathbb{S}^n$ be a symmetric matrix with eigenvalues $\lambda_{\min} = \lambda_1 \leq \ldots \leq \lambda_n = \lambda_{\max}$, and associated eigenvectors $e_1, \ldots, e_n$, we have:

$$\lambda_k(M) = \min_{\substack{S \\ \dim(S)=n-(k-1)}} \max_{\substack{x \in S \\ x \neq \mathbf{0}}} \frac{x^\intercal M x}{x^\intercal x}$$

and in particular:

$$\lambda_k(M) = \max_{\substack{x \perp e_1 \\ \vdots \\ x \perp e_{k-1} \\ x \neq \mathbf{0}}} \frac{x^\intercal M x}{x^\intercal x}.$$

In the following problem, the feasible region does not depend on the instance so we abuse notation by not writing it as a constant function. We call this problem an eigenpair problem because a pair of r^*, x^*, i.e. the optimal value and an optimal solution, corresponds to an eigenpair: the eigenvalue and an associated eigenvector.

COURANT-FISCHER EIGENPAIR	
Instance:	A symmetric matrix $L \in \mathbb{S}^n$ with $L\mathbf{1} = \lambda_{\min}(L)\mathbf{1}$.
Goal:	min.
Metric:	The Rayleigh quotient of L: $r : x \mapsto \frac{x^\intercal L x}{x^\intercal x}$.
Domain:	Non-zero vectors orthogonal to the eigenspace associated with $\lambda_{\min}(L)$: $X = \{x \in \mathbb{R}^n : x \neq \mathbf{0}, x \cdot \mathbf{1} = 0\}$.

In the optimization literature it is more common to write this problem in compact form. Indeed, for a given matrix L as defined above, we can write:

$$\min_{\substack{x \neq \mathbf{0} \\ x \perp \mathbf{1}}} \frac{x^\intercal L x}{x^\intercal x} \tag{2.1}$$

or by explicitly revealing which parameter is the input data:

$$\overbrace{L}^{\text{Instance}} \mapsto \underbrace{\overbrace{\min_{\substack{x \neq \mathbf{0} \\ x \perp \mathbf{1}}}}^{\text{Task}}}_{\text{Domain}} \overbrace{\frac{x^\intercal L x}{x^\intercal x}}^{\text{Objective}} \tag{2.2}$$

We underline our choice to define optimization problems as construction tasks, i.e. effectively finding x^*. This is in contrast with other texts where authors consider the optimization problem as the task of evaluating the optimal value, which here would be r^*. We believe that the distinction is important even if in the literature most optimal value problems are solved by algorithms which construct an optimal solution. Consider for example an optimization problem which combines a constant objective function with a non-empty domain defined in a non-trivial way. It is clear that evaluating the optimal value is trivial, it suffices to return the constant, while constructing a solution might be a much harder problem. To elaborate on this difference, we turn to the study of reductions which formalizes the possible links between problem variants, and even between completely different problems.

2.1.3 Reductions

On top of the optimal value evaluation problem, it is always possible to define at least one decision problem associated with a given optimization problem. The simplest way is to add an extra parameter to the instance which represents a test value, and to transform an optimization goal into the corresponding upper bound or lower bound question. For example, by reusing the definitions of r and X from the Courant-Fischer eigenpair problem we create the associated decision problem:

COURANT-FISCHER EIGENVALUE BOUND	
Instance:	An integer-valued symmetric matrix $L \in \mathbb{S}^n$ with $L\mathbf{1} = \lambda_{\min}(L)\mathbf{1}$ and a tentative bound $b \in \mathbb{Z}$.
Question:	Does there exist $x \in X$ such that $r(x) \leq b$?

It is easy to see that if we have an algorithm that can solve the eigenpair problem, then we can use it to solve the eigenvalue bound. This is a central notion in computational complexity called a reduction. Formally speaking if we have two problems A and B then we say that A is reducible to B if an hypothetical algorithm that solves B could be used to design an algorithm which solves A. We denote this relationship by $A \leq_T B$ where the T subscript stands for Turing reduction. In our example we have the following:

COURANT-FISCHER EIGENVALUE BOUND $\leq_T$ COURANT-FISCHER EIGENPAIR.

If we denote by $\mathrm{alg}_{\mathrm{CFE}}$ an hypothetical algorithm which solves the Courant-Fischer eigenpair problem and by r_M the Rayleigh quotient of a matrix M, then we can design the following algorithm.

Algorithm 1: BOUND CHECKER

Data: an integer-valued symmetric matrix $L \in \mathbb{S}_n$ such that $L\mathbf{1} = \lambda_{\min}(L)\mathbf{1}$, and a bound $b \in \mathbb{Z}$
Result: True (yes) or False (no)

```
x* = alg_CFE(L)
return r_L(x*) ≤ b
```

It is clear that Algorithm 1 returns the correct answer for the Courant-Fischer eigenvalue bound problem.

Note that this reduction holds in general for any optimization problem O and associated decision problem D_O:

$$D_O \leq_T O$$

where we simply replace $\mathrm{alg}_{\mathrm{CFE}}$ by alg_O an hypothetical algorithm to solve O, and r_L by the metric function m of the optimization problem.

The optimal value problem E_O is what is often used in textbooks to define optimization problems and consists in finding the optimal value of a problem instead of constructing an optimal solution. Outside of this fine distinction, O and E_O share the same instance, goal, metric, and domain. For this reason, a relationship similar to that of D_O and O holds between the optimal value problem E_O and the associated decision problem D_O.

Indeed $D_O \leq_T E_O$ is obtained directly by an algorithm which returns the Boolean value $\mathrm{alg}_{E_O} \leq b$. Furthermore $E_O \leq_T D_O$ can be obtained by using dichotomic search over b until the optimal value is found. Recall that for this reduction to terminate we have to require that the codomain of the metric function m should be countable, e.g. $\mathbb{Z}$ or $\mathbb{Q}$. We have chosen to not impose this restriction in our definition of optimization problems to match common

practice in the continuous optimization literature. Nevertheless, when the metric function has a countable codomain the optimal value problem and the associated decision problem are reducible to each other, which we denote by saying that E_O and D_O are equivalent or $E_O \equiv_T D_O$.

A reader might argue that a optimization problem asks more than a simple question and as such stating that a simple question is reducible to a complex question could be seen as vacuous. However we will see later that some apparently unrelated problems can be connected via a reduction.

Beyond the practical aspect of using known algorithms as primitives to solve new problems, reductions are used to categorize problems into classes. Before we introduce some well-known classes of problems, we are going to describe another relationship between problems and solutions through the lens of verification.

2.1.4 Verifiers and proofs

We have so far looked at computational tasks that are concerned with the answer to a general question made specific by a particular instance. We call *verifier* any algorithm which certifies that an affirmative answer to a decision problem for a given instance is correct given an additional piece of data called a *proof* (or witness or certificate). Trivially, an algorithm which solves the original decision problem is a verifier that requires no proof at all. However, in some cases there might exist ways to certify the answer to a problem without needing to know how to solve it.

Let us consider our first example, the graph connectivity problem and define a proof of connectivity to be a collection of $s-t$ paths for any pair $s \neq t$ of nodes in the graph G. The following algorithm is a verifier of the graph connectivity problem.

Algorithm 2: PATH CHECKER

Data: an undirected graph $G = (V, E)$, a set of $s - t$ paths
$P = \{P_{u,v} : u \neq v \in V^2\}$
Result: True (yes) or False (no)
`/* define path reachability */`
1 $r : V^2 \times P \to \mathbb{B}$
2 $r : s, t, p \mapsto (\text{tail}(p_1) = s) \wedge \left(\bigwedge_{i=1}^{n-1} \text{head}(p_i) = \text{tail}(p_{i+1})\right) \wedge (\text{head}(p_n) = t)$
3 **return** $\bigwedge_{s \neq t \in V^2, p_{s,t} \in P} r(s, t, p_{s,t})$

It is easy to see why. If Algorithm 2 returns a True value then we are sure that for every pair of distinct nodes there is a path connecting them. The

verification here consists in checking that every pair of nodes is paired with its own path, and that the sequence of edges is indeed a path from the source to its destination.

Here the set of $n(n-1)/2$ paths is the proof while Algorithm 2 is the verifier. We notice that this algorithm requires a number of element comparisons and Boolean operations that is polynomial in the size of the instance, here n. So not only is the proof short in a quantifiable manner, but the algorithm verifying it is also fast, those two properties staying true no matter what instance we consider.

We are now equipped with all the prerequisites to describe complexity classes, categories of problems which share an equivalent level of computational hardness.

2.2 Complexity and efficiency

In this section we review common complexity classes and their interpretations. A complexity class is a set of computational problems which share a common property, for example all decision problems which can be solved by an algorithm running in time bounded by an exponential function in the instance size belong to a class called **EXP**. Despite the major breakthroughs obtained by theoretical computer scientists in the field of computational complexity, seemingly trivial inclusions between classes remain elusive since the inception of the field with the paper by Hartmanis and Stearns [HS65]. We illustrate this by stating the following inclusions between four complexity classes:

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP}.$$

where **P** is the class of decision problems which admit a polynomial time algorithm, **NP** is the class of decision problems which admit a polynomial time verifier operating on a polynomial-size proof, **PSPACE** is the class of decision problems which admit an algorithm whose running time is not restricted but which operate on a polynomial amount of memory, and **EXP** is the class we described above.

Thanks to the hierarchy theorem of Hartmanis and Stearns which separates **P** and **EXP**, i.e. $\mathbf{P} \subsetneq \mathbf{EXP}$, we know that at least one of the three inclusions must be proper. Despite numerous attempts, the status of each of these inclusions is an outstanding open problem. All existing research in computational complexity constitutes circumstantial evidence that would lead the research community to conjecture that all these inclusions are proper and in particular that $\mathbf{P} \neq \mathbf{NP}$.

We will look in particular at three complexity classes. First, we will describe **P**, the class of decision problems which can be solved efficiently, together with **NP**, the class of decision problems which can be verified efficiently. We will also highlight some reasons to believe that **P** and **NP** are separate, and that invite us to believe that a wide family of problems in **NP** are hard to solve. Then, we will see how one can construct analogues of **P** when additional resources are considered. Here we will look at the impact of randomness on efficient computation with **BPP** which is the class of decision problems which admit randomized algorithms that output the correct answer with probability at least $2/3$.

2.2.1 P: deciding correctly in polynomial time

The theory of algorithms gives a particular role to polynomials because of their fundamental property of being a family of functions that is closed under composition. In the context of algorithms, this means that any algorithm which runs in polynomial time can include other polynomial time algorithms as subroutines. On the contrary, if one uses an exponential time algorithm as subroutine without specific care, it is almost guaranteed that the running time of the overall algorithm will be at least exponential.

Polynomial-time reductions

This property invites us to consider reductions that are more specific than the ones we have discussed above and which can be used to design efficient algorithms.

Indeed, when establishing that a problem A is reducible to another problem B, if the reduction transforms instances of problem A into an instance of problem B that has identical answers and does so in polynomial time, then we say that $A \leq_p B$, i.e. A is polynomial-time reducible to B.

It is easy to see that to solve problem A, given an algorithm alg_B which could solve problem B in polynomial time, it would suffice to apply such a reduction to produce an instance of problem B, run alg_B on the transformed instance, and return the answer that has been computed.

This type of reduction is called many-one reduction or Karp reduction, after Richard Karp who made major contributions to the field of computational complexity.

The seminal paper of Edmonds [Edm65] on the graph matching problem first popularized the concept of polynomial time which, together with the design of

many polynomial-time reductions between problems, would quickly become the standard in the theory of algorithms.

Polynomial time algorithms

We now give the formal definition of a polynomial time algorithm. Given a decision problem D defined over a set of instances $i \in I$, a polynomial time algorithm is a Turing machine which always terminates in a number of steps that is bounded by a polynomial in the size of the instance $|i|$. We also require that the algorithm is correct in the sense that the corresponding Turing Machine accepts on yes-instances $\{i \in I : D(i) = \text{yes}\}$ and rejects no-instances. The class of decision problems that admit a polynomial time algorithm is called $\mathbf{P}$.

The class $\mathbf{P}$ is often referred to as the set of problems that are easy to solve. This analogy is driven by two separate phenomena:

- even if one may fear that a problem would require a running time that is a very large polynomial, e.g. $O(n^{200})$, it has so far been the case that once a polynomial time algorithm has been designed and analyzed for a given problem, several independent teams of researchers design new algorithms whose running time is more modest, e.g. $O(n^7)$,

- almost all problems in $\mathbf{P}$ can be solved efficiently in practice.

However, it is still possible to argue that the cultural biases of the research community lead to the study of problems which exhibit enough structure to inspire efficient algorithms.

All the example problems presented in the previous sections are in $\mathbf{P}$: deciding whether a graph is connected, deciding whether the Rayleigh quotient of a given matrix is upper bounded by a test value, but also the problem of 2-edge connectivity that we briefly mentioned.

Easy optimization problems

By abuse of notation, we often say that an optimization problem O is easy or even that $O \in \mathbf{P}$ to denote that the natural decision problem D_O associated with O admits a polynomial time algorithm $D_O \in \mathbf{P}$. This abuse of notation is promoted by the common situation where algorithms effectively construct optimal solutions to decide an instance. Formally speaking, $\mathbf{PO}$ the class of optimization problems which admit polynomial time algorithms is related to $\mathbf{P}$ but is not entirely comparable to it as we previously discussed.

A major achievement in the theory of optimization is the proof by [Kha80] that the ellipsoid algorithm can be used to find solutions for linear optimization problems in polynomial time.

2.2.2 NP: efficient verification with short proofs

We have seen an example of a problem for which there exist witnesses or certificates that enable verification by algorithms called verifiers. To qualify the set of problems that admit efficient verification we need to determine the runtime of a given verifier but also the size of the certificates it can use. Just like problems in **P** admit algorithms that correctly return solutions efficiently, problems in **NP** admit algorithms that correctly check solutions efficiently. More formally, the class of decision problems that admit a polynomial time verifier using certificates of polynomial size is called **NP**. As we saw earlier, algorithms that solve a problem can be used as verifiers with an empty proof, i.e. $\mathbf{P} \subseteq \mathbf{NP}$.

Easily verifiable optimization problems

In the context of optimization, the class **NP** holds practical importance because a large number of natural optimization problems have associated decision problems which can be efficiently decided. Indeed, observe that a solution x^* of an optimization problem is by definition of polynomial size. Furthermore, if we consider a reasonable class of optimization problems O' which have an efficiently computable metric function whose codomain is countable, as well as efficient membership testing of their feasible region, then we can see that x^* is in fact a certificate and the verifier consists simply in testing that $x^* \in F(i)$ and $r(x^*) \lesseqgtr b$.

Now, let us review the trivial statement that for any optimization problem O we have $D_O \leq_T O$. The reduction we proposed simply used x^*, the output of alg_O to evaluate $m(x^*) \lesseqgtr b$. If we restrict it to a "nice" optimization problem O' with the aforementioned properties then it is easy to see that we constructed a Karp reduction, i.e. not only $D_{O'} \leq_T^P O'$ but also:

$$D_{O'} \leq_p O' \tag{2.3}$$

Hardness and completeness

It might occur that a problem P is expressive enough to subsume an entire class of problems **C**. Conversely, this means that P has to be as hard to solve as all of the problems it subsumes. Such problems are called **C**-hard which means

"at least as hard as **C** as a whole". Formally, this means that every problem of **C** is reducible to that particular problem. In the context of computational complexity, we consider reductions with respect to the use of limited resources such as time, space, or randomness, and as such we require that the reduction should not exceed the computational power of **C** which we denote as $\leq_T^{\mathbf{C}}$.

$$P \text{ is } \mathbf{C}\text{-hard} \iff \forall L \in \mathbf{C}, L \leq_T^{\mathbf{C}} P.$$

In some cases, a complexity class **C** is unified by a set of reductions to one of its members. A **C**-complete problem is a **C**-hard problem that also belongs to **C**.

NP-completeness and NP-hardness

Two major results in computational complexity regarding **NP** are the **NP**-completeness of the circuit satisfaction problem proven by Cook in [Coo71]:

$$\forall D \in \mathbf{NP}, D \leq_T^P \textsc{Circuit satisfaction}.$$

and the **NP**-completeness of 21 problems established by Karp in [Kar72]. This was done by establishing that the circuit satisfaction problem is reducible to each decision problem. Some of these problems are decision problem associated with classical combinatorial optimization problems such as the maximum clique problem in graph theory, the minimum set cover problem, or the maximum cut problem.

$$\forall D \in K_{21}, \textsc{Circuit satisfaction} \leq_T^P D. \tag{2.4}$$

where K_{21} is the set of the 21 problems considered by Karp.

This result combined with the standard reduction from an optimization problem to its associated decision problem established that well-known combinatorial optimization problems were in fact **NP**-hard. Since that discovery, an overwhelming number of optimization problems have been proven to be **NP**-hard. The most direct approach to establish $\mathbf{P} = \mathbf{NP}$ is to design a polynomial time algorithm for a particular **NP**-hard problem. Despite numerous attempts, there has so far been no evidence that it could be possible. Given the practical importance of solving optimization problems, complexity theorists and practitioners have attempted to find ways to bypass this obstacle by broadening the definition of the word "solve", either by considering algorithms with access to additional resources, or even by allowing suboptimal solutions.

2.2.3 BPP: probabilistic correctness

Several generalizations of efficiency have been devised by researchers in computational complexity to include access to various resources. As many algorithms use randomness to construct solutions or to answer questions, some extensions of **P** take into account the notion of efficiency with regard to the use of random bits as introduced by Gill in [Gil77]. In this context, a randomized algorithm is not assumed to always be correct, even on the same input.

Deciding in polynomial time, often correctly

Keeping in mind the idea that time is the scarce resource, we require the Turing machine to terminate in a bounded number of steps but we allow for non-correctness in the following sense: the Turing Machine must accept yes-instances with probability at least $2/3$ (or any constant strictly greater than $1/2$). These algorithms are usually called Monte Carlo by opposition to the Las Vegas algorithms which are randomized algorithms with guaranteed correctness but only polynomial running time in expectation. The class **BPP**, for bounded-error probabilistic polynomial time, consists of decision problems which admit randomized algorithms which are polynomial time and are incorrect at most $1/3$ of the time.

The key property of such probabilistic algorithms is that their success rate can be amplified by independent repetition. We will consider pralg a probabilistic algorithm which correctly solves a decision problem D with probability at least $2/3$. This means that given an instance $i \in I$, we can define a random variable $\boldsymbol{x}$ as follows:

$$\boldsymbol{x} = \begin{cases} 1 & \text{if } i \text{ is a } a\text{-instance and } \mathrm{pralg}(i) = a \text{ where } a \in \{\text{yes}, \text{no}\} \\ 0 & \text{otherwise.} \end{cases}$$

By definition, $\boldsymbol{x}$ follows a Bernoulli distribution with parameter $p \geq 2/3$, that is $\Pr(\boldsymbol{x} = 1) \geq 2/3$. We can now simply run our probabilistic algorithm k times and return the majority answer.

In Algorithm 3 we denote by $\boldsymbol{a}_j$ the answer returned by $\mathrm{pralg}(i)$ which makes its analysis straightforward. We consider the $\boldsymbol{x}_j$ as a collection of k independent and identically distributed random variables with mean p. The sum of these j random variables has mean $\mu = kp$. The Chernoff bound, which will be a topic in Chapter 3, states that the probability of deviation of this sum

Algorithm 3: SAMPLE MAJORITY

Data: an instance $i \in I$ of a decision problem D,
and a number of repetitions $k \in \mathbb{N}_+$
Result: True (yes) or False (no)

```
Sample ∀j ∈ [k], a_j ~ pralg(i)
return (1/k) Σ_{j=1}^{k} a_j > 1/2
```

from its mean decreases exponentially fast:

$$\Pr\left(\sum_{j=1}^{k} \boldsymbol{x}_j < (1-\varepsilon)\mu\right) < \exp(-\varepsilon^2\mu/2).$$

Choosing $\varepsilon = 1/4$ we directly obtain that the probability that our majority algorithm makes a mistake is at most:

$$\Pr\left(\sum_{j=1}^{k} \boldsymbol{x}_j < \frac{k}{2}\right) < \exp(-kp/32) \leq \exp(-k/48)$$

where the last step comes from $p \geq 2/3$. This means that from an error probability of $1/3$ we have obtained an error probability no larger than $\exp(-k/48)$. In other words, this means that running our probabilistic algorithm pralg $O(\log n)$ times is enough to guarantee that the sample majority algorithm returns a correct answer with high probability, that is at least $1 - 1/n$. In essence, problems in **BPP** admit randomized algorithms whose error rate can be quickly reduced to a low value, which can be chosen in advance, at the cost of only moderate repetition.

The pervasiveness of probabilistic correctness

The concept of probabilistic correctness is also used when verifying solutions to problems. The most famous example of it is the **PCP** theorem by Arora et al. [AS98]. Standing for probabilistically checkable proofs, $\mathbf{PCP}_{c,s}[r,q]$ is a family of complexity classes which correspond to decision problems that admit a verifier using a certain number r of random bits as well as a certain number q of queries to a certificate which is not necessarily bounded in size. Furthermore, the verifier must accept a proof of a yes-instance with probability at least c (completeness) and mistakenly accept an incorrect proof of a no-instance with probability at most s (soundness). For a given pair (r,q), we

obtain a complexity class denoted by $\mathbf{PCP}[r, q]$ which is by convention a shorthand for $\mathbf{PCP}_{1,1/2}[r, q]$. Arora's surprising theorem states that a problem which admits an efficient deterministic verifier working with a short proof also admits an efficient probabilistic verifier working with a proof of potentially exponential length, albeit queried a constant number of times:

$$\mathbf{NP} = \mathbf{PCP}[0, \text{poly}(n)] = \mathbf{PCP}[O(\log n), O(1)] \tag{2.5}$$

where $\mathbf{PCP}[0, \text{poly}(n)]$ is the standard definition of $\mathbf{NP}$ that we have discussed earlier. This alternative definition of the class $\mathbf{NP}$ unveils the importance of randomness and outlines the relationship of $\mathbf{NP}$ with other classes. Indeed $\mathbf{P}$ and $\mathbf{BPP}$, among many other complexity classes, can be described via $\mathbf{PCP}$. A deterministic algorithm which solves a decision problem requires no access to a proof, i.e. access to an empty proof, and no randomness:

$$\mathbf{P} = \mathbf{PCP}[0, 0]. \tag{2.6}$$

Furthermore, a probabilistic algorithm which can use a polynomial amount of randomness and fail at most $1/3$ of the time gives the following equality:

$$\mathbf{BPP} = \mathbf{PCP}_{c=2/3,s=1/3}[\text{poly}(n), 0]. \tag{2.7}$$

Recall our earlier example of using an optimal solution x^* as a certificate for the decision problem associated with a $\mathbf{NP}$-hard optimization problem where the deterministic verifier corresponds to solving a membership problem $x^* \in F(i)$ and testing the bound $m(x^*) \lesseqgtr b$. The $\mathbf{PCP}$ theorem tells us that there is another certificate, much larger than x^*, and a probabilistic verifier which correctly checks the certificate at least $2/3$ of the times by querying a constant number of bits of the proof.

2.3 Approximation

Optimization problems have a particular geometric flavor. It is easy to imagine that an optimization problem implicitly asks for an algorithm which has to explore a space of feasible solutions. Faced with the bad news that most optimization problems are $\mathbf{NP}$-hard, it is legitimate to believe that it might be possible for some problems to efficiently explore some of the space of their feasible solutions and obtain good, albeit suboptimal, solutions. Indeed, this image is the inspiration for the design of many approximation algorithms as can be seen for example in the book by Williamson and Shmoys [WS11]; [GM12].

In this section we will explore the formal meaning of efficient approximation.

When relaxing the assumption that a solution must be optimal, it is implicitly assumed that a suboptimal solution should be at least "good enough". In the context of approximation algorithms, this corresponds to the notion of proving that a solution is not far, in the sense of the objective function, from the optimal solution value. In line with the conservative approach of worst-case analysis, we will discuss worst-case a priori guarantees which constitute the most common approach used to analyze approximation algorithms.

It is also possible to relax the assumption that an algorithm must always return a feasible solution. While there are multiple ways to generalize this guarantee, we will focus on the feasibility analogue to probabilistic correctness. In essence, a randomized algorithm designed to solve optimization problems might sometimes return an infeasible solution. In an effort towards characterizing efficient computation, we will require similar error rates as the ones described for $\mathbf{BPP}$.

Finally, we will see how we can combine these two approaches to obtain a definition of efficient randomized approximation algorithms.

2.3.1 Worst case *a priori* guarantees

Originally motivated by the conjectured $\mathbf{P} \subset \mathbf{NP}$, researchers have attempted to design algorithms for which analysis certifies that they return solutions that are guaranteed to be good. Subsequently, this research program has evolved towards the exploration of the possible compromises between the running time of an algorithm and the quality of its solutions.

Approximation ratio

Using a worst-case approach similar in spirit to the classical approach for the running time of algorithms, we give a formal definition of an approximation algorithm. Given an instance $i \in I$ of an optimization problem $O = (I, F, m, \mathrm{opt})$, we say that an algorithm is a $f(i)$-approximation algorithm when it always returns a feasible solution whose value is within a factor $f(i)$ of the optimal solution value. By convention, we consider that $f(i) \geq 1$ which means that if O is a maximization problem, i.e. $\mathrm{opt} = \max$, any solution x returned by the algorithm have value at least $m(x) \geq m(x^*)/f(i)$. Correspondingly, if $\mathrm{opt} = \min$ then we have $m(x) \leq f(i)\, m(x^*)$. In both cases, this allows us to define the approximation ratio obtained by an algorithm alg producing a feasible solution

x as:

$$r_{\text{alg}} = \max_{i \in I} \max \left(\frac{m(x)}{m(x^*)}, \frac{m(x^*)}{m(x)} \right). \tag{2.8}$$

It is clear that this ratio generally does not have a closed-form expression, nor can it be efficiently computed. In practice, the analysis of a given algorithm will attempt to establish an upper bound on this approximation ratio, with the hope that the ratio be as close to 1 as possible.

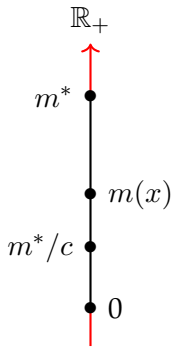

Figure 2.1: The value of the solution returned by an approximation algorithm for a maximization problem

Figure 2.1 is an illustration of the value of a solution x returned by a c-approximation algorithm for an optimization for which we assume that the codomain of the objective function is non-negative, that is $m : S \to \mathbb{R}_+$. The parts of the axis drawn in red describe zones where there cannot exist any feasible solution i.e. no solution exists neither above m^*, by maximality nor below 0, by assumption.

Approximation classes

It is possible to distinguish problems within **P** by their worst-case running time complexity, e.g. all problems which admit a $O(\log n)$ algorithm, or an $O(n)$ algorithm, then a $O(n^2)$ algorithm, etc. Similarly, **NP**-hard optimization problems can be classified according to the existence of a c-approximation algorithm for some constant $c > 1$, or problems which admit a $O(\log n)$-approximation, then a $O(\sqrt{n})$-approximation, and so on. This gives birth to a family of complexity classes called approximation classes.

An approximation class that is in essence very close to efficient exact optimization **PO** is the class **FPTAS** which describes the set of optimization

problems that admit fast algorithms whose approximation ratio can be controlled by an additional input. These algorithms are approximation schemes, i.e. for any desired precision $\varepsilon > 0$, they produce $(1+\varepsilon)$-approximate solutions. Furthermore, a fully polynomial time approximation scheme is an algorithm which runs in time bounded by a polynomial in both n the instance size and $1/\varepsilon$ the required precision. It is known that if $\mathbf{P} \neq \mathbf{NP}$ then:

$$\mathbf{PO} \subsetneq \mathbf{FPTAS} \tag{2.9}$$

Assuming this conjecture, it is established that the vast majority of **NP**-hard optimization problems do not admit approximation schemes, even less powerful versions where the running time of the algorithm is only polynomial in n, i.e. potentially with an exponential or worse dependence on $1/\varepsilon$. Before we move on to the next category of approximation classes, let us mention a few well-known optimization problems which admit approximation schemes. The most famous example of a problem in **FPTAS** is most assuredly the knapsack problem which was one of the 21 **NP**-hard problem analyzed by Karp in his seminal paper. One can also interpret the proof by Kachiyan [Kha80] that linear optimization is in **PO** to mean that the ellipsoid method when used to solve linear optimization problems is an approximation scheme which run in time polynomial in n but also polynomial in $\log(1/\varepsilon)$. This algorithm enjoys an even stronger guarantee than a standard fully polynomial time approximation scheme. The higher degree of precision allows the use of a rounding algorithm which transforms a $(1+\varepsilon)$-approximate solution into an exact solution, establishing that linear optimization is in **PO**.

The vast majority of optimization problems studied in the literature only admit approximation ratios that are fixed, in the sense that there is no algorithm with controllable precision. This is represented by **APX** which, like **PCP**, is a family of complexity classes parameterized by a value. For example, the class $O(1)$-**APX**, sometimes simply denoted by **APX**, is the class of all optimization problems which admit a constant factor approximation algorithm. Among Karp's 21 problems, the maximum cut problem for example is known to be in $O(1)$-**APX**, see for example the randomized approximation algorithm given by Goemans and Williamson [GW95] based on a SDP relaxation or previous approaches based on a LP relaxation [VK07]. But there is also $\log$ **APX**, poly**APX**, and beyond. Again, if we assume that $\mathbf{P} \neq \mathbf{NP}$ then we have a succession of proper inclusions:

$$\mathbf{PO} \subsetneq \mathbf{FPTAS} \subsetneq \mathbf{APX} \subsetneq \log\text{-}\mathbf{APX} \subsetneq \mathrm{poly}\text{-}\mathbf{APX}. \tag{2.10}$$

which can be interpreted as the following statement. If we can guarantee a precise approximation for a given problem then we can "forget" this precision

and provide a more coarse bound on the quality of solutions which can be efficiently constructed. In simpler words, a precise algorithm is trivially at least as good as a less precise algorithm. Conversely, being able to provide a coarse approximation does not help in general with constructing precise approximations.

This invites us to consider the hardness analogue of approximation classes. Recall that the reductions used to define a class must not be more powerful than the resources a class has access to. In the case of **NP** for example, it meant that reductions had to be polynomial time algorithms.

In the context of efficient approximation, we also consider guarantees as a resource. This justifies the following definition: we say that an optimization problem O_A is reducible to another optimization problem O_B if there exists a pair of reductions combining:

- a polynomial time algorithm which transforms an instance of O_A into an instance of O_B
- a polynomial time algorithm which transforms a feasible solution x_B for O_B into a feasible solution x_A for O_A.

Furthermore, we qualify this pair of reductions to be approximation-preserving in the sense that a bound on the value of a solution x_B implies some other bound on the value of a solution x_A.

More precise versions of approximation-preserving reductions allow for the definition of approximation hardness for the classes we have mentioned in this section. This can lead to theorems such as: the maximum cut problem is **APX**-hard [PY91] and furthermore it is **APX**-complete. In essence, the **APX**-hardness of an optimization problem establishes that there exists no algorithm which provides a variable approximation factor. If we consider another example, the minimum set cover problem is $\log$-**APX**-complete and as such the assumption that $\mathbf{P} \neq \mathbf{NP}$ denies the existence of any constant factor approximation for this problem.

We will now consider relaxing perfect feasibility by studying randomized algorithms for optimization problems.

2.3.2 Probabilistic feasibility

We have seen earlier that **BPP** is the class of problems which admit an efficient algorithm which returns a correct answer with high probability. A simple variation of this definition extends the concept of probabilistic correctness to the domain of optimization. Indeed we say that a problem is in **BPPO** if it admits

a randomized algorithm which may return infeasible solutions, albeit with probability no larger than $1/3$. Like earlier, this means that such an algorithm can be boosted by repetition to obtain feasibility with high probability. Formally, if we denote by $\boldsymbol{x}$ the random variable which corresponds to the output of a randomized algorithm pralg run on instance $i \in I$ with $|i| = n$ and $F(i)$ the feasible region, then pralg is an efficient randomized algorithm if we have:

$$\mathbb{P}(\boldsymbol{x} \in F(i)) \geq 1 - \frac{1}{n}. \tag{2.11}$$

We will now see how to these two concepts can be combined to obtain efficient randomized approximation algorithms. Indeed, if we have an efficient randomized algorithm pralg for a given optimization problem $O = (I, F, m, \mathrm{opt})$ and whose output is a random solution $\boldsymbol{x}$ satisfying property 2.11 then we can define the expected approximation ratio of pralg. The formula closely resembles the deterministic approximation ratio, except for the value of the approximate solution which is replaced by the expected value of the random solution. Like earlier, we denote by m the metric function of the optimization problem:

$$r^{\mathbb{E}}_{\mathrm{pralg}} = \max_{i \in I} \mathbb{E}\, \frac{m(\boldsymbol{x})}{m(x^*)} \tag{2.12}$$

We now have all the required notions to present the relaxation and randomized rounding framework which is a generic strategy to design randomized approximation algorithms for **NP**-hard optimization problems. This is the design strategy that we will employ in this thesis.

2.3.3 Relaxation and rounding

We describe the relaxation and rounding framework informally because it admits many variants. For simplicity we consider maximization problems but it is easy to reverse the construction for minimization problems. In essence, given a maximization problem $O = (I, F, m, \max)$, this framework invites us to consider the following algorithm:

1. Design a polynomial time algorithm which transforms problem O into another maximization problem $O' = (I, F', m, \max)$.
2. Prove that the O' is a **relaxation** of O, that is

   $$\forall i \in I, F(i) \subseteq F'(i).$$

   and that if we denote by the relaxed feasible region $S' = \{F'(i) : i \in I\}$ then the objective function is properly defined on it, i.e. $m : S' \to \mathbb{Q}_+$.

3. Prove that O' can be efficiently solved in some manner, e.g. $O' \in \mathbf{PO}$, $O' \in \mathbf{BPPO}$, $O' \in \mathbf{FPTAS}$, or $O' \in \mathbf{APX}$. We denote the corresponding (randomized) algorithm by $\mathrm{alg}_{\mathrm{relax}}$.

4. Prove an upper bound of the form:

$$m(x^*) \leq m(x')$$

   where x^* is an optimal solution of O and x' of O' is a efficiently computable solution of O'.

5. Design a polynomial time (randomized) algorithm round which transforms a relaxed solution $x' \in F'(i)$ into a solution $x \in F(i)$. In combinatorial optimization $F'(i)$ is often a continuous space while $F(i)$ is a discrete space, which is why it is called the **rounding** step.

6. Prove that $m(x) \geq m(x')/f(i)$ for some efficiently computable function $f : I \to \mathbb{Q}_+$.

We summarize the bounds that must be established in Figure 2.2. For simplicity we have also assumed that the objective function has a non-negative codomain $m : S' \to \mathbb{Q}_+$. We adopt the following convention: values to the left of the axis cannot be computed efficiently unless $\mathbf{P} = \mathbf{NP}$ whereas values to the right of the axis are efficiently computable. From the figure it is clear that x is a $f(i)$-approximate solution which can be computed efficiently by the following Algorithm 4:

Algorithm 4: GENERIC RELAXATION & ROUNDING

Data: an instance $i \in I$ of O
Result: a $f(i)$-approximate feasible solution of O

```
/* Construct the relaxed optimization problem        */
```
1 $O' = \mathrm{relax}(O, i)$
```
/* Solve it on instance i                            */
```
2 $x' = \mathrm{alg}_{\mathrm{relax}}(O'(i))$
```
/* Round the solution                                */
```
3 $x = \mathrm{round}(x')$
4 **return** x

This very general strategy has been used to design powerful approximation algorithms for **NP**-hard optimization problems for which the approximation ratio is either the best known or actually matches a corresponding impossibility

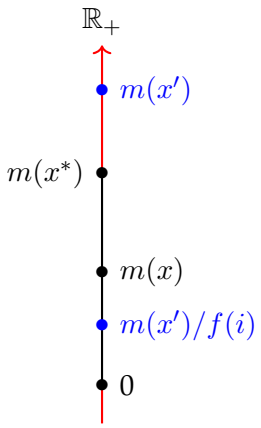

Figure 2.2: The value of the solution returned by an relaxation and rounding algorithm for a maximization problem

theorem. In the next section, we will detail the analysis of an approximation algorithm for a **NP**-hard optimization problem, the unique coverage problem. Then, we will sketch a hardness proof which states that the approximation ratio we have obtained is tight if we assume a plausible computational complexity conjecture. This example is meant to illustrate the critical phenomenon of hardness in approximation.

2.4 Critical thresholds in hardness of approximation

In the context of approximation, the goal of computational complexity is to determine the best ratio for which there exists a polynomial time algorithm. In essence, one can view the approximation ratio of an optimization problem as a tipping point between two phases of matter. In physics, water is known to abruptly switch from a liquid to a gaseous phase when the temperature exceeds 100°C under normal conditions of pressure. In hardness of approximation, the minimum set cover problem goes from being solvable in polynomial time to being **NP**-hard when the a priori guarantee we request becomes strictly smaller than $\log n$. To further the analogy with phase transitions, just like in condensed-matter physics, only a handful of problems have precisely identified threshold.

We first give a few representative examples of current results in hardness of approximation. Then we focus on a single example which we will study in more depth. The unique coverage problem will serve as an illustration

of a common phenomenon that can be observed among many optimization problems. We will start by describing the problem statement and quickly discuss its practical importance. Then, we will show that this optimization problem admits a randomized $O(\log m)$-approximation algorithm where m denotes the number of possible subsets to choose from in the instance. Finally, we give the sketch of a proof that this ratio is essentially the best possible assuming plausible computational complexity conjectures.

2.4.1 A varied landscape

In the introduction of this section we have contrasted the complete knowledge of the threshold for which the minimum set cover problem becomes hard to approximate with the caveat that few problems are as well understood as this one.

However, it is sometimes the case that stronger conjectures than $\mathbf{P} \neq \mathbf{NP}$ imply a more precise landscape of hardness phase transitions. Indeed, consider the generic approximation algorithm given by the sum of squares method of Parrilo [Par03] or its optimization dual the moment hierarchy of Lasserre [Las01]. This abstract algorithm, which is a particular case of the relaxation and rounding framework we have described above, applies to the maximization variants of the large class of constraint satisfaction problems. This class of problems includes among many others the maximum cut problem, the maximum satisfiability problem, and the maximum coverage problem. Surprisingly, the sum of squares algorithm produces approximation ratios which match the best known algorithms that had been designed specifically for each individual optimization problem. Furthermore, if we assume the unique games conjecture [Kho02], which is a much stronger statement than $\mathbf{P} \neq \mathbf{NP}$, then these approximation ratios are the best possible [Rag08]. This highlights the power of both the sum of squares algorithm and the expressiveness of the unique games conjecture.

In contrast with the above results, some **NP**-hard optimization problems like the densest k-subgraph problem are far from being understood in the sense of approximation. The densest k-subgraph is defined as following. Since a subgraph G_S is defined over k nodes, its density $|E_S|/|V_S| = |E_S|/k$ is simply proportional to its number of edges.

DENSEST k-SUBGRAPH [FPK01]	
Instance:	An undirected graph $G = (V, E)$ and a size $k \in \mathbb{Z}$.
Goal:	max.
Metric:	The size of the subgraph $G[S] = (V_S, E_S)$: $m : S \mapsto \lvert E_S \rvert$.
Domain:	Subgraphs of order k: $S \subseteq V : \lvert S \rvert = k$.

It is almost saddening to learn that the best known approximation algorithms for the densest k-subgraph problem produce $O(\sqrt[4]{n})$-approximate solutions while the corresponding impossibility theorem assuming a plausible complexity conjecture states that the problem does not admit a polynomial time approximation scheme. If we review the approximation classes we presented earlier, this means that anything from a constant-factor approximation to a $\sqrt[4]{n}$-approximation could be the best possible. Even when considering stronger conjectures, the gap between the current best approximation ratio and the current best impossibility theorem remains wide open.

In a somewhat unexpected turn of events, the seemingly innocuous extension of the problem to larger domain of subgraphs of order at least k has profound effects on our understanding of the approximation complexity of the problem. This variant we have just defined, the densest at-least-k-subgraph problem, admits a 2-approximation algorithm which is the best possible, assuming a variant of the unique games conjecture.

These observations point us towards the possibility that hardness of approximation, like many critical systems, is very sensitive to the specificity of a given problem. Indeed, for many optimization problems, it is rarely trivial to establish any result, be it positive or negative. To illustrate the depth of this discipline, we have chosen to present the design and analysis of a randomized approximation algorithm for the unique coverage problem given by Demaine et al. [Dem+08]. Their algorithm illustrates some of the concepts that we have discussed so far, and ideally serves as a helpful tutorial on the design of approximation algorithms.

2.4.2 The unique coverage problem

The unique coverage problem is a variant of several optimization problems related to set covers. The most well-known being the minimum set cover problem, which was part of Karp's 21 **NP**-hard problems, as well as the maximum coverage problem which can be seen as the "opposite" of the set cover problem. All set cover problems are concerned with instances defined by a set of n elements, $E = \{e_1, \ldots, e_n\}$, sometimes referred to as the universe,

and a set of m subsets of E. Generally speaking the optimization task is to obtain a large number of elements of the universe by selecting some of the m subsets. The task essentially asks to balance subsets of potentially different sizes which may or may not intersect each other.

All variants of these set covering problems aim at finding a good compromise between the number of elements covered, which should be high, and the number of subsets utilized, which should be low. Optimization problem variants make this balance more precise by setting one particular goal as a constraint which defines the feasible region while the other goal becomes the objective function. For example, the minimum partial set cover is a problem where feasible solutions must cover at least k elements and the objective is to do so at a minimum cost, considering that each subset is equipped with a non-negative cost. On the other hand, the maximum coverage problem asks to maximize the number of covered elements, subject to the constraint that the total cost of selecting subsets is no more than some budget B.

Here, we study the unique coverage problem which introduces a variant of the objective function found in the maximum coverage problem: the elements that are covered by a single subset are the only ones which contribute to the total profit. The unique coverage problem is simply the task of finding how to maximize the number of uniquely covered elements.

UNIQUE COVERAGE [DEM+08]

Instance: a set of n elements $E = \{e_1, \ldots, e_n\}$, a set S of of m subsets of E, where $S = \{X_1, \ldots, X_m\}$, a profit for each element $p : E \to \mathbb{N}$.

Goal: max.

Metric: the total profit of uniquely covered elements:

$$m(C) = \sum_{\substack{e \in E \\ |\{X \in C : e \in X\}| = 1}} p(e)$$

Domain: any subset C of S, i.e. $C \in \mathcal{P}(S)$.

Set cover problems can be seen as the prototypical examples of resource allocation problems. They can be applied in practical settings where services must be deployed to provide some commodity to a set of clients. Indeed, the unique coverage problem originated from a practical optimization task in wireless networks where clients who are covered by more than one radio frequency experience a loss of quality of service. This leads to the natural objective of wanting to favor the deployment of base stations in such a way

that the system serves the largest possible number of clients without radio interference.

Aside from practical applications, set covers are also useful as tools in computational complexity. Indeed, sets and subsets are very flexible mathematical objects and can be used in reductions to play the role of more specific objects. As a consequence, establishing positive and negative results on the hardness of approximation of such problems produces ripple effects in the landscape of complexity.

2.4.3 A randomized $O(\log m)$-approximation algorithm

The unique coverage problem is a maximization variant of the minimum set cover problem which, unlike the maximum coverage problem, does not contain any budget constraint. The feature which makes the problem non-trivial is that the value of a covered element depends on how many times it is covered by the chosen subsets, i.e. only elements that are covered once contribute to the objective to maximize.

The following randomized approximation algorithm is an example of an algorithm which always returns a feasible solution but whose value may vary. This is a stronger guarantee than the randomized algorithms that we had been considering so far. In the context of approximation, we are however still looking for a guarantee on the expected value of a solution returned by such a randomized algorithm.

Algorithm 5: SAMPLE BEST GROUP

Data: a set of elements $E = \{e_1, \dots, e_n\}$, a set $S = \{X_1, \dots, X_m\}$ of subsets of E, and a profit $p : E \to \mathbb{N}$
Result: a random subset of S

`/* count how many times each element is covered */`
$m : \mathcal{P}(S) \times E \to \mathbb{N}$
$m : C, e \mapsto |\{X \in C \mid e \in X\}|$
`/* group elements by coverage on an exponential scale */`
$E^{(i)} = \{e \in E \mid m(S, e) \in [2^i, 2^{i+1})\}$
`/* find group with highest total profit */`
$\mu = \operatorname{argmax}_{i \in \mathbb{N}^*} \sum_{e \in E^{(i)}} p(e)$
`/* keep each subset with probability 1/2^μ */`
Sample $\forall X \in S, \boldsymbol{s}_X \sim \text{Ber}\,(1/2^\mu)$
$U = \{X \in S \mid \boldsymbol{s}_X = 1\}$
return U

Theorem 6. *(Randomized $O(\log m)$-approximation for the unique coverage problem)* *The expected value of a cover returned by Algorithm 5 is within $c/\log m$ of the optimal solution value of the unique coverage problem for some constant $c > 0$.*

Proof. The first step in analyzing the performance of this randomized approximation algorithm is to describe how profit is split among groups.

If we represent the cover system given as initial data as a bipartite graph $G = (E, S, \{eX \mid e \in E, X \in S, e \in X\})$ we can read the coverage of each element as the degrees of the nodes in partition E. Notice that the maximum degree of nodes in E is bounded: $\deg(e) \leq m$.

Thus, groups corresponds to nodes whose degree is in one of the intervals $[2^k, 2^{k+1})$ for some $k \in \mathbb{N}$. Since 2^k is at most m, we have created at most $\log m$ groups.

Given that the number of groups is at most $\log m$ it is easy to see that the group μ, defined to be the group of elements with highest total profit, contains at least a $1/\log m$ fraction of the total profit of all elements, that is:

$$\sum_{e \in E^{(\mu)}} p(e) \geq \frac{\sum_{e \in E} p(e)}{\log m} \tag{2.13}$$

That bound can be attained in the case where all groups possess an equal share of the total profit.

The second step of the proof is to quantify the number of uniquely covered elements in cover U. For this we will again focus on the group μ which has highest total profit. The sampling rule to construct cover U is a simple Bernoulli distribution where subsets are kept with probability $1/2^\mu$. We can now write down the probability that an element e in group μ is covered uniquely in the cover U obtained by the sampling procedure. The event that e is covered by one and only one set can be described as the union of k independent events where in each event one of the k subsets covers e and all remaining $k-1$ subsets have been discarded by the sampling procedure.

$$\Pr(m(U, e) = 1) = \overbrace{k \cdot}^{\text{ways}} \underbrace{(1/2^\mu)^1}_{\text{one success}} \overbrace{\cdot}^{\text{and}} \underbrace{(1 - 1/2^\mu)^{k-1}}_{\text{all other failures}}. \tag{2.14}$$

First, we know that $k \geq 2^\mu$ which gives us a lower bound on ways to select a single subset.

$$k \geq 2^\mu \tag{2.15}$$

Now we want to find a lower bound on the probability of $k-1$ subsets being dropped where k appears as an exponent of a probability. In this case, we use the upper bound $k \leq 2^{\mu+1} - 1$ or equivalently $k - 1 \leq 2^{\mu+1}$ to obtain:

$$(1 - 1/2^{\mu})^{k-1} \geq (1 - 1/2^{\mu})^{2^{\mu+1}} \tag{2.16}$$

Combining these two inequalities we obtain a lower bound on the exact probability depending only on μ:

$$\Pr(m(U, e) = 1) \geq 2^{\mu} \cdot 1/2^{\mu} \cdot (1 - 1/2^{\mu})^{2^{\mu}+1} \tag{2.17}$$

$$\Pr(m(U, e) = 1) \geq (1 - 1/2^{\mu})^{2^{\mu}+1} \tag{2.18}$$

Assuming that $\mu \geq 2$ since otherwise we would not need to remove subsets, we have the following:

$$\Pr(m(U, e) = 1) \geq (1 - 1/2^2)^5 = \frac{243}{1024} \approx 0.237 \tag{2.19}$$

This bound on the probability that any given element of that group is uniquely covered can be interpreted as an expected fraction of uniquely covered elements in group μ. Let $E_U = \{e \in E \,|\, m(U, e) = 1\}$ be the elements uniquely covered by U, we have that:

$$\mathbb{E} \sum_{e \in E_U} p(e) \geq \frac{243}{1024} \sum_{e \in E^{(\mu)}} p(e). \tag{2.20}$$

We now combine the above inequality with inequality 2.13 to obtain the desired guarantee on the performance of Algorithm 5, that is if we denote by p^*_{UC} the optimal solution value of the unique coverage problem, we have that:

$$\mathbb{E} \sum_{e \in E_U} p(e) \geq c \frac{p^*_{\text{UC}}}{\log m} \tag{2.21}$$

where we used the trivial statement that $\sum_{e \in E} p(e) \geq p^*_{\text{UC}}$ and set $c = 243/1024$. □

In the above proof, we have used what could be seen as very loose bounds. First, we have only considered uniquely covered elements within the group with highest profit. However, one might hope that U might cover uniquely some elements in other groups. Second, we have used a very coarse upper bound on the optimal solution value of the unique coverage problem: the case

where the instance data consists of mutually disjoint subsets i.e. returning the entire set of subsets would be optimal.

However, this seemingly loose analysis results in what is essentially the best possible guarantee under a plausible computational complexity assumption.

2.4.4 Hardness of approximating unique coverage

In this section, we give a sketch of the proof that the balanced bipartite stable problem is reducible to the unique coverage problem.

Bipartite independent sets

Bipartite independent set problems are a family of computational problems related to graph theory which have been used in many reductions to establish hardness of approximation. Demaine et al. [Dem+08] in particular highlight the fact that these problems can be modified easily to fit several computational complexity assumptions. At its core, a bipartite independent set problem is defined over a bipartite graph $G = (A \cup B, E)$ whose partitions are of equal size, i.e. $|A| = |B| = n$. An independent set in this context corresponds to two subsets of $I_A \subseteq A$ and $I_B \subseteq B$ such that the induced subgraph $G[I_A \cup I_B]$, or $G[I]$ for short, is an independent set, i.e. $\forall u, v \in V(G[I]), uv \notin E(G[I])$. We say that G admits an (a, b)-BIS if there exists I_A and I_B with $|I_A| = a$ and $|I_B| = b$.

As we briefly mentioned in the previous section, set cover problems can be interpreted as problems on bipartite graphs where the left partition represents the set of subsets and the right partition represents the set of elements. Furthermore, we can represent membership $e \in X$ by an edge eX. The proof of Demaine et al. is indeed a randomized reduction which makes use of this representation. As such, we continue to identify an instance of the unique coverage problem with the bipartite graph $U = (E \cup S, \{eX : e \in E, X \in S, e \in X\})$.

Promises and gaps

The idea is then to introduce a variant of the bipartite independent set problem where it is possible to quantify very precisely the size of the bipartite independent on each side of the partition. This kind of problem belongs to the category of promise problems which split instances in three sets:

- yes-instances which must be accepted,
- no-instances which must be rejected,
- and irrelevant-instances are the rest.

Promise problems are of special importance in hardness of approximation because they can be used to establish gaps. For example, consider a promise problem which separates instances of an maximization problem into yes-instances represented solutions with value at least g and no-instances that are at most b. When $g > b$ it suffices to prove that this promise problem is **NP**-hard to rule out the existence of any approximation algorithm with ratio better than g/b unless $\mathbf{P} \neq \mathbf{NP}$.

A randomized reduction

This is the strategy employed by Demaine et al. Here, they consider the following promise problem:

PROMISE OF BIPARTITE INDEPENDENT SET	
Instance:	a bipartite graph $G = (A \cup B, E)$ with $\lvert A\rvert = \lvert B\rvert = n$, size parameters for yes-instances and no-instances: $l_{\text{yes}}, r_{\text{yes}}, l_{\text{no}}, r_{\text{no}} \in (0, 1)$ with $l_{\text{no}} < l_{\text{yes}}$ and $r_{\text{yes}} < r_{\text{no}}$.
Accept:	G admits a $\left(n^{l_{\text{yes}}}, n/(\log n)^{r_{\text{yes}}}\right)$-BIS.
Reject:	G does not admit a $\left(n^{l_{\text{no}}}, n/(\log n)^{r_{\text{no}}}\right)$-BIS.

They then randomly construct an instance of the unique cover problem by constructing a random graph based on the bipartite graph $G = (A \cup B, E)$. The procedure is relatively technical but amounts to the following steps:

- Sample a random graph based on G where each edge independently remains with probability $n^{-l_{\text{yes}}}$. This ensures that every node in the random graph has degree $n^{1-l_{\text{yes}}}$ in expectation. The remaining edges are called good edges.
- Create $O((l_{\text{yes}} - l_{\text{no}}) \log n)$ copies of this graph and fuse all nodes in A. The result is bipartite graph which has a left partition with n nodes and a right partition of $O(n \log n)$ nodes.
- For every of the $O(\log n)$ copies, add an edge set between A and a copy B_i which corresponds to the original edge set E, and for every successive copy the edge set of their predecessor but with every edge being discarded with probability $1/2$. The remaining edges are called bad edges.

Figuratively speaking, the random graph thus created is an unbalanced bipartite graph with n nodes on the left and $O((l_{\text{yes}} - l_{\text{no}}) \log n)$ on the right. The

size parameters however have a great influence over the size of the random graph:

- the gap between the yes- and no-parameters of the left partition determine how many copies are created in the right partition,
- the absolute value of the yes-parameter of the left partition determines the density of the bipartite random graph.

This construction attempts to identify the unbalanced bipartite independent sets in G with elements that are uniquely covered by good edges.

In the case of a yes-instance, G admit a large bipartite independent set which we denote by $I^* = I_A^*, I_B^*$ where I^* is a $\left(n^{l_{\text{yes}}}, n/(\log n)^{r_{\text{yes}}}\right)$-BIS. In this case, it is possible to count the number of good edges coming out of I_A^*: there are $n^{l_{\text{yes}}}$ nodes and each node has an expected good degree $n^{1-l_{\text{yes}}}$, this means that in expectation n good edges exit I_A^*. It is then possible to show that a significant fraction of elements are uniquely covered by good edges, namely at least a $\Omega(n/(\log n)^{r_{\text{yes}}})$ fraction of each of the $O((l_{\text{yes}} - l_{\text{no}}) \log n)$ copies. This establishes the following. If we denote by u^* the optimal solution of the unique coverage problem and $f(l)$ a constant term depending on the left-parameters, then with high probability we have:

$$G \text{ is a yes-instance } \implies u^* \geq f(l)\, n(\log n)^{(1-r_{\text{yes}})}.$$

In the case of a no-instance, G does not admit a large bipartite independent set. This can be used to show that in every graph copy on the right side of the unique coverage instance, no more than a small fraction of elements can be covered uniquely. With additional technical details, it is possible to show the following:

$$G \text{ is a no-instance } \implies u^* \leq f(l)\, n(\log n)^{(1-r_{\text{no}})}.$$

These two results produce a hardness gap of value $\Omega(1/(\log n)^{(r_{\text{yes}}-r_{\text{no}})})$

To complete the hardness proof, we are left to find the adequate parameters l and r which suffice to prove that the promise of bipartite independent set problem is hard.

Summary

In this chapter, we have explored several fundamental concepts in the theory of algorithms and in computational complexity with a strong focus on optimization problems. The main observation is that approximation sometimes exhibit

critical behavior and that whether a problem can be solved efficiently can depend on a small number of parameters, here for example the approximation ratio.

The picture we have painted might ambiguous as it is not clear if knowledge on efficient algorithms or hardness proofs can transfer directly to the optimization problem we are interested in. However, while individual results might stay confined to a particular problem, the general methodology of both computational complexity and the analysis of algorithms are the tools that we believe are appropriate to study a problem in its full generality.

In the context of the defense against epidemics, we offer two complementary points of view on how to use approximation algorithms and hardness proofs in practice. First, establishing approximation guarantee for an algorithm should be seen as an early indicator that an approach is worth pursuing, maybe to be combined with heuristics or human expertise. On the other hand, hardness proofs should be welcomed as warning signs. As is well-known, the worst-case analysis methodology does not reflect the reality of instances solved in practice, but it can serve as a windmill which signals a coming storm.

Now that we have established the reason why we want to solve an optimization problem, and presented the methodology necessary to establish an appropriate solution, we must turn to the mathematical objects most likely to help us design and analyze our wanted algorithm.

The question becomes "Which concept can become an efficient constructive algorithm for our problem?"

3 Random graphs

Introduction

The whodunit, which stands for "who has done it?", is the most prominent subgenre of detective stories. It usually describes the investigation around a crime, and is focused on the the goal, shared between the detective and the reader, of finding the identity of the perpetrator. In this thesis, we need to find the most likely candidate, within the world of mathematics, of the disappearance of an epidemic. Recall that to suppress the spread of an epidemics, we are looking for an efficient algorithm which would modify the spectral properties of the graph underlying the contact network of an epidemic model. In this context, we focus our attention on the influence of randomness on the spectrum of graphs.

This chapter is an introduction to random graphs as well as a tutorial on the use of the concentration of measure to construct complex combinatorial objects through sampling. In order to give a concrete example of the concepts and tools underlying the results presented in this thesis, we will derive a simple proof of a sufficient condition for the $G(n, p)$ random graph model to be connected. This example will serve at the same time as an illustration of the concentration of measure phenomenon as well as a proof of the effectiveness of spectral graph theory, that is using linear algebra to understand properties on graphs.

We start by giving a short history of the concept of concentration of measure and how it was adopted in combinatorics and theoretical computer science. Then, we present the most famous concentration inequality, the Chernoff bound, and give a quick sketch of its proof. Subsequently, we detail the proof of Tropp's matrix Chernoff bound which is a direct generalization of the Chernoff bound for random matrices. This serves not only as evidence of the generality of the concentration of measure phenomenon but also provides us with adequate tools to study random graphs via spectral graph theory. We take a short break from probability theory to recall basic results from spectral graph theory in order to facilitate the exposition of the next section on the $G(n, p)$ random graph model. Finally, we apply all the tools and techniques that we have introduced to derive a sufficient condition for a random graph to be connected. This is a constructive version of the well-known theorem on the

connectivity threshold of random graphs.

The final goal of this chapter is to establish the existence of a phase transition in connectivity for random graphs obtained from the $G(n, p)$ model at $p = O(\log n/n)$. Below an expected degree of $O(\log n)$, connectivity does not concentrate whereas above this threshold almost every random graph is connected.

3.1 The concentration of measure phenomenon

The foundations of probability theory have focused on general results regarding the asymptotic convergence of empirical distributions to fixed distributions. In the 19$^{\text{th}}$ century, Pierre-Simon Laplace obtained a good approximation of the number of heads obtained after flipping a fair coin a large number of times. This was an early application of an earlier result by Abraham de Moivre in the late 18$^{\text{th}}$ century which would then receive particular attention and formal proofs. Indeed, the central limit theorem, established in 1901 by Aleksandr Lyapunov, describes the convergence in distribution of a sum of independent and identically distributed (henceforth i.i.d.) random variables to a normal distribution of the same variance. Similar theorems pertaining to different kinds of distributions were obtained throughout the 20$^{\text{th}}$ century. They shone light on so-called asymptotic distributions i.e. the limit behavior of a large number n of random variables as n goes to infinity. No matter how informative, such theorems cannot be directly turned into randomized algorithms as they do not provide a precise picture of what happens for a particular value of n.

While a large part of classical probability theory has focused on establishing similar asymptotic convergence results, a small field of research focused on analyzing a large but finite number of random variables, often in geometrical settings. This line of work can be traced to a theorem by Paul Lévy in 1919 which states that continuous multivariate functions on the n-dimensional unit sphere which do not depend greatly on individual coordinates are almost constant. Later, in 1970, Vitali Milman gave a simple proof of Dvoretsky's theorem on the existence of ellipsoids within any convex body of sufficiently high dimension by applying Lévy's result to random subspaces of the convex body. His finding revealed that the vast majority of random subspaces were almost ellipsoids. This was the first application of what would be called the concentration of measure phenomenon.

Following this event of far-reaching cultural importance in the world of mathematics, several mathematicians including Michel Ledoux [Led01] and Michel Talagrand [Tal+96] at the turn of the 21$^{\text{st}}$ century worked on establish-

ing concentration of measure for a wide variety of probabilistic objects. The common structure among those objects is simply that they can be characterized by a large number of random variables which are weakly dependent if not independent from one another.

In practice, the theorems that have been established within the concentration of measure framework provide powerful non-asymptotic information on the behavior of mostly regular multivariate functions. They take the form of general inequalities that can be tuned to fit a particular mathematical situation and as such, they are applied in many settings. In particular, concentration inequalities have had a deep impact in the fields of combinatorics and the theory of algorithms. Broadly speaking, asymptotic theorems have been used to prove the existence of certain combinatorial objects, see for example the book by Alon and Spencer [AS04]. On the other hand, non-asymptotic theorems have been successfully used to construct combinatorial objects via randomized algorithms, see for example Motwani and Rhagavan's book [MR10]. We will now see two examples of these concentration inequalities which are of practical importance in the design of randomized algorithms.

3.2 Concentration inequalities

In this section we will draw parallels between an elementary result in non-asymptotic probability theory, the Chernoff bound, and a more advanced generalization of this theorem in the context of random matrices.

The Chernoff bound had been established by Herman Rubin, then in a weaker version by Herman Chernoff in 1952 [Che+52]. Both were inspired by previous work by Bernstein who derived the closely related Bernstein bounds [Ber24] in the early 1920s. However, the Chernoff bound only gained fame after a result by Claude Shannon depending on a erroneous application of the central limit theorem was salvaged by this new bound. In the aftermath of this event, the bound became known as the Chernoff bound, as described in Chernoff's autobiographical account of the events [Lin+14]. The matrix Chernoff bound on the other hand is a recent result obtained by Joel A. Tropp in 2010 [Tro12] in the wake of his discovery of inequalities applying to the matrix analogue of the Laplace transform of a sum of random variables.

3.2.1 The classical Chernoff bound

The classical Chernoff bound is a major tool in non-asymptotic analysis. It describes how a sum of independent non-negative random variables which

share a common upper bound behave when they are summed together. In other words, this bound describes that it is unlikely for a sum of random variables to be far apart from its expected value and that the chance of this event happening decreases exponentially relative to the distance between the random variable and its expectation.

Theorem 7. *(The scalar Chernoff bound) [Ber24]*
Let $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m$ be independent, scalar random variables that are non-negative and bounded:

$$\forall i \in [n], 0 \leq \boldsymbol{x}_i \leq u$$

and denote their sum by $\boldsymbol{x} = \sum_{i=1}^{m} \boldsymbol{x}_i$ with

$$\mu_{\min} \leq \mathbb{E}\,\boldsymbol{x} \leq \mu_{\max}$$

then for any $\varepsilon \in [0, 1]$, we have:

$$\Pr[\boldsymbol{x} \leq a] \leq \min_{t>0} e^{ta} \prod_{i=1}^{m} \mathbb{E} \exp(-t\boldsymbol{x}_i)$$
$$\Pr[\boldsymbol{x} \geq a] \leq \min_{t>0} e^{-ta} \prod_{i=1}^{m} \mathbb{E} \exp(t\boldsymbol{x}_i)$$

from which it is possible derive the more directly applicable bounds:

$$\Pr[\boldsymbol{x} \leq (1-\varepsilon)\mu_{\min}] \leq 1 \cdot \left(\frac{e^{-\varepsilon}}{(1-\varepsilon)^{1-\varepsilon}} \right)^{\mu_{\min}/u} \leq \exp\left(-\varepsilon^2 \frac{\mu_{\min}}{2u} \right),$$
$$\Pr[\boldsymbol{x} \geq (1+\varepsilon)\mu_{\max}] \leq 1 \cdot \left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{1+\varepsilon}} \right)^{\mu_{\max}/u} \leq \exp\left(-\varepsilon^2 \frac{\mu_{\max}}{3u} \right).$$

This theorem can be derived simply by applying Markov's inequality on the Laplace transform of the sum $\boldsymbol{x}$. While we will not prove it in this document, it can be shown that these two bounds are tight.

Proof. We first notice that the event of a random variable being bounded by a value a is equivalent to an inequality between the Laplace transform of both quantities. Recall that the Laplace transform relative to a value $t > 0$ is a monotonically decreasing function given by:

$$x \mapsto \exp(-tx)$$

For any $t > 0$, these two probabilities are thus equal:

$$\Pr[\boldsymbol{x} \leq a] = \Pr[\exp(-t\boldsymbol{x}) \geq \exp(-ta)].$$

We can now invoke Markov's inequality which gives an upper bound on the probability for a random variable to be lower bounded by a positive constant, which for a given random variable $\boldsymbol{y}$ and a constant $\alpha > 0$ reads as follows:

$$\Pr[\boldsymbol{y} \geq \alpha] \leq \frac{\mathbb{E}\,\boldsymbol{y}}{\alpha}.$$

Applying Markov's inequality on the random variable $\exp(-t\boldsymbol{x})$ and the positive constant $\exp(-ta)$ gives the following bound:

$$\Pr[\boldsymbol{x} \leq a] \leq \frac{\mathbb{E}\exp(-t\boldsymbol{x})}{\exp(-ta)} = e^{ta}\,\mathbb{E}\exp(-t\boldsymbol{x})$$

which we expand by exposing $\boldsymbol{x}$ as the sum of m independent random variables:

$$\Pr[\boldsymbol{x} \leq a] \leq e^{ta} \prod_{i=1}^{m} \mathbb{E}\exp(-t\boldsymbol{x}_i).$$

The above bound, as we take the minimum when $t > 0$, gives us:

$$\Pr[\boldsymbol{x} \leq a] \leq \min_{t>0} e^{ta} \prod_{i=1}^{m} \mathbb{E}\exp(-t\boldsymbol{x}_i) \tag{3.1}$$

which is the most general formulation of the Chernoff bound.

We let the reader verify that it suffices to take specific parameters $a = (1-\varepsilon)\mu_{\min}$ and $t = \log(1-\varepsilon)$ and apply classical inequalities of the logarithm to obtain the easily applicable Chernoff lower bound described in Theorem 7. Furthermore, similar reasoning can be applied to obtain the upper bound instead. □

Despite extensive effort in both the matrix analysis community and the quantum physics community, matrix versions of this bound did not come until the early 2010s when Joel A. Tropp in [Tro12] derived the matrix Chernoff bound by applying a powerful result in matrix analysis called Lieb's theorem [Lie73]; [Eps73].

3.2.2 The matrix Chernoff bound

The surprising feature of Tropp's matrix Chernoff bound is the striking similarity it shares with the classical scalar bound we just described.

However, before we introduce this matrix Chernoff bound, we need to recall a fundamental linear algebraic concept which extends the notion of non-negativity to Hermitian matrices. Indeed, Hermitian matrices are intimately related to non-negative vectors since the spectral theorem implies that all their eigenvalues are real numbers. Pushing the analogy further, when one diagonalizes a Hermitian matrix, the diagonal of eigenvalues can be seen as a vector. The notion of non-negativity for Hermitian matrices corresponds to this vector of eigenvalues being non-negative.

More formally, a Hermitian matrix M of size n is said to be positive semidefinite if all its eigenvalues are non-negative which we write as follows:

$$M \succeq 0 \iff \forall i \in [n], \lambda_i(M) \geq 0. \tag{3.2}$$

This concept extends the notion of inequality to Hermitian matrices in the following sense, called Löwner's partial order. Let two Hermitian matrices A and B of size n, we have that:

$$A \succeq B \iff A - B \succeq 0. \tag{3.3}$$

Let us review what these matrix inequalities mean in the context of the extreme values of the spectrum of a matrix. For example, if we say that an Hermitian matrix M satisfies:

$$0 \preceq M \preceq t \cdot I$$

it means from the first inequality that $\lambda_{\min}(M) \geq 0$ by definition. The second inequality implies that $\lambda_{\max}(M) \leq t$ which we can see from diagonalizing $M = P^\intercal \Lambda P$, i.e. Spec$(M)$ the set of eigenvalues of M is the same as Spec(Λ) the spectrum of a diagonal matrix:

$$\begin{aligned} t \cdot I \succeq M &\iff t \cdot I - M \succeq 0 \\ &\iff t \cdot I - P^\intercal \Lambda P \succeq 0 \\ t \cdot I \succeq M &\iff P^\intercal (t \cdot I - \Lambda) P \succeq 0 \end{aligned}$$

using the definition of positive semidefinite matrices we know in particular that the smallest eigenvalue satisfies the following inequality:

$$\begin{aligned} t \cdot I \succeq M &\iff \lambda_{\min}(t \cdot I - \Lambda) \geq 0 \\ &\iff t - \lambda_{\max}(M) \geq 0 \\ t \cdot I \succeq M &\iff t \geq \lambda_{\max}(M) \end{aligned}$$

where we have applied the standard relationship between opposite matrices and extreme eigenvalues: $\lambda_{\min}(-M) = -\lambda_{\max}(M)$.

Notice that there is no direct way to describe $\lambda_{\min}(M) \leq t$ or $\lambda_{\max}(M) \geq t$ with the Löwner partial order.

Theorem 8. *(The matrix Chernoff bound) [Tro12]*

Let $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_m$ be independent, square matrix random variables of size n such that

$$\forall i \in [m], \mathbf{0} \preceq \boldsymbol{X}_i \preceq u \cdot I_n$$

and denote their sum by $\boldsymbol{X} = \sum_{i=1}^{m} \boldsymbol{X}_i$ with:

$$\mu_{\min} \cdot I_n \preceq \mathbb{E}\,\boldsymbol{X} \preceq \mu_{\max} \cdot I_n.$$

For any $\varepsilon > 0$, we have:

$$\Pr[\lambda_{\min}(\boldsymbol{X}) \leq a] \leq \min_{t>0} e^{ta} \operatorname{tr} \bigodot_{i=1}^{m} \mathbb{E} \exp_\sigma(-t\boldsymbol{X}_i),$$

$$\Pr[\lambda_{\max}(\boldsymbol{X}) \geq a] \leq \min_{t>0} e^{-ta} \operatorname{tr} \bigodot_{i=1}^{m} \mathbb{E} \exp_\sigma(t\boldsymbol{X}_i)$$

where $A \odot B = \exp_\sigma(\log_\sigma(A) + \log_\sigma(B))$ is an operation on positive definite matrices.

Furthermore, these inequalities can be used to derive the more readily usable bounds:

$$\Pr[\lambda_{\min}(\boldsymbol{X}) \leq (1-\varepsilon)\mu_{\min}] \leq n \cdot \left(\frac{e^{-\varepsilon}}{(1-\varepsilon)^{1-\varepsilon}}\right)^{\mu_{\min}/u} \leq n \cdot \exp\left(-\varepsilon^2 \frac{\mu_{\min}}{2u}\right),$$

$$\Pr[\lambda_{\max}(\boldsymbol{X}) \geq (1+\varepsilon)\mu_{\max}] \leq n \cdot \left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{1+\varepsilon}}\right)^{\mu_{\max}/u} \leq n \cdot \exp\left(-\varepsilon^2 \frac{\mu_{\max}}{3u}\right).$$

Proof. The proof of the matrix Chernoff bound follows closely from the proof for the classical Chernoff bound. However, we will need matrix equivalents of the exponential and the logarithm functions.

Thankfully, we can invoke the powerful spectral mapping theorem [Rud91] which, given a function $f : \mathbb{R} \to \mathbb{R}$, constructs a function $f_\sigma : \mathbb{S}_n \to \mathbb{S}_n$ where for any symmetric matrix $M \in \mathbb{S}_n$, $M = P^\intercal \operatorname{diag}(\lambda_1, \ldots, \lambda_n)P$, we have:

$$f_\sigma(M) = P^\intercal \operatorname{diag}(f(\lambda_1), \ldots, f(\lambda_n))P. \tag{3.4}$$

In particular, we get the matrix version of the exponential function:

$$\exp_\sigma : \mathbb{S}_n \to \mathbb{S}_n^{++} \tag{3.5}$$

$$\exp_\sigma : M \mapsto P^\intercal \operatorname{diag}(e^{\lambda_1}, \ldots, e^{\lambda_n})P \tag{3.6}$$

where $\mathbb{S}_n^{++}$ is the set of positive definite matrices, i.e. $M \succ 0 \iff \lambda_{\min}(M) > 0$. We also obtain the matrix logarithm:

$$\log_\sigma : \mathbb{S}_n^{++} \to \mathbb{S}_n \tag{3.7}$$
$$\log_\sigma : M \mapsto P^\intercal \operatorname{diag}(\log(\lambda_1), \ldots, \log(\lambda_n))P. \tag{3.8}$$

The spectral mapping theorem guarantees that properties of these functions extend to the domain of symmetric matrices e.g. $\log_\sigma \exp_\sigma M = M$.

This lets us use Lieb's powerful theorem on the matrix operation $\odot$ which we define as follows. Given $A, B \in \mathbb{S}_n^{++}$ two positive definite matrices, we have the following

$$A \odot B = \exp_\sigma(\log_\sigma(A) + \log_\sigma(B)). \tag{3.9}$$

This operation enjoys several properties which make it similar to scalar multiplication: it is associative, commutative, and compatible with matrix-valued functions obtained from the spectral mapping theorem.

In particular, Lieb's theorem [Lie73]; [Eps73] establishes the following. For any fixed symmetric matrix H, the function

$$f : \mathbb{S}_n^{++} \to \mathbb{R}$$
$$f : M \mapsto \operatorname{tr} \exp_\sigma(\log_\sigma(M) + H)$$

is concave. This means in particular that the trace of the $\odot$ product of a matrix with any given fixed matrix $B \in \mathbb{S}_n^{++}$, e.g.

$$A \mapsto \operatorname{tr}(A \odot B)$$

is concave. Applying Jensen's inequality, i.e. $\mathbb{E}\, f(\boldsymbol{x}) \leq f(\mathbb{E}\, \boldsymbol{x})$ for a concave function f, on the trace of the $\odot$ product between a random matrix $\boldsymbol{A}$ and a fixed matrix $B \in \mathbb{S}_n^{++}$ we obtain that:

$$\mathbb{E} \operatorname{tr}(\boldsymbol{A} \odot B) \leq \operatorname{tr}(\mathbb{E}\, \boldsymbol{A} \odot B)$$

and since the $\odot$ product is associative and commutative we obtain:

$$\mathbb{E} \operatorname{tr}(\boldsymbol{A}_1 \odot \ldots \odot \boldsymbol{A}_m) \leq \operatorname{tr}(\mathbb{E}\, \boldsymbol{A}_1 \odot \ldots \odot \mathbb{E}\, \boldsymbol{A}_m). \tag{3.10}$$

Armed with these tools from matrix analysis, we can replicate the proof steps of the Chernoff bound as follows. First we form the Laplace transform of the smallest eigenvalue of our random matrix:

$$\Pr[\lambda_{\min}(\boldsymbol{X}) \leq a] = \Pr[\exp(-t\lambda_{\min}(\boldsymbol{X})) \geq \exp(-ta)].$$

By positive homogeneity of the smallest eigenvalue we can bring the $t > 0$ inside and apply Markov's inequality:

$$\Pr[\lambda_{\min}(\boldsymbol{X}) \leq a] \leq \frac{\mathbb{E}\exp(-\lambda_{\min}(t\boldsymbol{X}))}{\exp(-ta)} = e^{ta}\,\mathbb{E}\exp(-\lambda_{\min}(t\boldsymbol{X})).$$

Using the well-known relationship between extreme eigenvalues and sign-flipping $\lambda_{\min}(A) = -\lambda_{\max}(-A)$ we have:

$$\Pr[\lambda_{\min}(\boldsymbol{X}) \leq a] \leq e^{ta}\,\mathbb{E}\exp(\lambda_{\max}(-t\boldsymbol{X})).$$

Then, by the spectral mapping theorem applied on the exponential function, i.e. $\exp(\lambda_i(M)) = \lambda_i(\exp_\sigma(M))$, we have that:

$$\Pr[\lambda_{\min}(\boldsymbol{X}) \leq a] \leq e^{ta}\,\mathbb{E}\,\lambda_{\max}(\exp_\sigma(-t\boldsymbol{X})).$$

To recover the $\odot$ product we use a very coarse upper bound on the largest eigenvalue of a positive definite matrix. Indeed, since a positive definite matrix has positive eigenvalues, the largest eigenvalue must be bounded by the trace i.e. the sum of all eigenvalues:

$$\Pr[\lambda_{\min}(\boldsymbol{X}) \leq a] \leq e^{ta}\,\mathbb{E}\,\mathrm{tr}\exp_\sigma(-t\boldsymbol{X}).$$

We are left with exposing our random matrix $t\boldsymbol{X}$ as a sum of m random matrices $t\boldsymbol{X}_1, \ldots, t\boldsymbol{X}_m$. Using the property that matrix logarithm and matrix exponential are inverses, we express each $t\boldsymbol{X}_i = \log_\sigma \exp_\sigma t\boldsymbol{X}_i$. This allows us to reveal the $\odot$ product in the above inequality:

$$\Pr[\lambda_{\min}(\boldsymbol{X}) \leq a] \leq e^{ta}\,\mathbb{E}\,\mathrm{tr}(\exp_\sigma(-t\boldsymbol{X}_1) \odot \ldots \exp_\sigma(-t\boldsymbol{X}_m)).$$

or using $\bigodot$ by analogy with the product symbol $\prod$:

$$\Pr[\lambda_{\min}(\boldsymbol{X}) \leq a] \leq e^{ta}\,\mathbb{E}\bigodot_{i=1}^{m}\mathrm{tr}\exp_\sigma(-t\boldsymbol{X}_i).$$

Jensen's inequality combined with Lieb's theorem described above in 3.10 give us the most general expression of the matrix Chernoff bound:

$$\Pr[\lambda_{\min}(\boldsymbol{X}) \leq a] \leq e^{ta}\,\mathrm{tr}\bigodot_{i=1}^{m}\mathbb{E}\exp_\sigma(-t\boldsymbol{X}_i)$$

and in particular when optimizing over $t > 0$:

$$\Pr[\lambda_{\min}(\boldsymbol{X}) \leq a] \leq \min_{t>0} e^{ta}\,\mathrm{tr}\bigodot_{i=1}^{m}\mathbb{E}\exp_\sigma(-t\boldsymbol{X}_i). \tag{3.11}$$

Let us compare the above bound with the scalar version in 3.1:

$$\Pr[\lambda_{\min}(\boldsymbol{X}) \leq a] \leq \min_{t>0} e^{ta} \operatorname{tr} \bigodot_{i=1}^{m} \mathbb{E} \exp_{\sigma}(-t\boldsymbol{X}_i)$$

$$\Pr[\boldsymbol{x} \leq a] \leq \min_{t>0} e^{ta} \quad \prod_{i=1}^{m} \mathbb{E} \exp(-t\boldsymbol{x}_i)$$

As with the previous proof, we let the reader derive the more easily applicable bound presented in Theorem 8 by using standard logarithm inequalities (which naturally extend to the matrix logarithm!) and the following coarse upper bound on the trace of a symmetric matrix of size n that is $\operatorname{tr} M \leq n \cdot \lambda_{\max}(M)$. Like in the previous proof, the appropriate parameters to recover the bound featured in the theorem are $a = (1-\varepsilon)\mu_{\min}$ and $t = \log(1-\varepsilon)$. □

3.3 An interlude of spectral graph theory

It might have appeared to the reader that the mathematical tools we have presented are far from combinatorial objects such as graphs and discrete properties such as connectivity. However, it is important to remember that graphs have been studied from the point of view of their associated matrices well before the establishment of graph theory as a mathematical discipline. Often used to model neighborhood in the statistical physics literature, matrices have the advantage of being amenable to the numerous techniques and tools developed in linear algebra.

We underline a major methodological caveat with this approach: several distinct matrices can be associated with the same graph, and it is not clear a priori which type of matrix is appropriate to study a given property. To simplify our exposition we limit ourselves to two commonly studied matrices: the adjacency matrix and the Laplacian matrix.

3.3.1 The adjacency matrix

The adjacency matrix is a data structure in which edges of a graph are represented as entries of a square matrix. Given an undirected graph $G = (V, E)$ it is given by:

$$A_{ij} = \begin{cases} 1 & \text{if } ij \in E \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that this matrix has real entries and is symmetric i.e. $A_{ij} = A_{ji}$. The spectral theorem for real symmetric matrices tells us that A can be diagonalized i.e. the operator represented by A can be written as a weighted sum of simple operators. In the case of real symmetric matrices the weights, called eigenvalues, are real numbers. This is in contrast with arbitrary diagonalizable real matrices which generally admit complex eigenvalues.

We note a few properties of the adjacency matrix. The first property we have mentioned in passing in Chapter 1 is the Perron-Frobenius theorem [Per07] which states, in the case of undirected graphs, that the largest eigenvalue of an adjacency matrix must be strictly positive $\lambda_{\max}(A) > 0$ and that there exists a corresponding eigenvector whose components are strictly positive as well. Since we consider undirected graphs without loops, the diagonal of A is filled with 0 entries. Combining this information on the trace of an adjacency matrix with the Perron-Frobenius theorem, we also know that there must be at least one strictly negative eigenvalue.

3.3.2 The Laplacian matrix

The Laplacian matrix of a graph is the discrete equivalent of the Laplace operator used in physics to model the propagation of heat or other theories that involve potentials such as electrical flows. In a sense, while the continuous Laplacian gives information on the local extrema and saddle points of a continuous multivariate function, the Laplacian matrix contains information on how a certain notion of flow propagates along the underlying graph.

The Laplacian matrix L is defined as follows:

$$L = D - A$$

where D is the degree matrix of the graph and A is the adjacency matrix we introduced above. The degree matrix is simply the diagonal matrix given by:

$$D_{ij} = \begin{cases} \deg(i) & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that L is a diagonally dominant matrix i.e. the sum of the magnitude of each non-diagonal entry in a row or column is never more than the magnitude of the diagonal. By construction, the non-diagonal entries originate from the subtraction of the adjacency matrix in which all magnitudes are equal to 1 (the entries are equal to -1). In this case the Laplacian matrix can be easily seen as having equal values of magnitude in both the diagonal value

and the number of non-diagonal values in a row or column, the two of them counting up to the degree of a given node.

The Laplacian matrix is a sum of real symmetric matrices, thus it is also a real symmetric matrix.

We can then apply the Gershgorin circle theorem [Var10] which states that real symmetric diagonally dominant matrices are positive semidefinite i.e. all eigenvalues of L are non-negative.

In the specific case of L we have an additional information on its spectrum. Indeed consider how a Laplacian matrix operates on the all-ones vector $\mathbf{1}$ by evaluating the matrix-vector product $L\mathbf{1}$. Each row of this product corresponds to the sum of the row entries of L which we know sum up to 0 by construction:

$$\forall i \in V, (L\mathbf{1})_i = \deg(i) \underbrace{-1 \ldots -1 \ldots -1}_{\deg(i) \text{ times}} = 0$$

that is $L\mathbf{1}$ is the all-zeroes vector $\mathbf{0}$. Using the fact that $\mathbf{0} = 0\mathbf{1}$, i.e. the scalar 0 multiplying the all-ones vector, we have that $\mathbf{1}$ is an eigenvector of L with associated eigenvalue 0:

$$L\mathbf{1} = 0\mathbf{1}.$$

With this knowledge in hand, we are going to attempt to study how combinatorial properties of a graph can be understood through linear algebraic concepts.

3.3.3 Algebraic connectivity

From the above considerations, we know that the eigenvalues of the Laplacian matrix are as follows: $\mu_n \geq \ldots \mu_i \geq \mu_2 \geq \mu_1 = 0$ and that at least the largest eigenvalue $\mu_n > 0$. Somewhat surprisingly, the smallest eigenvalue μ_1 has multiplicity equal to the number of connected components in the graph.

We will now look at a more specific statement on μ_2, the second smallest eigenvalue of the Laplacian matrix, which is called the algebraic connectivity of the graph.

Theorem 9. *(Algebraic Connectivity) [Fie73]*

Let $G = (V, E)$ be an undirected graph and L be its associated Laplacian matrix. We have the following equivalence:

$$G \text{ is connected} \iff \mu_2 > 0.$$

Recall that the the right-hand side of this equivalence amounts to saying that the eigenvectors associated with μ_1 belong to a one-dimensional space. Indeed we know that all other eigenvalues of L are non-negative and that $\mu_1 = 0$. We will use this property to prove Theorem 9.

Proof. To prove the direct implication ($\Longrightarrow$), we assume that G is a connected graph and that x is an eigenvector of L associated with the eigenvalue 0 i.e. $Lx = \mathbf{0}$. This means that the quadratic form of L with respect to x is equal to 0:

$$x^\intercal Lx = 0.$$

Via direct evaluation it is easy to see that the quadratic form of L is given by:

$$x^\intercal Lx = \sum_{ij \in E} (x_i - x_j)^2.$$

From these two considerations, we have that $\forall ij \in E, x_i = x_j$. Furthermore, since G is connected, there exists a path connecting every pair of nodes u, v in the graph. Thus, along each of these paths, we have identical values of x for each node. This lets us conclude that $\forall u, v \in V, x_u = x_v$ and that all eigenvectors associated with 0 must be of the form $\alpha\mathbf{1}$ for some non-zero α, i.e. they belong to a one-dimensional space.

We will prove the reverse implication ($\Longleftarrow$) by contradiction. Assume that the eigenvectors associated with μ_1 belong to a one-dimensional space and that G is not connected i.e. there exists a connected component $C \subset V$. We can define the indicator vector of C as $\mathbf{1}_C$ defined as follows:

$$\mathbf{1}_C^{(i)} = \begin{cases} 1 & \text{if } i \in C \\ 0 & \text{otherwise.} \end{cases}$$

Computing the matrix-vector product $L\mathbf{1}_C$ gives us the following: each row of the product that corresponds to a node in C corresponds to the sum of the row entries of L while each row that corresponds to a node in $V \setminus C$ is equal to 0.

$$(L\mathbf{1}_C)_i = \begin{cases} \deg(i) \underbrace{-1 \ldots -1 \ldots -1}_{\deg(i) \text{ times}} = 0 & \text{if } i \in C \\ 0 & \text{otherwise.} \end{cases}$$

Again we use the fact that $\mathbf{0} = 0\mathbf{1}_C$ to claim that $\mathbf{1}_C$ is a second eigenvector of L associated with the eigenvalue $\mu_1 = 0$ which is not colinear to the all-ones vector $\mathbf{1}$ which we know is already an eigenvector associated with 0. This is in contradiction with the assumption that eigenvectors belong to a one-dimensional space. □

From Theorem 9 we have seen an example of a global combinatorial property of a graph which is reflected in a numerical property of an associated matrix. We can now translate questions regarding the connectivity of random graphs into questions about the spectrum of random matrices. In the following sections we will study the connectivity of random graphs generated from the $G(n,p)$ model through the spectrum of their associated random matrices.

3.4 The $G(n,p)$ random graph model

In the early 1960s, Edgar Gilbert on one hand [Gil59] and Paul Erdös and Alfréd Rényi on the other hand [ER60] introduced closely related random graph models. The $G(n,p)$ model introduced by Gilbert is asymptotically equivalent to the Erdös-Rényi model for most values of p and as such, both models have often been referred to as the Erdös-Rényi model. However, in practice, the $G(n,p)$ model is the one that is predominantly used because it describes random graphs with independent random variables.

The $G(n,p)$ model can be formally defined as follows: a random graph $\mathbf{G} \sim G(n,p)$ is a graph on a node set V with $|V| = n$ and edges ij with some probability $\Pr(ij \in E) = p$. Informally, the $G(n,p)$ model represents graphs with n nodes where each possible edge has equal probability of being present in the edge set. Each random edge can be seen as an independent and identically distributed Bernoulli random variable with parameter p.

The fact that edges are independent in the $G(n,p)$ model might hint at the fact that the concentration of measure phenomenon could apply to many properties of such random graphs. Indeed, we will see that it is the case. But first let us consider some basic properties of the random graphs generated by the $G(n,p)$ model.

The first observation one can make is that there are $n(n-1)/2$ possible edges in a graph of n nodes and that in expectation a p fraction of them will be present in a realization of the random model. Indeed the average degree of a random graph $\mathbf{G}$ is:

$$\mathbb{E}\,\bar{d}(\mathbf{G}) = \frac{(n-1)p}{2} \approx O(np).$$

This property can be improved to hold with high probability by using the Chernoff bound 7 and remains true for any value of p or n.

Before we discuss more advanced properties of random graphs, it is important to highlight the difference between the apparently similar statements: "a graph G satisfies property P" and "a random graph $\mathbf{G}$ satisfies property P".

3.4.1 Properties of random graphs

In structural graph theory, graph classes are a common framework used to explore important properties of graphs. The goal is to categorize graph themselves but also computational tasks that may leverage structural information. Let us look at a few examples of graph classes. Sometimes, a class is determined having its members admit a specific subgraph, e.g. Hamiltonian graphs contain a simple path as a partial subgraph which spans their entire node set. In many cases [RS04], it is the absence of a subgraph that defines the class. Take for example planar graphs which are defined as graphs with no $K_{3,3}$ or K_5 minor (a type of subgraph obtained by node and edge contractions). We can also mention triangle-free graphs or even perfect graphs which admit neither odd holes or anti-holes as induced subgraphs. Another way one can define graph classes is by considering quantitative properties such as cubic graphs in which all nodes have degree 3. Sometimes the quantity might not even be known to be computable in polynomial time such as in the case of graphs which have parameters derived from the solution of an **NP**-hard optimization problem. Consider for example the treewidth problem which asks, given an undirected graph, to compute a tree decomposition whose width is minimum [ACP87]. This directly leads to the class of bounded treewidth graphs for which the size of an optimal solution to the treewidth problem is bounded above by a constant. And as we saw earlier in section 3.3 a class might be determined in several equivalent ways: a connected graph is a graph for which any two nodes are connected by a path or equivalently a graph with strictly positive algebraic connectivity. In all these examples a given graph either belongs to a class or it does not in the same way one could consider set membership. In the case of random graphs the concept of membership to a class is less clear-cut.

To contrast graph classes with random graph models in more details, recall that the object of study in random graph models is an infinite family of graphs (usually with a node set of size n where n is going to infinity) that can be grouped under a defining fixed statistic. In the $G(n, p)$ model we are concerned with, we study graphs with independent random edges such that nodes have an average degree of $O(np)$. Determining whether such random graphs belong to a class amounts to studying properties of random graphs with varying values of p. This is to contrast with the fact that any graph on n nodes is in the support of the $G(n, p)$ model and as such all classes are covered albeit with very small probability.

Another issue arises with the question of the relevance of random graph models to study particular realizations of a model. Indeed whether a given graph *belongs* to a particular random graph model is not even clearly defined.

One may want to consider looking for a model with maximum likelihood that could have generated a given graph but the fact that random graph models are infinite families blurs the definition of maximum likelihood.

Despite these issues, Erdös and Rényi have discovered that 0-1 laws apply to random graph models such as the $G(n, p)$ model.

One of the major achievements of random graph theory is the determination of threshold values of p for which a certain property holds. In a sense, one can see such results as probabilistic versions of the graph classes from structural graph theory. To be more precise, let us consider the so-called 0-1 properties which were studied in the orignal paper by Erdös and Rényi [ER60]. A graph property can be called 0-1 if in the $G(n, p)$ model the property holds almost surely (goes to 1 as n goes to infinity) for all $p > t$ and almost never holds (holds with probability that goes to 0 as n goes to infinity) otherwise. It is easy to see that not all properties are 0-1, but in particular it is remarkable that monotone properties, those that persist as we add more edges to a graph, all admit a critical threshold [FK96]; [Bol01].

3.5 The connectivity threshold

In this section we discuss the first result by Erdös and Rényi on the subject [ER60] which characterizes when a random graph from the $G(n, p)$ model is connected. Connectivity, that is the existence of a path between any two nodes in a graph, is maybe one of the simplest global graph properties. Furthermore, it is easy to see that connectivity is maintained as one adds more edges to a given graph. This means that connectivity is a monotone property and as such admits a critical threshold. In this section, we will illustrate the use of concentration inequalities to derive the order of magnitude of the connectivity threshold i.e. the value p which separates random graphs that are almost surely connected from random graphs that are almost surely disconnected.

Let us give a general idea of the proof we will show in details. First, we study a random matrix, the Laplacian matrix of a graph from the $G(n, p)$ model. Because we know that connectivity is related to the spectrum of the Laplacian matrix through Theorem 9, we proceed in two steps. The first step consists in studying a random matrix whose spectrum is related to the Laplacian matrix. The second step is to prove necessary conditions to apply the matrix Chernoff bound following Theorem 8 and interpret the result as a sufficient condition for a random graph to be connected.

3.5.1 The Laplacian matrix of a random graph

In this section we look at the $G(n,p)$ model through the lens of its Laplacian matrix. As mentioned earlier, the Laplacian matrix of a graph $G=(V,E)$ is a positive semidefinite matrix which can be defined as follows:

$$L(G)_{ij} = \begin{cases} \deg(i) & \text{if } i=j \\ -1 & \text{if } ij \in E \\ 0 & \text{otherwise.} \end{cases}$$

Such matrix can be represented as a sum of elementary matrices. If we consider the Laplacian matrix $L^{(ij)}$ of a graph with a single edge $G_{ij}=(V,\{ij\})$ we have the following Laplacian matrix:

$$L^{(ij)} = E_{ii} + E_{jj} - E_{ij} - E_{ji}.$$

where the E_{ab} matrices are the standard basis of $M_{n,n}$, i.e. square matrices of size n that have one entry equal to 1 in position (a,b) and 0 otherwise.

The Laplacian of G is thus the sum of the Laplacians of each G_{ij} for $ij \in E$.

$$L = \sum_{ij \in E} \underbrace{E_{ii} + E_{jj} - E_{ij} - E_{ji}}_{L^{(ij)}} \tag{3.12}$$

A random matrix $\boldsymbol{L}$ corresponding to a $G(n,p)$ random graph would combine definition 3.12 with, for every possible edge $i \neq j \in V \times V$, an independent random variable $\boldsymbol{x}_{ij} \sim \text{Ber}(p)$ drawn from a Bernoulli distribution of parameter p:

$$\boldsymbol{L} = \sum_{i \neq j \in V \times V} \boldsymbol{x}_{ij} L^{(ij)} \tag{3.13}$$

We now give preliminary results on the spectrum of $L^{(ij)}$ which will be useful in the next section. First, recall that the all-ones vector $\mathbf{1}$ is always an eigenvector for any Laplacian matrix and is associated with eigenvalue 0. Second, reusing the argument in the proof of Theorem 9, we can interpret $L^{(ij)}$ as the Laplacian of an undirected graph with $n-2$ isolated nodes i.e. the graph G_{ij} has $n-1$ connected components. Finally, by a trace argument, we know that the remaining eigenvalue must be equal to the trace, that is 2. The set of eigenvalues of the matrix $\text{Spec}(L^{(ij)})$ is thus:

$$\text{Spec}(L^{(ij)}) = \{0^{n-1}, 2\}$$

which gives us exact bounds on the spectrum of each summand:

$$0 \preceq L^{(ij)} \preceq 2I. \tag{3.14}$$

3.5.2 Shifting the spectrum of a Laplacian matrix

Recall that the matrix Chernoff bound described in section 3.2.2 gives probabilistic estimates on the extreme values of the spectrum of a sum of positive semidefinite matrices i.e. it only gives bounds on the smallest and largest eigenvalues. In our case however, we are looking for an estimate of the second smallest eigenvalue.

Fortunately, we have precise information on the "bottom" part of the eigenspace of Laplacian matrices. Indeed, the all-ones vector $\mathbf{1}$ is an eigenvector of any Laplacian matrix and is always associated with the eigenvalue 0. This means that the second smallest eigenvalue can be computed explicitly through the Courant-Fischer variational characterization of eigenvalues, see for example [Bha13]. In particular, for any Laplacian matrix L if we denote by μ_2 the second smallest eigenvalue of L then it is the optimal solution value of the following optimization problem:

$$\mu_2(L) = \min_{\substack{x \neq \mathbf{0} \\ x \perp \mathbf{1} \\ x \in \mathbb{R}^n}} \frac{x^\intercal L x}{x^\intercal x}. \tag{3.15}$$

Using our knowledge of the eigenspace of Laplacian matrices, we will relate this variational characterization to the smallest eigenvalue of a different matrix. Intuitively speaking, the goal would be to analyze a "shifted" version of a Laplacian matrix whose smallest eigenvalue would correspond to the second smallest eigenvalue of our original matrix.

To construct this shifted Laplacian, we introduce a partial isometry: a matrix which preserves norms whenever it is applied to a vector orthogonal to its null space. Here, we choose a partial isometry whose null space is $\mathbf{1}$, the eigenvector associated with the smallest eigenvalue of Laplacian matrices. In order to "peel" off the space associated with eigenvector $\mathbf{1}$, we choose a rectangular matrix which would compress a Laplacian matrix of size n into a space of dimension $n-1$ i.e. our partial isometry should be a $n-1$ by n matrix. A simple necessary and sufficient condition [Hea67] for a matrix A to be a partial isometry is that the product with its conjugate transpose A^* would be idempotent $AA^* = I$. This informs us in choosing a $n-1$ by n matrix R which satisfies the following conditions:

$$RR^* = I_{n-1} \text{ and } R\mathbf{1} = \mathbf{0}. \tag{3.16}$$

This partial isometry allows us to analyze the second smallest eigenvalue of a Laplacian matrix as the smallest eigenvalue of a transformed Laplacian. Indeed, we can rewrite the Courant-Fischer formula 3.15 for μ_2 with a simple change of variables such that any vector $\forall x \in \mathbb{R}^n$ with $x \neq \mathbf{0}$ and $x \perp \mathbf{1}$ is

written as $x = R^* y$ for some $y \in \mathbb{R}^{n-1}$. This form makes it explicit that a vector $x \perp \mathbf{1}$ is really $(n-1)$-dimensional since we can assume for example that $x_n = -\sum_{i \neq n} x_i$. Thus we have for any Laplacian matrix L that:

$$\mu_2(L) = \min_{\substack{y \neq 0 \\ y \in \mathbb{R}^{n-1}}} \frac{y^\intercal R L R^* y}{y^\intercal R R^* y}. \tag{3.17}$$

and using the fact that R is a partial isometry, i.e. $RR^* = I_{n-1}$, we obtain the following identity:

$$\mu_2(L) = \min_{\substack{y \neq 0 \\ y \in \mathbb{R}^{n-1}}} \frac{y^\intercal R L R^* y}{y^\intercal y} = \lambda_{\min}(RLR^*). \tag{3.18}$$

This property lets us interpret bounds on the smallest eigenvalue of $R\boldsymbol{L}R^*$ as bounds on the second smallest eigenvalue of $\boldsymbol{L}$, that is for any $t \in \mathbb{R}$:

$$\Pr(\mu_2(\boldsymbol{L}) \leq t) = \Pr(\lambda_{\min}(R\boldsymbol{L}R^*) \leq t). \tag{3.19}$$

We can now focus on obtaining the technical prerequisites for using the matrix Chernoff bound on the shifted random Laplacian matrix $R\boldsymbol{L}R^*$.

3.5.3 Bounds on every summand

The first step is to obtain bounds on the spectrum of the summands that add up to our random matrix $R\boldsymbol{L}R^*$. For this purpose, we invoke the conjugation principle which states that for any pair of Hermitian matrices A, B of size n and any arbitrary k by n matrix M we have the following implication:

$$A \preceq B \implies MAM^* \preceq MBM^*. \tag{3.20}$$

Applying this principle to matrices $L^{(ij)}$ we preserve the bounds on their spectrum:

$$0 \preceq L^{(ij)} \preceq 2I_n \implies 0 \preceq RL^{(ij)}R^* \preceq 2I_{n-1}$$

which gives us that:

$$\forall i \neq j \in V \times V, 0 \preceq RL^{(ij)}R^* \preceq 2I_{n-1}. \tag{3.21}$$

3.5.4 Smallest eigenvalue of the expected matrix

We can now study the expectation of the shifted random Laplacian. If we denote by $K_n = (V, \{ij : i \neq j, i \in V, j \in V\})$ the complete graph on n nodes we have:

$$\mathbb{E}\, R\boldsymbol{L}R^* = pRL(K_n)R^*. \tag{3.22}$$

Recall that the Laplacian of a complete graph is a n by n matrix where diagonal values are equal to $n-1$ and off-diagonal values are equal to -1. By some rearranging and recalling that a matrix of ones can be written as a tensor product between two all-ones vectors, i.e. $U = \mathbf{1}\mathbf{1}^*$, we have:

$$L(K_n) = \underbrace{(n-1)I_n}_{\text{diagonal values}} + \underbrace{(I_n - \mathbf{1}\mathbf{1}^*)}_{\text{off-diagonal values}}$$

that is

$$L(K_n) = nI_n - \mathbf{1}\mathbf{1}^*$$

which lets us compute the expectation of $R\boldsymbol{L}R^*$ in a straightforward manner:

$$\begin{aligned}
\mathbb{E}\, R\boldsymbol{L}R^* &= p\, R(nI_n - \mathbf{1}\mathbf{1}^*)R^* \\
&= p\,(nR - \underbrace{R\mathbf{1}}_{\mathbf{0}}\mathbf{1}^*)R^* \\
&= np\, RR^* \\
\mathbb{E}\, R\boldsymbol{L}R^* &= np\, I_{n-1}.
\end{aligned}$$

Somewhat surprisingly, the expectation of the shifted random Laplacian is a diagonal matrix. However, this does not mean that a realization of this random matrix would resemble it in any other way than having a similar spectrum. In expectation, all eigenvalues of $R\boldsymbol{L}R^*$ are identical which means that in particular:

$$\lambda_{\min}(\mathbb{E}\, R\boldsymbol{L}R^*) = np. \tag{3.23}$$

3.5.5 Applying the matrix Chernoff bound

We now have the prerequisites to apply Theorem 8 to our sum of independent positive semidefinite random matrices $\sum_{i \neq j} \boldsymbol{x}_{ij} RL^{(ij)} R^*$.

First, from 3.21, we know that each summand $\boldsymbol{x}_{ij} RL^{(ij)} R^*$ has its spectrum upper bounded by 2 which means that $u = 2$. Also, from 3.23, we know that the smallest eigenvalue of the expected random matrix is exactly $\lambda_{\min}(\mathbb{E}\, R\boldsymbol{L}R^*) = np$, so the mean bounds are $\mu_{\min} = \mu_{\max} = np$. Finally, we replace $(1-\varepsilon)$ by

some $\eta > 0$ in the statement of the matrix Chernoff bound in Theorem 8. This yields:

$$\Pr(\lambda_{\min}(R\boldsymbol{L}R^*) \leq \eta\, np) \leq (n-1)\left(\frac{e^{\eta-1}}{\eta^\eta}\right)^{np/2}.$$

Using identity 3.19 this bound applies to the original random Laplacian:

$$\Pr(\mu_2(\boldsymbol{L}) \leq \eta\, np) \leq (n-1)\left(\frac{e^{\eta-1}}{\eta^\eta}\right)^{np/2}.$$

and recalling the limit of the following function:

$$\lim_{x\to 0} \frac{e^{x-1}}{x^x} = \frac{1}{e}$$

we know as $\eta \to 0$ that:

$$\Pr(\mu_2(\boldsymbol{L}) \leq 0) \leq (n-1)\exp\left(-\frac{np}{2}\right). \tag{3.24}$$

The only remaining step is to find a condition on p such that the right-hand side of this inequality is at most $1/n$ which would imply that the algebraic connectivity of $G(n,p)$ is positive with high probability, i.e. $\Pr(\mu_2(\boldsymbol{L}) > 0) \geq 1 - 1/n$.

Rearranging the following inequality:

$$(n-1)\exp\left(-\frac{np}{2}\right) \leq \frac{1}{n}$$

and taking the logarithm on both sides, we obtain:

$$\log n + \log(n-1) - np/2 \leq 0$$

which we simplify into:

$$p \geq 4\frac{\log n}{n}. \tag{3.25}$$

This result can be seen as a randomized algorithm to construct a connected graph with average degree np for any $p \geq 4\log n/n$ with high probability.

Summary

With the simple application of a generic matrix concentration inequality we have obtained a non-asymptotic version of the connectivity threshold of $G(n, c/n)$ random graphs:

Theorem 10. *Non-asymptotic connectivity threshold [Tro+15]*
A random graph $G(n, c/n)$ is connected with high probability if $c = \Omega(\log n)$, that is:

$$\Pr(G(n, \alpha \log n/n) \text{ is connected}) \geq 1 - \frac{1}{n} \text{ if } \alpha \geq 4.$$

Furthermore it is possible to prove an almost reciprocal result with scalar concentration inequalities. For some $c < 1$, we have:

$$\Pr(G(n, \alpha \log n/n) \text{ is connected}) \leq \frac{1}{n} \text{ if } \alpha < c.$$

Together, these constitute a non-asymptotic version of the celebrated theorem of Erdös and Rényi:

Theorem 11. *The connectivity threshold [ER60]*
The threshold for a random graph $G(n, c/n)$ to be connected is $c = \log n$, that is:

$$\lim_{n\to\infty} \Pr(G(n, \alpha \log n/n) \text{ is connected}) = \begin{cases} 1 \text{ if } \alpha > 1 \\ 0 \text{ if } \alpha < 1 \end{cases}$$

In more abstract terms, the Theorem 10 we have presented is a sufficient condition for a random Laplacian matrix to concentrate around its expected value. We underline the fact that the Theorem 11 due to Erdös and Rényi is more precise than the result we have presented, i.e. $\alpha > 1$ while in our case we ask for $\alpha \geq 4$. However, a non-asymptotic statement can directly be transformed into a randomized algorithms because it holds with high probability for any value of n.

Now that we have explored a very powerful concept which allows to construct specific combinatorial objects via random sampling, it remains to see how to apply it to the problem of finding a graph with good spectral properties such that a given epidemic would be guaranteed to die out in a short amount of time.

The question left for us to answer is "How to tie everything together?"

4 Intermission

In this thesis we explore to what extent it is possible to quickly reconfigure networks to suppress spreading malware. We have learned from Chapter 1 that adjusting the spectral properties of a contact graph is an indirect way to suppress a epidemic. From Chapter 2 we have made precise the notion of algorithms that are guaranteed to be efficient, even in the case of problems that are believed to be hard to solve. Finally, we have seen in Chapter 3 that it is possible to construct matrices with specific spectral properties via randomized algorithms.

However, network reconfiguration has traditionally not been a topic associated with network security. Indeed, threats can happen over short time scales while changing the configuration of network elements or modifying a topology have traditionally been operations undertaken over a long period of time. To understand the relevance of this security countermeasure, we explore how the network security landscape has evolved in the light of novel technologies that aim to make networks more controllable.

4.1 The advent of controllable networks

In this section, we start by giving a short history of computer systems, how they are used in computer networks, and how the use of software to abstract away hardware elements has contributed towards more widespread automation at every layer.

4.1.1 Systems as abstraction

The history of computer systems can be seen as a quest towards the abstraction of all hardware resources. For example, one can interpret the invention of the first high-level programming languages and compilers in the 1950s as the abstraction of hardware architecture from programming. This abstraction has proved itself to be essential to the development and modern practice of programming which is overwhelmingly undertaken in high-level programming languages.

Another example of abstraction can be seen in operating systems which provide interfaces for different peripherals such as screens, printers, input methods, among many others. The UNIX operating system developed in the 1970s was designed to unify all peripherals with a single multipurpose abstraction: the file system. Interaction with peripherals would be represented as writing to or reading from specific files. A large majority of operating systems in use at the time of writing this thesis are derivatives of UNIX that adhere to this abstraction.

With the exponential growth of computing power described by Moore's law, it was possible as early as the 1960s to abstract away physical hardware by emulation. The resulting software would allow for the creation and management of virtual machines. Contemporary practice in operations consists in isolating different applications within virtual machines or containers and managing them without reference to the underlying hardware.

This trend depicts a unifying theme: the introduction of a new category of computer program abstracts away one layer of a computer system. Once this abstraction is accepted, the vast majority of programs depend on it and opens up opportunities for further abstraction.

4.1.2 The case of network resources

Despite the efforts of projects such as the Plan 9 operating system which attempted to include network resources in the file system abstraction, network elements have long remained detached from the increasing "softwarization" of computer systems. It can be said that software abstractions have thrived in areas related to personal computers and individual servers, while software dedicated to larger infrastructure has largely remained in the hands of hardware vendors.

However, the rapid expansion of the Internet and the concentration of computing power in the hands of resourceful multinational companies have provided incentives to bring network resources to the software layer.

Software technologies dedicated to abstract networking resources originated from organizations seeking to simplify the management of data centers or supercomputing facilities. In particular, network function virtualization (NFV) and software-defined networking (SDN) have gained wider usage among a growing number of Internet service providers as well as Internet companies.

Network function virtualization (NFV) is a set of technologies aimed at facilitating the deployment of various network equipment usually found at the nodes of a network. NFV leverages existing virtualization techniques to deploy software equivalents of existing pre-programmed hardware appliances includ-

ing for example load balancing equipment, firewalls, or intrusion detection devices.

Software defined networking (SDN) on the other hand is a set of technologies which allow network administrators to dissociate the control plane (the logic that decides how traffic is going to be routed) from the data plane (which is tasked to implement the routes planned by the control plane). Unlike traditional networks, a SDN control plane can also manage several disjoint data planes.

The central benefit provided by SDN is the automated management of data plane elements via a software platform.

In traditional networks, the data plane is composed of physical hardware such as switches or routers. Such elements are often configured outside of the network, to be later connected with other network elements. Finally, while hardware elements allow some monitoring, it is usually specific to the hardware constructor and accessible only via specialized interfaces.

With SDN, configuration, and especially re-configuration, can be done after a data plane element has been installed. This opens the door to changing the behavior of data plane elements in almost real time, thereby modifying the network topology.

When combining network function virtualization and software-defined networking, application-specific network topologies can be spawned from utility servers which might not even be located in the same physical location. From the point of view of the application, the entire network is abstracted away, and only physical properties remain, such as the latency incurred by geographical distance. Monitoring is also unified under a common interface which informs decisions in how to modify the network to increase the overall quality of service.

4.1.3 Network security

Unlike the software technologies deployed at the scale of personal computers or individual servers, the infrastructure necessary to manage computer networks remain in the hands of large organizations. For this reason, the adoption of software technologies to increase automation in the networking world depends on different socio-economic factors. The major argument in favor of these technologies comes from the benefits organizations can collect from improving their existing operations or from providing new services to their users.

Network security is a growing concern among many actors, including clients of Internet service providers. The two network-level software technologies we have mentioned, NFV and SDN, can improve network security in many

ways. Here we consider two examples: centralized monitoring and automatic deployment of countermeasures.

4.2 Epidemic models for network security

Centralized monitoring is a simple but effective improvement over current practice in traditional networks. In software-defined networking environments, standardized monitoring information is facilitated by the virtual nature of network elements, from virtual switches in the data plane to authentication systems in the application layer. Furthermore, centralized monitoring opens the door to the systematic use of online statistical analysis of the overall performance of a network and also to the use of machine learning models to predict traffic for example. In practice, the OpenFlow protocol has been widely adopted among SDN users to facilitate the real-time collection of metrics from switches to their controller, see for example the sampling-based architecture proposed by Chowdhury et al. [Cho+14].

A common way to represent threats propagating over a network is to use variants of the epidemic models found in the mathematical epidemiology literature. Owing to the diversity of network security situations, diverse epidemic models have been used to model, simulate, and analyze threats at varying degrees of granularity. With the help of statistics, it is possible not only to simulate but also to infer the most likely models that fit with observations of the network.

We start by reporting existing applications of epidemic models in the field of network security and then discuss the possibility of leveraging time series information to fit these models to recorded data.

4.2.1 Modeling computer viruses and propagating threats

Standard epidemic models such as the SIS model discussed in Chapter 1 have been used to represent the spread of computer viruses in networks systems with homogeneous connections such as small-scale local area networks where nodes are fully connected, i.e. the topology is a complete graph, see for example the use of a SEIRS model to represent the spread of malicious files in a network [MS07]. However, the applicability of these models is limited and the community of researchers in network security has often be a proponent of models that take into account heterogeneous connectivity such as Mickens and Noble in their study of mobile networks [MN05].

Before the systematic study of networked epidemic models as described

in Chapter 1, many attempts at taking into account the heterogeneity of real networks were put forward, such as studies that augment standard epidemic models with average degree information [Yan+13].

We note that networked epidemic models have rarely been used in the network security literature so far. Instead they have been mostly featured in papers exploring the spread of abstract processes such as rumors propagating over a social network, ideas being shared by a network of innovators, or products increasing in popularity in a market, see for example the references in the survey by Nowzari et al. [NPP16].

4.2.2 Learning model parameters

The possibility of having close to real-time metrics transmitted from every node in a network to their SDN controller invites us to consider the use of statistical methods to determine whether traffic or the state of every node corresponds to presupposed models. Following the common statistical practice of maximum likelihood estimation of parameters, it is possible to fit models to observations.

In particular in the case of networked epidemic models, it is sufficient to have access to time series of node states to find the maximum likelihood parameters of a networked SIS model. Ruhi in a research project [Ruh17] describes two such estimators using time series containing the last two time steps per node. Similar methods can be generalized for various networked models beyond the SIS model and be made more robust by considering a longer time period than two time steps.

From a different methodological point of view, machine learning models can also help in finding appropriate model parameters. This line of work has especially been fruitful when combining artificial neural networks together with differential equations solvers, such as in the case of neural ordinary differential equations [Che+18]. These neural network models can be used to learn parameters of a differential equations even in the absence of some observations. Indeed, while it is ultimately desirable to have perfectly reliable monitoring, monitoring information of particular nodes could be missing at some time step.

4.3 Reactive countermeasures to threats

Countermeasures have always been devised to protect computer networks by mitigating potential or existing threats. However, SDN offers the possibility to quickly identify threats and to counter them with adequate responses.

4.3.1 Deploying security appliances

The most common combination of SDN and NFV consists in architectures which are tasked to deploy virtual firewalls in order to protect a network of virtual machines, such as in the VNGuard framework of Deng et al. [Den+15]. Unlike physical firewall appliances, virtual firewalls can be reconfigured in real-time and their rules, which specify what incoming traffic they must block, can be updated at the same time as an attack threatening the network, e.g. a distributed denial of service attack.

Virtual intrusion detection systems can also be deployed depending on preliminary analysis of detected threats, primarily anomaly detection systems that produce early warnings and recommend more advanced investigation. However, the provisioning and management of such virtual network functions must be done in an efficient manner, which remains an open problem [Lui+15].

4.3.2 Topology reconfiguration

A tenet of SDN is that virtual networks should be re-configurable at virtually no cost. While zero cost is usually not possible in practice, existing SDN architectures still promote the use of dynamic network topologies that evolve over time to respond to client demands but also to defend against threats.

Unlike traditional networks whose design rely on over-provisioning resources such as bandwidth and backup routes to improve quality of service both in terms of rate and reliability, SDN networks can create a dedicated route for a specific client demand, allowing for guaranteed service levels for specific needs, e.g. emergency services.

To defend against threats, it can be useful to keep the network from becoming predictable to an outside attacker. In a paper by Kampanakis et al. [KPB14] the authors describe the use of automated moving target defense strategies in SDN networks. This strategy corresponds to having network elements change their behavior, e.g. some nodes change IP addresses, others slightly modify the protocols they use to communicate, in the hope of making the so-called attack surface of network elements less exploitable.

Simpler countermeasures include for example rerouting denial of service traffic or simply re-configuring switches to drop certain packets on purpose. Finally, isolating nodes that have been deemed suspicious can be done via SDN, mimicking the “honey pot” strategy used by “white hat” security experts, see for example the Honeymix framework proposed by Han et al. [Han+16a].

4.4 A novel network security framework

Following the design goals of software-defined networking that we have mentioned in the previous sections, we introduce a novel network security framework based on networked epidemic models and optimization algorithms.

4.4.1 Finding a temporary topology reconfiguration

In this framework we consider a SDN architecture which contains three core components. Before we go into the specifics of every component, we assume that the SDN controller maintains at all times a perfectly reliable view of the current software defined topology.

The first component of our framework is an anomaly detection system. For each node, an anomaly detection system must detect whether the node is behaving as expected and return time series of these predictions to the SDN controller. Such anomaly detection systems can be chosen to have very low CPU and memory footprint and be trained offline or using network simulators. In particular, we consider the use of one-class support vector machines which have been known to perform well for novelty detection in time series [MP03].

The second component is a set of maximum likelihood estimators, each attempting to fit the time series data to a particular networked epidemic model. Depending on the desired robustness to the spread of anomalies, this epidemic estimation system returns the most likely epidemic model or the most alarming one. This can be done by extending the work of Ruhi [Ruh17] to the large family of networked epidemic models covered in the paper by Prakash et al. [Pra+11].

Finally, the third component is a topology optimizer system which takes as input the current topology defined by the controller and the epidemic model returned by the epidemic estimation system. The epidemic model can be summarized as a single number $\epsilon(\pi)$, function of the multiple epidemic parameters π_i learned by the second component.

We now use the results presented in Chapter 1 to define the task that must be accomplished by the topology optimizer system. We interpret the statement that a given epidemic must die out in a short amount of time (logarithmic in the number of nodes) if the spectral radius of the graph representing the network topology is no more than the value $\epsilon(\pi)$. In particular, this means that modifying the topology for a short amount of time is guaranteed to remove the epidemic from the network without further intervention.

The downside of this approach comes from the fact that some edges or nodes must be temporarily removed from the network, no matter how short that

time period could be. However, we argue that this is close in spirit to the automated quarantine techniques promoted in Han et al. [Han+16a], with the added guarantee that our strategy is guaranteed to be short, if we assume that the parameter estimation of the epidemic model is accurate.

4.4.2 Contributions

The central aspect of this thesis is the analysis of the topology optimizer system which forms the core of the network security system we have presented above. Our contributions can be separated into two categories.

- From a theoretical point of view, we design a randomized approximation algorithm for the optimization problem corresponding to our topology optimizer system and provide evidence that the performance guarantee we have derived is tight for this algorithm.
- From an experimental point of view, we develop a novel testing methodology for the evaluation of randomized approximation algorithms inspired by existing methodologies used in machine learning. In particular, this methodology allows us to compare algorithms that may be randomized and may return infeasible, approximate solutions.

Our theoretical contributions are gathered in Chapter 5 which is based on work submitted to the Journal of Combinatorial Optimization [BBG20] in response to an invitation to expand our paper presented at the 12$^{\text{th}}$ Annual International Conference on Combinatorial Optimization and Applications [BBG18]. Contributions to the experimental evaluation of algorithms are gathered in Chapter 6 in preparation for a future submission to the 18$^{\text{th}}$ Symposium on Experimental Algorithms [BBG].

5 Approximating the maximum spectral subgraph

5.1 Introduction

Modifying the topology of a network to mitigate the spread of an epidemic with epidemiological constant λ amounts to the **NP**-hard problem of finding a partial subgraph with spectral radius bounded above by λ. A simple objective could be to try to preserve as much as possible of the original graph and as such to find a partial subgraph with maximum number of edges. A software-defined network (SDN) capable of real-time topology reconfiguration can then use an algorithm for finding such subgraph to quickly remove spreading malware threats without deploying specific security countermeasures.

In this chapter, we propose a novel randomized approximation algorithm based on the relaxation and rounding framework that achieves a $O(\log n)$ approximation in the case of finding a subgraph with spectral radius bounded by $\lambda \in (\log n, \lambda_{\max}(G))$ where $\lambda_{\max}(G)$ is the spectral radius of the input graph and n its number of nodes. We combine this algorithm with a maximum matching algorithm to obtain a $O(\log^2 n)$-approximation algorithm for all values of λ. We also describe how the mathematical programming formulation we give has several advantages over previous approaches which attempted at finding a subgraph with minimum spectral radius given an edge removal budget. Finally, we show that the analysis of our randomized rounding scheme is essentially tight by relating it to classical results from random graph theory.

5.1.1 Context

In recent years, a sequence of results [Cha+08]; [GMT05]; [VOK09]; [Wan+03] have established a relationship between the convergence of Markovian models representing an epidemic spreading over a network and the spectral characteristics of the underlying graph. The generalization of these theorems by Prakash et al. [Pra+11] states that in the case of a graph G and an epidemic model with epidemiological characteristic λ, fast convergence of the Markovian model to its absorbing state is guaranteed if the spectral radius of the graph

$\lambda_{\max}(G) < \lambda$. This has led the mathematical epidemiology community to look for algorithms that modify the topology of a network to ensure that a given epidemic converges rapidly to extinction.

At the same time, the software-defined networking (SDN) paradigm has transformed network administration by allowing real-time statistics [SG12] and topology reconfiguration [WNS12]. This new paradigm has deep consequences for the management of network security as it is now possible for a SDN controller to automatically detect malware spreading over its network via machine learning [MKK11] and react to such threat by deploying adequate security countermeasures. In this work we are following epidemiological practice and propose to use topology modification as a disease-agnostic countermeasure to the spread of malware in networks.

We are looking to preserve as much as possible the existing network topology by keeping the largest number of edges in the graph while guaranteeing that a given epidemic of epidemiological characteristic λ would rapidly disappear. For this purpose, we introduce the maximum spectral subgraph problem (MSSP) defined formally as follows. Denoting by $\lambda_{\max}(G)$ the spectral radius of G, i.e. the largest eigenvalue of its adjacency matrix A, we have:

MAXIMUM SPECTRAL SUBGRAPH PROBLEM (MSSP)	
Instance:	an undirected graph $G = (V, E)$ and an epidemic model with critical threshold $1 \leq \lambda < \lambda_{\max}(G)$.
Goal:	max.
Metric:	the size of the partial subgraph graph: $\lvert E' \rvert$.
Domain:	any partial subgraph $H = (V, E')$ such that $E' \subset E$ with bounded spectrum $\lambda_{\max}(H) \leq \lambda$.

5.1.2 Related work

Spectral graph theory has often been a decisive tool in the design and analysis of algorithms. However, to the best of our knowledge, surprisingly few computational problems have been defined in terms of finding graphs with appropriate spectrum. The mathematical epidemiology community has proposed and analyzed several problems related to the spectrum of the adjacency matrix [Sah+15]; [Van+11] while systems and control researchers have considered optimization problems related to the spectrum of the Laplacian matrix [GB06]. In a separate effort, the theoretical computer science community has focused on problems related to the design of expander graphs and graphs with high

algebraic connectivity i.e. the second smallest eigenvalue of the Laplacian matrix [Kol+10]; [Mos08]. In this line of research, all problems are **NP**-hard and the algorithms proposed in the literature are often simple to state. We contrast this with the fact that their analysis can be involved and yet, to the best of our knowledge, only amount to conditional approximation guarantees. Throughout this chapter we qualify approximation algorithms by their performance guarantee $r > 1$ which means that they return solutions whose value is at least a fraction $1/r$ of the optimal value for maximization problems or at most a factor r of the optimal value for minimization problems.

A minimization version of MSSP has been studied by Saha et al. [Sah+15] where the task is to remove the minimum amount of edges from a graph G such that the resulting subgraph H satisfies $\lambda_{\max}(H) \leq \lambda$. They give a $(1+\varepsilon, \varepsilon^{-1}\log n)$ bi-criteria approximation algorithm which guarantees that if an optimal solution is to remove k edges to achieve a spectral radius less than or equal to λ then the algorithm will remove $O(\varepsilon^{-1}\log n)$ times more edges (with $n = |V|$ the number of nodes in G) and returns a graph with spectral radius less than or equal to $(1+\varepsilon)\lambda$. It is important to notice that if used "as is" the subgraph returned by the algorithm may not lead to the extinction of the epidemic. Zhang et al. [Zha+15] study the problem of maximizing the drop in spectral radius $\lambda_{\max}(G) - \lambda_{\max}(H)$ where H is a subgraph of G obtained by deleting at most k edges. Their randomized algorithm, inspired by the relaxation and rounding framework, has the following conditional guarantees: if the weighted graph obtained from the solution of the relaxed semidefinite programming problem has maximum weighted degree $\Delta^* = \Omega(\log^4 n)$, then the returned subgraph satisfies the constraint on the number of edge deletions in expectation and, with high probability, the remaining graph has a spectral radius within an additive $O(\sqrt{\Delta^*})$ factor of the optimal solution. If the condition on the maximum weighted degree is not satisfied, they do not obtain any performance guarantee. Here again, a drop in spectral radius is not guaranteed to result in the rapid extinction of an epidemic.

In this chapter we introduce the maximum spectral subgraph problem (MSSP) and our main contribution is the design of a $O(\log^2 n)$-approximation algorithm for MSSP obtained by combining a randomized algorithm based on the relaxation and rounding framework with a maximum matching algorithm. We also describe some shortcomings of existing mathematical programming formulation for variants of MSSP that attempt at minimizing the spectral radius of a given graph within a prescribed edge deletion budget. Finally we provide evidence that the analysis of our randomized algorithm is essentially tight by connecting it to classical results in random graph theory.

The rest of this chapter is organized as follows. In section 5.2 we recall some

simple facts from spectral graph theory and introduce appropriate notations and known results. In section 5.3 we describe our relaxation and rounding algorithm and illustrate its usage on star graphs. Then, in section 5.4, we prove its approximation ratio for the range $\lambda \in (\log n, \lambda_{\max}(G))$ in general graphs. In section 5.5 we show that a maximum matching is a $O(\lambda^2)$-approximation algorithm for MSSP. In section 5.6 we study the tightness of the analysis done in section 5.4 by looking at the performance of our relaxation and rounding algorithm for the range $\lambda < \log n$. Finally, perspectives and concluding remarks are provided in section 5.7.

5.2 Preliminaries

We review here useful facts about the spectrum of adjacency matrices of graphs. Unless specified, all graphs are assumed to be undirected. Recall that the adjacency matrix A of a graph $G = (V, E)$ is a symmetric matrix defined as follows:

$$A_{ij} = \begin{cases} 1 & \text{if } ij \in E \\ 0 & \text{otherwise} \end{cases}$$

Theorem 12. *(General bounds) [Van10]*

Given a graph $G = (V, E)$, we denote by $\Delta(G)$ its largest degree. The spectral radius of the graph, defined as the largest eigenvalue of its adjacency matrix, lies between the following quantities:

$$\max\left(\sqrt{\Delta(G)}, \frac{2|E|}{|V|}\right) \leq \lambda_{\max}(G) \leq \Delta(G) \tag{5.1}$$

5.2.1 Computational complexity

The problem of deciding whether there exists a subgraph with at least k edges and spectral radius at most λ was studied by van Mieghem et al. [Van+11]. We can see that it is the decision problem associated with both MSSP and the problem of minimum edge removal introduced by Saha et al. [Sah+15] that was mentioned in section 5.1.1. Van Mieghem et al. proved that their problem was **NP**-complete by showing that the Hamiltonian path problem is reducible to it. It follows from this result that MSSP is **NP**-hard. Furthermore, the Hamiltonian path problem remains **NP**-complete on specific graph classes such as bipartite graphs, planar graphs, and even on more restricted classes

such as the cubic subgraphs of the square grid graph. For further details, we direct the reader to the references in Bazgan et al. [BST99].

The reduction uses a fact from the field of extremal spectral graph theory (see for example [TT17]): the path graph $P_{|V|}$ on $|V|$ nodes is the graph with minimum spectral radius among all connected graphs with $|V|$ nodes and $|V|-1$ edges. Setting $\lambda = \lambda_{\max}(P_{|V|}) = 2\cos(\pi/(|V|+1))$ and $k = |V|-1$ completes the reduction. Recall that while the spectral radius of a graph might be an irrational number, verifying a candidate solution amounts to checking whether the eigenvalues of a given adjacency matrix are bounded above by a given value which can be done in polynomial time to any precision [PC99].

Note that if the bound on the spectral radius $\lambda = 1$, then MSSP becomes the maximum matching problem which can be solved in polynomial time. Indeed, from Theorem 12, it is easy to see that the problem consists in finding a subgraph of degree at most 1 with maximum number of edges. Furthermore, note that all undirected graphs that are not matchings have a spectral radius larger than or equal to $\sqrt{2}$ which is the spectral radius of the path graph on 3 nodes. From this consideration, we will study the range where the bound on the spectral radius is meaningful, that is $\sqrt{2} \leq \lambda < \lambda_{\max}(G)$.

We now present our algorithm based on the relaxation and randomized rounding framework.

5.3 Relaxation and matrix randomized rounding

The relaxation and randomized rounding framework [RT87] is a general algorithmic technique composed of two steps. The first step corresponds to solving a continuous relaxation of the original combinatorial problem, often represented by some mathematical programming formulation. The second step is then to sample a discrete solution based on an optimal solution of the relaxed problem. This technique has resulted in the design of a large number of approximation algorithms for a broad range of combinatorial problems and has been the cornerstone of the application of the sum of squares hierarchy developed by Lasserre [Las01] and Parrilo [Par03] in combinatorial optimization. There are often two steps in the analysis of a relaxation and randomized rounding algorithm: finding a tight relaxation of the original problem that is solvable in polynomial time and proving that the random discrete solution is feasible with high probability.

Here we propose a mathematical programming formulation of MSSP that uses semidefinite programming (SDP) to model the constraint on the spectral radius. While linear programming allows to define optimization problems

with non-negative vector variables written $x \geq 0$, SDP extends to the larger class of problems with positive semidefinite matrix variables written $X \succeq 0$ i.e. all eigenvalues of X are non-negative: $\forall\, i \in [1, n],\ \lambda_i(X) \geq 0$. In the literature, SDP is often use to strengthen a linear programming relaxation, such as in the celebrated result of Goemans and Williamson for the maximum cut problem [GW95]. However, this is not the case in our work as the semidefinite constraint is a natural consequence of the problem statement being concerned with the largest eigenvalue of a matrix.

Given an input graph $G = (V, E)$ and a bound on the spectral radius λ, we write the following semidefinite programming problem with binary variables:

$$\begin{aligned} \max \quad & \sum_{ij \in E} y_{ij} \\ \text{s.t.} \quad & \sum_{ij \in E} y_{ij} A_{ij} \preceq \lambda I \\ & \sum_{j \in \Gamma(i)} y_{ij} \leq \lambda^2,\ \forall i \in V \\ & y_{ij} \in \{0, 1\},\ \forall\, ij \in E \end{aligned} \tag{SDP$_{0,1}$}$$

where A_{ij} is the adjacency matrix of the graph $G_{ij} = (V, \{ij\})$ with a single edge ij and I is the identity matrix of size $|V|$. The decision variables y_{ij} represent whether an edge ij belongs to the subgraph when $y_{ij} = 1$ or not when $y_{ij} = 0$. Recall that for a n by n square matrix, $M \preceq tI \iff \forall\, i \in [1, n],\ \lambda_i(M) \leq t$. The semidefinite constraint ensures that the adjacency matrix of the subgraph, defined by the y_{ij} variables equal to 1, has its spectral radius bounded above by λ. The linear constraint ensures that the degree of each node $i \in V$ in the subgraph is bounded above by λ^2. Note that this constraint is redundant given that the general bounds of Theorem 12 state that the maximum degree of a graph is bounded above by the square of its spectral radius i.e. $\Delta \leq \lambda^2_{\max}$. However this is in general not the case with weighted graphs, which will be discussed in section 5.3.3.

The continuous relaxation of Problem (SDP$_{0,1}$) is obtained by relaxing integer constraints into box constraints. We underline that the semidefinite constraint does not originate from the relaxation as is the case for some problems which relax vector variables with quadratic constraints into a SDP problem e.g. the one used in the algorithm given by Goemans and Williamson for the maximum

cut problem [GW95]. Our relaxation is limited to the binary variables.

$$\begin{aligned} \max \quad & \sum_{ij \in E} y_{ij} \\ \text{s.t.} \quad & \sum_{ij \in E} y_{ij} A_{ij} \preceq \lambda I \\ & \sum_{j \in \Gamma(i)} y_{ij} \leq \lambda^2, \ \forall i \in V \\ & y_{ij} \in [0,1], \ \forall \, ij \in E \end{aligned} \tag{SDP$_{\lambda\Delta}$}$$

Since we are using Problem SDP$_{\lambda\Delta}$ as the basis of a sampling procedure which will presumably incur some approximation factor, it is not necessary to obtain an optimal solution. Instead, we can use polynomial-time algorithms which return a feasible solution whose value is within an arbitrary small additive factor $\varepsilon > 0$ of the optimal value, see for example Vanderberghe and Boyd [VB96]. In this sense, solving Problem SDP$_{\lambda\Delta}$ can be done in polynomial time. We note that the computational complexity of semidefinite programming, that is solving SDP problems exactly, is not fully understood yet, see for example the article of de Klerk and Vallentin [KV16] or the survey of Laurent and Rendl [LR05]. For some small $\varepsilon > 0$, this allows us to state our relaxation and randomized rounding algorithm which is then guaranteed to terminate in polynomial time. Denoting by $\boldsymbol{x} \sim \text{Ber}(\mu)$ the fact that $\boldsymbol{x}$ is a random variable following a Bernoulli distribution of mean μ, we have:

Algorithm 6: RELAXATION & RANDOMIZED ROUNDING

Data: $G = (V, E)$, $\sqrt{2} \leq \lambda < \lambda_{\max}(G)$, and $r > 1$.
Result: $H = (V, E')$ such that $\lambda_{\max}(H) \leq \lambda$ with some probability p.

1 $y^* \leftarrow \arg$ Problem (SDP$_{\lambda\Delta}$)
2 Sample $\forall \, ij \in E$, $\boldsymbol{x}_{ij} \sim \text{Ber}(y^*_{ij}/r)$
3 **return** $H = (V, \{ \, ij \in E : \boldsymbol{x}_{ij} = 1 \, \})$

We will now turn to a simple application of Algorithm 6 to the case of star graphs and determine the adequate sampling factor r that results in a feasible solution with high probability i.e. $p = 1 - 1/n$ where $n = |V|$.

5.3.1 The case of star graphs

Before giving the complete analysis of our relaxation and randomized rounding algorithm we focus on a specific class of input graphs to illustrate the methodology of relaxation and randomized rounding but also to highlight the

importance of the degree constraint in our proposed mathematical programming formulation.

Recall that a star graph $S_n = K_{1,n}$ is a graph with $V = \{0, \dots, n\}$ and $E = \{(0,1), \dots, (0,n)\}$. It is a well-known fact from spectral graph theory, see for example the textbook by Chung [Chu97], that the spectral radius of a star equals the square root of its number of edges i.e. $\lambda_{\max}(S_n) = \sqrt{n}$. More generally, it is easy to see that a weighted star graph S_w, where each edge ij is associated with weight w_{ij}, has spectral radius $\lambda_{\max}(S_w) = ||w||_2 = \sqrt{\sum_{ij \in E} w_{ij}^2}$. Notice that we recover the non-weighted case by setting every weight to be 1. Using this property, we determine that the number of edges in an optimal solution of Problem ($\text{SDP}_{0,1}$) is exactly $\lfloor \lambda^2 \rfloor$ edges. We denote the optimal value of MSSP on a star graph S_n and parameter λ by $\text{opt}(S_n, \lambda) = \lfloor \lambda^2 \rfloor$.

To analyze the gap between the combinatorial problem and our relaxation, we now compute the value of an optimal solution of Problem ($\text{SDP}_{\lambda\Delta}$). First, we can use the above definition of the spectral radius of a star graph to replace the semidefinite constraint by $||y||_2 \leq \lambda$. Second, we interpret the degree constraint as a constraint on the ℓ_1-norm of y. This means that we can compute an optimal solution of Problem ($\text{SDP}_{\lambda\Delta}$) by solving the following second-order cone programming problem:

$$\begin{aligned} \max_{y \in [0,1]^{|E|}} \quad & ||y||_1 \\ \text{s.t.} \quad & ||y||_2 \leq \lambda \\ & ||y||_1 \leq \lambda^2 \end{aligned} \tag{$\text{SOCP}_{\lambda\Delta}$}$$

By a geometrical argument, an optimal solution of this problem has value at most λ^2 and that can be achieved by any y such that $||y||_2 \leq \lambda$, e.g. the uniform solution where $\forall ij \in E, y^*_{ij} = \lambda^2/n$ has ℓ_2-norm $||y^*||_2 = \lambda^2/\sqrt{n}$. This solution is feasible since we assume that $\lambda < \lambda_{\max}(S_n) = \sqrt{n}$ which is easily seen to imply that $\lambda^2/\sqrt{n} \leq \lambda$. We denote by $\text{opt}_{\text{rel}}(S_n, \lambda) = \lambda^2$ the optimal value of the relaxation.

We now have a complete description of the integrality gap g_{S_n} of our relaxation for star graphs. The integrality gap is the largest ratio between the optimal value of the continuous relaxation and the optimal value of the original, combinatorial, problem:

$$g_{S_n} \stackrel{\text{def}}{=} \max_{S_n, \lambda} \frac{\text{opt}_{\text{rel}}(S_n, \lambda)}{\text{opt}(S_n, \lambda)} = \frac{\lambda^2}{\lfloor \lambda^2 \rfloor} \leq \frac{3}{2} \tag{5.2}$$

where the last inequality comes from the fact that $\lambda \geq \sqrt{2}$.

5.3.2 Erdös-Rényi stars

Now that we have solved a continuous relaxation of MSSP, we will use the computed optimal solution y^* to sample a discrete solution, here a random subgraph $S_{\boldsymbol{x}}$ of the original star graph S_n. For this purpose we introduce for each edge ij an independent random variable $\boldsymbol{x}_{ij} \sim \text{Ber}(y^*_{ij}/r)$. By definition the *random* number of edges $\boldsymbol{x}$ of the random subgraph $S_{\boldsymbol{x}}$ is a sum of independent Bernoulli random variables with mean $\mathbb{E}\,\boldsymbol{x} = \sum_{ij} y^*_{ij}/r = \lambda^2/r$.

If for some $r > 1$ the random subgraph $S_{\boldsymbol{x}}$ satisfies the spectral radius constraint with high probability, i.e. $p = 1 - O(1/n)$, then we would have a polynomial time randomized r-approximation algorithm. We obtain the following approximation algorithm in the case of star graphs:

Theorem A. *(Randomized $O(1)$-approximation for star graphs)*
Given a star graph S_n, a bound on the spectral radius $\lambda \geq \sqrt{2}$, and an optimal solution y^* of Problem (SOCP$_{\lambda\Delta}$), the random partial subgraph $S_{\boldsymbol{x}}$ obtained by keeping edges according to independent random variables $\boldsymbol{x}_{ij} \sim$ Ber(y^*_{ij}/r) is a feasible solution of MSSP *with probability $p \geq 2/3$ whose expected value is within a factor $r = 4$ of the optimal value.*

Proof. As is common practice in the analysis of randomized algorithms [MR10], we use the Chernoff bound to get an estimate of the probability that our sampled solution is feasible. Recall that the Chernoff bound, Theorem 7 in Chapter 3, gives an upper bound on the probability that a sum of independent random variables exceeds a certain value.

Using the general formulation of the bound, we write the following. Let $\boldsymbol{x} = \sum_{i=1}^{n} \boldsymbol{x}_i$ where each $\boldsymbol{x}_i$ is an independent random variable. Then, for a given value $a > 0$, the following estimate holds:

$$\Pr(\boldsymbol{x} \geq a) \leq \min_{t>0} e^{-ta} \prod_{i=1}^{n} \mathbb{E}\exp(t\boldsymbol{x}_i).$$

We directly apply the Chernoff bound on the random number of edges $\boldsymbol{x} = \sum_{ij\in E} \boldsymbol{x}_{ij}$ for the value $a = \lambda^2$.

First we start with a simple bound on the moment generating function of each summand. Denoting the parameter of each Bernoulli random variable by $p_{ij} = y^*_{ij}/r$, we have:

$$\mathbb{E}\exp(t\boldsymbol{x}_{ij}) = p_{ij}e^t + (1 - p_{ij})$$
$$\mathbb{E}\exp(t\boldsymbol{x}_{ij}) = 1 + p_{ij}(e^t - 1)$$

using the simple inequality $1+x \leq \exp(x)$ when $x > 0$ we obtain the following:

$$\mathbb{E}\exp(t\boldsymbol{x}_{ij}) \leq \exp\left(\frac{y^*_{ij}}{r}(e^t - 1)\right)$$

This allows us to write the following inequality:

$$\Pr(\boldsymbol{x} \geq \lambda^2) \leq \min_{t>0} e^{-t\lambda^2} \exp\left(\sum_{ij\in E} \frac{y^*_{ij}}{r}(e^t - 1)\right)$$

$$\Pr(\boldsymbol{x} \geq \lambda^2) \leq \min_{t>0} \exp\left(\frac{\lambda^2}{r}(e^t - 1) - t\lambda^2\right).$$

The minimum of the r.h.s. is attained at $t = \log r$ under the condition that $t > 0$ from which we deduce that $r = 1 + h$ for some $h > 0$. The bound then simplifies into:

$$\Pr(\boldsymbol{x} \geq \lambda^2) \leq \exp\left(\lambda^2\left(\frac{h}{1+h} - \log(1+h)\right)\right).$$

While there is a tighter bound, we choose $r = 1 + h = 4$ to produce the following simple expression which remains valid for any $\lambda \geq \sqrt{2}$:

$$\Pr(\boldsymbol{x} \geq \lambda^2) \leq \exp(-0.64\lambda^2) \leq \frac{1}{3}$$

which concludes the proof of Theorem A. Recall indeed that the event $\lambda_{\max}(S_{\boldsymbol{x}}) \geq \lambda$ is equivalent to the event $\boldsymbol{x} \geq \lambda^2$. □

Since our success probability for a single sample $p \geq 2/3$ we can amplify it by repetition in polynomial time to obtain a solution of expected value $\lambda^2/4$ and such that the solution is feasible with high probability $p = 1 - O(1/n)$.

To summarize the case of star graphs, our relaxation and randomized rounding algorithm is a polynomial time algorithm which returns with high probability a feasible star graph of expected size $\lambda^2/4$.

5.3.3 Without the degree constraint

It is important to notice that the degree constraint played a significant role in obtaining a constant factor approximation in the case of star graphs. Reusing the same analysis as in section 5.3.1 we can see that Problem (SDP$_{\lambda\Delta}$) without the degree constrained is equivalent to the following problem:

$$\begin{aligned} \max_{y\in[0,1]^{|E|}} \quad & ||y||_1 \\ \text{s.t.} \quad & ||y||_2 \leq \lambda \end{aligned} \tag{SOCP$_\lambda$}$$

By a geometrical argument, we notice that the uniform solution $\forall ij \in E, y_{ij} = \lambda/\sqrt{n}$ is the unique optimal solution of Problem (SOCP$_\lambda$). It follows that the associated optimal value $\mathrm{opt}_{\mathrm{rel}'}(S_n, \lambda) = \lambda\sqrt{n}$.

In that case, the integrality gap of the relaxation given by Problem (SOCP$_\lambda$) is

$$g'_{S_n} \overset{\text{def}}{=} \max_{S_n,\lambda} \frac{\mathrm{opt}_{\mathrm{rel}'}(S_n, \lambda)}{\mathrm{opt}(S_n, \lambda)} = \frac{\lambda\sqrt{n}}{\lfloor \lambda^2 \rfloor} = O\left(\frac{\sqrt{n}}{\lambda}\right) \tag{5.3}$$

which translates into a much higher $r = O(g'_{S_n})$ than the constant obtained in section 5.3.1. Indeed, the integrality gap for general graphs cannot be lower than the one derived for a specific class of graphs. Problem formulations focusing on minimizing the spectral radius given an edge deletion budget cannot a priori bound the maximum degree of the resulting weighted graph. This additional information is a key advantage over problems that optimize the spectral parameter.

We are now ready to describe our matrix randomized rounding whose analysis follows a similar structure to the one for star graphs. However we need to use more powerful concentration inequalities than the Chernoff bound to obtain bounds on the spectral radius of the random matrix we sample. This sampling can be seen as a special case of inhomogeneous $G(n, p)$ random graphs.

5.4 Spectral subgraphs in general graphs

In order to extend the analysis of Algorithm 6 to arbitrary graphs we turn to more advanced concentration inequalities that describe the behavior of random matrices and in particular their spectrum. Fortunately, recent results in the analysis of random matrices (cf. the survey by Tropp [Tro+15]) provide tail bounds for the largest eigenvalue of random matrices. These results are directly applicable to the analysis of Algorithm 6 for finding the sampling factor r that guarantees that the returned solution is feasible with high probability $p = 1 - 1/n$.

We start by presenting the generic matrix Bernstein bound and its application to adjacency matrices following the work of Radcliffe and Chung [CR11]. Finally we give the proof that Algorithm 6 is a randomized $O(\log n)$-approximation algorithm with the following property:

Theorem B. *(Randomized $O(\log n)$-approximation when $\lambda \geq \log n$)*

Given a graph $G = (V, E)$ with $|V| = n$, a bound on the spectral radius $\lambda \geq \log n$, and an optimal solution y^* of Problem (SDP$_{\lambda\Delta}$), the random subgraph H obtained by

keeping edges $ij \in E$ according to independent random variables $\boldsymbol{x}_{ij} \sim Ber(y^*_{ij}/r)$ is a feasible solution of MSSP with probability $p \geq 2/3$ whose expected value is within a factor $r = O(\log n)$ of the optimal value.

5.4.1 Following the matrix Bernstein bound

The matrix Bernstein bound is a generalization of the classical Bernstein bound to the setting of independent random matrices. The theorem states the following:

Theorem 13. *(Matrix Bernstein) [CR11]*
Let $\boldsymbol{X} = \sum_i \boldsymbol{X}_i$ where each summand $\boldsymbol{X}_i$ is an independent symmetric random matrix of size n which is centered, $\mathbb{E}\,\boldsymbol{X}_i = 0$, and bounded in spectral norm, $\lambda_{\max}(\boldsymbol{X}_i) \leq L$. We define the matrix variance of $\boldsymbol{X}$ by $v(\boldsymbol{X}) = \lambda_{\max}(\sum_i \mathbb{E}\,\boldsymbol{X}_i^2)$. The following tail inequality holds:

$$\Pr(\lambda_{\max}(\boldsymbol{X}) \geq a) \leq n \exp\left(-\frac{a^2}{2v(\boldsymbol{X}) + 2La/3}\right). \tag{5.4}$$

The output of Algorithm 6 corresponds to a random adjacency matrix $\boldsymbol{A}$ which is the sum of independent random adjacency matrices each corresponding to an edge in the random graph. Let $A_{ij} = (E_{ij} + E_{ji})$ where the E_{ij} form the canonical basis for $M_{n,n}$ and denote by $\boldsymbol{x}_{ij}$ a Bernoulli random variable of mean y^*_{ij}/r. We have the following:

$$\boldsymbol{A} = \sum_{ij \in E} \boldsymbol{x}_{ij} A_{ij} \tag{5.5}$$

Note that our random adjacency edges have non-zero mean $\mathbb{E}\,\boldsymbol{x}_{ij}A_{ij} = (y^*_{ij}/r)A_{ij}$. Fortunately, applying Weyl's inequalities on $\boldsymbol{A}$ and $\mathbb{E}\,\boldsymbol{A}$ will give us control over the spectral radius of $\boldsymbol{A}$ by proxy.

Theorem 14. [Bha13] *(Weyl's inequalities)*
Let X and Y be two symmetric matrices, for all $\varepsilon > 0$:

$$\lambda_{\max}(X - Y) \leq \varepsilon \implies |\lambda_{\max}(X) - \lambda_{\max}(Y)| \leq \varepsilon \tag{5.6}$$

We will use Theorem 14 with the specific value $\varepsilon = (1-1/r)\lambda$ on our centered random adjacency matrix to obtain the adequate bound on the spectral radius of $\boldsymbol{A}$. The centered random adjacency matrix is:

$$\boldsymbol{A} - \mathbb{E}\,\boldsymbol{A} = \sum_{ij \in E} \left(\boldsymbol{x}_{ij} - \frac{y^*_{ij}}{r} \right) A_{ij}$$

For this, we consider the event where $\boldsymbol{A}$ has greater spectral radius than $\mathbb{E}\,\boldsymbol{A}$ and drop the absolute value:

$$\lambda_{\max}(\boldsymbol{A} - \mathbb{E}\,\boldsymbol{A}) < \left(1 - \frac{1}{r}\right)\lambda \implies \lambda_{\max}(\boldsymbol{A}) - \lambda_{\max}(\mathbb{E}\,\boldsymbol{A}) < \left(1 - \frac{1}{r}\right)\lambda$$

and by feasibility of an optimal solution of the relaxed SDP, i.e. Problem ($\mathrm{SDP}_{\lambda\Delta}$), we have $\lambda_{\max}(\mathbb{E}\,\boldsymbol{A}) \leq \lambda/r$ which gives:

$$\lambda_{\max}(\boldsymbol{A} - \mathbb{E}\,\boldsymbol{A}) < \left(1 - \frac{1}{r}\right)\lambda \implies \lambda_{\max}(\boldsymbol{A}) < \lambda.$$

From the general bounds of Theorem 12 we know that the spectral radius of the centered adjacency matrix of a random edge ij is either y^*_{ij}/r (no edge) or $1 - y^*_{ij}/r$ (one edge) which lets us bound the spectrum of each summand. In the worst case we have, for each edge ij:

$$\lambda_{\max}\left(\left(\boldsymbol{x}_{ij} - \frac{y^*_{ij}}{r}\right) A_{ij}\right) \leq \max\left(\frac{y^*_{ij}}{r}, 1 - \frac{y^*_{ij}}{r}\right) \leq 1 \tag{5.7}$$

5.4.2 Proof of Theorem B

We start by computing $v(\boldsymbol{A} - \mathbb{E}\,\boldsymbol{A})$ the matrix variance, see for example the monograph by Tropp [Tro+15], of our centered random adjacency matrix :

$$v(\boldsymbol{A} - \mathbb{E}\,\boldsymbol{A}) = \lambda_{\max}\left(\sum_{ij \in E} \mathrm{Var}(\boldsymbol{x}_{ij} A_{ij})\right).$$

With the basic properties of the scalar variance $\mathrm{Var}(\boldsymbol{x}_{ij} A_{ij}) = \mathrm{Var}(\boldsymbol{x}_{ij}) A_{ij}^2$ and a simple property of the square of the adjacency matrix of an edge $A_{ij}^2 = D_i + D_j$ where $D_v = E_{vv}$ is a diagonal matrix, we obtain a clean expression for the variance of the centered adjacency matrix as the spectral radius of the matrix

of degree variances. In essence, each edge in the graph contributes its variance to its two incident nodes.

$$\begin{aligned} v(\boldsymbol{A} - \mathbb{E}\,\boldsymbol{A}) &= \lambda_{\max}\left(\sum_{ij\in E} \operatorname{Var}(\boldsymbol{x}_{ij})(D_i + D_j)\right) \\ &= \max_{i\in V} \sum_{j\in\Gamma(i)} \frac{y^*_{ij}}{r}\left(1 - \frac{y^*_{ij}}{r}\right) \\ v(\boldsymbol{A} - \mathbb{E}\,\boldsymbol{A}) &\leq \max_{i\in V} \sum_{j\in\Gamma(i)} \frac{y^*_{ij}}{r} \end{aligned}$$

where we have used classical formulas for the variance of a Bernoulli random variable, see for example the textbook by Bertsekas and Tsitsiklis [BT02]. By feasibility of an optimal solution of the relaxation, the degree constraint holds which means that $\max_{i\in V} \sum_{j\in\Gamma(i)} y^*_{ij} \leq \lambda^2$ and gives:

$$v(\boldsymbol{A} - \mathbb{E}\,\boldsymbol{A}) \leq \frac{\lambda^2}{r}.$$

We now fulfill all the prerequisites to apply Theorem 13, the matrix Bernstein bound, on $\boldsymbol{A} - \mathbb{E}\,\boldsymbol{A}$ and $L = 1$. To explicitly describe the fact that the approximation ratio $r > 1$ we introduce as earlier $h > 0$ such that $r = 1 + h$. We apply the Bernstein bound for the value $a = (h/(1+h))\lambda$:

$$\begin{aligned} \Pr\left(\lambda_{\max}(\boldsymbol{A} - \mathbb{E}\,\boldsymbol{A}) \geq \frac{h}{1+h}\lambda\right) &\leq n\exp\left(-\frac{1}{2}\frac{a^2}{v(\boldsymbol{A} - \mathbb{E}\,\boldsymbol{A}) + \frac{a}{3}}\right) \\ &= n\exp\left(-\frac{1}{2}\frac{a^2}{\frac{\lambda^2}{1+h} + \frac{1}{3}\frac{h}{1+h}\lambda}\right) \\ &\leq n\exp\left(-\frac{1}{2}\frac{h^2}{(1+h)^2}\,\frac{\lambda^2}{\frac{\lambda^2}{1+h} + \frac{1}{3}\frac{h}{1+h}\lambda}\right). \end{aligned}$$

We simplify the above expression to obtain:

$$\begin{aligned} \Pr\left(\lambda_{\max}(\boldsymbol{A} - \mathbb{E}\,\boldsymbol{A}) \geq \frac{h}{1+h}\lambda\right) &\leq n\exp\left(-\frac{1}{2}\frac{h^2}{(1+h)^2}\,\frac{\lambda^2}{\frac{\lambda}{1+h}(\lambda + h/3)}\right) \\ &= n\exp\left(-\frac{1}{2}\,\frac{h^2}{1+h}\,\frac{\lambda}{\lambda + h/3}\right). \end{aligned}$$

As in the case of star graphs, we will derive possible values for r (resp. for h) such that the probability of our subgraph H being infeasible is less than $1/3$. For this, we attempt to derive an upper bound for the argument of the exponential as $n\exp(-x) \leq \frac{1}{3}$ implies that $x \geq \log 3n$.

We are looking for values of h and λ such that the following inequality holds:

$$\frac{1}{2}\frac{h^2}{1+h}\frac{\lambda}{\lambda + h/3} \geq \log 3n$$

We start by deriving a lower bound on λ function of h. In the above inequality, $\lambda/(\lambda + h/3)$ can be arbitrarily small if h is unbounded. To prevent this, we impose that, for a certain constant $c > 0$:

$$\frac{1}{2}\frac{\lambda}{\lambda + h/3} \geq c$$

which implies that

$$\lambda \geq \frac{2c}{3-6c}h.$$

Choosing $c = 1/4$ gives us the condition that $\lambda \geq h/3$.

Now we are left with finding the value of h such that:

$$\frac{1}{4}\frac{h^2}{1+h} \geq \log 3n.$$

For all values of n, it is sufficient to take $h = 10\log n$ which completes the proof. □

Algorithm 6 is a randomized algorithm which returns a feasible solution with probability greater than $2/3$ and of expected value within a $(1 + 10\log n)$ factor of the optimal value whenever $\lambda \geq \log n$. Recall that the success probability of such an algorithm can be amplified to high probability in polynomial time. We now turn to a different algorithm to handle the range $\lambda \in [\sqrt{2}, \log n]$.

5.5 Maximum matching

After designing an approximation algorithm for MSSP for the range of the spectral bound $\lambda \in [\log n, \lambda_{\max}(G))$, we turn to the well-studied maximum matching problem: finding a subgraph M consisting of the maximum number of non-adjacent edges in a given graph G. The number of edges in M is often called the matching number $\nu(G)$ of the graph. We use a spectral generalization of a classical lower bound on the matching number due to Stevanović [Ste10] which states the following:

Theorem 15. [Ste10] *(Spectral lower bound on the matching number)*
Given a graph $G = (V, E)$ we have the following lower bound:

$$\nu(G) \geq \frac{|E|}{\lambda_{\max}^2(G) - 1}.$$

This static lower bound can be immediately turned into an approximation algorithm since computing a maximum matching can be done in polynomial time.

Algorithm 7: MAXIMUM MATCHING

Data: $G = (V, E)$, $\sqrt{2} \leq \lambda \leq \lambda_{\max}(G)$
Result: $H = (V, E')$ such that $\lambda_{\max}(H) \leq \lambda$
1 **return** $H = \arg \nu(G)$

Theorem C. *(Approximation by maximum matching)*
Given $G = (V, E)$ and a spectral bound $\lambda > 0$, a maximum matching of G is a $(\lambda^2 - 1)$-approximation for MSSP.

Proof. Denoting by H^* an optimal solution of MSSP for a graph G and spectral bound λ, we know that H^* is a partial subgraph of G which implies $\nu(G) \geq \nu(H^*)$. We also know that H^* is feasible i.e. $\lambda_{\max}(H^*) \leq \lambda$. Combining these two statements together with the lower bound of Stevanović, we obtain the following inequality:

$$\nu(G) \geq \nu(H^*) \geq \frac{\mathrm{opt}(G, \lambda)}{\lambda^2 - 1}$$

which shows that the size of a maximum matching is within a factor of $\lambda^2 - 1$ of an optimal solution of MSSP. Furthermore any matching has spectral radius equal to 1 i.e. is trivially feasible. □ □

Used in the range $\lambda \in [\sqrt{2}, \log n]$ a maximum matching is a $O(\log^2 n)$-approximation algorithm in the worst-case. We then combine Algorithm 6 with Algorithm 7 to obtain a $O(\log^2 n)$-approximation algorithm for all values of λ.

It is not entirely satisfactory to need a different algorithm for the range $\lambda \in [\sqrt{2}, \log n]$ and one might hope that stronger concentration inequalities would extend the performance guarantee of Algorithm 6 to such values of λ. However, we will see now that even if that were the case, the relaxation and randomized rounding algorithm cannot achieve an essentially better approximation ratio.

5.6 Independent rounding: an intrinsic $\Omega(\log n)$ barrier

In this section we describe an inherent limitation of our approach based on the relaxation and randomized rounding framework for small values of λ. While we are not able to show that Algorithm 6 is an approximation algorithm for $\lambda < \log n$, we show that assuming it were an r-approximation algorithm, its approximation ratio would necessarily be at least $\Omega(\log n)$ for constant $\lambda > 0$:

Theorem D. *(No better than $\Omega(\log n/\lambda^3)$)*
Given a complete graph K_n, a bound on the spectral radius $\lambda \geq \sqrt{2}$, we can construct an optimal solution $\forall ij \in E, y^*_{ij} = \lambda/(n-1)$ of Problem (SDP$_{\lambda\Delta}$) such that the random graph $G(n, \lambda/(n-1)r)$ is a feasible solution of MSSP *with probability $p \geq 2/3$ for $r = \Omega(\log n/\lambda^3)$.*

Since we are attempting at finding an algorithm-specific lower bound, we exhibit a family of instances for which there is no integrality gap but instead what we call a rounding gap: a random graph sampled from a fractional solution does not concentrate around its expected graph i.e. its spectral radius is $O(\sqrt{\log n})$ instead of $O(\lambda)$.

5.6.1 Cliques and the $G(n,p)$ random graph model

For this purpose, let us consider an instance composed of $G = K_n$ a complete graph on n nodes and $\lambda > 0$ a constant. We can easily obtain an optimal solution of SDP$_{\lambda\Delta}$ for this case. We start by showing that the uniform solution where $\forall ij \in E, y^*_{ij} = \lambda/(n-1)$ is feasible. Since the spectral radius is positive homogeneous, we have:

$$\lambda_{\max}\left(\sum_{ij\in E} \frac{\lambda}{n-1} A_{ij}\right) = \frac{\lambda}{n-1}\lambda_{\max}(A)$$

and we know that $\lambda_{\max}(A) = n-1$ since the complete graph is $(n-1)$-regular. This tells us that the above uniform solution is compatible with the semidefinite constraint as its spectral radius is exactly λ. It is also easy to see that the solution is a weighted regular graph of degree λ, trivially respecting the degree constraint: every node in the graph must have weighted degree at most λ^2.

To evaluate this solution, we start by noticing that the upper bound on the number of edges of an graph with known spectral radius, which can be deduced from the inequality of Theorem 12 relating average degree and

spectral radius, can be extended to weighted graphs. Using the Courant-Fischer definition of eigenvalues, it suffices to compute the Rayleigh quotient of an adjacency matrix A with the all-ones vector $\mathbf{1}$ to obtain a lower bound on the largest eigenvalue of the matrix:

$$\frac{\mathbf{1}^\intercal A \mathbf{1}}{\mathbf{1}^\intercal \mathbf{1}} \leq \max_{x \neq 0} \frac{x^\intercal A x}{x^\intercal x}.$$

Evaluating the left-hand side of this inequality and using the hand-shaking lemma yields the desired relationship. We apply this result on the weighted adjacency matrix $\sum_{ij} y^*_{ij} A_{ij}$ to obtain:

$$\frac{2}{n} \sum_{ij \in E} y^*_{ij} \leq \lambda \text{ that is } \sum_{ij \in E} y^*_{ij} \leq \frac{\lambda}{2} n.$$

On the other hand we simply compute the value of the uniform solution which attains the above upper bound:

$$\sum_{ij \in E} y^*_{ij} = \frac{n(n-1)}{2} \frac{\lambda}{n-1} = \frac{\lambda}{2} n.$$

The fact that the uniform solution is optimal for complete graphs has deep consequences on the applicability of our algorithm. Indeed in this case the independent rounding scheme with sampling factor r can be seen as the classical random graph model $G(n,p)$ [Gil59] with parameters $n = |V|$ and $p = O(\lambda/nr)$.

5.6.2 Proof of Theorem D

To prove Theorem D we leverage known probabilistic results on $G(n,p)$ random graphs. A broad family of results have been obtained for this random graph model and in particular Krivelevich and Sudakov [KS03] give a sharp estimate of their spectral radius.

Theorem 16. *[KS03] (Spectral radius of a sparse $G(n,p)$ random graph)*
For any $\lambda > 0$, let $p = \lambda/n$, we have that asymptotically almost surely:

$$\lambda_{\max}(G(n,p)) = (1 + o(1)) \max\left(np, \sqrt{\Delta_{n,p}}\right)$$

where $\Delta_{n,p}$ is a random variable representing the maximum degree of the random graph.

This theorem can be interpreted as saying that in this regime of p the spectral radius of a random graph matches the simple lower bound on the spectral radius of Theorem refgeneral-bounds. Indeed, the expected average degree $np = \lambda$ is the random analogue of the average degree while the square root of the random maximum degree $\sqrt{\Delta_{n,p}}$ corresponds to its static counterpart.

Furthermore, the random maximum degree is well understood for $p = \lambda/n$. Indeed a corollary of the above theorem states that for constant $\lambda > 0$, the average degree $np = O(1)$ and as such the spectral radius of $G(n,p)$ is entirely determined by its maximum degree. Indeed in this regime, see for example the textbook by Bollobás [Bol01], the maximum degree of $G(n,p)$ is:

$$\Delta(G(n,p)) = \frac{\log n}{\log \log n}$$

which implies that the spectral radius is almost surely the square root of the maximum degree:

$$\lambda_{\max}(G(n,p)) = (1+o(1))\sqrt{\frac{\log n}{\log \log n}}.$$

In order to understand how the maximum degree of $G(n,p)$ is affected by the edge probability p, we use a special case of a result from Bollobás [Bol80] to obtain an upper bound on the maximum degree that holds asymptotically as $n \to \infty$:

$$\Pr\left(\Delta_{n,p} < np + \sqrt{2np(1-p)\log n}\left(1 - \frac{\log\log n}{4\log n} - \frac{\log 2\sqrt{\pi}\log(3/2)}{2\log n}\right)\right) = \frac{2}{3}.$$

Using the simple upper bound $p(1-p) \leq p$ we can simplify it to obtain:

$$\Pr\left(\Delta_{n,p} < np + \sqrt{2np\log n}\left(1 - \frac{\log\log n}{4\log n} - \frac{\log 2\sqrt{\pi}\log(3/2)}{2\log n}\right)\right) = \frac{2}{3}.$$

Recall that in an independent randomized rounding procedure we sample each edge with probability $p/r = \lambda/nr$ for a certain sampling factor $r > 1$. Replacing p by p/r in the above expression, it is easy to see that in order to make the $\sqrt{2np\log n}$ term small compared to the np term in the asymptotic expression of the maximum degree, the sampling factor must be at least:

$$r = \Omega\left(\frac{\log n}{\lambda^3}\right).$$

Indeed, for such r, the bound on the maximum degree becomes asymptotically:

$$\Pr\left(\Delta_{n,\lambda/nr} < \frac{\lambda^4}{\log n} + (1-o(1))\sqrt{\left(1-\frac{\lambda^4}{n\log n}\right)\frac{\lambda^4\log n}{\log n}}\right) = \frac{2}{3}$$

which simplifies into:

$$\Pr\left(\Delta_{n,\lambda/nr} < \frac{\lambda^4}{\log n} + (1-o(1))\lambda^2\right) = \frac{2}{3}.$$

Following Theorem 16 by Krivelevich and Sudakov, we have that if $r = \Omega(\log n/\lambda^3)$ then asymptotically with probability 2/3:

$$\lambda_{\max}(G(n,\lambda/nr)) = (1-o(1))\lambda$$

in which case the random graph $G(n,\lambda/nr)$ is a feasible solution of MSSP whose value is within a factor r of the optimal value. With positive probability, there exists one graph such that Algorithm 6 can be a valid randomized approximation algorithm only if $r = \Omega(\log n/\lambda^3)$. □

We know that Algorithm 6 is a $O(\log n)$-approximation algorithm for $\lambda \geq \log n$ and one can hope that this extends to $\lambda < \log n$. Theorem D shows us that we cannot hope to obtain a better ratio than $\Omega(\log n)$ in general for this algorithm which indicates that our analysis, while incomplete, essentially matches the guarantee provided by our algorithm. Somewhat surprisingly, the performance guarantee of Algorithm 6 seems to be directly related to statistical properties of random graphs and not to the existence of an integrality gap as is commonly the case in the analysis of relaxation and rounding algorithms [WS11]. To the best of our knowledge, very few randomized approximation algorithms have been shown to exhibit bounded performance as a consequence of what we have called a rounding gap, see for example the paper by Bandeira [Ban18]. Based on these results, we posit that this phenomenon arises in the case of properties related to matrix norms. Indeed the maximum degree of an undirected graph is at the same time the ℓ_1-norm and the ℓ_∞-norm of its adjacency matrix whereas the spectral radius is its ℓ_2-norm.

5.7 Conclusion and perspectives

We have introduced the maximum spectral subgraph problem and designed a randomized $O(\log^2 n)$-approximation algorithm based on the relaxation and

rounding framework to solve it. Although we do not know of a matching lower bound, we have shown that the analysis of the relaxation and randomized rounding portion of our algorithm is essentially tight.

In terms of lower bounds, we currently do not have any result regarding hardness of approximation, but we are actively exploring this direction. To the best of our knowledge, no inapproximability results have been established for problems related to the spectrum of a graph. Indeed, **NP**-hardness results found in the literature [Mos08]; [Van+11] are based on reductions which relate extremal values in spectral graph theory to classical computational problems. These reductions cannot be directly extended to obtain an approximation gap.

Without a better lower bound than **NP**-hardness, we are compelled to find new techniques to improve our current upper bound. First, the continuous relaxation used in Algorithm 6 is rather natural aside from the redundant degree constraints. It would be interesting to see if stronger relaxations could be used to obtain more information about the random graph e.g. strong bounds on the variance of the random degrees. For this purpose we would like to consider a sum-of-squares relaxation for the binary semidefinite programming problem. Indeed, Nie [Nie11] has given an extension of the classical sum-of-squares hierarchy to include positivity certificates for matrix variables. This relates to the question of generalizing the results of Raghavendra [Rag08] on maximum constraint satisfaction problems where constraints apply to at most k variables to maximum constraint satisfaction problems with spectral constraints which, by definition, involve all variables at once. Aside from strengthening the relaxation, there is opportunity for improvement in developing more precise tail bounds on the spectrum of random adjacency matrices following recent results by van Handel [Han17] as well as by Le, Levina, and Vershynin [LLV17]. On a separate note, we are currently working on applying the method of conditional probabilities to derandomize Algorithm 6 in order to obtain a deterministic approximation algorithm. The analysis of section 5.5 focuses on the maximum matching problem as a way of computing a feasible solution for the range $\lambda \in [\sqrt{2}, \log n)$. It is natural to wonder whether the degree constrained subgraph problem which is a polynomial-time solvable generalization of the maximum matching problem [Sch03] with a constraint of the form $\Delta \leq \lambda$ could be proven to return a better solution, and possibly match the $O(\log n)$ ratio obtained by Algorithm 6.

Finally, we are also interested in applying a similar strategy to the problem of adding the smallest number of edges to reach a given algebraic connectivity i.e. a lower bound on the second smallest eigenvalue of the Laplacian matrix of the graph. This problem, proven **NP**-hard by Mosk-Aoyama [Mos08], is a variant of the problem of finding the maximum algebraic connectivity given

an edge addition budget proposed by Ghosh and Boyd [GB06]. While Kolla et al. have designed an approximation algorithm with conditional guarantees [Kol+10] for the original problem, we hope that our methodology could apply to the variant and lead to an unconditional approximation ratio.

6 Experimental design for randomized approximation algorithms

6.1 Introduction

Epidemics are a growing threat in telecommunication networks as well as in social networks. However, the recent discovery of the central role of network topology in the survival of epidemics together with the development of new technology like software-defined networking (SDN) which enable real-time network topology modification provide opportunities to defend against this new threat. Indeed, several algorithms have been proposed to compute, given an epidemic, an appropriate network topology modification which guarantees a short time to extinction for the epidemic. In this study we focus on the maximum spectral subgraph problem which consists in finding a partial subgraph with maximum number of edges such that its spectral radius is bounded above by an input parameter representing the intrinsic speed of the epidemic.

The primary objective of this chapter is to establish adequate tools to empirically determine the performance of a recently proposed randomized approximation algorithm for the maximum spectral subgraph problem based on semidefinite programming and matrix randomized rounding. For this purpose, we describe an experimental methodology using sequential optimization techniques to analyze the impact of a variety of practical algorithmic improvements to a direct implementation of the algorithm. We also provide preliminary empirical evidence of the competitiveness of the approximation algorithm compared to heuristics derived from algorithms proposed in the literature.

We start by reviewing the literature in the experimental study of algorithms which warns the research community in regarding the standards employed to establish scientific truth. Then, we survey existing statistical techniques that can enhance the reliability, validity, and reproducibility of experiments and discuss their applicability to computer experiments. Finally we propose an experimental setup to assess the performance of randomized approximation

algorithms based on sequential optimization techniques for the design of experiments.

6.2 The scientific method in experimental algorithmics

Computer science is a young scientific discipline whose practices have been inherited from the domains that led to the theory and practical implementation of computers such as mathematics and logic on one hand, physics and electrical engineering on the other hand. It has also received influence from fields where computers have been used to facilitate science such as applied mathematics, physics, chemistry, biology, and more recently the social sciences. Most of these scientific disciplines attempt to follow the scientific method which is characterized by the elaboration of hypotheses, often originating from a theory, followed by the design and conduct of experiments which have the goal of adding weight to a given hypothesis or refuting it instead.

However, according to Johnson [Joh02], the current practice of experimental research in algorithms for combinatorial optimization is far from meeting modern standards of reliability, statistical significance, and reproducibility. This state of affairs is at odds with the extreme degree of control available to researchers when conducting computer experiments, which is beyond any of the natural sciences. While strict methodology exists and is promoted by notable researchers in the community [BP14]; [BHL17]; [McG01] such techniques are rarely used in practice.

Instead each field of computing seems to develop their own methodology, such as the metaheuristics community which places a premium over solving difficult problem instances. In this context, tuning parameters to match specific instances and hiding implementation details to maintain a competitive advantage over other teams of researchers is valued more than reliability of results, and goes against the concept of scientific results being reproducible by the community.

In other communities such as numerical optimization, there is a long standing tradition of experimental studies for specific parameters such as the condition number, sparsity patterns or more complex structural information. The comparison of available methods can be done in a rigorous manner, see for example COCO platform which provides a common ground to compare continuous optimization solvers [Han+16b]. In communities focusing on discrete problems, it is common to see experimental studies that attempt to study a given implementation of an algorithm with ad-hoc methodological tools. For example, consider the famous study of an efficient implementation of the Lin–

Kernighan heuristic for the traveling salesman problem by Helsgaun [Hel00]. In this paper, the author considers a fully public fixed benchmark, the TSPlib [Rei91], and performs independent runs of his implementation on every instance. The author has split the dataset based on the number of nodes present in each instance and considers 100 independent runs for small instances and 10 independent runs for large instances according to the arbitrary threshold of 1000 nodes. While similarly straightforward experimental choices can sometimes be made for implementations of algorithms meant to solve well-studied problems, we will see that this is not the case in general.

6.3 Algorithms as experimental subjects

In experimental sciences, a simple goal is often to gather evidence supporting that two phenomenons are associated with each other. This is in particular true when studying fixed systems that have controllable inputs and observable outputs. Consider for example experiments in physical sciences where initial conditions and the duration of the experiment are the inputs, nature plays the role of the system often modeled by a differential equation, and measurements are the outputs. In essence, such experiments in physical attempt to identify the correct parameters of dynamical processes that fit with observations.

Algorithms on the other hand are not fixed systems with standard inputs and outputs. Given different inputs, the number and types of operations executed by an algorithm can be vastly different. To understand why, it is important to recall the definitions used in the theory of algorithms. Analogous to the definition of algorithms, a computational problem is defined over a set of instances which is generally infinite. Indeed, what draws the line between a single computation, which could be arbitrarily long and complex, and an algorithm is that an algorithm is defined over an unspecified input, the instance. Instances are often infinite families of common mathematical objects such as functions, sets, combinatorial structures, or even mathematical expressions.

As such, the experimental study of efficiency in algorithms as a domain must follow practices to prevent it from degenerating into the study of efficiently carrying out a single, potentially long and complex, computation. Indeed, since it is possible to consider endless variations and types of algorithms, such study would eventually establish the superiority, in terms of running time, of a "computation" which simply reads out the answer.

The practical efficiency of an algorithm implementation must be characterized by its observed performance over a wide variety of instances. However two major difficulties arise from this requirement.

6.3.1 Performance metrics

The first complication is related to the ambiguity of summarizing performance over a set of runs, i.e. the definition and use of appropriate performance metrics. When several alternatives are compared, it is possible to rely on relative metrics such as the performance profiles proposed by Dolan and Moré [DM02]. In their paper, the authors propose to rate an optimization algorithm implementation s, for solver, by the ratio of the time $t_{p,s}$ it used to solve a given problem instance p with the time spent by the best solver among the alternatives S. The performance ratio denoted by $r_{s,p}$ is then given by

$$r_{s,p} = \frac{t_{p,s}}{\min_{s' \in S} t_{p,s'}}$$

and the performance profile ρ_s of an implementation s for a given set of instances P is the empirical estimate of the probability that a solver s is within a factor τ of the best possible performance ratio.

$$\rho_s(\tau) = \frac{|\{p \in P : r_{s,p} \leq \tau\}|}{|P|}.$$

This allows the comparison of different solvers by plotting out their performance profile for all values $\tau \in \mathbb{R}$ as a cumulative distribution function $\rho_s : \mathbb{R} \to [0, 1]$ and reading out which algorithm implementation has the highest probability of being the best out of the alternatives.

Because performance profiles handle running time as relative to the best solver, they cannot provide experimental evidence of the asymptotic behavior of algorithm implementations. Even then, performance profiles are statistics that focus primarily on the running time of exact optimization algorithms and relegates failure to produce a solution or approximate solutions as second concerns by modeling either as arbitrarily high performance ratios. It is also clear that this type of metrics is entirely dependent on the available alternatives and on the dataset of instances that was studied and provides limited venues for generalization.

6.3.2 Datasets

The second issue is concerned with instances and can be illustrated with the same example of Dolan and Moré's performance profiles. As algorithm implementations are man-made designs, it is at the same time legitimate and misguided for algorithm designers to select a particular algorithm given computed metrics, or to modify an implementation to achieve a better score. A

simple way to counter this tendency is to follow the classical approach in machine learning of splitting the data in three sets: the training set, used to fit the parameters of a given model, the validation set, which includes assessing the performance of hyper-parameters, and finally the testing set, which allows for an unbiased estimation of the performance of the model, see for example introductory texts in machine learning such as the book by Bishop [Bis06].

Most competitions organized by the research community such as the PACE challenge [Del+17] follow this practice by sharing a public dataset with competitors for them to tune their implementations and withholding a private dataset on which evaluation will be conducted. In this sense, researchers must find algorithm parameters that generalize to the unseen data, provided that both public and private datasets are sampled from the same distribution.

However, as we will see in more details, the task of constructing datasets and of choosing specific distributions comes with many pitfalls.

6.3.3 Random samples and randomized experiments

Thus, the concept of efficiency intrinsically relies on the property of algorithms to generalize to unseen instances, to borrow terms from the machine learning literature. This forces us to rethink the statistical concept of random sampling, that is producing a finite number of samples on which the experiment will be conducted such that the sample is representative of the overall population (or more generally distribution). Unfortunately, it is not clear a priori what distributions underlie undirected graphs or Boolean formulas, if any. For this reason, it is unlikely that sample bias correction methods [CM14] could be applied as-is.

In the statistical literature, most often when experiments are costly to conduct, it is difficult to obtain truly random samples and researchers are often bound to work with a subset of a given population they have access to. To assess the validity of a phenomenon researchers then conduct randomized experiments and apply statistical methods to determine whether the observed effect was real or probably due to chance, see for example the Rubin framework of potential outcomes [Rub74]. However, it is important to highlight that randomized experiments do not fix the discrepancy between the sample and the overall population, they instead remove bias in the effect observed for the sample. In particular, randomized experiments can minimize the impact of external factors on the outcome of an experiment.

6.4 Factors of influence

Before attempting to propose a concrete methodology to conduct experiments in approximate optimization, we need to identify key factors that may influence the preparation, cost, and outcome of experiments.

In natural sciences researchers are often looking for causal relationships that support or invalidate an hypothesized mechanism for a given phenomenon. For this they study the influence of a factor on a particular outcome. A common example would be the study of a group of patients, each choosing between different drugs and recovering, or not, from a disease. However, possible factors may influence both their choice and their recovery, for example gender or age. The goal of analysis is to unveil a causal relationship that is exempt from the influence of such confounding factors. For an introduction to statistical analysis with a focus on confounding factors see for example the handbook by Dallal [Dal+12] which portrays the way statistics are used by epidemiologists to establish causal relationships.

In computational experiments, the environment is almost entirely controlled by researchers, and as such the methodology used in natural sciences cannot be easily reused. However, there are many factors that can contribute to the outcome of the experiment. In the experimental study of algorithms in particular, experiment designers must find the proper balance between the exploration of alternatives scenarios that might shed more light on a given hypothesis and the cost incurred by multiplying experiments. Indeed not all factors involve

No matter the chosen experimental design, it is necessary to explicitly list out the multiple factors that can alter the running time of programs designed to produce approximate solutions to optimization problems. We summarize prominent factors and some of their properties in Table 6.1.

6.4.1 Problem instances

The first category of factors is related to instances and is the area where experiments in optimization differ the most from experiments in machine learning. Indeed, while practical machine learning tasks are generally evaluated on real life data [Den+09]; [Wan+18], many optimization problems are relevant on a very wide variety of instances. In this sense, experiments in optimization have a much heavier reliance on instance generation than in the context of machine learning.

In well-studied domains of optimization, datasets have been consolidated to gather relevant examples of instances. For example the SNDlib dataset

Category	Factor	Cost	Notes
Instance	Benchmarks	High	Often inherited from competitions for specific problems.
	Parameters	Low	Especially amenable to experimental design.
	Size	Med	Impacted by time constraints.
	Invariants	Med	Often necessitates custom instance generators.
Hardware	Architecture	High	Via on-demand computing platforms.
	Parallelism	Low	Toggling GPU, restricting number of threads/cores.
Software	Runtime	High	Via containers and virtualization.
	Implementation	High	Often inherited from existing libraries.
	Parameters	Low	Especially amenable to experimental design.

Table 6.1: Factors influencing experiments on approximate optimization

[Orl+10] contains several real-life networks meant to serve as instances of the survivable network design problem. Similarly the TSPlib [Rei91] collects instances of the traveling salesman problem. Even when assuming that these benchmarks provide a reliable way to compare algorithms in practice, most optimization problems, and especially newly defined problems such as the maximum spectral subgraph problem, do not come with existing benchmarks. The construction and maintenance of datasets are very demanding tasks which explains that most benchmarks are associated with community efforts such as competitions [Del+17] or industry sponsorship [B+07].

The large majority of combinatorial optimization problems studied in the literature are defined over instances which contain a combinatorial structure such as a set, a graph, a lattice, or a logical expression. However, they also often contain numerical parameters such as a number of paths to be found in a graph, weights that represent the profit gained from covering given elements, a total budget, and many more. Furthermore, the definition of combinatorial optimization problems often imposes very little restriction on what value these parameters could be. On the other hand, hardness results are often obtained when considering very specific choices of parameters.

A powerful notion in both the theory of algorithms and computational

complexity is the notion of instance invariants, more commonly known as parameters. The field of parameterized complexity is dedicated to studying the contribution of instance structure to solving problems and to design algorithms that can take advantage of that additional information [Cyg+15]. This approach can be successfully combined with approximation algorithms to obtain practical and efficient algorithms for **NP**-hard problems [Mar08]; [Baz+14]. However, measuring the impact of invariants requires the generation (or collection) of specific instances, which can become a large part of the total effort to conduct experiments. Sometimes, relevant parameters are **NP**-hard to compute exactly, e.g. treewidth or cliquewidth, and associated generation problems **NP**-hard as well.

6.4.2 Implementation of algorithms

The second category of factors that can greatly influence the progress of a set of experiments is the implementation of algorithms.

The hardware resources used to run an experiment can crucially alter the performance of specific experiments. We describe two representative examples. As long as the main memory is large enough to contain a given problem instance, memory access is rarely a limiting factor in the running time of a program implementing a candidate algorithm. However, as soon as the instance size necessitates disk access, it is common for the same program to incur running time penalties of at least one order of magnitude. Another example is the degree of parallelism that a given architecture can provide to a program. While it is generally preferable to make use of most available threads for embarrassingly parallel tasks, e.g. a parallel map operation, parallelism can be counterproductive in tasks where the amount of work expanded for memory transfers compensates the running time savings.

It is still possible to systematically explore the impact of hardware architecture by using on-demand computing platforms such as Amazon AWS, Microsoft Azure, DigitalOcean, and many others. These services provide a large selection of architectures, albeit at the price of having no control over the virtualization infrastructure used by such platforms.

The availability of specific hardware is another major factor in the performance of a given algorithm. Many efficient algorithms make use of linear algebra primitives which have been extensively optimized both in numerical libraries and hardware architectures. Such algorithms can experience dramatic speedups with compatible accelerators such as graphics processing units (GPU) [NBG08] or even more specialized components such as tensor processing units (TPU) [Jou+17]. The success of modern neural networks models is attributed

in a large part to efficient implementations of the back-propagation algorithm on GPU [RMN09] which enable several orders of magnitude improvements over standard implementations on CPU.

Intimately connected to specific hardware, a key factor in the performance of algorithm implementations is the reliance on software libraries offering high-performance implementations of common subroutines such as combinatorial algorithms like sorting, or numerical algorithms like eigenvalue computation [Vir+19]. However, such subroutines almost always come with optional parameters that alter the behavior of the program. For example, consider the `eigsh` function provided by the SciPy Python library which returns approximate eigenpairs of an Hermitian matrix. This Python function is essentially an interface to the ARPACK [LSY98] collection of Fortran77 subroutines to solve large scale eigenvalue problems. While it is possible to use the default settings provided by the SciPy library, the following optional parameters are available:

```
scipy.sparse.linalg.eigsh(A,
    k=6,
    M=None,
    sigma=None,
    which='LM',
    v0=None,
    ncv=None,
    maxiter=None,
    tol=0,
    return_eigenvectors=True,
    Minv=None,
    OPinv=None,
    mode='normal')
```

While not all parameters are critical to the running time of each call of this function in practice, the `maxiter` and `tol` parameters which default respectively to $10n$, where n is the size of the matrix `A`, and to machine precision, which is architecture dependent, provide qualitatively different trade-offs between the desired precision of eigenvalues and running time.

These apparently small differences in implementations can lead to considerable effective speedups, especially when the loss of precision (approximately sorted over sorted, approximate eigenvalue over machine precision eigenvalue computations) does not drastically modify the returned solution.

Unlike the study by Helsgaun [Hel00] we have mentioned above which considered a black-box implementation of a famous heuristic, we can expose these algorithmic variants and subroutine parameters to the toolbox of experimental design in order to produce fair and accurate assessments of the performance of an implementation.

6.5 Towards systematic parameter setting

The previous section highlighted the key factors which can influence the observed efficiency of an algorithm implementation run on a particular instance. In particular, many numerical factors and categorical factors can be seen as inputs of an experimental setup. In this section, we survey the possible techniques to select adequate parameters and comment on the popularity of these techniques in different research communities.

In recent years, the machine learning community has seen increased activity in the domain of parameter selection which corresponds in this context to finding appropriate hyper-parameters for given machine learning models whose parameters have been fit to a fixed set of instances, the training set. See for example the algorithms proposed by Bergstra et al. [Ber+11].

However, methodological practices widely differ based on the research community and on the sociological adherence to values associated with statistical evidence.

Hand-picked values: this corresponds to the usage of particular numerical values or combinatorial objects without justification. It can be seen for example in the operations research literature where the size of generated instance is often a multiple of 10.

"Grad student descent": this portmanteau between graduate student and gradient descent is the practice of exploiting human resources commonly found in research laboratories to guide the search for better parameters. It is often used in the metaheuristics community or the machine learning community in the context of competitions. It is commonly accompanied with other types of anti-replication behavior [Hut18] such as closed source software or partial publication of experimental runs.

Racing: this entirely empirical methodology proposes to exploit parallelism and early stopping to select adequate parameters [Lóp+16]. This practice has shown some success in fields with standardized benchmarks such as mathematical programming solvers but a posteriori theoretical explanations of its relevance have been rare so far.

Random search: the most common and most widely trusted technique for setting parameters is the use of randomness [BB12]. It is often combined with massive parallelism and racing techniques [Li+17]. This technique has been shown to thrive in high dimensional settings but it requires the experimenter to specify particular bounds and distributions which are not

necessarily part of the problem description. Like most techniques which require parameters, random search can suffer from a case of higher-order "hand-picked values".

Bayesian optimization and bandit algorithms: two well-understood mathematical frameworks for sequential optimization which enjoy theoretical performance guarantees [Mar14]; [Sha+16]. Both can be seen as instances of derivative-free optimization, that is algorithms that try to optimize a function with access to a function evaluation oracle. Because of its sequential nature, Bayesian optimization is not directly amenable to parallelism. Despite their mathematical qualities, there is an ongoing debate regarding their practical effectiveness in comparison with pure random search. Furthermore, both frameworks require experimenters to model their a priori knowledge of the experimental setups in terms of objective functions to guide the exploration of parameters. While this may seem to be a net loss compared to manually selecting parameters, the number of parameters is at the same time much lower than the original experimental setup and semantically richer since it encodes the goals of the experimenter. These two frameworks are especially effective in terms of obtaining statistically significant results from a small number of experiments which is not necessarily the case with random search. In terms of applicability, Bayesian optimization in the form of Gaussian process regression [Ras04] is the most flexible method and can be adapted to continuous, discrete, and even hierarchical data and can handle noisy experiments, constraints on exploration, and more. Bandit algorithms on the other hand can be seen as a very specific finite-dimensional analogue which enjoy much stronger theoretical guarantees [B+12].

Optimal design of experiments: a class of techniques which attempt to find estimators of maximum information [Sac+89]; [JKB06]. While Bayesian optimization under the name of kriging [Kle09]; [R+12]; [Che+14] is often considered part of optimal design, other techniques are centered around finding optimal parameters assuming strong prior information on the behavior of the model, which is often the case in statistics but not in machine learning or algorithmics.

6.5.1 Gaussian process regression

Gaussian process regression [Ras04] is the core of the modern applications of Bayesian optimization in the machine learning community to tune model hyper-parameters. However, several issues are included with the technique.

First, it is often difficult to select the parameters of the method itself called kernels which sometimes come with their own parameters. Kernels are used in Gaussian process regression to encode prior information on the experiment and there is ongoing effort to find simple rules to select them [Duv14].

Acquisition functions are another category of parameters of the method which describe the goal of the experimenter in finding the next experiment to conduct [Rus+18]. They represent the trade-off between exploration, learning about the unknown probability distribution, and exploitation, finding points where the distribution has high value. For example, an experimenter might want to select the experiment setup that is most likely going to minimize the uncertainty over the effect of a parameter. Such acquisition functions include variants of the upper confidence bound. Alternatively, it is possible to aim for an experimental setup which will find the expected best possible value according to the objective function. In practice, this is the most commonly used acquisition function, named the expected improvement function.

While Gaussian process regression is the most common choice for Bayesian optimization, alternatives such as t-Student processes [SWG14] and random forest processes have been shown to exhibit similar practical performance.

6.5.2 Libraries

A large number of open source software and libraries offer solutions to perform Bayesian optimisation with Gaussian process regression, Tree-structured Parzen Estimators (TPE), or in other cases model-free methods. A quick survey of the literature has shown that simple practitioners, i.e. researchers who do not develop alternative solutions, seem to use Hyperopt and Scikit-Optimize most frequently.

- Hyperopt [BYC13a]; [BYC13b] is an easy to use library providing tree-structure Parzen estimators optimization methods which allow for the exploration of different types of parameters (continuous, discrete, hierarchical).

- Hyperband [Li+17] is a random search library enhanced with racing techniques which can take advantage of parallel computing resources.

- Scikit-Optimize [Hea+18] is similar to Hyperopt but provides a large selection of competing methods including a simple implementation of random search.

- SMAC [HHL11] is a sequential optimization software suite created by a team focusing on using statistical methods for combinatorial optimization.

- PyMC3 [SWF16] is a general purpose library designed for probabilistic programming which includes a Gaussian process regression module.

6.6 Issues in experimental evaluation of algorithms

6.6.1 Qualitative differences

In the context of optimization, algorithms are expected to provide a solution, i.e. a mathematical object which satisfies a set of constraints and provides the best possible score in terms of the objective function. In that setting, it is relatively straightforward to compare different algorithms by measuring the time they take to reach an optimal solution. For **NP**-hard optimization problems, it is often more practical to set a fixed time limit and compare how many instances have been solved optimally as is commonly done for challenges [Del+17]; [BS18]. However, no clear standards have been established to compare optimization algorithms that are not necessarily correct, exact, or deterministic.

Correctness can be relaxed in at least two ways: the WALK algorithm proposed by Saha et al. [Sah+15] for example may return an unfeasible solution as the output graph satisfies $\lambda_{\max}(H) \leq (1+\varepsilon)\lambda$ instead of the required $\lambda_{\max}(H) \leq \lambda$. The $\lambda\Delta$-RR algorithm proposed by the present authors [BBG18] is analyzed as a Monte Carlo algorithm i.e. it has a positive probability of returning an unfeasible solution even if this probability can be controlled by the user. Furthermore, the algorithms we consider in this study are heuristics or approximation algorithms: none of them are guaranteed to return an optimal solution.

Additionally, algorithms can differ in the way they construct candidate solutions: while some algorithms start with a candidate solution and improve on it step by step, other methods must finish all their steps before returning anything. While this is not necessarily a problem in classical algorithm racing challenges, the fact that some iterative algorithms can be stopped at every step is a desired property in the context of approximation algorithms.

To complicate things even more, some algorithms can be naturally converted into distributed or parallel algorithms, whereas some others are inherently sequential.

In this complex setting, it is difficult to design fair and unbiased benchmarks to compare qualitatively different algorithms. However, we have so far only mentioned theoretical differences in algorithms and have not considered the context of modern software practice when implementing these algorithms.

6.6.2 Practical implementation details

Practical implementations of optimization algorithms are not direct equivalent of the algorithms that are analyzed in theory. Clever preprocessing techniques, tuning of hyperparameters, parallel implementations, and access to existing software are essential building blocks when implementing state-of-the-art optimization algorithms. We illustrate this fact by describing the details of an implementation of the $\lambda\Delta$-RR algorithm, that is Algorithm 8 presented in [BBG18].

Algorithm 8: $\lambda\Delta$ RELAXATION & RANDOMIZED ROUNDING

Data: $G = (V, E)$, $\sqrt{2} \leq \lambda < \lambda_{\max}(G)$.
Result: $H = (V, E')$ such that $\lambda_{\max}(H) \leq \lambda$ with probability $p = 1 - 1/|V|$.

1 **if** $\lambda \geq \log |V|$ **then**

2 $$y^* \leftarrow \arg \begin{cases} \max & \displaystyle\sum_{ij \in E} y_{ij} \\ \text{s.t.} & \displaystyle\sum_{ij \in E} y_{ij} A_{ij} \preceq \lambda I \\ & \displaystyle\sum_{j \in \Gamma(i)} y_{ij} \leq \lambda^2, \ \forall i \in V \\ & y_{ij} \in [0, 1], \ \forall\, ij \in E \end{cases}$$

3 Set $r = 1 + 10 \log |V|$

4 Sample $\forall\, ij \in E$, $\boldsymbol{x}_{ij} \sim \text{Ber}(y^*_{ij}/r)$

5 **return** $H = (V, \{\, ij \in E : \boldsymbol{x}_{ij} = 1 \,\})$

6 **else**

7 **return** $H = \arg \nu(G)$

8 **end**

We will now describe how this simple relaxation and rounding algorithm can be implemented in practice. We focus on the **if** branch of the algorithm.

The first step of the implementation is related to constructing and solving a semidefinite program (line 2). This is heavily dependent on the semidefinite solver employed. In order to consider our algorithm as a case study in relaxation and rounding implementations, we have focused on general-purpose semidefinite solvers and left for further work the investigation of more specific

solvers such as those based on the spectral bundle method [HR00], which can be used to solve SDP problems in dual form, i.e. maximizing a linear functional under a semidefinite constraint like in our problem. Instead, we have focused our attention on the SuperSCS first-order solver [SMP17]; [TP16]; [ODo+16] which has been shown to outperform a large panel of general-purpose approximate SDP solvers. SuperSCS accepts constraint matrices as input which we construct using the Numpy library and take care to not generate redundant degree constraints which would be trivially true for the given adjacency matrix. The SuperSCS solver comes with several parameters including: the desired precision `eps`, an optional warm-start solution `x`, the amount of memory used by the LBFGS subroutine `memory`, as well as a toggle to use the GPU to compute solutions of linear systems `gpu`. This solver makes use of the BLAS and LAPACK linear algebra subroutine packages. As we do not require an exactly optimal solution we can tune these parameters to minimize the time spent solving the SDP problem.

The second step of the implementation involves finding the smallest possible $r \in (1, 1 + 10 \log |V|)$ (line 3) which outputs feasible random graphs with relatively high probability. For this purpose we implement a dichotomy procedure which aims at minimizing r under the constraint of an empirical sample of `dichotomy-sample-size` graphs being at least `dichotomy-sample-tol` percent feasible with a stopping condition based on a threshold `dichotomy-improve` of relative improvement in r.

After this setting of r, we can sample subgraphs (line 4) and test their feasibility in a massively parallel fashion. We then select the best feasible subgraph that has been produced so far and return it (line 5. Testing the feasibility of a subgraph amounts to computing the largest eigenvalue of its adjacency matrix and comparing it to the bound λ. This can be done using third-party linear algebra software like ARPACK [LSY98], a high-performance implementation of the Implicitly Restarted Lanczos Method in Fortran77. This program which computes approximate solutions to the eigenvalue problem $Ax = \lambda x$ has many hyperparameters including `tol` a floating point number which represents the relative accuracy required when computing eigenvalues and acts as a stopping criterion. According to the ARPACK manual, if `tol` is set to 0.0 then the program uses machine precision for the accuracy.

Algorithm 8 is presented as a Monte Carlo algorithm but since feasibility can be checked in polynomial time, it can be converted in a Las Vegas algorithm. This flexibility allows us to adapt it to settings where time is not a hard constraint to benefit from its expected polynomial running time.

It is thus clear that algorithm implementations cannot be evaluated without taking into account: third-party software dependencies, tuning of hyperparam-

Step	Line	Software	Parameters	Parallel
construct SDP	2	Numpy	–	–
solve SDP	2	SuperSCS	`eps`, `x`, `memory`, `gpu`	GPU
select r	3	ARPACK	`ratio-feasibility-tol`	Joblib
sample subgraphs	4	ARPACK	–	Joblib
return best subgraph	5	–	–	reduce

Table 6.2: Details of the $\lambda\Delta$ algorithm

eters, or simple tricks developed to accelerate computation.

We note that experimental evaluation of algorithms should also be considered through the lens of recent theoretical developments. For example, the new result by Ahmadinejad et al. [Ahm+19] which provides nearly linear time algorithms for computing the largest eigenvalue of a non-negative matrix as well as its associated eigenvector.

6.7 Experiments

We start by describing our experimental setup and detail how we selected real-life network topologies together with random graph generators to provide accurate points of comparison between the different methods. In a second part we discuss the experimental results and summarize the evidence collected in relation to the questions we have asked.

6.7.1 Pilot experiments

In this section we describe several pilot experiments that highlight the difficulty of summarizing the performance of the implementation of our algorithm.

1. influence of solver precision
2. influence of the sampling parameters
3. approximation ratios obtained by different algorithms
4. effect of the degree constraint

The influence of solver precision

The first pilot experiment is conducted on synthetic graphs of moderate size ($|V|$ ranges from 150 to 2500, while $|E|$ ranges from 1470 to 39195) that represent two extreme behaviors in terms of degree distributions. The threshold λ was set to a logarithmic factor of the node size of each graph. The first group consists of 4 $G(n, p)$ random graphs, which are mostly homogeneous, while the second group is composed of 4 random graphs obtained via the Barabasí-Albert preferential attachment model [AB02], which are highly heterogeneous as their degree distribution follow a power law.

The influence of the SDP solver precision parameter `eps` on running time, cf. Figure 6.1, and solution quality, cf. Figure 6.2, seems to be puzzling at first glance. It is clear that the overall running time increases with the size of the instance, for example the number of edges $|E(G)|$ in the input graph. However, while this growth is modest in the case of homogeneous graphs, the Barabasí-Albert graphs show a striking increase in running time, which we suppose to be related to the size of the SDP program which necessitates several GBs of memory to be processed by the solver. We observe that by requesting a very coarse approximate solution, i.e. `eps` = 0.5, we abruptly lower the computational burden incurred by the SDP solving step. Other experiments not depicted here show on the contrary that high precision, i.e. `eps` = 0.01, is qualitatively comparable to medium precision: the cases where `eps` = 0.1 or 0.2 shown in Figure 6.1.

One could imagine that the benefits brought by the coarser SDP approximate solution in terms of running time would have to be compensated with discrete solutions of lower quality. This belief is partially contradicted by the fact that while the upper bound computed by the coarse SDP solution is not informative in the case of homogeneous graphs, the graphs sampled from that distribution turn out to share the same solution quality. Heterogeneous graphs display a more straightforward behavior. First, the a posteriori approximation factor is almost 1 as the graphs sampled from the SDP solutions are close to the upper bound, hence close to optimal. Furthermore, it is possible to notice a visible but moderate loss in solution quality as the solver precision goes down.

The effects of random sampling

In this second pilot experiment, we have reused the same groups of instances to provide a point of reference. This time, we have fixed the solver precision to moderate accuracy (`eps` = 0.2) and focused our attention on the randomized rounding procedure.

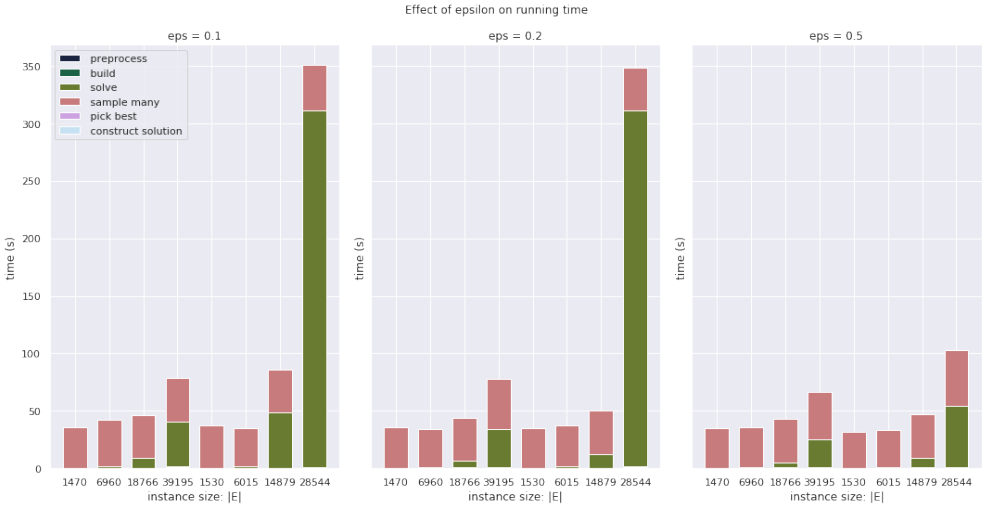

Figure 6.1: Overall running time is highly dependent on `eps`

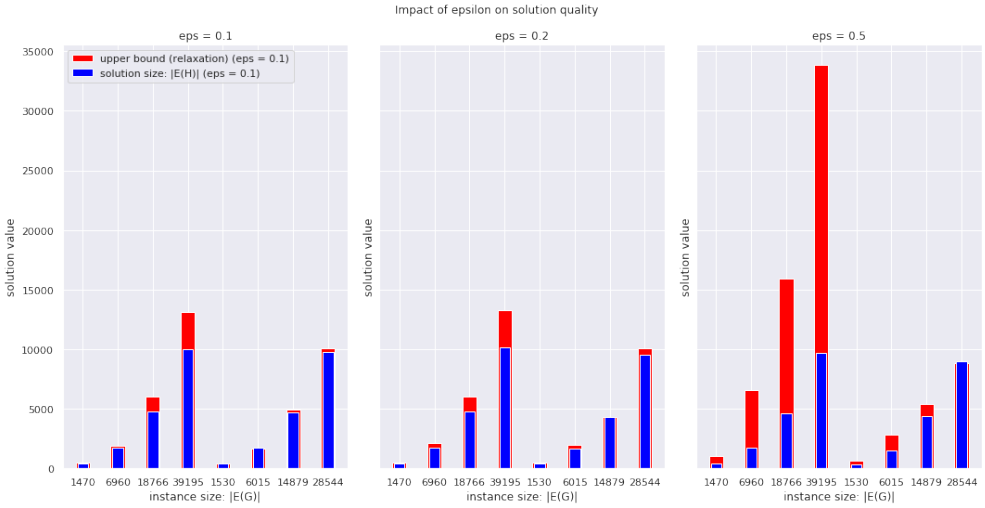

Figure 6.2: Solution quality is moderately affected by `eps`

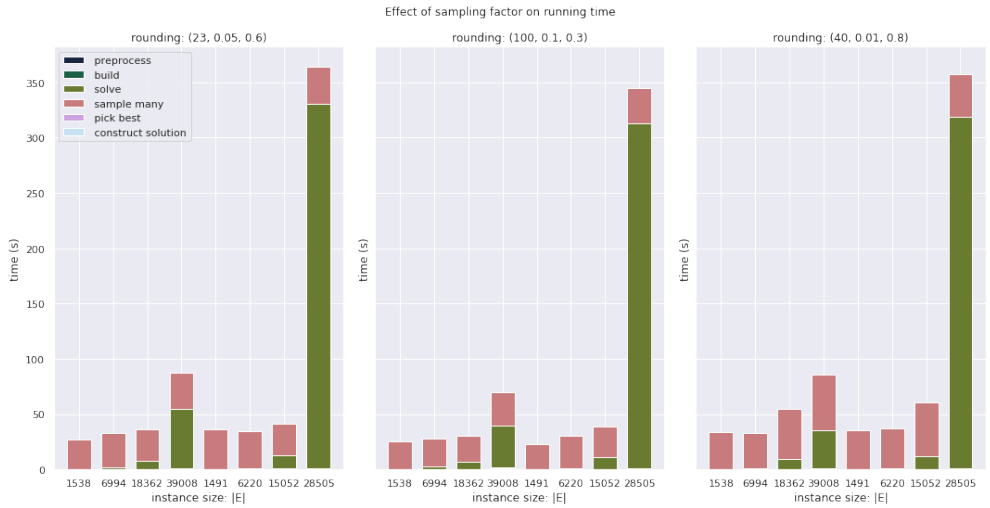

Figure 6.3: Overall running time is weakly affected by the sampling step

While randomized rounding is described in Algorithm 8 as simply drawing each edge ij from a Bernoulli random variable whose parameter is the ratio of the relaxed optimal solution y^*_{ij} and the sampling factor r, in practice the solution obtained from the SDP solver might not be optimal and the sampling factor is almost never equal to $O(\log n)$.

Instead, the implementation uses a variant of dichotomic search to select the appropriate sampling factor. This sub-routine depends on three parameters, `sample_count`, which determines how many graphs should be sampled at every attempt, `bisect_min_improvement` which is the termination condition for the dichotomy, and `bisect_error` which represents the highest proportion of infeasible subgraphs acceptable in an interval. Furthermore, since the interval in which to select the sampling factor starts at $[1, O(\log n)]$, we know that this algorithm can default to the sampling factor value from the proof which guarantees us a feasible solution with high probability.

In Figure 6.3, the variation in running time seems to stem from factors that are not related to sampling and appears to be mostly due to non-deterministic running time in the SDP solving phase. If we focus our attention to the third set of rounding parameters, which features a termination condition that is noticeably harder to reach than in the first two cases, we can observe that the more demanding termination condition translates into slightly longer running times and visibly longer sampling phases.

When looking at solution quality, we return to our previous observation in

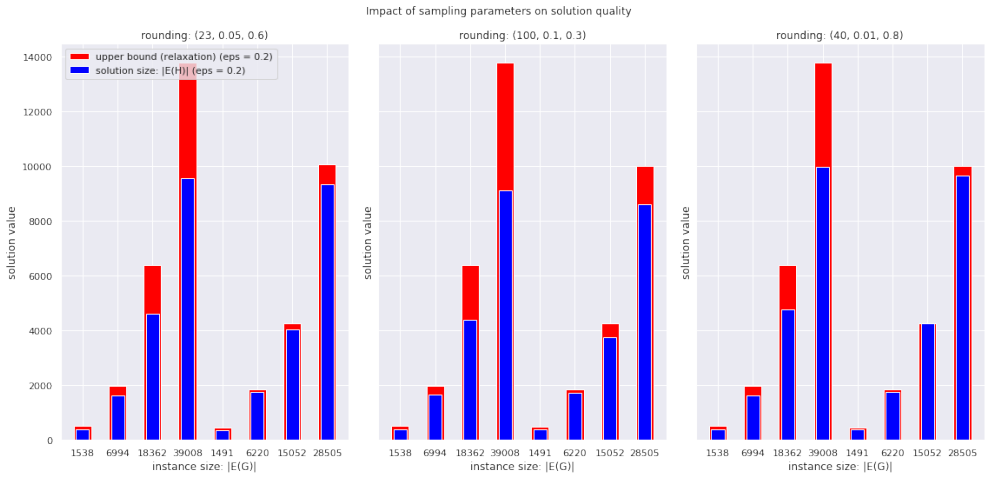

Figure 6.4: Solution quality can be improved by longer sampling phases

the first pilot experiment that the upper bounds obtained for homogeneous graphs are loose. It is interesting to notice that the longer sampling phase in the third set of sampling parameters has a noticeable effect on the quality of both groups of instances. On the contrary, while the second set of sampling parameters contains a higher number of samples per step, the termination condition is not restrictive enough to match the first and third sets. We also notice that the influence of the sampling phase seems to be more important on heterogeneous graphs than homogeneous graphs.

Approximation across algorithms

In this third pilot experiment, we consider a larger set of instances including synthetic instances, as well as some hand-picked instances from the SNAP dataset [LK14] for which the SDP program could fit in memory.

We have implemented a simple variant of Algorithm 8 that uses a feasible uniform solution to the SDP as base distribution for the sampling phase together with direct implementations of algorithms from van Mieghem et al. [Van+11] and Saha et al. [Sah+15].

The top five scatter plots in Figure 6.5 present the running time spent by the five algorithms we have studied to produce solutions. The leftmost plot shows that the running time of the uniform rounding algorithm we have described above is the most competitive. However, this is not surprising because the algorithm simply involves sampling many random graphs and checking if their

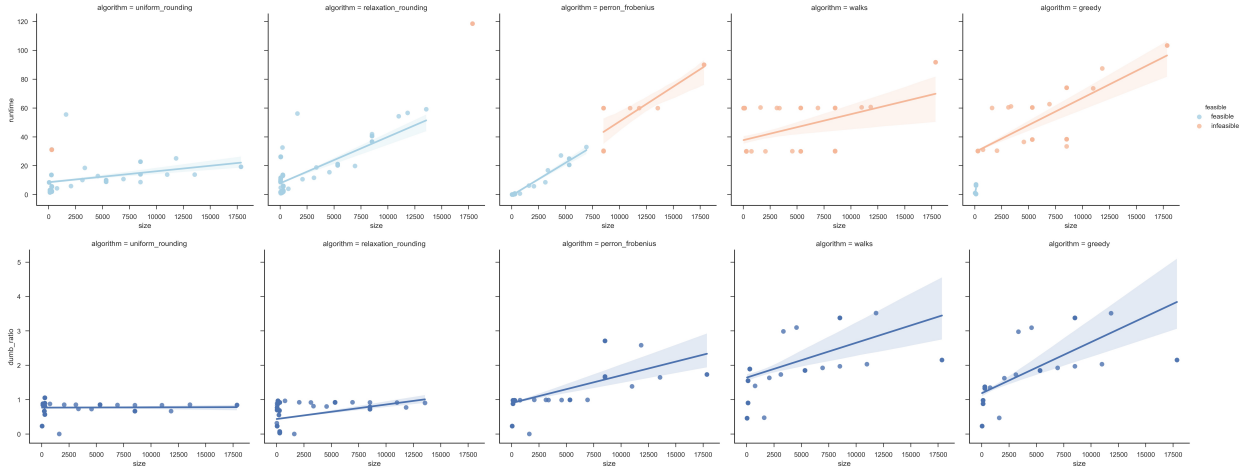

Figure 6.5: Randomized algorithms produce better and faster approximations

spectral radius exceeds or not the threshold. The second plot shows the limits of Algorithm 8 in terms of scalability as the SDP program can become too large to fit in memory. The running time of the relaxation and randomized rounding algorithm seems to follow the same slope as the next plot which depicts the running time of the Perron-Frobenius vector-based algorithm proposed by van Mieghem et al. Unfortunately, this greedy algorithm quickly returns infeasible solutions due to a large number of iterations which becomes prohibitive in cases where too many edges need to be removed from the original graph. The two remaining plots show that both the closed walks-based algorithm proposed by Saha et al. and the naive greedy algorithm which picks the edge that maximizes the decrease in spectral radius are not competitive in terms of running time and almost never manage to produce feasible solutions.

The bottom five scatter plots represent the a posteriori approximation ratio and seem to follow relatively similar slopes as the top ones. These ratios are computed on the basis of available information which includes the best feasible solution known across all algorithms and/or the best upper bound when it is available. The best upper bound defaults to the trivial upper bound which states that a graph on n nodes with spectral radius λ cannot have more than $\lambda/2n$. To represent both feasible and infeasible solutions on the same graphs, we have used the inverse of the approximation ratio which is less than 1 when the solution is feasible and more than 1 when the solution is infeasible. In this experiment, the quality of the solutions produced by Algorithm 8 is indistinguishable from those obtained by the simpler variant that does not

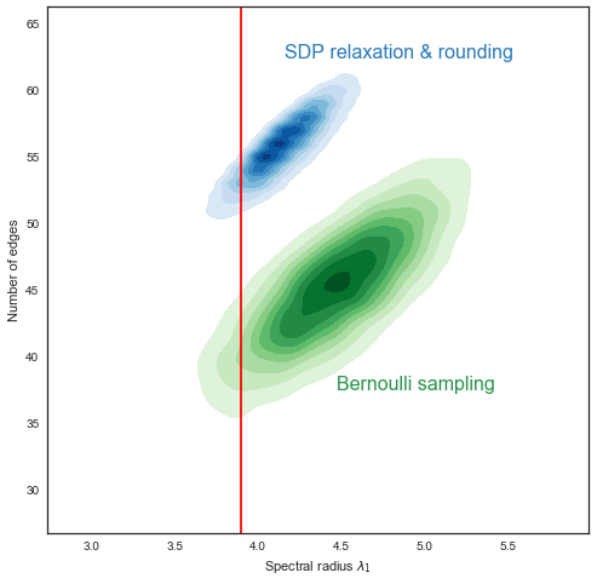

Figure 6.6: Solution quality benefits from a more optimal distribution

solve the SDP. This is due in part to the scale of these plots in the presence of coarsely approximate solutions and infeasible solutions.

We illustrate the difference of solution quality with Figure 6.6 which zooms in onto a single small problem instance where the red line represents the spectral radius threshold and the ovoid shapes are the random graphs sampled from the distributions obtained from the nearly-optimal SDP solution, cf. SDP relaxation & rounding, and the uniform solution, cf. Bernoulli sampling, respectively.

Importance of the degree constraint

The fourth pilot experiment is of smaller scale and illustrates the effect of the degree constraint on random graphs sampled from a near-optimal SDP solution. To see its effect, we design a small problem instance in which the graph is highly heterogeneous and necessitates the maximum degree constraint, i.e. the constraint is not trivial as in the large majority of graphs we have considered.

In Figure 6.7, the histogram on the left represents the distribution of the spectral radius of random graphs sampled without sampling factor, i.e. $r = 1$, from an optimal solution of Problem $\text{SDP}_{\lambda\Delta}$. We can see that a large number of random graphs are feasible since they stand on the left of the red line which stands for the threshold. The right histogram on the other hand shows that

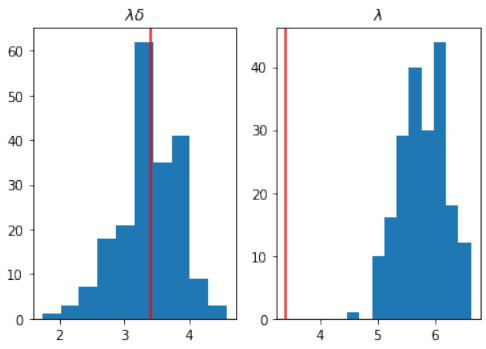

Figure 6.7: High degrees strongly impact feasibility

the spectral radius of the random graphs follows a similar almost-Gaussian distribution but none of which are feasible.

These pilot experiments illustrate that while some high-level characteristics of Algorithm 8 seem to appear from a few initial experiments, the large variations in terms of running time, upper bound quality, solution quality, and scalability show that the behavior of the implementation we have proposed is not straightforward and requires much more care. Indeed, being able to divide by a factor of three the running time of a difficult instance without losing a significant approximation factor is puzzling at best. It also remains to be seen what properties of graphs and degree distributions have the most impact on the SDP phase.

6.7.2 Experimental setup

We have implemented the above algorithms using the Python 3 programming language with extensive use of linear algebra packages like ARPACK and LAPACK exposed via the Scipy and the Numpy libraries. Semidefinite programs are written using the `cvxpy` library which also provides interfaces to a variety of third-party optimization solvers. We mainly use the open-source SuperSCS solver to find solutions to our semidefinite programs and in some cases employ the closed-source MOSEK solver as well as the open-source solver CVXOPT.

Experiments are managed via the Sacred Python library which provides facilities for defining, running, and monitoring arbitrary computational experiments. We have run the experiments on a MacBook Pro equipped with an Intel Core i5-6360U CPU with 2 hyper-threaded cores clocked at 2GHz and 8GB of main memory. Each algorithm is provided with a bounded time budget defined on an instance basis which aims at modeling the need for a quick de-

cision when calculating a counter-measure for an ongoing network epidemic. This follows existing practice when evaluating previously proposed algorithms in different settings such as disease control in Zhang et al. [Zha+15] but we underline that applications in disease control may benefit from computational time budgets counted in days whereas automated network security policies can be expected to react within a few minutes.

We start by exploring generic questions which hold for all algorithms:

A posteriori approximation: given an input graph and a target threshold, which algorithm produces a feasible subgraph with the maximum number of edges?

Parameter sensitivity: how does algorithm performance evolve as λ goes from being close to the original spectral radius to 1, its lowest possible value?

Instance sensitivity: do algorithms perform as well at varying levels of input graph sparsity? Since the spectral radius is bounded below by the average degree of the graph and bounded above by the maximum degree, does input graph regularity have an effect on algorithm performance?

Time/performance profile: how do algorithms benefit from running for a larger amount of time?

Algorithms based on the relaxation and rounding framework have only been briefly studied in the context of optimization problems related to spectral graph theory. One experiment in Zhang et al. [Zha+15] compares the runtime of a semidefinite programming formulation compared to other algorithms. However, we are interested in the practical usage of solutions obtained via randomized rounding methods which involves the consideration of several issues such as the fast computation of a feasible solution to the semidefinite program, a possible lack of feasible samples, as well as the question of setting appropriate scaling parameters. To this end we discuss the following questions:

Approximation tolerance: what is the behavior of random subgraph generators obtained from suboptimal solutions to the semidefinite program?

Feasibility: what is the proportion of feasible subgraphs returned by randomized algorithms? Does this proportion have a link with performance?

Anti-concentration: do random subgraph generators sometimes fail to produce feasible solutions with overwhelming probability? Do the predictions of Le et al. [LLV15] regarding sparse inhomogeneous Erdös-Renyí

graphs apply to the random subgraph generators relevant to the MAX λ-SPECTRAL SUBGRAPH problem?

Instances

Random graph generators are a useful tool when investigating questions relative to graph parameters. For example, Erdös-Renyí graphs can be used as a proxy for graph density by changing the probability for each edge to appear. It is also useful to combine different generators which provide radically different properties e.g. a comparison between random regular graphs and a combination of random power-law tree graphs can give an idea of the effect of regularity when running algorithms on graphs with a constant number of nodes and edges.

On the other hand, raw algorithm performance appears to be assessed in a practical manner when algorithms are confronted to real-life networks exhibiting features that can elude mathematical characterizations. For this reason we consider the Stanford Large Network Dataset Collection [LK14] which comprise not only social network topologies but also communication networks such as snapshots of the global network of Autonomous Systems. While networks of that scale might not be currently managed via software-defined networking, ongoing efforts by major technology companies as well as by Internet providers point at a near future where this setting would be realistic scenario.

Evaluating infeasible solutions

Our setting puts a high penalty on infeasible solutions since their deployment to a software-defined network would not change the propagation of the epidemic. Contrarily to previous works which mainly studied greedy algorithms and considered the spectral radius of the resulting subgraph to be a best effort objective, we want to qualify which algorithms are adequate i.e. algorithms which output actually secure network topologies. In the case of randomized algorithms, we will consider as adequate an algorithm which outputs at least one feasible solution in strictly more than $2/3$ of its runs.

6.8 Pre-registration

We have explored methodological frameworks and practical tools to design adequate experiments to assess the performance of randomized approximation algorithms.

First, we have highlighted the specificity of algorithms as experimental subjects and addressed the shortcomings associated with directly applying methodology related to natural processes. Then, we have discussed the key factors which contribute to the observed performance of algorithm implementations in practice.

Then we have presented several methods that help with selecting parameters as parts of an experimental setup. In particular, we have found that the family of Bayesian optimization methods and in particular Gaussian process regression is a good fit to study randomized approximation algorithms. Indeed by nature of the optimization task and the type of algorithm employed, each run can be noisy from the randomization, might fail if the algorithm does not return a feasible solution, and might take a significant amount of time to complete.

Finally we have provided extensive details of the direct implementation of our proposed algorithm and of the surprisingly many parameters that arise from a straightforward effort. We have detailed available computing resources as well as instance datasets and generators that can be used in the Bayesian optimization process.

This work serves as a pre-registration of experiments to come, which is another way to remove bias from real life applications the scientific method.

Conclusion

The growing threats to network security are being met with countermeasures allowed by an increasing degree of monitoring and control over the network. In this thesis, we have proposed a new countermeasure based on the reconfiguration of the network topology. In particular, we have designed an algorithm which, given the topology as an undirected graph and an information on the characteristic speed of a propagating malware, can select an adequate topology which is guaranteed to remove the epidemic in a short amount of time. This topology modification necessitates to remove existing links in the network but since the extinction of the epidemic is known to be fast, the links can be disabled temporarily.

The advantages of the algorithm we have proposed are numerous, first it can be easily implemented by reusing existing semidefinite programming solvers as well as more specific optimization methods, and readily adapted to make use of parallelism both in the optimization step and in the sampling step by leveraging the capabilities of graphical processing units. Also, our algorithm is flexible and can be easily modified to provide coarse-grained solutions very fast or higher quality solutions if time and computational resources permit. Furthermore, its analysis can be specialized to specific classes of graphs and random graphs which may yield better approximation ratios or additional algorithmic insights.

Theoretically, the maximum spectral subgraph is interesting in itself as a canonical example of optimization in spectral graph theory. Many variants of this problem could be considered, ranging from different objectives such as taking into account the variability of importance among edges, or other spectral properties such as the algebraic connectivity. Additional constraints such as required edges or entire subgraphs could be considered in contexts where critical subnetworks need to be maintained at all times during an epidemic. More generally, other optimization problems based on spectral graph theory might also admit good approximate algorithms using the relaxation and randomized rounding framework.

On the topic of lower bounds, it would be insightful to investigate the computational power of the sum of squares algorithm on the maximum spectral subgraph problem of which our algorithm is the simplest example, i.e. the first

level of the hierarchy of relaxations. A more ambitious goal would be to prove some hardness result assuming some reasonable computational complexity conjecture. Alternatively, one could study the parameterized complexity of the maximum spectral subgraph problem, although it is not entirely clear how to handle potentially irrational eigenvalues as parameters.

Finally, the implementation of a SDN module, which would combine statistical methods to infer the most likely epidemic parameters together with an algorithm which designs an epidemic-repelling topology, could be studied experimentally either in simulators or in toy networks to assess the practicality and impact of this type of countermeasure.

Bibliography

[AB02] Réka Albert and Albert-László Barabási. "Statistical mechanics of complex networks". In: *Reviews of modern physics* 74.1 (2002), p. 47.

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[ACP87] Stefan Arnborg, Derek G Corneil, and Andrzej Proskurowski. "Complexity of finding embeddings in ak-tree". In: *SIAM Journal on Algebraic Discrete Methods* 8.2 (1987), pp. 277–284.

[Ahm+19] AmirMahdi Ahmadinejad, Arun Jambulapati, Amin Saberi, and Aaron Sidford. "Perron-Frobenius Theory in Nearly Linear Time: Positive Eigenvectors, M-matrices, Graph Kernels, and Other Applications". In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2019, pp. 1387–1404.

[AS04] Noga Alon and Joel H Spencer. *The probabilistic method*. John Wiley & Sons, 2004.

[AS98] Sanjeev Arora and Shmuel Safra. "Probabilistic checking of proofs: A new characterization of NP". In: *Journal of the ACM (JACM)* 45.1 (1998), pp. 70–122.

[B+07] James Bennett, Stan Lanning, et al. "The netflix prize". In: *Proceedings of KDD cup and workshop*. Vol. 2007. New York, NY, USA. 2007, p. 35.

[B+12] Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. "Regret analysis of stochastic and nonstochastic multi-armed bandit problems". In: *Foundations and Trends® in Machine Learning* 5.1 (2012), pp. 1–122.

[Ban18] Afonso S Bandeira. "Random Laplacian matrices and convex relaxations". In: *Foundations of Computational Mathematics* 18.2 (2018), pp. 345–379.

[Baz+14] Cristina Bazgan, Morgan Chopin, André Nichterlein, and Florian Sikora. "Parameterized approximability of maximizing the spread of influence in networks". In: *Journal of Discrete Algorithms* 27 (2014), pp. 54–65.

[BB12] James Bergstra and Yoshua Bengio. "Random search for hyper-parameter optimization". In: *Journal of Machine Learning Research* 13.Feb (2012), pp. 281–305.

[BBG] Cristina Bazgan, **Paul Beaujean**, and Éric Gourdin. "Experimental evaluation of an approximation algorithm for the maximum spectral subgraph problem". In preparation.

[BBG18] Cristina Bazgan, **Paul Beaujean**, and Éric Gourdin. "Relaxation and matrix randomized rounding for the maximum spectral subgraph problem". In: *Proceedings of the 12th International Conference on Combinatorial Optimization and Applications*. Best Student Paper. 2018, pp. 108–122.

[BBG20] Cristina Bazgan, Paul Beaujean, and Éric Gourdin. "An approximation algorithm for the maximum spectral subgraph problem". In: *Journal of Combinatorial Optimization* (2020), pp. 1–20.

[Ber+11] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. "Algorithms for hyper-parameter optimization". In: *Advances in neural information processing systems*. 2011, pp. 2546–2554.

[Ber24] Sergei Bernstein. "On a modification of Chebyshev's inequality and of the error formula of Laplace". In: *Ann. Sci. Inst. Sav. Ukraine, Sect. Math* 1.4 (1924), pp. 38–49.

[Bha13] Rajendra Bhatia. *Matrix analysis*. Vol. 169. Springer Science & Business Media, 2013.

[BHL17] Vahid Beiranvand, Warren Hare, and Yves Lucet. "Best practices for comparing optimization algorithms". In: *Optimization and Engineering* 18.4 (2017), pp. 815–848.

[Bis06] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[BN01] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. SIAM, 2001.

[Bol01] Béla Bollobás. *Random graphs*. 73. Cambridge University Press, 2001.

[Bol80] Béla Bollobás. "The distribution of the maximum degree of a random graph". In: *Discrete Mathematics* 32.2 (1980), pp. 201–203.

[BP14] Thomas Bartz-Beielstein and Mike Preuß. "Experimental analysis of optimization algorithms: Tuning and beyond". In: *Theory and Principled Methods for the Design of Metaheuristics*. Springer, 2014, pp. 205–245.

[Bra17] Fred Brauer. "Mathematical epidemiology: Past, present, and future". In: *Infectious Disease Modelling* 2.2 (2017), pp. 113–127.

[BS18] Edouard Bonnet and Florian Sikora. "The PACE 2018 Parameterized Algorithms and Computational Experiments Challenge: The Third Iteration". In: *13th International Symposium on Parameterized and Exact Computation (IPEC 2018)*. 2018.

[BST99] Cristina Bazgan, Miklos Santha, and Zsolt Tuza. "On the approximation of finding a(nother) Hamiltonian cycle in cubic Hamiltonian graphs". In: *Journal of Algorithms* 31.1 (1999), pp. 249–268.

[BT02] Dimitri P Bertsekas and John N Tsitsiklis. *Introduction to probability*. Vol. 1. 2002.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[BYC13a] James Bergstra, Dan Yamins, and David D Cox. "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms". In: *Proceedings of the 12th Python in Science Conference*. Citeseer. 2013.

[BYC13b] James Bergstra, Daniel Yamins, and David Daniel Cox. "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures". In: (2013).

[Cha+08] Deepayan Chakrabarti, Yang Wang, Chenxi Wang, Jurij Leskovec, and Christos Faloutsos. "Epidemic thresholds in real networks". In: *ACM Transactions on Information and System Security (TISSEC)* 10.4 (2008), p. 1.

[Che+14] Clément Chevalier, Julien Bect, David Ginsbourger, Emmanuel Vazquez, Victor Picheny, and Yann Richet. "Fast parallel kriging-based stepwise uncertainty reduction with application to the identification of an excursion set". In: *Technometrics* 56.4 (2014), pp. 455–465.

[Che+18] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. "Neural ordinary differential equations". In: *Advances in neural information processing systems*. 2018, pp. 6571–6583.

[Che+52] Herman Chernoff et al. "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations". In: *The Annals of Mathematical Statistics* 23.4 (1952), pp. 493–507.

[Cho+14] Shihabur Rahman Chowdhury, Md Faizul Bari, Reaz Ahmed, and Raouf Boutaba. "Payless: A low cost network monitoring framework for software defined networks". In: *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE. 2014, pp. 1–9.

[Chu97] Fan RK Chung. *Spectral graph theory*. 92. American Mathematical Soc., 1997.

[CM14] Corinna Cortes and Mehryar Mohri. "Domain adaptation and sample bias correction theory and algorithm for regression". In: *Theoretical Computer Science* 519 (2014), pp. 103–126.

[Coo71] Stephen A Cook. "The complexity of theorem-proving procedures". In: *Proceedings of the third annual ACM symposium on Theory of computing*. ACM. 1971, pp. 151–158.

[CR11] Fan Chung and Mary Radcliffe. "On the spectra of general random graphs". In: *The Electronic Journal of Combinatorics* 18.1 (2011), p. 215.

[Cyg+15] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Vol. 4. 8. Springer, 2015.

[Dal+12] Gerard V Dallal et al. *The little handbook of statistical practice*. 2012. URL: http://www.jerrydallal.com/LHSP/LHSP.htm (visited on 09/30/2019).

[Del+17] Holger Dell, Christian Komusiewicz, Nimrod Talmon, and Mathias Weller. "The PACE 2017 Parameterized Algorithms and Computational Experiments Challenge: The Second Iteration". In: *12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*. 2017.

[Dem+08] Erik D Demaine, Uriel Feige, MohammadTaghi Hajiaghayi, and Mohammad R Salavatipour. "Combination can be hard: Approximability of the unique coverage problem". In: *SIAM Journal on Computing* 38.4 (2008), pp. 1464–1483.

[Den+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[Den+15] Juan Deng, Hongxin Hu, Hongda Li, Zhizhong Pan, Kuang-Ching Wang, Gail-Joon Ahn, Jun Bi, and Younghee Park. "VNGuard: An NFV/SDN combination framework for provisioning and managing virtual firewalls". In: *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. IEEE. 2015, pp. 107–114.

[DH00] Odo Diekmann and Johan Andre Peter Heesterbeek. *Mathematical epidemiology of infectious diseases: model building, analysis and interpretation*. Vol. 5. John Wiley & Sons, 2000.

[DM02] Elizabeth D Dolan and Jorge J Moré. "Benchmarking optimization software with performance profiles". In: *Mathematical programming* 91.2 (2002), pp. 201–213.

[Duv14] David Duvenaud. "Automatic model construction with Gaussian processes". PhD thesis. University of Cambridge, 2014.

[Edm65] Jack Edmonds. "Paths, trees, and flowers". In: *Canadian Journal of mathematics* 17.3 (1965), pp. 449–467.

[Eps73] H Epstein. "Remarks on two theorems of E. Lieb". In: *Communications in Mathematical Physics* 31.4 (1973), pp. 317–325.

[ER60] Paul Erdos and Alfréd Rényi. "On the evolution of random graphs". In: *Publ. Math. Inst. Hung. Acad. Sci* 5.1 (1960), pp. 17–60.

[FHL08] Uriel Feige, MohammadTaghi Hajiaghayi, and James R Lee. "Improved approximation algorithms for minimum weight vertex separators". In: *SIAM Journal on Computing* 38.2 (2008), pp. 629–657.

[Fie73] Miroslav Fiedler. "Algebraic connectivity of graphs". In: *Czechoslovak mathematical journal* 23.2 (1973), pp. 298–305.

[FK96] Ehud Friedgut and Gil Kalai. "Every monotone graph property has a sharp threshold". In: *Proceedings of the American mathematical Society* 124.10 (1996), pp. 2993–3002.

[FPK01] Uriel Feige, David Peleg, and Guy Kortsarz. "The dense k-subgraph problem". In: *Algorithmica* 29.3 (2001), pp. 410–421.

[GB06] Arpita Ghosh and Stephen Boyd. "Growing well-connected graphs". In: *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE. 2006, pp. 6605–6611.

[Gil59] Edgar N Gilbert. "Random graphs". In: *The Annals of Mathematical Statistics* 30.4 (1959), pp. 1141–1144.

[Gil77] John Gill. "Computational complexity of probabilistic Turing machines". In: *SIAM Journal on Computing* 6.4 (1977), pp. 675–695.

[GJ] Michael R Garey and David S Johnson. *Computers and intractability*. Vol. 29.

[GM12] Bernd Gärtner and Jiri Matousek. *Approximation algorithms and semidefinite programming*. Springer Science & Business Media, 2012.

[GMM71] Narendra S Goel, Samaresh C Maitra, and Elliott W Montroll. "On the Volterra and other nonlinear models of interacting populations". In: *Reviews of modern physics* 43.2 (1971), p. 231.

[GMT05] Ayalvadi Ganesh, Laurent Massoulié, and Don Towsley. "The effect of network topology on the spread of epidemics". In: *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*. Vol. 2. IEEE. 2005, pp. 1455–1466.

[GW95] Michel X Goemans and David P Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". In: *Journal of the ACM (JACM)* 42.6 (1995), pp. 1115–1145.

[Han+16a] Wonkyu Han, Ziming Zhao, Adam Doupé, and Gail-Joon Ahn. "Honeymix: Toward sdn-based intelligent honeynet". In: *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. ACM. 2016, pp. 1–6.

[Han+16b] Nikolaus Hansen, Anne Auger, Olaf Mersmann, Tea Tusar, and Dimo Brockhoff. "COCO: A platform for comparing continuous optimizers in a black-box setting". In: *arXiv preprint arXiv:1603.08785* (2016).

[Han17] Ramon van Handel. "Structured random matrices". In: *Convexity and Concentration*. Springer, 2017, pp. 107–156.

[HDS74] Morris W Hirsch, Robert L Devaney, and Stephen Smale. *Differential equations, dynamical systems, and linear algebra*. Vol. 60. Academic press, 1974.

[Hea+18] Tim Head et al. *scikit-optimize/scikit-optimize: v0.5.2*. Mar. 2018. DOI: 10.5281/zenodo.1207017. URL: https://doi.org/10.5281/zenodo.1207017.

[Hea67] John Z Hearon. "Partially Isometric Matrices". In: *Journal of Research of the National Bureau of Standards -B. Mathematics and Mathematical Physics* 71B.4 (1967), pp. 225–228.

[Hel00] Keld Helsgaun. "An effective implementation of the Lin–Kernighan traveling salesman heuristic". In: *European Journal of Operational Research* 126.1 (2000), pp. 106–130.

[HHL11] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. "Sequential model-based optimization for general algorithm configuration". In: *International Conference on Learning and Intelligent Optimization*. Springer. 2011, pp. 507–523.

[Hol12] Richard A Holmgren. *A first course in discrete dynamical systems*. Springer Science & Business Media, 2012.

[HR00] Christoph Helmberg and Franz Rendl. "A spectral bundle method for semidefinite programming". In: *SIAM Journal on Optimization* 10.3 (2000), pp. 673–696.

[HS65] Juris Hartmanis and Richard E Stearns. "On the computational complexity of algorithms". In: *Transactions of the American Mathematical Society* 117 (1965), pp. 285–306.

[Hut18] M Hutson. "Artificial intelligence faces reproducibility crisis." In: *Science (New York, NY)* 359.6377 (2018), p. 725.

[JKB06] Alexander I J. Forrester, Andy J Keane, and Neil W Bressloff. "Design and analysis of" Noisy" computer experiments". In: *AIAA journal* 44.10 (2006), pp. 2331–2339.

[Joh02] David S Johnson. "A theoretician's guide to the experimental analysis of algorithms". In: (2002).

[Jou+17] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. "In-datacenter performance analysis of a tensor processing unit". In: *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. IEEE. 2017, pp. 1–12.

[Kar72] Richard M Karp. "Reducibility among combinatorial problems". In: *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[Kha80] Leonid G Khachiyan. "Polynomial algorithms in linear programming". In: *USSR Computational Mathematics and Mathematical Physics* 20.1 (1980), pp. 53–72.

[Kho02] Subhash Khot. "On the power of unique 2-prover 1-round games". In: *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. ACM. 2002, pp. 767–775.

[Kle09] Jack PC Kleijnen. "Kriging metamodeling in simulation: A review". In: *European journal of operational research* 192.3 (2009), pp. 707–716.

[KM27] William Ogilvy Kermack and Anderson G McKendrick. "A contribution to the mathematical theory of epidemics". In: *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character* 115.772 (1927), pp. 700–721.

[Kol+10] Alexandra Kolla, Yury Makarychev, Amin Saberi, and Shang-Hua Teng. "Subgraph sparsification and nearly optimal ultrasparsifiers". In: *Proceedings of the forty-second ACM symposium on Theory of computing*. ACM. 2010, pp. 57–66.

[KPB14] Panos Kampanakis, Harry Perros, and Tsegereda Beyene. "SDN-based solutions for moving target defense network protection". In: *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. IEEE. 2014, pp. 1–6.

[KS03] Michael Krivelevich and Benny Sudakov. "The largest eigenvalue of sparse random graphs". In: *Combinatorics, Probability and Computing* 12.01 (2003), pp. 61–72.

[KV16] E. de Klerk and F. Vallentin. "On the Turing model complexity of interior point methods for semidefinite programming". In: *SIAM J. Optim.* 26.3 (2016), pp. 1944–1961. DOI: 10.1137/15M103114X.

[Las01] Jean B Lasserre. "Global optimization with polynomials and the problem of moments". In: *SIAM Journal on optimization* 11.3 (2001), pp. 796–817.

[Led01] Michel Ledoux. *The concentration of measure phenomenon*. 89. American Mathematical Soc., 2001.

[Li+17] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. "Hyperband: A novel bandit-based approach to hyperparameter optimization". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 6765–6816.

[Lie73] Elliott H Lieb. "Convex trace functions and the Wigner-Yanase-Dyson conjecture". In: *Les rencontres physiciens-mathématiciens de Strasbourg-RCP25* 19 (1973), pp. 0–35.

[Lin+14] Xihong Lin, Christian Genest, David L Banks, Geert Molenberghs, David W Scott, and Jane-Ling Wang. *Past, present, and future of statistical science*. CRC Press, 2014.

[LK14] Jure Leskovec and Andrej Krevl. *SNAP Datasets: Stanford Large Network Dataset Collection*. `http://snap.stanford.edu/data`. June 2014.

[LL01] Elliott H Lieb and Michael Loss. *Analysis*. Vol. 14. 2001.

[LLV15] Can M Le, Elizaveta Levina, and Roman Vershynin. "Sparse random graphs: regularization and concentration of the Laplacian". In: *arXiv preprint arXiv:1502.03049* (2015).

[LLV17] Can M Le, Elizaveta Levina, and Roman Vershynin. "Concentration and regularization of random graphs". In: *Random Structures & Algorithms* 51.3 (2017), pp. 538–561.

[Lóp+16] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. "The irace package: Iterated racing for automatic algorithm configuration". In: *Operations Research Perspectives* 3 (2016), pp. 43–58.

[Lot10] Alfred J Lotka. "Contribution to the theory of periodic reactions". In: *The Journal of Physical Chemistry* 14.3 (1910), pp. 271–274.

[LR05] Monique Laurent and Franz Rendl. "Semidefinite programming and integer programming". In: *Handbooks in Operations Research and Management Science* 12 (2005), pp. 393–514.

[LSY98] Richard B Lehoucq, Danny C Sorensen, and Chao Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. Vol. 6. Siam, 1998.

[Lui+15] Marcelo Caggiani Luizelli, Leonardo Richter Bays, Luciana Salete Buriol, Marinho Pilla Barcellos, and Luciano Paschoal Gaspary. "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions". In: *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE. 2015, pp. 98–106.

[Mar08] Dániel Marx. "Parameterized complexity and approximation algorithms". In: *The Computer Journal* 51.1 (2008), pp. 60–78.

[Mar14] Ruben Martinez-Cantin. "BayesOpt: A Bayesian Optimization Library for Nonlinear Optimization, Experimental Design and Bandits". In: *Journal of Machine Learning Research* 15 (2014), pp. 3915–3919.

[McG01] Catherine C McGeoch. "Experimental analysis of algorithms". In: *Notices of the AMS* 48.3 (2001), pp. 304–311.

[MKK11] Syed Akbar Mehdi, Junaid Khalid, and Syed Ali Khayam. "Revisiting traffic anomaly detection using software defined networking". In: *Proceedings of the International Workshop on Recent Advances in Intrusion Detection (RAID 2011)*. Springer. 2011, pp. 161–180.

[MN05] James W Mickens and Brian D Noble. "Modeling epidemic spreading in mobile environments". In: *Proceedings of the 4th ACM workshop on Wireless security*. ACM. 2005, pp. 77–86.

[Mos08] Damon Mosk-Aoyama. "Maximum algebraic connectivity augmentation is NP-hard". In: *Operations Research Letters* 36.6 (2008), pp. 677–679.

[MP03] Junshui Ma and Simon Perkins. "Time-series novelty detection using one-class support vector machines". In: *Proceedings of the International Joint Conference on Neural Networks, 2003.* Vol. 3. IEEE. 2003, pp. 1741–1745.

[MR10] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010.

[MS07] Bimal Kumar Mishra and Dinesh Kumar Saini. "SEIRS epidemic model with delay for transmission of malicious objects in computer network". In: *Applied Mathematics and Computation* 188.2 (2007), pp. 1476–1482.

[NBG08] John Nickolls, Ian Buck, and Michael Garland. "Scalable parallel programming". In: *2008 IEEE Hot Chips 20 Symposium (HCS)*. IEEE. 2008, pp. 40–53.

[Nie11] Jiawang Nie. "Polynomial matrix inequality and semidefinite representation". In: *Mathematics of Operations Research* 36.3 (2011), pp. 398–415.

[NPP16] Cameron Nowzari, Victor M Preciado, and George J Pappas. "Analysis and control of epidemics: A survey of spreading processes on complex networks". In: *IEEE Control Systems* 36.1 (2016), pp. 26–46.

[ODo+16] Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. "Conic optimization via operator splitting and homogeneous self-dual embedding". In: *Journal of Optimization Theory and Applications* 169.3 (2016), pp. 1042–1068.

[Orl+10] Sebastian Orlowski, Roland Wessäly, Michal Pióro, and Artur Tomaszewski. "SNDlib 1.0—Survivable network design library". In: *Networks* 55.3 (2010), pp. 276–286.

[Par03] Pablo A Parrilo. "Semidefinite programming relaxations for semialgebraic problems". In: *Mathematical programming* 96.2 (2003), pp. 293–320.

[PC99] Victor Y. Pan and Zhao Q. Chen. "The complexity of the matrix eigenproblem". In: *Proceedings of the ACM Symposium on Theory of Computing (STOC 1999)*. 1999, pp. 507–516.

[Per07] Oskar Perron. "Zur theorie der matrices". In: *Mathematische Annalen* 64.2 (1907), pp. 248–263.

[Pra+11] B Aditya Prakash, Deepayan Chakrabarti, Michalis Faloutsos, Nicholas Valler, and Christos Faloutsos. "Threshold conditions for arbitrary cascade models on arbitrary networks". In: *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE. 2011, pp. 537–546.

[PY91] Christos H Papadimitriou and Mihalis Yannakakis. "Optimization, approximation, and complexity classes". In: *Journal of computer and system sciences* 43.3 (1991), pp. 425–440.

[R+12] Olivier Roustant, David Ginsbourger, Yves Deville, et al. "DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization". In: *Journal of Statistical Software* 51.i01 (2012).

[Rag08] Prasad Raghavendra. "Optimal algorithms and inapproximability results for every CSP?" In: *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM. 2008, pp. 245–254.

[Ras04] Carl Edward Rasmussen. "Gaussian processes in machine learning". In: *Advanced lectures on machine learning*. Springer, 2004, pp. 63–71.

[Rei91] Gerhard Reinelt. "TSPLIB—A traveling salesman problem library". In: *ORSA journal on computing* 3.4 (1991), pp. 376–384.

[RMN09] Rajat Raina, Anand Madhavan, and Andrew Y Ng. "Large-scale deep unsupervised learning using graphics processors". In: *Proceedings of the 26th annual international conference on machine learning*. ACM. 2009, pp. 873–880.

[RS04] Neil Robertson and Paul D Seymour. "Graph minors. XX. Wagner's conjecture". In: *Journal of Combinatorial Theory, Series B* 92.2 (2004), pp. 325–357.

[RT87] Prabhakar Raghavan and Clark D Tompson. "Randomized rounding: a technique for provably good algorithms and algorithmic proofs". In: *Combinatorica* 7.4 (1987), pp. 365–374.

[Rub74] Donald B Rubin. "Estimating causal effects of treatments in randomized and nonrandomized studies." In: *Journal of educational Psychology* 66.5 (1974), p. 688.

[Rud91] Walter Rudin. *Functional Analysis*. McGrawHill, 1991.

[Ruh17] Navid Azizan Ruhi. "Inferring Epidemic Parameters in Networked SIS Epidemics". California Institute of Technology ACM 158 Project. 2017.

[Rus+18] Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. "A tutorial on Thompson sampling". In: *Foundations and Trends® in Machine Learning* 11.1 (2018), pp. 1–96.

[Sac+89] Jerome Sacks, William J Welch, Toby J Mitchell, and Henry P Wynn. "Design and analysis of computer experiments". In: *Statistical science* (1989), pp. 409–423.

[Sah+15] Sudip Saha, Abhijin Adiga, B Aditya Prakash, and Anil Kumar S Vullikanti. "Approximation algorithms for reducing the spectral radius to control epidemic spread". In: *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM. 2015, pp. 568–576.

[Sch03] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Vol. 24. Springer Science & Business Media, 2003.

[SG12] Seungwon Shin and Guofei Gu. "CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?)" In: *Proceedings of the IEEE International Conference on Network Protocols (ICNP 2012)*. IEEE. 2012, pp. 1–6.

[Sha+16] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. "Taking the human out of the loop: A review of bayesian optimization". In: *Proceedings of the IEEE* 104.1 (2016), pp. 148–175.

[Sla+08] Gordon Slade et al. "Probabilistic models of critical phenomena". In: *The Princeton companion to mathematics*. 2008, pp. 343–346.

[SMP17] P. Sopasakis, K. Menounou, and P. Patrinos. *SuperSCS: A fast and accurate conic optimization solver*. https://kul-forbes.github.io/scs/. Apr. 2017.

[Ste10] Dragan Stevanović. "Resolution of AutoGraphiX conjectures relating the index and matching number of graphs". In: *Linear Algebra and Its Applications* 8.433 (2010), pp. 1674–1677.

[SWF16] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. "Probabilistic programming in Python using PyMC3". In: *PeerJ Computer Science* 2 (2016), e55.

[SWG14] Amar Shah, Andrew Wilson, and Zoubin Ghahramani. "Student-t processes as alternatives to Gaussian processes". In: *Artificial Intelligence and Statistics*. 2014, pp. 877–885.

[Tal+96] Michel Talagrand et al. "A new look at independence". In: *The Annals of probability* 24.1 (1996), pp. 1–34.

[TP16] Andreas Themelis and Panagiotis Patrinos. "SuperMann: a superlinearly convergent algorithm for finding fixed points of nonexpansive operators". In: *arXiv preprint arXiv:1609.06955* (2016).

[Tro+15] Joel A Tropp et al. "An introduction to matrix concentration inequalities". In: *Foundations and Trends® in Machine Learning* 8.1-2 (2015), pp. 1–230.

[Tro12] Joel A Tropp. "User-friendly tail bounds for sums of random matrices". In: *Foundations of computational mathematics* 12.4 (2012), pp. 389–434.

[TT17] Michael Tait and Josh Tobin. "Three conjectures in extremal spectral graph theory". In: *Journal of Combinatorial Theory, Series B* 126 (2017), pp. 137–161.

[Van+11] Piet Van Mieghem, Dragan Stevanović, Fernando Kuipers, Cong Li, Ruud Van De Bovenkamp, Daijie Liu, and Huijuan Wang. "Decreasing the spectral radius of a graph by link removals". In: *Physical Review E* 84.1 (2011), p. 016101.

[Van10] Piet Van Mieghem. *Graph spectra for complex networks*. Cambridge University Press, 2010.

[Var10] Richard S Varga. *Geršgorin and his circles*. Vol. 36. Springer Science & Business Media, 2010.

[VB96] Lieven Vandenberghe and Stephen Boyd. "Semidefinite programming". In: *SIAM review* 38.1 (1996), pp. 49–95.

[Vir+19] Pauli Virtanen et al. "SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python". In: *arXiv e-prints*, arXiv:1907.10121 (July 2019), arXiv:1907.10121. arXiv: 1907.10121 [cs.MS].

[VK07] Wenceslas Fernandez de la Vega and Claire Kenyon-Mathieu. "Linear programming relaxations of maxcut". In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 2007, pp. 53–61.

[VOK09] Piet Van Mieghem, Jasmina Omic, and Robert Kooij. "Virus spread in networks". In: *IEEE/ACM Transactions on Networking (TON)* 17.1 (2009), pp. 1–14.

[Vol28] Vito Volterra. "Variations and fluctuations of the number of individuals in animal species living together". In: *ICES Journal of Marine Science* 3.1 (1928), pp. 3–51.

[Wan+03] Yang Wang, Deepayan Chakrabarti, Chenxi Wang, and Christos Faloutsos. "Epidemic spreading in real networks: An eigenvalue viewpoint". In: *Reliable Distributed Systems, 2003. Proceedings. 22nd International Symposium on*. IEEE. 2003, pp. 25–34.

[Wan+18] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. 2018, pp. 353–355.

[WG75] Henry William Watson and Francis Galton. "On the probability of the extinction of families". In: *The Journal of the Anthropological Institute of Great Britain and Ireland* 4 (1875), pp. 138–144.

[WNS12] Guohui Wang, TS Ng, and Anees Shaikh. "Programming your Network at Run-Time for Big Data Applications". In: *Proceedings of the Workshop on Hot Topics in Software Defined Networks (HotSDN 2012)*. ACM. 2012, pp. 103–108.

[WS11] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.

[Yan+13] Lu-Xing Yang, Xiaofan Yang, Jiming Liu, Qingyi Zhu, and Chenquan Gan. "Epidemics of computer viruses: A complex-network approach". In: *Applied Mathematics and Computation* 219.16 (2013), pp. 8705–8717.

[Zha+15] Yao Zhang, Abhijin Adiga, Anil Vullikanti, and B Aditya Prakash. "Controlling Propagation at Group Scale on Networks". In: *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE. 2015, pp. 619–628.

RÉSUMÉ

Les théories mathématiques en épidémiologie ont adopté l'usage de réseaux d'interaction pour modéliser la propagation d'une épidémie au sein d'une population de nœuds qui sont en contact s'ils sont reliés par une arête. Bien que des avancées majeures aient été réalisées pour concevoir des contre-mesures efficaces qui agissent directement sur les maladies, peu d'études en comparaison ont été effectuées pour tenter de modifier le réseau d'interaction lui-même.
Cette thèse étudie la possibilité de trouver une modification optimale d'un réseau de manière à stopper une épidémie qui s'y propagerait. Ce problème d'optimisation étant difficile du point de vue de la théorie de la complexité, nous proposons un algorithme probabiliste d'approximation qui est garanti de produire une modification stoppant l'épidémie en un temps limité. De plus, nous montrons que l'analyse du ratio d'approximation de cet algorithme est la meilleure possible pour un large ensemble d'instances.
Pour mesurer l'utilité pratique d'un tel algorithme, nous procédons à une analyse critique des méthodologies actuelles en évaluation expérimentale des algorithmes. En réponse, nous proposons une nouvelle méthodologie pour étudier des implantations d'algorithmes produisant des solutions potentiellement inexactes, sous-optimales et dont le comportement dépend de paramètres.

MOTS CLÉS

Optimisation combinatoire · Sécurité des réseaux · Matrices aléatoires · Algorithmes d'approximation · Théorie spectrale des graphes

ABSTRACT

The modern mathematical study of epidemics has adopted the concept of contact networks to model a disease spreading among nodes who may interact with each other along edges. While much progress has been made in designing effective countermeasures against epidemics by acting upon the disease, fewer studies have explored the use of modifying the contact network itself.
This thesis explores the possibility of finding an optimal modification of a network to stop an epidemic spreading over it. Because this optimization problem is computationally hard to solve, we design a randomized approximation algorithm by combining semidefinite programming together with matrix concentration inequalities which is guaranteed to return a network modification that stops the epidemic in a short amount of time. Furthermore, we give evidence that the analysis of this algorithm is tight in a large regime.
To understand the practical applicability of this algorithm, we analyze current practices in the experimental evaluation of algorithms and propose a new methodology to assess algorithms that may fail, may return approximate solutions, and may change behavior based on hyperparameters.

KEYWORDS

Combinatorial optimization · Network security · Random matrices · Approximation algorithms · Spectral graph theory