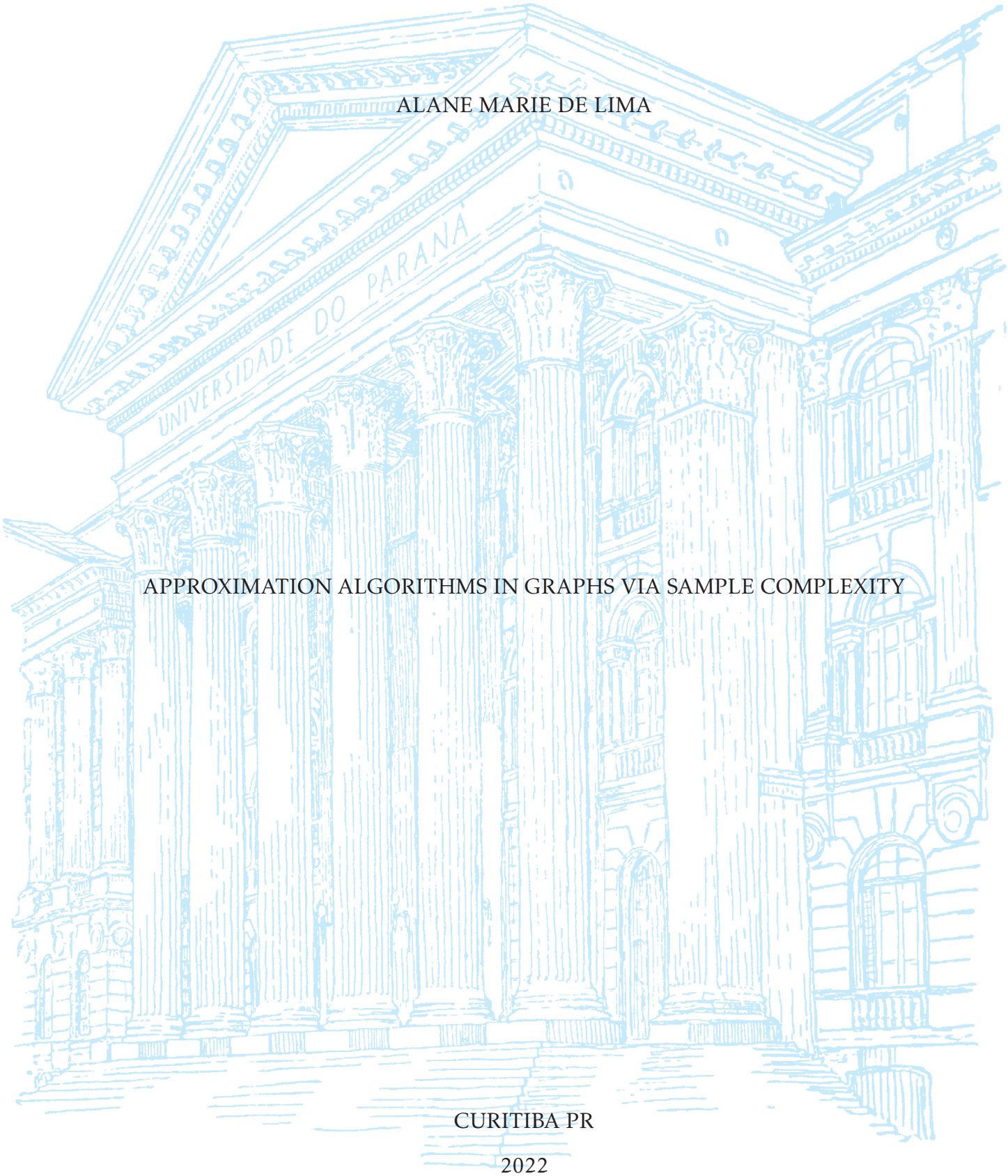
UNIVERSIDADE FEDERAL DO PARANÁ

ALANE MARIE DE LIMA



APPROXIMATION ALGORITHMS IN GRAPHS VIA SAMPLE COMPLEXITY

CURITIBA PR

2022

ALANE MARIE DE LIMA

APPROXIMATION ALGORITHMS IN GRAPHS VIA SAMPLE COMPLEXITY

Tese apresentada como requisito parcial à obtenção do grau de Doutor em Ciência da Computação no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: André L. Vignatti.

Coorientador: Murilo V. G. da Silva.

CURITIBA PR

2022

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA CIÊNCIA E TECNOLOGIA

Lima, Alane Marie de.
Approximation algorithms in graphs via sample complexity. / Alane Marie de Lima. – Curitiba, 2022.
1 recurso on-line : PDF.

Tese (Doutorado) – Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática.
Orientador: Prof. Dr. André L. Vignatti.
Coorientador: Prof. Dr. Murilo V. G. da Silva.

1. Ciência da computação. 2. Informática. 3. Teoria dos grafos. 4. Algorítmos. I. Vignatti, André L. II. Silva, Murilo V. G. da. III. Universidade Federal do Paraná. Programa de Pós-Graduação em Informática. IV. Título.

Bibliotecário: Nilson Carlos Vieira Júnior CRB-9/1797

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **ALANE MARIE DE LIMA** intitulada: **Approximation Algorithms in Graphs via Sample Complexity**, sob orientação do Prof. Dr. ANDRÉ LUÍS VIGNATTI, que após terem inquirido a aluna e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de doutora está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 28 de Outubro de 2022.

Assinatura Eletrônica
31/10/2022 12:00:02.0
ANDRÉ LUÍS VIGNATTI
Presidente da Banca Examinadora

Assinatura Eletrônica
01/11/2022 17:15:10.0
GUILHERME ALEX DERENIEVICZ
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
03/11/2022 19:20:45.0
MURILO VICENTE GONÇALVES DA SILVA
Coorientador(a)

Assinatura Eletrônica
01/11/2022 14:14:41.0
LEANDRO MIRANDA ZATESKO
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

Assinatura Eletrônica
31/10/2022 09:24:04.0
EDUARDO SANY LABER
Avaliador Externo (PONTIFICIA UNIVERSIDADE CATÓLICA - PUC/RIO)

To my beloved family and to all the ones that fight for science and education.

ACKNOWLEDGEMENTS

I could not forget to thank those who stretched out a hand to help me climb each step in the course of this journey, and that made me evolve not only professionally but also personally.

- Firstly, to God, the alpha and the omega;
- to my parents, Luiz and Raquel, and to my brother, Pedro Paulo, for all the affection, support, and unconditional love. Thank you for your understanding, for always encourage me to study, and for always believe in me. This achievement is for you!
- To my grandmother, Inocência, for the prayers;
- to my relatives of families Wolochatyi and Lima, for the support;
- to my advisors, André Vignatti, PhD and Murilo da Silva, PhD, for sparing no effort and dedication in advising me, and for trusting in my potential. I also thank you for the friendship that has grown in the past years. You are role-models that I want to follow in my prospective teaching and researching career!
- To the Federal University of Paraná (UFPR), for accepting me as contributor in the advance of science in our country;
- to my colleagues and friends from the research labs of UFPR, specially the Teoria da Computação, Otimização e Combinatória (TEORIA) research group, for the moments in the University Restaurant, for the coffee and chatting, for the "rolês", and for all the shared experiences, pains and joys;
- to the friends that I made in Guarapuava and Curitiba, for all the shared livelihood;
- to the invited professors of the qualification and the final examination board, Eduardo Laber, PhD, Leandro Zatesko, PhD, Guilherme Derenievicz, PhD, and David Menotti, PhD, for the words of support, and for the careful reading and valuable suggestions for the improvement of the quality of the work;
- to all my teachers, from school to graduate program, for all the lessons learned;
- to the Coordination for the Improvement of Higher Education Personnel (CAPES) for the scholarship (Proc. 88882.461738/2019-01), and for the National Council for Scientific and Technological Development (CNPq) for providing financial support at some conferences I have attended (Proc. 428941/2016-8 and 420079/2021-1);
- to the Center for Mathematical Modelling, for funding my attendance at XV Escuela de Verano en Matemáticas Discretas (Valparaiso, Chile), and to the University of Bonn, for funding my attendance at the Algorithmic Data Analysis School (Bonn, Germany).

"Although this may seen a paradox, all exact science is dominated by the idea of approximation."
— *Bertrand Russell*

RESUMO

Grafos de grande porte advém de diversos contextos em fenômenos naturais e sociais. Contudo, algoritmos que escalam em complexidade de tempo cúbica e até mesmo quadrática, quando executados nesses grafos, podem ser computacionalmente custosos na prática. Neste trabalho, apresentamos esquemas de aproximação aleatorizados de tempo próximo de linear para dois problemas em grafos onde não se tem algoritmo conhecido de tempo *estritamente subcúbico* no número de vértices. Adicionalmente, para um terceiro problema, também apresentamos um esquema de aproximação aleatorizado de tempo próximo de linear, mas para o caso em que o grafo de entrada tem diâmetro logarítmico. Os algoritmos propostos foram projetados utilizando análise de complexidade de amostra, teoria de dimensão Vapnik–Chervonenkis e médias de Rademacher. Seja $G = (V, E)$ um grafo com $n = |V|$ e $m = |E|$. O primeiro problema tratado neste trabalho é o de *centralidade de percolação* em um grafo G direcionado com pesos reais não-negativos. Tal medida quantifica a *importância* de um vértice em um grafo que está passando por um processo de contágio. Neste trabalho, apresentamos um algorimo de aproximação de tempo $\mathcal{O}(m \log n \log \mathrm{diam}_V(G))$ para estimar a centralidade de percolação de cada vértice de V, onde $\mathrm{diam}_V(G)$ é número máximo de vértices em um caminho mínimo em G. O segundo problema é uma versão relaxada do problema de computar *todos os caminhos mínimos entre todos os pares de vértices* (APSP). Nesta versão, iremos computar todos os caminhos de G que tenham "centralidade" pelo menos ε, para $0 < \varepsilon < 1$ constante, medida esta que está relacionada a uma generalização da centralidade de intermediação. O algoritmo proposto executa em tempo $\mathcal{O}(m \log n + (\mathrm{diam}_V(G))^2)$. O terceiro problema é o de *coeficiente de agrupamento local* de cada vértice de um grafo G. Neste trabalho propomos um algoritmo para este problema que roda em tempo $\mathcal{O}(\Delta \lg \Delta + m)$, onde Δ é o grau máximo de um vértice de um grafo. Finalmente, neste trabalho também apresentamos algoritmos de aproximação para os problemas do *conjunto dominante mínimo* (MDS) e da *cobertura por vértices mínima* (MVC). Em ambos aplicamos técnicas probabilísticas, contudo não utilizamos análise de complexidade de amostra nestes casos. Para estes problemas, obtivemos limitantes mais justos lidando com o problema diretamente em um modelo específico de grafo aleatório *lei de potência*. Em particular, mostramos que o fator de aproximação esperado para o problema MDS é assintoticamente no máximo 9.14, e para o problema MVC, o fator é assintoticamente menor do que 2. Tais valores superam os limitantes conhecidos da literatura.

Palavras-chave: Complexidade de amostra e limitantes de generalização (ACM-2012 10010072). Algoritmos de aproximação (ACM-2012 10010918). Teoria dos grafos (ACM-2012 10003633).

ABSTRACT

Very large graphs arise in many contexts in natural and social phenomena. Algorithms with time complexity that scales in cubic or even in quadratic time in such graphs, although polynomial, can be computationally expensive in practice. In this thesis we present near-linear time randomized approximation schemes for two problems in graphs that are not known to admit algorithms that are *truly subcubic* in the number of vertices. Additionally, for a third problem, we also present a near-linear time randomized approximation scheme, but under the assumption that the input graph has logarithmic diameter, which is a very common scenario for large graphs in real-world applications. Our algorithms have been built using tools from sample complexity analysis, theory of Vapnik–Chervonenkis dimension, and Rademacher Averages. Let $G = (V, E)$ be a graph with $n = |V|$ and $m = |E|$. The first problem we deal with is the *percolation centrality* in a directed weighted graph G, a measure that quantifies the *importance* of a vertex in a graph that is going through a contagious process. In this work we present an expected $\mathcal{O}(m \log n \log \mathrm{Diam}_V(G))$ time approximation algorithm for the estimation of the percolation centrality for all vertices of G, wherein $\mathrm{Diam}_V(G)$ is the maximum number of vertices in a shortest path in G. The second problem is a relaxed version of the *all-pairs shortest paths* (APSP) where we are interested in computing all shortest paths with "centrality" at least ε, where $0 < \varepsilon < 1$ is a constant. This centrality measure is a certain generalization of the betweenness centrality. We propose an algorithm for this relaxed problem that runs in $\mathcal{O}(m \log n + (\mathrm{Diam}_V(G))^2)$ time. The third problem corresponds to the computation of the *local clustering coefficient* of each vertex in a graph G, a measure that quantifies the degree in which a vertex is a part of a cluster in G. The algorithm that estimates the local clustering of each vertex in G runs in $\mathcal{O}(\Delta \lg \Delta + m)$, wherein Δ is the maximum degree of a vertex in a graph. Finally, we also propose approximation algorithms for the *minimum dominating set* (MDS) and the *minimum vertex cover* (MVC). However, for both problems we apply probabilistic techniques that are not related to sample complexity analysis, since we found tighter bounds for the approximation factor for these problems by attacking them directly using a particular random graph model for *p*ower-law graphs. In particular, we show that the expected approximation factor for the MDS problem is assymtotically at most 9.14, and for the MVC problem, the factor is assymptotically smaller than two. Such values outperforms the known bounds of the literature.

Keywords: Sample complexity and generalization bounds (ACM-2012 10010072). Approximation algorithms (ACM-2012 10010918). Graph theory (ACM-2012 10003633).

CONTENTS

1 INTRODUCTION

Real-world applications where the amount of data is extremely large demand a large amount of processing time, even for algorithms that scale in cubic or quadratic time. However, some problems may not admit *truly* subcubic-time algorithms (i.e. $\mathcal{O}(n^{3-c})$ algorithms, for some constant $c > 0$) for computing exact solutions (Williams and Williams, 2010). That is the case for problems related to the computation of certain *centrality measures* in networks, so the investigation of faster algorithms that return approximated solutions for such problems is a natural path to pursue.

One usual approach for the design of efficient approximation algorithms is the use of sampling strategies, where only a small part of the input is inspected. In such strategies, the intrinsic trade-off between the size of the sample and the quality and confidence of the approximation obtained by the algorithm can be studied using standard deviation bounds from probability theory. For instance, we can estimate a single value of a certain quantity of interest (e.g. the estimation of a measure of centrality of a given vertex in a graph) using, for example, Hoeffding, Chernoff, or Azuma bounds. However, when the problem requires the analysis of multiple values of interest simultaneously in the sample space, say, the estimation of the centrality of every vertex of a given graph, techniques based on these standard bounds along with the union bound become extremely loose. The referred issue can be dealt with the use of sample complexity analysis, which aims to capture the *combinatorial structure* of the problem. The idea is that one can find out the minimum size of a random sample required to the estimation of a certain quantity, consistent with the desired parameters of quality and confidence, taking into account some property of the input instance. These techniques produce tighter bounds, and hence, more efficient algorithms. More concretely, we show that the sample size for the estimation of certain measures of graphs depends on properties such as the number of edges, or the diameter of an input graph. Since typically real-world graphs are characterized for being sparse and having logarithmic diameter (Chung, 2010), one can benefit from these features to obtain an algorithm that has its time complexity reduced on these graphs.

The use of sample complexity tools — Rademacher averages, the Vapnik-Chervonenkis (VC) dimension and pseudo-dimension theories, the ε-net and ε-sample theorems — on the design of sampling-based algorithms, which are at the core of the theory of statistical learning, are of theoretical and practical interest when one is dealing with large real-world graphs in a variety of contexts (Abraham et al., 2011; Riondato and Kornaropoulos, 2016; Riondato and Upfal, 2018; Ducoffe et al., 2020). In the work of Riondato et al. the authors showed that these techniques are promising and practical for real-world graphs. In addition, Ducoffe et al. (2020) showed that algorithms based on these techniques lead to truly subquadratic-time algorithms for computing the diameter of some classes of unweighted graphs. We show how such techniques from sample complexity theory can be used in three problems in graphs, namely the *percolation centrality problem*, the *all-pairs shortest path (APSP) problem* in a relaxed version, and the *local clustering coefficient problem*.

The *percolation centrality* of a vertex is a measure related to a contagious process going through a network (Piraveenan et al., 2013). The intuition behind such centrality, roughly speaking, is to quantify the probability a vertex has to spread the contamination

among the network if such vertex gets infected. It has relevant applications related to epidemics – e.g. the COVID-19 disease transmission – and misinformation spreading. Computing this measure for every vertex of a graph using exact algorithms is a problem that is computationally expensive, since it generalizes the betweenness centrality (Anthonisse, 1971; Freeman, 1977), a measure whose computation is known to be reducible to the APSP (Abboud et al., 2014). This later problem may not admit a *truly* subcubic algorithm, as suggested by results on fine-grained complexity (Williams and Williams, 2010). In this work, we present an $\mathcal{O}(m \log n \log \text{diam}_V(G))$ approximation algorithm that returns, with probability at least $1 - \delta$, the estimation for the percolation centrality of every vertex in G within ε error, for constant $0 < \varepsilon, \delta < 1$, where $\text{diam}_V(G)$ denotes the vertex-diameter of G. Note that the percolation centrality is guaranteed to be within ε error for every single vertex, not only for the average case. We use pseudo-dimension theory to show a bound for the sample size that depends on $\text{diam}_V(G)$. Notice that in common real-world applications, the graph is sparse and it has logarithmic diameter and, thus, the time complexity of our approach becomes simply $\mathcal{O}(n \log n \log \log n)$. We also show a progressive sampling algorithm for the problem using Rademacher averages that, in the worst case, create a sample with the same size bound as the one for the fixed-size sample approach. To the best of our knowledge, our algorithms are the first sampling-based algorithms for approximating percolation centrality.

In a relaxed version of the APSP problem, we are asked, for every pair of vertices (u, v), to output, with high probability, the length of a shortest path between u and v. This depends on a certain measure of the "importance" of such shortest path, which we call *shortest path centrality*. In an application of computing optimal routes in a map, for instance, the intuition is to find the routes that appear as subroutes in the largest amount of other routes in such map. One of our main results regarding this problem is a bound on the sample size that depends only on the parameters of quality and confidence (i.e. ε and δ, respectively) of the desired solution, and not on the size of the graph or on any of its topological properties. This is due to the constant VC dimension of the range space that models the version of the APSP that we are approaching. The total running time of the proposed algorithm is $\mathcal{O}(m + n \log n + (\text{diam}_V(G))^2)$, for constant $0 < \varepsilon, \delta < 1$. In the case of sparse graphs with logarithmic diameter, this time complexity is reduced to $\mathcal{O}(n \log n)$.

The clustering coefficient of a vertex, called the *local clustering coefficient*, measures the degree in which such vertex is part of a cluster in a graph. It is related to the ratio of the number of triangles existing in the neighborhood of the target vertex to the total number of pair of nodes in the neighborhood. Consider a graph that represents a social network where each vertex is a person and each edge represents a relationship between two people. Intuitively, the local clustering coefficient measures the probability that a person has to form communities on the network by estimating the probability that a pair of friends of such person also become friends between each other. The measure was originally proposed by Watts and Strogatz (1998) in order to determine if a graph has the property of being *small-world*. In particular, we can say that the local clustering coefficient is a variant of the local triangle counting problem, and the exact computation of this task can be done by a matrix multiplication based algorithm that runs in time $\mathcal{O}(m^{2w/(w+1)})$, where w is around 2.373, which is associated with the cost of performing a matrix multiplication (Alon et al., 1997). In Chiba and Nishizeki (1985), the authors proposed a $\mathcal{O}(\alpha(G)m)$ algorithm, where $\alpha(G)$ is the arboricity of graph G, that does not use matrix multiplication. We present, in this work, an algorithm that

estimates within ε error, with probability at least $1-\delta$, the local clustering coefficient of each vertex of a graph in time $\mathcal{O}(\Delta \lg \Delta + m)$, where Δ is the maximum degree of a vertex in graph G. Besides the previously mentioned characteristics that real-world graphs typically have, many of them also have power-law vertex degree distribution, as shown by empirical studies from the late 1990's and early 2000's (Barabási and Albert, 1999; Kleinberg et al., 1999; Kumar et al., 2000; Kleinberg and Lawrence, 2001; Guelzim et al., 2002; Siganos et al., 2003; Eubank et al., 2004; Faloutsos et al., 2011). Thus, for power-law graphs such as the one described in the model of Aiello et al. (2001), if we assume $\Delta = o(\sqrt{n})$, then our algorithm has complexity time $\mathcal{O}(\sqrt{n} \lg \sqrt{n})$ in this case. We also show that the algorithm runs in time $\mathcal{O}(\Delta)$ if the input is a planar graph. In the case of bounded-degree graphs the running time is $\mathcal{O}(1)$ if a bound for the value of Δ is given as a part of the input, and $\mathcal{O}(n)$ otherwise.

We also present approximation algorithms for two $\mathcal{NP}$-hard problems, namely, the *minimum dominating set (MDS)* and the *minimum vertex cover (MVC)* problems in power-law graphs. The minimum dominating set problem corresponds to find the minimum set of vertices $D \subseteq V$ in a graph $G = (V, E)$ such that each $v \in V$ is either in D or has at least one neighbor in D. The minimum vertex cover problem consists of finding the minimum set $C \subseteq V$ such that each $e \in E$ has at least one endpoint in C.

The proposed approaches for both problems also rely on probabilistic techniques, but not particularly on sample complexity analysis. We aimed to solve the problem for the case of power-law graphs, obtaining better approximation factors by attacking the problem *directly* instead of considering a sample complexity approach such as the one presented in the work of Brönnimann and Goodrich (1995). In their work, the authors showed a deterministic polynomial-time method for the approximation factor of the minimum set cover of a graph G within at most $\mathcal{O}(d \log(d \text{ OPT}))$, where d is the VC dimension of the set system that models the problem and OPT is the optimal size of the solution, and within $\mathcal{O}(\log \text{ OPT})$ if d is constant.

Consider a domain U and a set I of subsets of U. The *set cover* is the problem of finding the minimum number of sets in I such that the union of such sets corresponds to U. In particular, the problems of the minimum set cover and the hitting set are equivalent. The *hitting set problem* consists of finding the smallest subset of U such that the elements in U intersect at least one subset in I. The definition of ε-net is a relaxation of the definition of a hitting set, so when dealing with sample complexity analysis in the design of algorithms for $\mathcal{NP}$-Hard problems, it is natural first establishing reductions to/from the hitting set problem. In our case, we have that MVC is a special case of the hitting set, and MVC can be reduced to MDS.

The MDS problem does not admit a polynomial-time approximation algorithm with a strictly sublogarithmic factor unless $\mathcal{P} = \mathcal{NP}$ (Raz and Safra, 1997). It is conjectured that the MVC problem does not admit a polynomial-time approximation algorithm with a factor smaller than 2 (Khot and Regev, 2008; Garey and Johnson, 1979). However, these bounds can be overcome when the input is a power-law graph.

We present upper bounds ϕ and ψ for the expected approximation factors of the MDS and the MVC problems, respectively, in power-law graphs modelled by a *generalized random graph* with expected degree distribution as the one presented in the work of Aiello et al. (2001). We show that a combination of a preprocessing on the neighborhood of vertices of degree one with a 2-approximation on the remaining part of the graph leads to bounds that outperform previous results from literature. In particular, we show that ϕ is asymptotically at most 9.14, a tighter bound than the

one of Gast et al. (2015). For the value of ψ, we show that ψ is asymptotically strictly smaller than 2, improving the previous results of Gast et al. (2015) and Vignatti and Silva (2016). We show in Figures 6.1 and 6.2, in Chapter 6, the comparative results between our bounds and the ones presented by the aforementioned authors.

The text is organized as follows: in Chapter 2 we establish the notation and provide the graph theoretical definitions used in this work (Section 2.1) as well as the main concepts of sample complexity theory which are the groundwork of our algorithms (Section 2.2); in Chapters 3, 4, and 5 we present our analytical results on the percolation centrality, the relaxed APSP, and the local clustering coefficient problems, respectively; in Chapter 6 we show our analytical results on the minimum dominating set and the minimum vertex cover problems; in Chapter 7 we present the final remarks with directions for future work.

The contents of Chapter 3 are published in a conference paper (Lima et al., 2020) and a journal article (Lima et al., 2022a), and the contents of Chapter 5 are published in a conference paper (Lima et al., 2022b). A previous version of the contents of Chapter 4 is published in a conference paper (Lima et al., 2021b), and an updated version is under review in a journal. The contents of Chapter 6 are also under review in a journal. Other results related to the main theme of this thesis are published in journals (Lima et al., 2019, 2021a).

2 THEORETICAL FOUNDATIONS

In this chapter, we present the main definitions and notation used in this work with respect to graphs (Section 2.1) and with respect to sample complexity theory (Section 2.2).

2.1 GRAPH PRELIMINARIES

In this section, we present the main concepts of graph theory which are common for the problems that we are approaching in this work. An in-depth exposition of such concepts can be found in the book of Bondy and Murty (1976).

Let $G = (V, E)$ be a (directed or undirected) graph and let ω be a function from E to an enumerable subset of $\mathbb{R}_{\geq 0}$. Without loss of generality, we assume that G is (strongly) connected, since all results in this document can be applied to the connected components of the graph if it is disconnected. We also assume that the edges weights can be represented in linear size on G. A *path* is a sequence of vertices $P = (v_1, v_2, \ldots, v_k)$ such that $v_i \neq v_{i+1}$ and $(v_i, v_{i+1}) \in E$, for $1 \leq i < k$. If $u = v_1$ and $v = v_k$, such path is referred to as a (u, v)-*path*. We define E_P as the set of edges of P. The *shortest path* from u to v in G is the (u, v)-path such that the sum of the weights of the edges in E_P is minimized.

The set of all shortest paths from u to v in G is denoted $\mathcal{C}_{uv}$. For a given path $P \in \mathcal{C}_{uv}$, let $\mathrm{Inn}(P)$ be the set of *inner* vertices of P, that is, $\mathrm{Inn}(P) = \{w \in P : w \notin \{u, v\}\}$. Consider a shortest (u, v)-path P, and let u' and v' be two vertices of P, with u' closer to u and v' closer to v. The subpath of P starting in u' and ending in v' is called a (u', v')-*subpath of* P. The (immediate) *predecessor* of v in a shortest (u, v)-path P, denoted $\mathrm{pred}_P(v)$, is the vertex $w \in \mathrm{Inn}(P)$ such that $(w, v) \in E_P$. The *diameter* of G, denoted diam_G, is the size of the largest shortest path in G. The *vertex-diameter*, denoted $\mathrm{diam}_V(G)$, is the maximum number of vertices in a shortest path of G.

2.2 SAMPLE COMPLEXITY THEORY

We present in this section the main definitions, examples, and results related to sample complexity theory that we use in this work. In Section 2.2.1 we introduce the Vapnik-Chervonenkis dimension (VC dimension) and pseudo-dimension theories; in Section 2.2.3 we present the ε-net and ε-sample theorems; in Section 2.2.4 we introduce the Rademacher averages and their relation with the progressive sampling; in Section 2.2.5 we present a brief overview of the applications of sample complexity theory in a variety of problems.

An in-depth exposition on VC Dimension, pseudo-dimension theories, ε-sample, ε-net theorems, and Rademacher averages can be found in the books of Anthony and Bartlett (2009), Mohri et al. (2012), Shalev-Shwartz and Ben-David (2014), and Mitzenmacher and Upfal (2017).

2.2.1 VC dimension

The VC dimension theory, introduced in 1971 by Vladmir Vapnik and Alexey Chervonenkis (Vapnik and Chervonenkis, 1971), is a measure of the expressiveness of a space of functions, and it was central to relate the size of a sample with the convergence of empirical averages to their true expectations. It was firstly introduced in learnability by Haussler and Welzl (1986) and Blumer et al. (1989), being in the core of the Probably Approximately Correct (PAC) Framework of Kearns et al. (1994). It has applications that extend the context of learning algorithms, such as computational geometry, database management, and graph algorithms (see Section 2.2.5).

Definition 1 (Range space)**.** *A* range space *is a pair $\mathcal{R} = (X, \mathcal{I})$, where X is a domain (finite or infinite) and $\mathcal{I}$ is a collection of subsets of X, called* ranges*.*

For a given $S \subseteq X$, the *projection* of $\mathcal{I}$ on S is the set $\mathcal{I}_S = \{S \cap I : I \in \mathcal{I}\}$. If $|\mathcal{I}_S| = 2^{|S|}$ then we say S is *shattered* by $\mathcal{I}$. The VC dimension of a range space is the size of the largest subset S that can be shattered by $\mathcal{I}$, as presented by the following definition.

Definition 2 (VC dimension)**.** *The* VC dimension *of a range space $\mathcal{R} = (X, \mathcal{I})$, denoted by $VCDim(\mathcal{R})$, is $VCDim(\mathcal{R}) = \max\{d : \exists S \subseteq X$ such that $|S| = d$ and $|\mathcal{I}_S| = 2^d\}$.*

The definition above does not imply that every set of cardinality d is shattered by $\mathcal{I}$, but it implies that if one shows that every set of cardinality at least $d + 1$ is not shattered by $\mathcal{I}$, then VCDim($\mathcal{R}$) is no larger than d. In fact, computing the VC dimension of a range space over a finite domain, in general, is $\Sigma_3^{\mathcal{P}}$-Complete (Schaefer, 1999).

Consider the following example. Let $X = \mathbb{R}$ and $\mathcal{I} = \{[a, b] : a, b \in \mathbb{R}\}$ such that $[a, b] = \{x \in \mathbb{R} : a \leq x \leq b\}$. A classical result transcribed in Theorem 1 states that this range space has bounded VC dimension although it has infinite set of elements and infinite set of ranges.

Theorem 1. *Let $\mathcal{R} = (X, \mathcal{I})$ be a range space with $X = \mathbb{R}$ and $\mathcal{I} = \{[a, b] : x \in \mathbb{R}$ and $a \leq x \leq b\}$. Then $VCDim(\mathcal{R}) = 2$.*

Proof. First we show that VCDim($\mathcal{R}$) ≥ 2 by demonstrating that there exists a set of cardinality 2 which is shattered by $\mathcal{I}$. Consider the set $S = \{2, 4\}$. Then $|\mathcal{I}_S| = 2^2 = 4$ since every subset of S can be generated as follows:

- $S \cap [0, 1] = \emptyset$;
- $S \cap [1, 3] = \{2\}$;
- $S \cap [3, 5] = \{4\}$;
- $S \cap [1, 5] = \{2, 4\}$.

Now we show that VCDim($\mathcal{R}$) < 3. Consider a set $S = \{a, b, c\}$ such that $a, b, c \in \mathbb{R}$ and $a < b < c$. Assume, for the sake of contradiction, that S is shattered by $\mathcal{I}$. By definition, $\{a, c\} \in \mathcal{I}_S$, and then for $x, y \in \mathbb{R}$, a range $[x, y] \in \mathcal{I}$ must exist so that $S \cap [x, y] = \{a, c\}$, $x < a$, and $y \geq c$ (otherwise, $a, c \notin ([x, y] \cap S)$. Hence, $x < a < b < c$ and $b \in \{[x, y] \cap S\}$, a contradiction since we have supposed $[x, y] \cap S = \{a, c\}$. Therefore, S is not shattered by $\mathcal{I}$ and VCDim($\mathcal{R}$) $= 2$. □

Consider the range space $\mathcal{R} = (X, \mathcal{I})$ wherein $X = \mathbb{R}^d$ and $\mathcal{I}$ is the set of half-spaces in d dimensions. Theorem 3 states that $\text{VCDim}(\mathcal{R}) = d+1$, for $d \in \mathbb{N}$. To illustrate the idea, we first present the classical result stated in Theorem 2 that $\text{VCDim}(\mathcal{R}) = 3$ for $X = \mathbb{R}^2$.

Theorem 2. *Let $\mathcal{R} = (X, \mathcal{I})$ be a range space with $X = \mathbb{R}^2$ where $\mathcal{I}$ is the set of linear half-spaces in $\mathbb{R}^2$. Then VCDim$(\mathcal{R}) = 3$.*

Proof (sketch). We first show $\text{VCDim}(\mathcal{R}) \geq 3$ by depicting in Figure 2.1 a set of three non-colinear points that is shattered by $\mathcal{I}$.

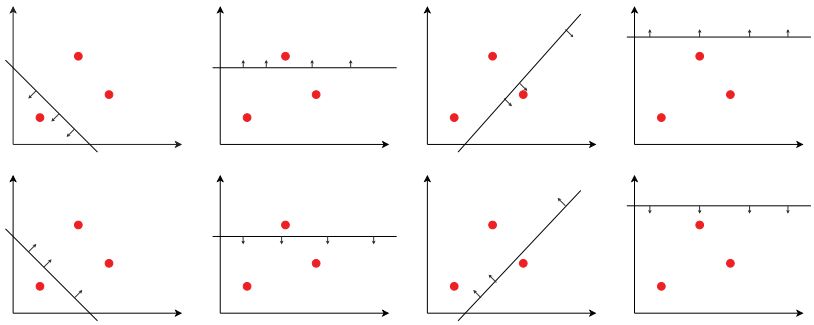


Figure 2.1: Possible configurations of linear half-spaces which shatter the set of red points

To show that $\text{VCDim}(\mathcal{R}) < 4$, we have to consider three cases:

a) a set of three colinear points, which cannot be shattered, since there is no way to separate the middle point from the other two by any half-space;

b) a point that lies inside the convex hull defined by the other three points, which cannot be shattered since there is no way to separate it from the other points by any half-space;

c) a set of four points that define a convex hull which cannot be shattered since there is no way to separate the red points from the blue points by any half-space as demonstrated in Figure 2.2.

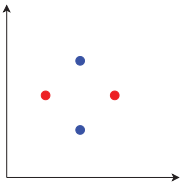


Figure 2.2: No set of four points can be shattered by any half-space.

Therefore, $\text{VCDim}(\mathcal{R}) = 3$. □

Theorem 3. *Let $\mathcal{R} = (X, \mathcal{I})$ be a range space with $X = \mathbb{R}^d$ where $\mathcal{I}$ is the set of half-spaces with dimension d, for $d \in \mathbb{N}$. Then VCDim$(\mathcal{R}) = d + 1$.*

Proof. Let $f_{w,b}(x) = \text{sgn}(w^T x + b)$ be a function that defines a half-space in $\mathbb{R}^d$ for some $w, x \in \mathbb{R}^d$ and $b \in \mathbb{R}$, where sgn is the sign function. We will first show that $\text{VCDim}(\mathcal{R}) \geq d + 1$ by defining a set of points X that is shattered by $\mathcal{I}$.

Let $X = \{\mathbf{0}, \boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_d}\}$ be such set, where $\mathbf{0}$ is the origin of $\mathbb{R}^d$ and each $\boldsymbol{x_i}$, for $i \in \{1, \ldots, d\}$, are the vectors of the standard basis of $\mathbb{R}^d$ (i.e. $x_1 = (1, 0, \ldots, 0)$, $x_2 = (0, 1, 0, \ldots, 0)$ and so on). Consider a labeling of the points in X in $1's$ and $-1's$ defined as $y_0, y_1, \ldots, y_d$ where y_0 is the label of $\mathbf{0}$ and y_i is the label of x_i, for $i \in \{1, \ldots, d\}$.

Let $b = 0.5y_0$ and consider the vector $w = (w_1, \ldots, w_d)$, where $w_i = y_i$ for $i \in \{1, \ldots, d\}$. Note that the inner product of w^T by x_i is equal to w_i. Then, $f_{w,b}(x_i) = \text{sgn}(w_i + 0.5y_0)$. We have the following cases:

a) $w_i = 1$ and $y_0 = 1$, and $f_{w,b}(x_i) = \text{sgn}(1 + 0.5) = 1$

b) $w_i = -1$ and $y_0 = 1$, and $f_{w,b}(x_i) = \text{sgn}(-1 + 0.5) = -1$

c) $w_i = 1$ and $y_0 = -1$, and $f_{w,b}(x_i) = \text{sgn}(1 - 0.5) = 1$

d) $w_i = -1$ and $y_0 = -1$, and $f_{w,b}(x_i) = \text{sgn}(-1 - 0.5) = -1$

The function $f_{w,b}(x_i)$ always correctly computes the label of x_i, and therefore, any half-space defined by this function shatters the set X. So $\text{VCDim}(\mathcal{R}) \geq d + 1$.

To show that $\text{VCDim}(\mathcal{R}) < d + 2$, recall Radon's Theorem (Radon, 1921), which states that any set $S \subseteq \mathbb{R}^d$ of size $d + 2$ can be partitioned in two disjoint sets S_1 and S_2 such that the convex hull of S_1 intersects with the convex hull of S_2.

Consider a set $X = \{x_1, \ldots, x_{d+2}\}$ of $d + 2$ points of $\mathbb{R}^d$ and a partition of X in two disjoint sets X_1 and X_2. Without loss of generality, give a labelling to the points in X such that every point in X_1 has the label 1 and every point in X_2 has the label -1. If there is a half-space that separates X_1 from X_2, then the space that contains X_1 is convex (and so does the space of X_2), and consequently, the points in the convex hull of X_1 have label 1 and correspondingly, the points in the convex hull of X_2 have label -1.

However, by Radon's Theorem, there is a partition of X in two disjoint sets X_1' and X_2' such that the convex hull of X_1' intersects with the convex hull of X_2'. Hence, if $X_1 = X_1'$ and $X_2 = X_2'$, then we have a contradiction that X can be separated by the given labeling. Therefore, X cannot be always separated in two disjoint sets with no intersecting convex hulls, and then X cannot be shattered by $\mathcal{I}$, which gives $\text{VCDim}(\mathcal{R}) < d + 2$.

Finally, then, $\text{VCDim}(\mathcal{R}) = d + 1$. □

In this work, we are usually interested in range spaces defined over discrete and finite sets. In Chapter 14.2.1 of the book of Mitzenmacher and Upfal (2017) the authors give such an example, whose we present in Theorem 4. Let a *monotone Boolean conjunction* be defined as a Boolean conjunction that has only non-negated variables. Consider a set of n Boolean variables $Y = \{y_1, y_2, \ldots, y_n\}$ and the set $U = \{0, 1\}^n$ of all possible true assignments to the variables in the natural way. Let $M = \{f_{Y'} : Y' \subseteq Y\}$, where $f_{Y'}$ is a monotone conjunction function defined for Y', and $\mathcal{I}_{Y'}$ is the set of assignments that satisfy $f_{Y'}$, i.e. $\mathcal{I}_{Y'} = \{x \in U : f_{Y'} \in M \text{ and } f_{Y'}(x) = 1\}$.

Let $\mathcal{R} = (U, \mathcal{I})$ be a range space with $X = U$ and $\mathcal{I} = \{\mathcal{I}_{Y'} : f_{Y'} \in M\}$. Then $\text{VCDim}(\mathcal{R}) = n$. Before presenting the idea of the proof described in Mitzenmacher and Upfal (2017), consider the following example for a set of three Boolean variables $Y = \{y_1, y_2, y_3\}$ and the set $S \subseteq U$ such that $S = \{(0, 1, 1), (1, 0, 1), (1, 1, 0)\}$, where the $\emptyset$ is associated to a formula that has no variables (empty conjunction). Then S can be shattered as follows:

- $S \cap \mathcal{I}_{y_1, y_2, y_3} = \emptyset$;

- $S \cap \mathcal{I}_{y_2,y_3} = \{(0,1,1)\}$;
- $S \cap \mathcal{I}_{y_3} = \{(0,1,1),(1,0,1)\}$;
- $S \cap \mathcal{I}_{y_1,y_3} = \{(1,0,1)\}$;
- $S \cap \mathcal{I}_{y_2} = \{(0,1,1),(1,1,0)\}$;
- $S \cap \mathcal{I}_{y_1,y_2} = \{(1,1,0)\}$;
- $S \cap \mathcal{I}_{y_1} = \{(1,0,1),(1,1,0)\}$;
- $S \cap \emptyset = \{(0,1,1),(1,0,1),(1,1,0)\}$.

Theorem 4. *For the range space $\mathcal{R} = (X,\mathcal{I})$ where $X = U$ and $\mathcal{I} = \{\mathcal{I}_{Y'} : f_{Y'} \in M\}$, $VCDim(\mathcal{R}) = n$.*

Proof. We will show that $\text{VCDim}(\mathcal{R}) = n$ for a subset $S \subseteq X$ of size n where each assignment in S has exactly one zero and S has no repeated assignments. That is,

$$S = \{(0,1,\ldots,1),(1,0,1,\ldots,1),\ldots,(1,1,\ldots,1,0)\}.$$

Then each subset $S' \subseteq S$ can be generated by the intersection of S with $\mathcal{I}_f$ such that $f = y_{i_1} \wedge y_{i_2} \wedge \ldots \wedge y_{i_j}$, where $y_{i_1}, y_{i_2}, \ldots, y_{i_j}$ are the variables that have value 1 in all $x \in S'$. The whole set S can be generated with the intersection of the empty conjunction, which by definition is always *true*. Hence, S is shattered by $\mathcal{I}$, and then $\text{VCDim}(\mathcal{R}) \geq n$.

Now observe that 2^n intersections of S with each $\mathcal{I}_{Y'} \in \mathcal{I}$ are necessary to shatter S, and the size of $\mathcal{I}$ is 2^n. If the size of S was $n+1$, then 2^{n+1} intersections in the form $S \cap \mathcal{I}_{Y'}$ would be necessary to shatter S. But $|\mathcal{I}| = 2^n$, which is a contradiction.

Therefore, $\text{VCDim}(\mathcal{R}) < n+1$ and finally, $\text{VCDim}(\mathcal{R}) = n$. □

2.2.2 Pseudo-dimension

In a range space $\mathcal{R} = (X,\mathcal{I})$, let $f_I : X \to \{0,1\}$ be the indicator function associated to the range $I \in \mathcal{I}$, i.e. for a given $x \in X$, the function f_I either returns 1 if x is inside the range I or it returns 0.

Let $\mathcal{F} = \{f_I : I \in \mathcal{I}\}$. When modeling the problem that we have at hand in terms of a range space $\mathcal{R} = (X,\mathcal{F})$ such that $\mathcal{F}$ contains functions with codomain in the range $[0,1]$, then we need to use *pseudo-dimension theory*, which is a generalization of the VC dimension theory. We have to create a new range space $\mathcal{R}' = (D,\mathcal{F}^+)$ where $D = X \times [0,1]$, and for each $f \in \mathcal{F}$, we create a set $R_f = \{(x,t) : x \in X \text{ and } t \leq f(x)\}$.

Definition 3 (Pseudo-dimension)**.** *Let $\mathcal{R} = (X,\mathcal{F})$ and $\mathcal{R}' = (D,\mathcal{F}^+)$ be range spaces, where $\mathcal{F}^+ = \{R_f : f \in \mathcal{F}\}$. The* pseudo-dimension *of $\mathcal{R}$, denoted by $PD(\mathcal{R})$, corresponds to the VC dimension of $\mathcal{R}'$, i.e. $PD(\mathcal{R}) = VCDim(\mathcal{R}')$.*

Definition 3 implies that pseudo-dimension is a generalization of VC dimension.

The next two lemmas, proved by Riondato and Upfal (2018), present constraints on the sets that can be shattered by a range set $\mathcal{F}^+$.

Lemma 1 (Riondato and Upfal (2018), Lemma 3.7)**.** *Let $B \subseteq D$ be a set that is shattered by $\mathcal{F}^+$. Then, B can contain at most one $(d,y) \in D$, for each $d \in X$.* □

Lemma 2 (Riondato and Upfal (2018), Lemma 3.8)**.** *Let $B \subseteq D$ be a set that is shattered by $\mathcal{F}^+$. Then, B does not contain any element in the form $(d,0) \in D$, for each $d \in X$.* □

2.2.3 ε-nets, ε-samples, empirical averages, and ε-representative samples

Consider a range space $\mathcal{R} = (X, \mathcal{I})$. The following combinatorial object, called ε-net, is useful when one wants to find a sample $S \subseteq U$ that intersects every range in $\mathcal{I}$ of a sufficiently large size.

Definition 4 (ε-net)**.** *Given $0 < \varepsilon < 1$, a set S is called ε-net w.r.t. a range space $\mathcal{R} = (X, \mathcal{I})$ and a probability distribution π on X if*

$$\forall I \in \mathcal{I}, \quad \Pr_{\pi}(I) \geq \varepsilon \Rightarrow |I \cap S| \geq 1.$$

Consider, for example, the unit square Q in $[0,1]\times[0,1]$. In Example 1 we depict, in Figure 2.3, a set of eight points inside Q which is an ε-net for Q, where $\varepsilon = 1/4$.

Example 1. *Consider a range space $\mathcal{R} = (X, \mathcal{I})$ where $X = \mathbb{R}^2$, $\mathcal{I}$ is the set of closed filled rectangles (there is one range for each product of closed intervals), and consider the unit square Q in $[0,1]\times[0,1]$. Then a set N of eight points inside Q with the configuration showed in Figure 2.3 is an $1/4$-net of Q.*

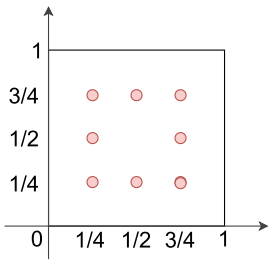


Figure 2.3: A configuration of eight points inside $Q = [0,1]\times[0,1]$ that is an ε-net of Q, where $\varepsilon = 1/4$.

Consider a rectangle P that covers at least $1/4$ of the area of Q. Then N is an $1/4$-net of the unit square Q, since P has a non-empty intersection with at least one of the points in N.

The definition of ε-sample is a stronger notion, as it not only intersects ranges of a sufficiently large size but it also guarantees the right relative frequency of each range in $\mathcal{I}$ within the sample S.

Definition 5 (ε-sample)**.** *Given $0 < \varepsilon < 1$, a set S is called ε-sample w.r.t. a range space $\mathcal{R} = (X, \mathcal{I})$ and a probability distribution π on X if*

$$\forall I \in \mathcal{I}, \quad \left| \Pr_{\pi}(I) - \frac{|S \cap I|}{|S|} \right| \leq \varepsilon.$$

A polling about presidential election is an example of ε-sample, where X is the set of the electorate from the country and there is one range $I \in \mathcal{I}$ for each one of the candidates, where each range contains people who vote on the candidate represented by such set. If the previous polling done on a group of people S observed that the percent of votes in the polling for each candidate I, i.e. $|S \cap I|/|S|$, differs from at most 0.01 from the original percent given by the results of the election, then S is a 0.01-sample for the range space defined by $\mathcal{R} = (X, \mathcal{I})$. If the ranges $I \in \mathcal{I}$ are all disjoint, then the 0.01-sample returns an accurate approximation to the results of an usual election. On the other hand, if we allow intersections between the ranges, then the approximations

would be related to the results of an unusual election where each person can vote in more than one candidate.

When computing ε-samples for a given range space $\mathcal{R} = (X, \mathcal{I})$, we typically build a sample S from elements of X. If $\mathcal{R}$ has the *uniform convergence* property, defined below, then a goal is to find the minimum size that S must have to be an ε-sample, with high probability, with respect to such range space.

Definition 6 (Mitzenmacher and Upfal (2017), Definition 14.8)**.** *Consider a range space $\mathcal{R} = (X, \mathcal{I})$. We say that $\mathcal{R}$ has the* uniform convergence *property if for every $\varepsilon, \delta > 0$ there is a sample size $r = r(\varepsilon, \delta)$ such that, for every distribution π over X, a sample S of r elements of X sampled with respect to π is a ε-sample. This holds with probability at least $1 - \delta$.*

One can obtain lower bounds for the size of S via the probabilistic method. However, these bounds can be improved if the VC dimension of the range space that models the problem at hand, denoted d, is finite (Theorem 7 states bounds that are based on d).

A more general definition of ε-sample, called ε-representative set, is given below for the context where, for a given domain X, there is a family of functions $\mathcal{F}$ from X to $\mathbb{R}_{>0}$ such that $\mathcal{F} = \{f_I : I \in \mathcal{I}\}$. Let S be a collection of r elements from X sampled with respect to a probability distribution π.

Definition 7. *For each $f \in \mathcal{F}$, we define the expectation of f and its empirical average as L_X and L_S, respectively, i.e.*

$$L_X(f) = \mathbb{E}_{x \sim \pi}[f(x)]$$

and

$$L_S(f) = \frac{1}{r} \sum_{s \in S} f(s).$$

Definition 8. *Given $0 < \varepsilon < 1$, a set S is called* ε-representative *w.r.t. some domain X, a family of functions $\mathcal{F}$, and a probability distribution π if for all $f \in \mathcal{F}$,*

$$|L_S(f) - L_X(f)| \leq \varepsilon.$$

As observed by Riondato and Upfal (2018), by the linearity of expectation we have that the expected value of the empirical average $L_S(f)$ is equal to $L_X(f)$. By the *law of large numbers*, $L_S(f)$ converges to its true expectation as r goes to infinity, since $L_S(f)$ is the empirical average of r random variables sampled independently and identically with respect to π. However, this law provides no information about the value $|L_S(f) - L_X(f)|$ for any sample size. Thus, the results from the VC dimension and pseudo-dimension theories become important to provide bounds on the size of the sample that guarantees that the maximum deviation of $|L_S(f) - L_X(f)|$ is within ε with probability at least $1 - \delta$, for given $0 < \varepsilon, \delta < 1$.

Estimating the bound that guarantees that for a single $f \in \mathcal{F}$ the value of $|L_S(f) - L_X(f)|$ is within ε with probability at least $1 - \delta$ can be done by the use of Hoeffding's bound (Theorem 5).

Theorem 5 (Mitzenmacher and Upfal (2017), Theorem 4.12)**.** *Let $X_1, \ldots, X_r$ be r independent random variables such that for all $1 \leq i \leq r$, $\mathbb{E}[X_i] = \mu$, and $\Pr(a \leq X_i \leq b) = 1$, where $a, b \in \mathbb{R}$. Then*

$$\Pr\left(\left|\frac{1}{r}\sum_{i=1}^{r} X_i - \mu\right| \geq \varepsilon\right) \leq 2e^{-2r\varepsilon^2/(b-a)^2}.$$

□

If each X_i is the random variable for the value of f to a point in X such that $\Pr(a \leq X_i \leq b) = 1$, then $L_X(f) = \mathbb{E}_{x\sim\pi}[f(x)] = \mu$ and $L_S(f) = \frac{1}{r}\sum_{i=1}^{r} X_i$, and

$$\Pr(|L_S(f) - L_X(f)| \geq \varepsilon) \leq 2e^{-2r\varepsilon^2/(b-a)^2}.$$

Consider, for example, a function $f : X \to [0,1]$. Then

$$\Pr(|L_S(f) - L_X(f)| \geq \varepsilon) \leq 2e^{-2r\varepsilon^2} \leq \delta.$$

The minimum sample size r which guarantees that the estimations will be within the desired parameters ε and δ is obtained by applying Boole's Inequality (Theorem 6), known as *union bound*, to all functions in $\mathcal{F}$.

Theorem 6 (Mitzenmacher and Upfal (2017), Lemma 1.2)**.** *For any finite or countably infinite sequence of events $E_1, E_2, \ldots$*

$$\Pr\left(\bigcup_{i\geq 1} E_i\right) \leq \sum_{i\geq 1}^{\infty} \Pr(E_i).$$

□

If the size of $\mathcal{F}$ in our example is n, then using the union bound,

$$\begin{aligned} n \cdot 2e^{-2r\varepsilon^2} &\leq \delta \\ 2r\varepsilon^2 &\leq \ln\delta - \ln(2n) \\ r &\geq \frac{\ln 2 + \ln n - \ln\delta}{\varepsilon^2}. \end{aligned}$$

Note that the bound above depends on the size of $\mathcal{F}$. This bound can be improved if we take into consideration the combinatorial structure of the range space defined for the problem that we are dealing with. An upper bound to the pseudo-dimension of a range space allows us to build an ε-representative set (or ε-net) and to guarantee the uniform convergence property. This is stated in Theorem 7.

Theorem 7 (Har-Peled and Sharir (2011), Theorem 2.12)**.** *Given $0 < \varepsilon, \delta < 1$, let $\mathcal{R} = (X, \mathcal{F})$ be a range space with $VCDim(\mathcal{R}) \leq d$, a probability distribution π on the domain X, and let c be a universal positive constant.*

1. *A collection $S \subseteq X$ sampled w.r.t. π with $|S| = \frac{c}{\varepsilon^2}\left(d + \ln\frac{1}{\delta}\right)$ is ε-representative with probability at least $1 - \delta$.*

2. *A collection $S \subseteq X$ sampled w.r.t. π with $|S| = \frac{c}{\varepsilon}\left(d\ln\frac{1}{\varepsilon} + \ln\frac{1}{\delta}\right)$ is an ε-net with probability at least $1 - \delta$.* □

In the work of Löffler and Phillips (2009), it has been proven that the constant c is approximately 1/2.

The following combinatorial object, called a *relative (p, ε)-approximation*, is useful in the context when one wants to find a sample $S \subseteq X$ that estimates the size of

ranges in $\mathcal{I}$, with respect to an adjustable parameter p, within $\varepsilon \Pr_\pi(I)$, for $0 < \varepsilon, p, \delta < 1$. This holds with probability at least $1 - \delta$, for $0 < \delta < 1$, where π is a distribution on X and $\Pr_\pi(I)$ is the probability of a sample from π belongs to I.

Definition 9 (relative (p, ε)-approximation, (Riondato and Kornaropoulos (2016), Definition 5))**.** *Given $0 < p, \varepsilon < 1$, a set S is called a (p, ε)-approximation w.r.t. a range space $\mathcal{R} = (X, \mathcal{I})$, and a distribution π on X if for all $I \in \mathcal{I}$,*

(i) $\left| \Pr_\pi(I) - \frac{|S \cap I|}{|S|} \right| \leq \varepsilon \Pr_\pi(I), \quad$ *if* $\Pr_\pi(I) \geq p$,

(ii) $\frac{|S \cap I|}{|S|} \leq (1 + \varepsilon)p, \quad$ *otherwise.*

An upper bound to the VC dimension of a range space allows to build a sample S that is a (p, ε)-approximation set.

Theorem 8 (Har-Peled and Sharir (2011), Theorem 5)**.** *Given $0 < \varepsilon, \delta, p < 1$, let $\mathcal{R} = (X, \mathcal{I})$ be a range space with $VCDim(\mathcal{R}) \leq d$, a given distribution π on X, and let c be a universal positive constant. A collection $S \subseteq X$ sampled w.r.t. π with*

$$|S| \geq \frac{c}{\varepsilon^2 p} \left(d \log \frac{1}{p} + \log \frac{1}{\delta} \right)$$

is a relative (p, ε)-approximation with probability at least $1 - \delta$, where c is an absolute positive constant. □

2.2.4 Progressive Sampling and Rademacher Averages

Finding a bound on the sample size that is tight may be a complicated task depending on the problem that we have at hand. Hence, making use of progressive sampling becomes an alternative to this issue (Provost et al., 1999). In this approach, the process starts with a small sample that progressively increases until the desired accuracy of the solution is achieved. The combination of an appropriate scheduling for the sample increase with an efficient-to-evaluate stopping condition (i.e. knowing when the sample is large enough) leads to an improvement in time for the estimation of the value of interest in comparison to a fixed-size sample approach (Riondato and Upfal, 2018).

The VC dimension and pseudo-dimension theories measure the range space independently of the probability distribution on the input space. The use of Rademacher averages, however, leads to a stopping condition for the progressive sampling approach that takes into consideration the input distribution on the range set. This theory lies in the core of statistical learning theory, although their applications extend beyond the context of learning algorithms.

Before describing the formal definition of the main concepts and definitions on Rademacher averages, we will consider the following application. Let A be a classifier algorithm and X a set of objects to be classified. The function $c : X \to \{-1, 1\}$ labels each object in X as *positive* (label with value 1) or *negative* (labels with value -1) according to the given classification problem. The algorithm receives as input a training set $(x_1, c(x_1)), ..., (x_m, c(x_m))$, with size m, where x_i is chosen according to the distribution π, and $c(x_i)$ is the correct classification of x_i. Let $\mathcal{C}$ be the set of possible classifications of the objects, called *hypotheses*. The algorithm returns a hypothesis $h \in \mathcal{C}$ which is a function $h : X \to \{-1, 1\}$. Each h represents a partition of X.

We say that the *training error* of a hypothesis h is the fraction of examples incorrectly classified by h. Formally,

$$\hat{\text{err}}(h) = \frac{1}{m}|\{i : h(x_i) \neq c(x_i), 1 \leq i \leq m\}|.$$

Since h and c are functions with codomain in $\{-1, 1\}$, then

$$\frac{1 - c(x_i)h(x_i)}{2} = \begin{cases} 0, \text{ if } c(x_i) = h(x_i) \\ 1, \text{ if } c(x_i) \neq h(x_i) \end{cases}$$

and we can rewrite

$$\begin{aligned} \hat{\text{err}}(h) &= \frac{1}{m}\sum_{i=1}^{m}\frac{1 - c(x_i)h(x_i)}{2} = \frac{1}{2m}\sum_{i=1}^{m}\frac{1}{2}(1 - c(x_i)h(x_i)) \\ &= \frac{m}{2m} - \frac{1}{2m}\sum_{i=1}^{m}c(x_i)h(x_i) = \frac{1}{2} - \frac{1}{2m}\sum_{i=1}^{m}c(x_i)h(x_i). \end{aligned}$$

The value $\frac{1}{m}\sum_{i=1}^{m}c(x_i)h(x_i)$ represents the correlation between h and c. That is, if both always return the same value, then the correlation between them is 1. Otherwise, if they always have different answers, then the value of the correlation is -1. Hence, the hypothesis which minimizes the error is the one that maximizes the correlation.

A way of creating a random partition of X is creating Rademacher variables, that is, the set $\sigma = (\sigma_1, ..., \sigma_m)$ with $\Pr(\sigma = 1) = \Pr(\sigma = -1) = 1/2$. The hypothesis h which maximizes

$$\mathbb{E}_\sigma = \mathbb{E}\left[\max_{h \in \mathcal{C}}\frac{1}{m}\sum_{i=1}^{m}\sigma_i h(x_i)\right],$$

for a random set of Rademacher variables is the one that aligns best with σ.

We will consider the extreme cases to bound the values of $\mathbb{E}_\sigma$. If $|\mathcal{C}| = 1$, then

$$\mathbb{E}_\sigma = \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m}\sigma_i h(x_i)\right]$$

and in this case $\frac{1}{m}\sum_{i=1}^{m}h(x_i)\mathbb{E}[\sigma_i] = \frac{1}{m}\sum_{i=1}^{m}h(x_i)0 = 0$. If $\mathcal{C}$ shatters $\{x_1, ..., x_m\}$, that is, $|\mathcal{C}| = 2^m$, then the value of $\mathbb{E}_\sigma$ is 1, since $\mathcal{C}$ always contains a hypothesis such that $h(x_i) = \sigma_i$, for all $1 \leq i \leq m$ and for any set of Rademacher variables σ. Hence, $0 \leq \mathbb{E}_\sigma \leq 1$, and then the most expressive hypotheses are the ones where $\mathbb{E}_\sigma$ is close to 1.

Being more general, consider a set of real-valued functions $\mathcal{F}$ (that in the problem on learning, played the role as the set of hypotheses). Consider that each point of the input is defined to a probability space with distribution π. Consider a sample S and the computation of the maximum deviation of $L_S(f)$ from the true expectation of f, for all $f \in \mathcal{F}$. That is, $\sup_{f \in \mathcal{F}}|L_S(f) - L_X(f)|$.

Suppose that the expressiveness of $\mathcal{F}$ will be measured by only looking on set S. We will split set S into two sets S_1 and S_2 such that, for $\sigma = (\sigma_1, ..., \sigma_m) \in \{\pm 1\}^m$, we have $S_1 = \{x_i : \sigma_i = 1\}$ and $S_2 = \{x_j : \sigma_i = -1\}$. Then we will measure

$$\sup_{f \in \mathcal{F}}(L_{S_1}(f) - L_{S_2}(f)).$$

We have that

$$L_{S_1}(f) - L_{S_2}(f) = \frac{1}{|S_1|}\sum_{x_i \in S_1} f(x_i) - \frac{1}{|S_2|}\sum_{x_j \in S_2} f(x_j).$$

On average, $|S_1| = |S_2| \approx |S|/2$, then

$$L_{S_1}(f) - L_{S_2}(f) \approx \frac{2}{m}\left(\sum_{x_i \in S_1} f(x_i) - \sum_{x_j \in S_2} f(x_j)\right) = \frac{2}{m}\left(\sum_i \sigma_i f(x_i)\right).$$

The *empirical Rademacher average* of $\mathcal{F}$ is defined as follows.

Definition 10. *Consider a sample $S = \{z_1, \ldots, z_r\}$ and a distribution of r Rademacher random variables $\sigma = (\sigma_1, \ldots, \sigma_r)$, i.e. $\Pr(\sigma_i = 1) = \Pr(\sigma_i = -1) = 1/2$ for $1 \leq i \leq r$. The empirical Rademacher average of a family of functions $\mathcal{F}$ w.r.t. to S is defined as*

$$\tilde{R}_r(\mathcal{F}, S) = \mathbb{E}\left[\sup_{f \in \mathcal{F}} \frac{1}{r}\sum_{i=1}^{r} \sigma_i f(z_i)\right].$$

At the heart of the algorithm for the problem presented in Chapter 3, the stopping condition for the progressive sampling depends on the Rademacher average of the sample. For the connection of the empirical Rademacher average with the value of $\sup_{f \in \mathcal{F}} |L_S(f) - L_X(f)|$, we use the bound of Boucheron et al. (2005) and the bound previously introduced by Riondato and Upfal (2018), which extended the bound of Oneto et al. (2013) to the supremum of its absolute value to functions with codomain in $[0, 1]$ and uniform probability distribution on the input.

Theorem 9 (Boucheron et al. (2005), Theorem 3.2)**.** *With probability at least $1 - \delta$,*

$$\sup_{f \in \mathcal{F}} |L_S(f) - L_X(f)| \leq 2\tilde{R}_r(\mathcal{F}, S) + \sqrt{\frac{2\ln(2/\delta)}{r}}.$$

□

Theorem 10 (Oneto et al. (2013), Theorem 3.11)**.** *With probability at least $1 - \delta$,*

$$\sup_{f \in \mathcal{F}} |L_S(f) - L_X(f)| \leq 2\tilde{R}_r(\mathcal{F}, S) + \frac{\ln\frac{3}{\delta} + \sqrt{(\ln\frac{3}{\delta} + 4r\tilde{R}_r(\mathcal{F}, S))\ln\frac{3}{\delta}}}{r} + \sqrt{\frac{\frac{3}{\delta}}{2r}}.$$

□

The exact computation of $\tilde{R}_r(\mathcal{F}, S)$ depends on an extreme value, i.e. the supremum of deviations for all functions in $\mathcal{F}$, which can be expensive and not straightforward to compute over a large (or infinite) set of functions (Mitzenmacher and Upfal, 2017). For this reason, we use the bound described in Theorem 11, which is a variant of the Massart's Lemma (Theorem 14.22 from Mitzenmacher and Upfal (2017)). It is a function that is convex, continuous in $\mathbb{R}_{\geq 0}$, and that can be efficiently minimized by standard convex optimization methods.

Consider the vector $v_f = (f(z_1), \ldots, f(z_r))$ for a given sample of r elements, denoted by $S = \{z_1, \ldots, z_r\}$, and let $\mathcal{V}_S = \{v_f : f \in \mathcal{F}\}$.

Theorem 11 (Riondato and Upfal (2018), Theorem 3.4)**.** *Let* $w : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ *be the function*

$$w(s) = \frac{1}{s} \ln \sum_{v_f \in \mathcal{V}_S} \exp\left(\frac{s^2 \|v_f\|_2^2}{2r^2}\right).$$

Then $\tilde{R}_r(\mathcal{F}, S) \leq \min_{s \in \mathbb{R}_{\geq 0}} w(s)$. □

2.2.5 Related Work on Applications of Sample Complexity, VC dimension, and Rademacher averages

In this section we present a brief overview of results of the contents presented in this section applied to a variety of scenarios.

Regarding $\mathcal{NP}$-Hard problems, the work of Brönnimann and Goodrich (1995) provided approximation algorithms for the hitting set and set cover problems on systems with bounded VC dimension. In the context of learning, Sontag (1998) obtained results on the VC dimension of neural network architectures. In the context of communication complexity, Bhattacharya et al. (2022) used VC dimension on the communication measure of a set system.

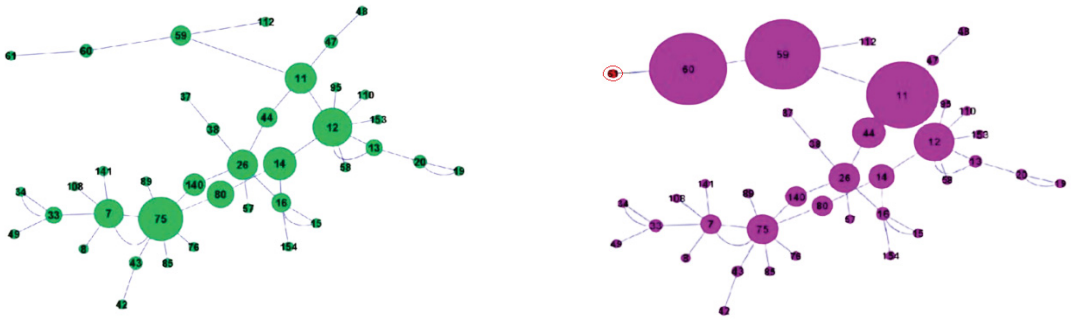
There are some work related to database applications. Benedikt and Libkin (2002) used VC dimension in the context of constraint query languages to get approximations for volume operators. Gross-Amblard (2003) worked on the problem of watermarking of databases or XML documents while preserving queries on a language $\mathcal{L}$. They showed that no watermarking scheme can be obtained if the VC dimension of a range space defined on sets in $\mathcal{L}$ is maximal but not bounded. Riondato et al. (2011) bounded the VC dimension of a range space defined on outcomes of a collection of SQL queries. Riondato and Upfal (2014) designed an approximation algorithm using VC dimension bounds on frequent itemset mining. They were pioneers on using VC dimension in knowledge discovery. They extended these results by using Rademacher averages into a progressive sampling approach (Riondato and Upfal, 2015), being pioneers on using these techniques on a pattern mining problem. Santoro et al. (2020) used VC dimension and Rademacher averages in the modeling of an approximation algorithm to the frequent sequential patterns on massive datasets. Riondato and Vandin (2020) were the first authors to use pseudodimension theory in the field of subgroup discovery. More specifically, in this work they approximate the most interesting subgroups from a random sample of a transactional dataset.

In the context of graph problems, Kranakis et al. (1997) proved results on VC dimension for a variety of problems in range spaces where the domain is the set of vertices of a graph and the ranges are induced by paths, neighborhoods and stars, and connected sets of edges. The work of Kleinberg (2004), Kleinberg et al. (2004), and Gandhi et al. (2010) contains results on the use of VC dimension theory and the ε-net theorem in problems related to security in networks. More specifically, Gandhi et al. (2010) applied these techniques on monitoring sensor networks, and Kleinberg (2004) and Kleinberg et al. (2004) treated the problem of detecting edge failures of a given network. Abraham et al. (2011) bounded the VC dimension of a range space $\mathcal{R} = (X, \mathcal{I})$ where X is the set of vertices of a graph and each range on $\mathcal{I}$ represents an unique shortest path, so the vertices inside in a range are the vertices from its respective shortest path. The authors proved that $\text{VCDim}(\mathcal{R}) \leq 2$. The work by Bousquet et al. (2015) uses

VC dimension to bound the size of identifying codes, i.e. subset of vertices C such that every vertex of the graph is uniquely identified by the set of its neighbors within the C. Riondato and Kornaropoulos (2016) provided an approximation to the betweenness centrality using VC dimension and ε-sample theorem in the modeling of the problem. They also designed a progressive sampling algorithm to the betweenness centrality using pseudo-dimension and Rademacher averages (Riondato and Upfal, 2018), being pioneers on using these techniques in graph mining. Ducoffe et al. (2020) obtained a truly subquadratic time algorithm for the computation of the diameter of some classes of unweighted graphs. They generalize these classes as the graphs with constant *distance VC dimension*, that is, the VC dimension of the ball hypergraph from a graph G. They proposed a randomized algorithm using ε-net construction that either computes the diameter of graph G, if its distance VC dimension is at most d, or concludes that it is larger than a fixed value k.

3 PERCOLATION CENTRALITY PROBLEM

The importance of a vertex in a graph can be quantified using centrality measures. In this chapter we deal with the *percolation centrality*, a measure relevant in applications where graphs are used to model a contagious process in a network (e.g. disease transmission or misinformation spreading). Centrality measures can be defined in terms of local properties, such as the vertex degree, or global properties, such as the betweenness centrality or the percolation centrality. The betweenness centrality of a vertex v (Freeman, 1977), roughly speaking, is the fraction of shortest paths containing v as an intermediate vertex. The percolation centrality generalizes the betweenness centrality by allowing weights on the shortest paths, and the weight of a shortest path depends on the disparity between the degree of contamination of the two ending vertices of such path (Figure 3.1).



(a) Betweenness centrality

(b) Percolation centrality

Figure 3.1: Comparison between the values of betweenness and percolation centralities of the Alberta network (Piraveenan et al., 2013). The vertices sizes match the centrality values. In b), the red vertex corresponds to an infected vertex.

The study of the percolation phenomenon in a physical system was introduced by Broadbent and Hammersley (1957) in the context of the passage of a fluid in a medium. In graphs, percolation centrality was proposed by Piraveenan et al. (2013), where the medium are the vertices of a graph G and each vertex v in G has a *percolation state* (reflecting the *degree of contamination* of v). The percolation centrality of v is a function that depends on the topological connectivity and the states of the vertices of G. The best-known algorithms that exactly compute the betweenness centrality for all vertices of a graph depends on computing all its shortest paths (Riondato and Kornaropoulos, 2016) and, consequently, the same applies in the computation of percolation centrality. The fastest algorithm for this task for weighted graphs, proposed by Williams (2014), runs in $\mathcal{O}\left(n^3/2^{c\sqrt{\log n}}\right)$ time, for some constant c. Currently it is a central open problem in graph theory whether this problem can be solved in $\mathcal{O}(n^{3-c})$, for some $c > 0$, and the hypothesis that there is no such algorithm is used in hardness arguments in some works (Abboud and Williams, 2014; Abboud et al., 2018). In the particular case of sparse graphs, which are common in many applications, the complexity of the exact computation for the betweenness centrality can be improved to $\mathcal{O}(n^2)$. However, the same is not known to be the true for percolation centrality and no subcubic-time algorithm is known even in such restricted scenario.

The main contributions of this chapter are approximation algorithms to estimate the percolation centrality of all vertices of a graph. A main theme we deal with is the fact that for large scale graphs, even algorithms with time complexity that scales in quadratic-time are inefficient in practice, and high-quality approximations obtained with high confidence are usually sufficient in real-world applications. Riondato and Upfal (2018) observed that keeping track of the exact centrality values, which may change continuously, provides little information gain. So, the idea is to sample a subset of all shortest paths in the graph so that, for any fixed constants $0 < \varepsilon, \delta < 1$, they obtain values within ε from the exact value with probability $1 - \delta$.

In this chapter we describe techniques based on Pseudo-dimension theory applied to percolation centrality (Lima et al., 2022a) and on Rademacher Averages applied to betweenness centrality (Riondato and Upfal, 2018). We show that this combination can be further developed for giving an approximation algorithm for the percolation centrality based on a progressive sampling strategy. The idea is that the algorithm iteratively increases the size of a sample of shortest paths used for estimating the percolation centrality until the desired accuracy is achieved. The stop condition depends on the Rademacher Averages of the current sample of shortest paths. One of the consequences of the approach based on Rademacher Averages is that such technique is sensitive to the input distribution, so it can provide tighter bounds for certain inputs. Additionally, even if no assumption is made on the input distribution, we show that with the use of pseudo-dimension theory on the sample analysis we can obtain a sample size that is tighter than the one given by standard Hoeffding and union-bound techniques, and never worse than the sample size given by the fixed-size sample algorithm.

We have in mind both a theoretical and a practical perspective. More precisely, we show that the estimation of the percolation centrality can be computed in $\mathcal{O}(m \log n \log \mathrm{diam}_V(G))$ expected time, where $\mathrm{diam}_V(G)$ is the maximum number of vertices in a shortest path of G. Note that since many real-world graphs are sparse and have logarithmic diameter, the time complexity of the algorithm for such graphs is $\mathcal{O}(n \log n \log \log n)$. In the practical front, in Section 3.5, we give the relation between the quality and confidence constants and the sample size required for meeting the approximation guarantee and, in fact, our experimental evaluation shows that our algorithms produce results that are orders of magnitude better than the guarantees given by the referred theoretical analysis.

We note that Riondato and Upfal (2018) use pseudo-dimension theory for the betweenness problem in order to make use of Rademacher averages. In both of our approximation algorithms to the percolation centrality (the one using a fixed-size sample and the one using progressive sampling), we need pseudo-dimension theory by the very nature of the problem, since percolation functions are real-valued and VC dimension does not apply in our scenario.

This chapter is organized as follows: Section 3.1 states the definitions and notation used; in Section 3.2 we model the problem in terms of a range space; in Section 3.3 we compute the bound for its pseudo-dimension and we describe the fixed-size sample and the progressive sampling algorithms to estimate the percolation centrality of every vertex $v \in G$; in Section 3.4 we prove the correctness and running time of our proposed algorithms; in Section 3.5 we present the results of the experimental evaluation of our algorithms.

3.1 PRELIMINARIES

Given a directed weighted graph $G = (V, E)$ and the percolation states $0 \leq x_v \leq 1$ for each $v \in V$, we say a vertex v is *fully percolated* if $x_v = 1$, *non-percolated* if $x_v = 0$, and *partially percolated* if $0 < x < 1$. We say that a path from u to w is *percolated* if $x_u - x_w > 0$. Let $\sigma_{uw} = |\mathcal{C}_{uw}|$, recalling that $\mathcal{C}_{uw}$ is the set of all shortest paths from u to w. We denote $\sigma_{uw}(v)$ as the number of shortest paths from u to w that a vertex v is internal to. The percolation centrality is defined below.

Definition 11 (Percolation Centrality)**.** *Let* $R(x) = \max\{x, 0\}$. *Given a graph* $G = (V, E)$ *and percolation states* $x_v, \forall v \in V$, *the* percolation centrality *of a vertex* $v \in V$ *is defined as*

$$p(v) = \frac{1}{n(n-1)} \sum_{\substack{(u,w)\in V^2 \\ u\neq v\neq w}} \frac{\sigma_{uw}(v)}{\sigma_{uw}} \frac{R(x_u - x_w)}{\sum\limits_{\substack{(f,d)\in V^2 \\ f\neq v\neq d}} R(x_f - x_d)}.$$

The definition originally presented by Piraveenan et al. (2013) does not have the normalization factor $\frac{1}{n(n-1)}$, introduced in this thesis with the purpose of defining a proper probability distribution in Section 3.2. This normalization preserves the original relation among the vertices centralities.

3.2 PSEUDO–DIMENSION AND PERCOLATED SHORTEST PATHS

In this section we model the percolation centrality estimation problem in terms of a range set of the percolated shortest paths. We first describe the range space defined for the fixed-size sample algorithm (Section 3.2.1), where the points of the domain are shortest paths. Next, we present a range space where the domain corresponds to the pairs of vertices of G (Section 3.2.2). This modification is necessary in the progressive sampling algorithm.

3.2.1 Range Space defined for the Fixed-Size Sample Algorithm

In this section we model the percolation centrality in terms of a range space of the percolated shortest paths. For a given graph $G = (V, E)$ and the percolation states x_v for each $v \in V$, let $X = S_G$, with $n = |V|$, where

$$S_G = \bigcup_{\substack{(u,w)\in V^2 \\ u\neq w}} \mathcal{C}_{uw}.$$

For each $v \in V$, there is a set $\tau_v = \{p \in X : v \in \text{Inn}(p)\}$. For a pair $(u, w) \in V^2$ and a path $p_{uw} \in X$, let $f_v : X \to [0, 1]$ be the function

$$f_v(p_{uw}) = \frac{R(x_u - x_w)}{\sum\limits_{\substack{(f,d)\in V^2 \\ f\neq v\neq d}} R(x_f - x_d)} \mathbb{1}_{\tau_v}(p_{uw}),$$

where $\mathbb{1}_{\tau_v}(p_{uw})$ is the indicator function that returns 1 if $v \in \text{Inn}(p_{uw})$ — and hence, p_{uw} is in the interval τ_v of vertex v — and 0 otherwise. The function f_v gives the proportion

of the percolation between u and w to the total percolation in the graph if $v \in \mathrm{Inn}(p_{uw})$. We define $\mathcal{F} = \{f_v : v \in V\}$.

Let $D = X \times [0,1]$. For each $f_v \in \mathcal{F}$, there is a range

$$R_v = R_{f_v} = \{(p_{uw}, t) : p_{uw} \in X \text{ and } t \leq f_v(p_{uw}).\}$$

Note that each range R_v contains the pairs (p_{uw}, t), where $0 < t \leq 1$ such that $v \in \mathrm{Inn}(p_{uw})$ and

$$t \leq \frac{R(x_u - x_w)}{\sum\limits_{\substack{(f,d)\in V^2 \\ f\neq v\neq d}} R(x_f - x_d)}.$$

We define $\mathcal{F}^+ = \{R_v : f_v \in \mathcal{F}\}$.

Each $p_{uw} \in X$ is sampled according to the function $\pi(p_{uw}) = \frac{1}{n(n-1)}\frac{1}{\sigma_{uw}}$. In order to see that this is a valid probability distribution, note that

$$\begin{aligned}\sum_{p_{uw}\in U} \pi(p_{uw}) &= \sum_{p_{uw}\in U} \frac{1}{n(n-1)}\frac{1}{\sigma_{uw}} = \sum_{u\in V}\sum_{\substack{w\in V \\ w\neq u}}\sum_{p\in\mathcal{C}_{uw}} \frac{1}{n(n-1)}\frac{1}{\sigma_{uw}} = \sum_{u\in V}\sum_{\substack{w\in V \\ w\neq u}} \frac{1}{n(n-1)}\frac{\sigma_{uw}}{\sigma_{uw}} \\ &= \frac{1}{n(n-1)}\sum_{u\in V}\sum_{\substack{w\in V \\ w\neq u}} 1 = \frac{1}{n(n-1)}\sum_{u\in V}(n-1) = 1\,.\end{aligned}$$

We state in the next theorem that $\mathbb{E}[f_v(p_{uw})] = p(v)$ for all $v \in V$.

Theorem 12. *For $f_v \in \mathcal{F}$ and for all $p_{uw} \in X$, such that each p_{uw} is sampled according to the probability function $\pi(p_{uw})$, we have $\mathbb{E}[f_v(p_{uw})] = p(v)$.*

Proof. For a given graph $G = (V, E)$ and for all $v \in V$, we have from Definition 7

$$\begin{aligned}L_X(f_v) = \mathbb{E}_{p_{uw}\sim\pi}[f_v(p_{uw})] &= \sum_{p_{uw}\in X} \pi(p_{uw}) f_v(p_{uw}) \\ &= \sum_{p_{uw}\in X} \frac{1}{n(n-1)}\frac{1}{\sigma_{uw}} \frac{R(x_u - x_w)}{\sum\limits_{\substack{(f,d)\in V^2 \\ f\neq v\neq d}} R(x_f - x_d)} \mathbb{1}_{\tau_v}(p_{uw}) \\ &= \frac{1}{n(n-1)} \sum_{\substack{u\in V \\ u\neq v}} \sum_{\substack{w\in V \\ w\neq v\neq u}} \sum_{p\in\mathcal{C}_{uw}} \frac{1}{\sigma_{uw}} \frac{R(x_u - x_w)}{\sum\limits_{\substack{(f,d)\in V^2 \\ f\neq v\neq d}} R(x_f - x_d)} \mathbb{1}_{\tau_v}(p) \\ &= \frac{1}{n(n-1)} \sum_{\substack{u\in V \\ u\neq v}} \sum_{\substack{w\in V \\ w\neq v\neq u}} \frac{\sigma_{uw}(v)}{\sigma_{uw}} \frac{R(x_u - x_w)}{\sum\limits_{\substack{(f,d)\in V^2 \\ f\neq v\neq d}} R(x_f - x_d)} \\ &= \frac{1}{n(n-1)} \sum_{\substack{(u,w)\in V^2 \\ u\neq v\neq w}} \frac{\sigma_{uw}(v)}{\sigma_{uw}} \frac{R(x_u - x_w)}{\sum\limits_{\substack{(f,d)\in V^2 \\ f\neq v\neq d}} R(x_f - x_d)} = p(v).\end{aligned}$$

□

Let $S = \{p_{u_1w_1}, \ldots, p_{u_rw_r}\}$ be a collection of r shortest paths sampled independently from X. Next, we define $\tilde{p}(v)$, the estimation to be computed, as the empirical average from Definition 7:

$$\tilde{p}(v) = L_S(f_v) = \frac{1}{r} \sum_{p_{u_i w_i} \in S} f_v(p_{u_i w_i}) = \frac{1}{r} \sum_{p_{u_i w_i} \in S} \frac{R(x_{u_i} - x_{w_i})}{\sum\limits_{\substack{(f,d) \in V^2 \\ f \neq v \neq d}} R(x_f - x_d)} \mathbb{1}_{\tau_v}(p_{u_i w_i}).$$

3.2.2 Range Space defined for the Progressive Sampling Algorithm

In this section we describe a modification in the range space so that we can use the bound of Riondato and Upfal (2018) in our progressive sampling algorithm, since an uniform probability distribution in the points of the domain is required in this case.

For a given graph $G = (V, E)$ and the percolation states x_v, for each $v \in V$, let $X' = V \times V$. Let $f_v : X' \to [0, 1]$ be the function

$$f_v(u, w) = \frac{R(x_u - x_w)}{\sum\limits_{\substack{(f,d) \in V \times V \\ u \neq w}} R(x_f - x_d)} \frac{\sigma_{uw}(v)}{\sigma_{uw}}.$$

We define $\mathcal{F}' = \{f_v : v \in V\}$. There is one range

$$R_v = \{((u, w), t) : (u, w) \in X' \text{ and } t \leq f_v(u, w)\}$$

for each $f_v \in \mathcal{F}'$. We define $\mathcal{F}'^{+} = \{R_v : f_v \in \mathcal{F}'\}$. Each $(u, w) \in X'$ is sampled with probability $\pi(u, w) = \frac{1}{n(n-1)}$, which is a valid probability distribution.

For a collection $S' = \{u_1 w_1, \ldots, u_r w_r\}$ of r pairs of vertices sampled independently and identically from X', the estimation $\tilde{p}(v)$ to be computed is the empirical average of f_v, according to Definition 7:

$$\tilde{p}(v) = L_{S'}(f_v) = \frac{1}{r} \sum_{(u_i, w_i) \in S'} f_v(u_i, w_i) = \frac{1}{r} \sum_{(u_i, w_i) \in S'} \frac{R(x_{u_i} - x_{w_i})}{\sum\limits_{\substack{(f,d) \in V \times V \\ f \neq v \neq d}} R(x_f - x_d)} \frac{\sigma_{uw}(v)}{\sigma_{uw}}.$$

For each $v \in V$, the value $\tilde{p}(v)$ can be defined as $\|\mathbf{v}_v\|/r$, where

$$\mathbf{v}_v = (f_v(u_1, w_1), \ldots, f_v(u_r, w_r)).$$

We denote the set $\mathcal{V} = \{\mathbf{v}_v : v \in V\}$. Note that $|\mathcal{V}| \leq |V|$, since there may be different vertices u' and v' with $\mathbf{v}_{u'} = \mathbf{v}_{v'}$.

3.3 APPROXIMATION TO THE PERCOLATION CENTRALITY

In this section we present sampling algorithms for approximating the percolation centrality of every vertex of a graph using either a fixed-size sample approach (Section 3.3.1) or a progressive sampling approach (Section 3.3.2). The correctness and running time of the algorithms rely on the sample size given by Theorems 7 and 9. We prove an upper bound to the VC dimension of the range space $\mathcal{R}$ in Theorem 13 in order to bound the sample size. We are aware that the main idea in the proof is similar to the proof of a result for a different range space on the shortest paths obtained in the work

of Riondato and Kornaropoulos (2016) using VC dimension. For the sake of clarity, instead of trying to fit their definition to our model and use their result, we found it easier stating and proving the theorem directly for our range space.

Theorem 13. *Let $\mathcal{R} = (X, \mathcal{F})$ and $\mathcal{R}' = (D, \mathcal{F}^+)$ be the corresponding range spaces for the domain and range sets defined in Section 3.2, and let $diam_V(G)$ be the vertex-diameter of G. We have*

$$PD(\mathcal{R}) = VCDim(\mathcal{R}') \leq \lfloor \lg(diam_V(G) - 2) \rfloor + 1.$$

Proof. Let $\mathrm{VCDim}(\mathcal{R}') = k$. Recall that $\mathrm{VCDim}(\mathcal{R}')$ must be finite, i.e. $k \in \mathbb{N}$, because X and $\mathcal{F}$ are finite. Then, there is $S \subseteq D$ such that $|S| = k$ and S is shattered by $\mathcal{F}^+$. From Lemmas 1 and 2, we know that for each $p_{uw} \in X$, there is at most one pair (p_{uw}, t) in S for some $t \in (0, 1]$ and there is no pair in the form $(p_{uw}, 0)$. By the definition of shattering, each $(p_{uw}, t) \in S$ must appear in 2^{k-1} different ranges in $\mathcal{F}^+$. On the other hand, each pair (p_{uw}, t) is in at most $|p_{uw}| - 2$ ranges in $\mathcal{F}^+$, since $(p_{uw}, t) \notin R_v$ either when $t > f_v(p_{uw})$ or $v \notin \mathrm{Inn}(p_{uw})$. Considering that $|p_{uw}| - 2 \leq \mathrm{diam}_V(G) - 2$, we have

$$\begin{aligned} 2^{k-1} &\leq |p_{uw}| - 2 \leq \mathrm{diam}_V(G) - 2 \\ k - 1 &\leq \lg(\mathrm{diam}_V(G) - 2). \end{aligned}$$

Since k must be integer, $k \leq \lfloor \lg(\mathrm{diam}_V(G) - 2) \rfloor + 1 \leq \lg(\mathrm{diam}_V(G) - 2) + 1$. Finally,

$$\mathrm{PD}(\mathcal{F}) = \mathrm{VCDim}(\mathcal{F}^+) = k \leq \lfloor \lg(\mathrm{diam}_V(G) - 2) \rfloor + 1 \,. \qquad \square$$

By Theorem 12 and Definition 7, $L_X(f_v) = p(v)$ and $L_S(f_v) = \tilde{p}(v)$, respectively, for each $v \in V$ and $f_v \in \mathcal{F}$. Thus, $|L_S(f_v) - L_X(f_v)| = |\tilde{p}(v) - p(v)|$, and by Theorems 7 and 13, a sample of size $\left\lceil \frac{c}{\varepsilon^2} \left(\lfloor \lg(\mathrm{diam}_V(G) - 2) \rfloor + 1 + \ln \frac{1}{\delta} \right) \right\rceil$ suffices to our algorithm, for given constants $0 < \varepsilon, \delta < 1$.

If we had used a Hoeffding bound, we would have

$$\Pr(|\tilde{p}(v) - p(v)| \geq \varepsilon) \leq 2\exp(-2r\varepsilon^2)$$

for a sample of size r and for each $v \in V$. Applying the union bound for all $v \in V$, the value of r must satisfy $2\exp(-2r^2\varepsilon^2)n \geq \delta$, which leads to $r \leq \frac{1}{2\varepsilon^2}(\ln 2 + \ln n + \ln(1/\delta))$. Even though $\mathrm{diam}_V(G)$ might be as large as n, we note that the bound given in Theorem 13 is tighter since it depends on the combinatorial structure of G, which gives a sample size tailored for it. For instance, if $\mathrm{diam}_V(G) = \ln n$ (which is common in many real-world graphs, in particular power-law graphs), we have that $\mathrm{VCDim}(\mathcal{R}) \leq \lfloor \lg(\ln n - 2) \rfloor + 1$.

In particular, the problem of computing the diameter of G is not known to be easier than the problem of computing all of its shortest paths (Aingworth et al., 1996), so obtaining an exact value for the diameter would defeat the whole purpose of using a sampling strategy that avoids computing all shortest paths. However, a bound on $\mathrm{diam}_V(G)$ given by a folklore 2–approximation on unidirected graphs is enough and it can be efficiently computed (Riondato and Upfal, 2018). If G is directed, we use this strategy in the underlying undirected graph. Note that this method does not guarantee the desired approximation factor in the directed case, however, in practice the algorithm gives very tight bounds as we show in our experimental evaluation in Section 3.5. We also observe that the diameter can be approximated within smaller factors, but even for a $(\frac{3}{2}, \frac{3}{2})$-approximation algorithm (Aingworth et al., 1996), i.e. an algorithm that outputs

a solution of size at most $\frac{3}{2} \cdot \text{diam}(G) + \frac{3}{2}$, the complexity is $\tilde{O}(m\sqrt{n} + n^2)$, what would also be a bottleneck to our algorithm. Furthermore, since in our case we do not need the largest shortest path, but simply the value of the diameter, and we take logarithm of this value, the 2-approximation is sufficient.

3.3.1 Fixed–size sample algorithm

Given a directed weighted graph $G = (V, E)$ and the percolation states x_v, for each $v \in V$, as well as the quality and confidence parameters $0 < \varepsilon, \delta < 1$, assumed to be constants (they do not depend on the size of G, respectively), Algorithm 2 works as follows. At the beginning of the execution the approximated value $\text{diam}_V(G)$ for the vertex-diameter of G is obtained, in line 1, by a folklore 2-approximation described by Riondato and Kornaropoulos (2016), if the graph is undirected, as previously mentioned. If the input graph is directed, we can use the same approximation algorithm, ignoring the edges directions. According to Theorem 13, this value is used to determine the sample size, denoted by r, in line 2.

The value $\text{minus\_sum}[v] = \sum_{(f,d)\in V^2 : f \neq v \neq d} R(x_f - x_d)$, for each vertex $v \in V$, which is necessary to compute $\tilde{p}(v)$, is obtained in line 4 by the linear-time dynamic programming strategy presented in Algorithm 1. The correctness of Algorithm 1 is not self-evident, so we provide a proof of its correctness in Theorem 14.

A pair $(u, w) \in V^2$ is sampled uniformly and independently, and then a shortest path p_{uw} between u and w is sampled independently in $\mathcal{C}_{uw}$ in lines 11–15. For a vertex $z \in \text{Inn}(p_{uw})$, the value $(1/r) \cdot (R(x_u - x_w)/\text{minus\_sum}[z])$ is added to $\tilde{p}(z)$.

Algorithm 1 getPercolationDifferences(A, n)

Input: Array A, sorted in non-decreasing order, and $n = |A|$.
Output: The value $\text{sum} = \sum_{i=1}^{n} \sum_{j=1}^{n} R(A[j] - A[i])$ and the array $\left\{\text{minus\_sum}[k] = \sum_{i=1, i\neq k}^{n} \sum_{j=1, j\neq k}^{n} R(A[j] - A[i]) : k \in \{1, ..., n\}\right\}$, such that $R(z) = \max\{z, 0\}$.

```
1: sum ← 0
2: minus_sum[i] ← 0, ∀i ∈ {1,...,n}
3: svp ← (0, 0, ..., 0)
4: for i ← 2 to n do
5:     svp[i] ← svp[i − 1] + A[i − 1]
6:     sum ← sum + (i − 1)A[i] − svp[i]
7: svp[n + 1] ← svp[n] + A[n]
8: for i ← 1 to n do
9:     minus_sum[i] ← sum − A[i](2i − n − 2) − svp[n + 1] + 2svp[i]
10: return sum, minus_sum
```

Theorem 14. *For an array A of size n, sorted in non-decreasing order, Algorithm 1 returns for sum and minus_sum$[k]$, respectively, the values $\sum_{i=1}^{n} \sum_{j=1}^{n} R(A[j]-A[i])$ and $\sum_{\substack{i=1 \\ i\neq k}}^{n} \sum_{\substack{j=1 \\ j\neq k}}^{n} R(A[j]-A[i])$, for each $k \in \{1, \ldots, n\}$.*

Proof. By the definition of sum, we have that

$$\text{sum} = \sum_{i=1}^{n} \sum_{j=1}^{n} R(A[i] - A[j]) = \sum_{i=1}^{n} \sum_{j=1}^{n} R(A[j] - A[i]) = \sum_{i=1}^{n} \sum_{j=1}^{n} \max\{A[j] - A[i], 0\}.$$

Since A is sorted, then $\max\{A[j]-A[i],0\}=0$ if $j<i$. Hence, if we consider only the values $j\geq i$, then sum $=\sum_{i=1}^{n}\sum_{j=i}^{n}(A[j]-A[i])$.

A similar step can be applied to the values of the array minus_sum, and then for all indices $k\in\{1,\ldots,n\}$,

$$\text{minus\_sum}[k]=\sum_{\substack{i=1\\ i\neq k}}^{n}\sum_{\substack{j=1\\ j\neq k}}^{n}\max\{A[j]-A[i],0\}=\sum_{\substack{i=1\\ i\neq k}}^{n}\sum_{\substack{j=i\\ j\neq k}}^{n}(A[j]-A[i]).$$

The recurrences below follow directly from lines 5 and 6, where sum_k denotes the value of sum at the beginning of the k–th iteration of the algorithm.

$$\text{svp}[k]=\begin{cases}0, & \text{if } k=1\\ \text{svp}[k-1]+A[k-1], & \text{otherwise.}\end{cases}$$

$$\text{sum}_k=\begin{cases}0, & \text{if } k=1\\ \text{sum}_{k-1}+(k-1)A[k]-\text{svp}[k], & \text{otherwise.}\end{cases}$$

The solutions to the above recurrences are, respectively,

$$\text{svp}[k]=\sum_{i=1}^{k-1}A[i]\quad\text{and}\quad sum_k=\sum_{i=1}^{k}((i-1)A[i]-\text{svp}[i]).$$

The value sum is then correctly computed in lines 4–6, since

$$\begin{aligned}\text{sum}&=\sum_{i=1}^{n}\sum_{j=i}^{n}(A[j]-A[i])=\sum_{i=1}^{n}\sum_{j=i}^{n}A[j]-\sum_{i=1}^{n}\sum_{j=i}^{n}A[i]\\
&=\sum_{i=1}^{n}\sum_{j=i}^{n}A[j]-\sum_{i=1}^{n}(n-i+1)A[i]\\
&=\sum_{j=1}^{n}\sum_{i=1}^{j}A[j]-\sum_{i=1}^{n}(n-i+1)A[i]\\
&=\sum_{j=1}^{n}jA[j]-\sum_{i=1}^{n}(n-i+1)A[i]=\sum_{i=1}^{n}iA[i]-\sum_{i=1}^{n}(n-i+1)A[i]\\
&=\sum_{i=1}^{n}(i-1)A[i]-\sum_{i=1}^{n}(n-i)A[i]=\sum_{i=1}^{n}(i-1)A[i]-\sum_{i=1}^{n}\sum_{j=1}^{i-1}A[j]\\
&=\sum_{i=1}^{n}\left((i-1)A[i]-\sum_{j=1}^{i-1}A[j]\right)=\sum_{i=1}^{n}((i-1)A[i]-\text{svp}[i]).\end{aligned}$$

Finally, minus_sum is also correctly computed in lines 8 and 9, since

$$\begin{aligned}
\text{minus_sum}[k] &= \sum_{\substack{i=1\\ i\neq k}}^{n}\sum_{\substack{j=i\\ j\neq k}}^{n}(A[j]-A[i]) = \sum_{i=1}^{n}\sum_{j=i}^{n}(A[j]-A[i]) \\
&-\left(\sum_{j=1}^{k-1}(A[k]-A[j]) + \sum_{j=k+1}^{n}(A[j]-A[k])\right) \\
&= \text{sum} - \left(\sum_{j=1}^{k-1}A[k] - \sum_{j=k+1}^{n}A[k] - \sum_{j=1}^{k-1}A[j] + \sum_{j=k+1}^{n}A[j]\right) \\
&= \text{sum} - \left((k-1)A[k] - (n-(k+1)+1)A[k] - \sum_{j=1}^{k-1}A[j] + \sum_{j=k+1}^{n}A[j]\right) \\
&= \text{sum} - \left((2k-n-1)A[k] + \sum_{j=1}^{n}A[j] - \sum_{j=1}^{k-1}A[j] - A[k] - \sum_{j=1}^{k-1}A[j]\right) \\
&= \text{sum} - \left((2k-n-2)A[k] + \sum_{j=1}^{n}A[j] - 2\sum_{j=1}^{k-1}A[j]\right) \\
&= \text{sum} - (2k-n-2)A[k] - \text{svp}[n+1] + 2\text{svp}[k].
\end{aligned}$$

□

The main idea of the fixed-size sample algorithm described in Algorithm 2 is shown below.

step 1. Sample a pair of vertices $(u,w) \in V \times V$ uniformly and independently (with replacement) at random;

step 2. Compute the set $\mathcal{C}_{uw}$ of shortest paths from u to w;

step 3. Sample a shortest path $p_{uw} \in \mathcal{C}_{uw}$ independently with probability $1/\sigma_{uw}$. In this step, start a backward traversing from w as follows. Do $t \leftarrow w$, and while $t \neq u$, sample a predecessor z of t with probability σ_{uz}/σ_{ut}; increase the estimation for the percolation centrality $\tilde{p}(z)$ by $\frac{1}{r}\frac{R(x_u - x_w)}{\sum_{(f,d)\in V\times V: f\neq v\neq d} R(x_d - x_f)}$, and do $t \leftarrow z$;

step 4. Repeat the steps above r times, where $r = \frac{c}{\varepsilon^2}\left(\lfloor \lg(\text{diam}_V(G)-2)\rfloor + 1 + \ln\frac{1}{\delta}\right)$;

step 5. Return the set $\{\tilde{p}(v) : \forall v \in V\}$.

Algorithm 2 PercolationCentralityApproximation(G,x,ε,δ)

Input: Graph $G = (V, E)$ with $n = |V|$, percolation states $\boldsymbol{x}$, accuracy constant $0 < \varepsilon < 1$, confidence constant $0 < \delta < 1$.
Output: Approximation $\tilde{p}(v)$ for the percolation centrality of all vertices $v \in V$.

```
diam_V(G) ← getVertexDiameter(G)
r ← ⌈(c/ε²)(⌊lg diam_V(G) − 2⌋ + 1 − ln δ)⌉
sort x    // after sorted, x = (x_1, x_2, ..., x_n)
minus_sum ← getPercolationDifferences(x, n)
for i ← 1 to r do
    sample u ∈ V with probability 1/n
    sample w ∈ V with probability 1/(n − 1)
    C_uw ← allShortestPaths(u, w)
    if C_uw ≠ ∅ then
        t ← w
        while t ≠ u do
            sample z ∈ P_u(t) with probability σ_uz / σ_ut
            if z ≠ u then
                p̃[z] ← p̃[z] + (1/r) · R(x_u − x_w) / minus_sum[z]
            t ← z
return p̃[v], ∀v ∈ V
```

3.3.2 Progressive Sampling Algorithm

When comparing the fixed-size sample algorithm with a progressive sampling approach, we can build an algorithm using the range space defined by $\mathcal{R} = (X, \mathcal{F})$ in Section 3.2 and the bound in Theorem 3.2 of Boucheron et al. (2005) as the stopping condition for sampling. However, the corresponding initial size to the sample schedule obtained by this bound is greater than the value given by Theorems 7 and 13 for a fixed-size sample approach, so we use the range space defined by $\mathcal{R}' = (X', \mathcal{F}')$ in Section 3.2.2, which has an uniform probability distribution on the domain X'. We note that if there is only one shortest path between any pair of vertices, then we can guarantee $\mathrm{PD}(\mathcal{R}') \leq \lfloor \lg(\mathrm{diam}_V(G) - 2) \rfloor + 1$; otherwise, the Pseudo–dimension of D is the same as the one obtained by Hoeffding and union bounds (as proven in Lemma 4.5 of Riondato and Upfal (2018)). Despite this issue, we show in the experimental evaluation that in practice, this range space can lead to improvements on the running time of an approximation algorithm that uses a progressive sampling schedule.

The schedule is defined as follows. Let S_1 be the initial sample size and $\delta_1 = \delta/2$. At this point, the only information available about the empirical Rademacher average of S_1 is that $\tilde{R}_r(\mathcal{F}', S_1) \geq 0$. Plugging this with the r.h.s. of the bound in Theorem 10, which has to be at most ε, we have

$$\frac{2\ln(6/\delta)}{|S_1|} + \sqrt{\frac{\ln(6/\delta)}{2|S_1|}} \leq \varepsilon$$

$$\frac{4\ln^2(6/\delta)}{|S_1|^2} - \frac{4\ln(6/\delta)\varepsilon}{|S_1|} + \varepsilon^2 \leq \frac{\ln(6/\delta)}{2|S_1|},$$

giving

$$|S_1| \geq \frac{(1 + 8\varepsilon + \sqrt{1 + 16\varepsilon})\ln(6/\delta)}{4\varepsilon^2} \quad .$$

There is no fixed strategy for scheduling. Provost et al. (1999) conjecture that a geometric sampling schedule is optimal (although we do not need such assumption), i.e. the one that $S_i = g^i S_1$, for each $i \geq 1$ and for constant $g > 1$. In our algorithm we follow this result, as previously indicated in (Riondato and Upfal, 2018).

Given $0 < \varepsilon, \delta < 1$, let $(|S_i|)_{i \geq 1}$ be a geometric sampling schedule with starting sample size defined above. We present the outline of the progressive sampling algorithm for estimating the percolation centrality with probability $1 - \delta$. Consider the table $\tilde{p}$ with the centrality estimation.

The following steps are repeated for each $i \geq 1$, summarizing the behavior of Algorithm 3. For the sake of clarity, $S_0 = \emptyset$.

step 1. Create a sample of $k = |S_i| - |S_{i-1}|$ elements of $V \times V$ chosen uniformly and independently (with replacement) at random;

step 2. For each pair of vertices $(u, w) \in \{S_i - S_{i-1}\}$, compute the set $\mathcal{C}_{uw}$ of shortest paths from u to w. Let z be an inner vertex of some shortest path between u and w. Increase the value $\tilde{p}(z)$ by $\frac{R(x_u - x_w)}{\sum_{\substack{(f,d) \in V \times V \\ f \neq v \neq d}} R(x_d - x_f)} \frac{\sigma_{uw}(z)}{\sigma_{uw}}$;

step 3. Compute the bound to $\tilde{R}_r(\mathcal{F}', S_i)$ by minimizing the function defined in Theorem 11. If it satisfies the stopping condition defined in Theorem 9, then return the set $\{\tilde{p}(v)/|S_i| : \forall v \in V\}$. Otherwise, increase the size of S_i until it has size $|S_{i+1}|$, increase i, and return to step 1.

Step 1 is trivial. Step 2 requires the computation and update of inner vertices to some shortest path from u to w and the computation of minus_sum$[v] = \sum_{(f,d) \in V^2 : f \neq v \neq d} R(x_f - x_d)$ for each $v \in V$. The former task can be computed in time $\mathcal{O}(m + n \log n)$ following the steps of Riondato and Upfal (2018): an adaptation on Dijkstra's algorithm in $G = (V, E)$, for each sampled pair of vertices (u, w), is executed for the computation of shortest paths. The adaptation, discussed in Lemma 3 of Brandes (2001), works as follows. Let z be an inner vertex of some shortest path from u to w. The modified Dijkstra stores the distance $d(u, z)$ from u to z in a shortest path from u to w. After $\mathcal{C}_{uw}$ is computed, the set of inner vertices in some path $p \in \mathcal{C}_{uw}$ is sorted in inverse order of $d(u, z)$. The value $\sigma_{uw}(z)$ corresponds to $\sigma_{uz}\sigma_{zw}$, where σ_{uz} is returned by the modified Dijkstra's algorithm and $\sigma_{zw} = \sum_{y : z \in P_u(y)} \sigma_{yw}$, where $P_u(w)$ is the set of immediate predecessors of u in a shortest path from u to w.

In the calculation of minus_sum$[v] = \sum_{(f,d) \in V^2 : f \neq v \neq d} R(x_f - x_d)$, for each $v \in V$, which are necessary to compute $\tilde{p}(v)$, we perform the linear-time dynamic programming strategy presented in Algorithm 1.

On Step 3, let $S_i = \{(u_1, w_1), \ldots, (u_j, w_j)\}$ be the sample of size j obtained after the execution of the i-th iteration of the progressive sampling algorithm and let $\mathbf{v}_v$ be the vector $\mathbf{v}_v = \left(f_v(u_1, w_1), \ldots, f_v(u_j, w_j)\right)$, for all $v \in V$. The ℓ_1 and ℓ_2 norms of each $\mathbf{v}_v$ are stored on the hash tables $\mathcal{V}_1$ and $\mathcal{V}_2$, respectively. The set $\mathcal{V}$, represented as a hash table,

keeps the values of $\mathcal{V}_2$ with no repetition to the computation of ω_s in line 21, which is the bound for the empirical Rademacher average of S_i obtained by the function defined in Theorem 11. For each inner vertex v to be updated, Algorithm 4 checks if the value associated to $\mathcal{V}_2[v]$ in $\mathcal{V}$ is greater than zero. If yes, the value in $\mathcal{V}[\mathcal{V}_2[v]]$ is increased by one; otherwise, a new key with $\mathcal{V}_2[v]$ is created in $\mathcal{V}$. The value of $\tilde{p}(v)$ corresponds to $\mathcal{V}_1[v]/|S_i|$.

Algorithm 4, in line 19, stores each value in $\mathcal{V}$ in a sparse way and without repetition. It also updates the map $\mathbf{v}[v]$, for each $v \in V$.

Algorithm 3 PercolationCentralityApproximation(G,x,ε,δ)

Input: Graph $G = (V,E)$ with $n = |V|$, percolation states x, accuracy constant $0 < \varepsilon < 1$, confidence constant $0 < \delta < 1$, sample scheduling $(S_i)_{i\geq 1}$.
Output: Approximation $\tilde{p}(v)$ for the percolation centrality of all vertices $v \in V$.

1: $\mathcal{V}, \mathcal{V}_1, \mathcal{V}_2 \leftarrow$ hash tables
2: $\mathcal{V}_1[v] \leftarrow 0, \mathcal{V}_2[v] \leftarrow 0, \text{minus\_sum}[v] \leftarrow 0, \forall v \in V$
3: sort x // after sorted, $x = (x_1, x_2, \ldots, x_n)$
4: minus_sum $\leftarrow$ getPercolationDifferences(x,n)
5: $|S_0| \leftarrow 0$
6: $i \leftarrow 0$
7: **repeat**
8: $\quad i \leftarrow i+1$
9: $\quad$ **for** $l \leftarrow 1$ **to** $|S_i| - |S_{i+1}|$ **do**
10: $\quad\quad$ sample $u \in V$ with probability $1/n$
11: $\quad\quad$ sample $w \in V$ with probability $1/(n-1)$
12: $\quad\quad \mathcal{C}_{uw} \leftarrow$ allShortestPaths(u,w)
13: $\quad\quad$ **if** $\mathcal{C}_{uw} \neq \emptyset$ **then**
14: $\quad\quad\quad$ **for** $z \in P_u(w)$ **do**
15: $\quad\quad\quad\quad \sigma_{zw} \leftarrow 1$
16: $\quad\quad\quad$ **for** each z inner to some shortest path from u to w in inverse order of $d(u,w)$ **do**
17: $\quad\quad\quad\quad \sigma_{uw}(z) \leftarrow \sigma_{uz}\sigma_{zw}$
18: $\quad\quad\quad\quad$ updateSetV$\left(\mathcal{V}, z, \mathcal{V}_1, \mathcal{V}_2, \frac{R(x_u - x_w)}{\text{minus_sum}[z]} \frac{\sigma_{uw}(z)}{\sigma_{uw}}\right)$
19: $\quad\quad\quad\quad$ **for** $y \in P_u(z)$ **do**
20: $\quad\quad\quad\quad\quad \sigma_{yw} \leftarrow \sigma_{yz} + \sigma_{zw}$
21: $\quad \omega_s \leftarrow \min_{s\in\mathbb{R}_{\geq 0}} \frac{1}{s} \ln \sum_{q\in\mathcal{V}} \exp \frac{s^2 q}{2|S_i|^2}$
22: $\quad \delta_i \leftarrow \delta/2^i$
23: $\quad \eta \leftarrow 2\omega_s + \frac{\ln(3/\delta_i) + \sqrt{(\ln(3/\delta_i) + 4|S_i|\omega_s)\ln(3/\delta_i)}}{|S_i|} + \sqrt{\frac{\ln(3/\delta_i)}{2|S_i|}}$
24: **until** $\eta > \varepsilon$
25: $\tilde{p}[v] \leftarrow \mathcal{V}_1[v]/|S_i|, \forall v \in V$
26: **return** $\tilde{p}[v], \forall v \in V$

3.4 CORRECTNESS AND RUNNING TIME ANALYSIS

In Theorems 15 and 16 we prove the correctness and running time of Algorithm 3, which can be extended to the fixed size sample Algorithm 2.

Theorem 15. *Algorithm 3 returns with probability at least* $1 - \delta$ *an approximation* $\tilde{p}(v)$ *to* $p(v)$, *for each* $v \in V$, *such that* $\tilde{p}(v)$ *is within* ε *error.*

Algorithm 4 UpdateSetV($\mathcal{V}$,z,$\mathcal{V}_1$,$\mathcal{V}_2$,r_z)

Input: Table $\mathcal{V}$, vertex z, table $\mathcal{V}_1$, table $\mathcal{V}_2$, real value r_z.

```
 1: v ← V2[z]
 2: v' ← v + r_z^2
 3: if v' ∉ V then
 4:     V[v'] ← 1
 5: else
 6:     V[v'] ← V[v'] + 1
 7: if v > 0 and V[v] ≥ 1 then
 8:     V[v] ← V[v] − 1
 9: if v > 0 and V[v] = 0 then
10:     remove V[v]
11: V1[z] ← V1[z] + r_z
12: V2[z] ← V2[z] + r_z^2
```

Proof. Let l be the number of iterations of the loop in lines 7–24 of Algorithm 3. Consider the sample $S_l = \{(u_1, w_1), \ldots, (u_r, w_r)\}$ of size r obtained after the last iteration of such loop where the stopping condition is satisfied, where each (u_j, w_j) is a pair in $V \times V$, for $1 \leq j \leq r$. Let η_i be the value obtained in line 23 on the i-th iteration, where $1 \leq i \leq l$, and let ω_s be the optimum value of the function defined in Theorem 11, which is an upper bound to the empirical Rademacher average of the sample S_l and which is computed by a linear-time procedure of Johnson (2014). Then

$$\eta_l = 2\omega_s + \frac{\ln(3/\delta_l) + \sqrt{(\ln(3/\delta_l) + 4|S_l|\omega_s)\ln(3/\delta_l)}}{|S_l|} + \sqrt{\frac{\ln(3/\delta_l)}{2|S_l|}}$$

is the value such that $\eta_l \leq \varepsilon$ for the input graph $G = (V, E)$ and for fixed constants $0 < \varepsilon, \delta < 1$.

Let E_i be the event where $\sup_{v \in V} |\tilde{p}(v) - p(v)| > \eta_i$ in iteration i. We need the event E_i occurring with probability at most δ for some iteration i. That is, we need

$$\Pr(\exists i \geq 1 \text{ s.t. } E_i \text{ occurs}) \leq \sum_{i=1}^{\infty} \Pr(E_i) \leq \delta,$$

where the inequality comes from union bound. Setting $\Pr(E_i) = \delta/2^i$, we have

$$\sum_{i=1}^{\infty} \Pr(E_i) = \delta \sum_{i=1}^{\infty} \frac{1}{2^i} = \delta.$$

For each iteration i in 7–24, the pair (u_j, w_j) is sampled with probability $\frac{1}{n(n-1)}$ in lines 10 and 11, for $1 \leq j \leq r$, and the set $\mathcal{C}_{u_j w_j}$ is computed by Dijkstra's algorithm (line 12).

The value of $\frac{R(x_{u_j} - x_{w_j})}{\text{minus\_sum}[z]} \frac{\sigma_{u_j w_j}(z)}{\sigma_{u_j w_j}}$, for each inner vertex z of a shortest path $p \in \mathcal{C}_{u_j w_j}$ found on the backtracking procedure (lines 16–20) is added to $\mathcal{V}_1[z]$ by Algorithm 4 in line 18. The correctness of the procedure in lines 19–20 can be checked in Lemma 3 of Brandes (2001).

The value of minus_sum[z] is correctly computed in line 4 as shown in Theorem 14. Then, at the end of Algorithm 3,

$$\tilde{p}(z) = \frac{1}{r} \sum_{(u,w)\in S_l} \frac{R(x_u - x_w)}{\sum\limits_{\substack{(f,d)\in V\times V \\ f\neq z\neq d}} R(x_f - x_d)} \frac{\sigma_{uw}(z)}{\sigma_{uw}}$$

which corresponds to $\tilde{p}(z) = \frac{1}{r} \sum\limits_{p_{uw}\in S_l} f_z((u,w))$.

Since $\eta_l \leq \varepsilon$, $L_{S_l}(f_v) = \tilde{p}(v)$, and $L_{X'}(f_v) = p(v)$ (Theorem 12) for all $v \in V$, and $f_v \in \mathcal{F}$, then $\Pr(|\tilde{p}(v) - p(v)| \leq \varepsilon) \geq 1 - \delta$ (Theorem 10). □

Theorem 16. *Given a weighted graph $G = (V, E)$ with $n = |V|$ and $m = |E|$, and a sample of size $r = \frac{c}{\varepsilon^2}(\lfloor \lg diam_V(G) - 2 \rfloor + 1) - \ln \delta)$, Algorithm 3 has expected running time $\mathcal{O}(m \log^2 n)$.*

Proof. We sample the vertices u and w in lines 10 and 11, respectively, in linear time.

Sorting the percolation states array x (line 3) can be done in $\mathcal{O}(n \log n)$ time and the execution of Algorithm 1 on the sorted array x (line 4) has running time $\mathcal{O}(n)$. Before the loop in lines 16–20 start, the vertices in G are sorted according to $d(u,w)$ in reverse order, which takes $\mathcal{O}(n \log n)$. The complexity analysis of the procedure in 16–20 proceeds as follows. Once $|P_u(z)| \leq d_G(z)$, where $d_G(z)$ denotes the degree of z in G and $P_u(z)$ is the set of predecessors of z in the shortest paths from u to w, and since this loop is executed at most n times if all the vertices of G are inner to some shortest path between u and w, the total running time of these steps corresponds to $\sum_{v\in V} d_G(v) = 2m = \mathcal{O}(m)$.

The execution of Algorithm 4 in line 18 has $\mathcal{O}(1)$ expected running time, since the sets $\mathcal{V}$, $\mathcal{V}_1$, and $\mathcal{V}_2$ are stored as hash tables and operations of insertion, deletion, and search on these structures take $\mathcal{O}(1)$ time in average. Line 21 is executed by an algorithm that is linear in the size of the sample (Johnson, 2014). The loop in lines 7–24 runs at most r times, since $|\tilde{p}(v) - p(v)| \leq \varepsilon$ for all $v \in V$, with probability $1 - \delta$, when the sample has size r (Theorem 7). The Dijkstra algorithm which is executed in line 12 has running time $\mathcal{O}(m + n \log n) = \mathcal{O}(m \log n)$, so the total expected running time of Algorithm 3 is $\mathcal{O}(n \log n + r \max(m, m \log n)) = \mathcal{O}(n \log n + r(m \log n)) = \mathcal{O}(r(m \log n)) = \mathcal{O}(m \log^2 n)$. □

Corollary 1. *Given an unweighted graph $G = (V, E)$, with $n = |V|$ and $m = |E|$, and a sample of size $r = \frac{c}{\varepsilon^2}(\lfloor \lg diam_V(G) - 2 \rfloor + 1) - \ln \delta)$, Algorithm 3 has expected running time $\mathcal{O}((m + n) \log n)$.*

Proof. The proof is analogous to the one of Theorem 16, with the difference that the shortest paths between a sampled pair $(u,w) \in V \times V$ will be computed by the breadth-first search (BFS) algorithm, which has running time $\mathcal{O}(m + n)$. □

We observe that, even though it is an open problem whether there is a $\mathcal{O}(n^{3-c})$ algorithm for computing all shortest paths in weighted graphs, in the unweighted case there is a $\mathcal{O}(n^{2.38})$ (non-combinatorial) algorithm for this problem (Seidel, 1995). However, even if this algorithm could be adapted to compute betweenness/percolation centrality (what is not clear), our algorithm obtained in Corollary 1 is still faster.

3.5 EXPERIMENTAL EVALUATION

In this section we present the experimental evaluation on the percolation centrality problem on the fixed-size sample and the progressive sampling algorithms. Since, as far as we know, our algorithms are the first estimation algorithms for percolation centrality, we compare our results with the best known algorithm for computing the same measure in the exact case. As expected, our algorithms are many times faster than the exact algorithm, since the exact computation of the percolation centrality takes $\mathcal{O}(n^3)$ time, which is extremely time consuming. Additionally, a main advantage of our algorithms is that they output an estimation with a very small error. In fact, for every one of the networks used in our experiment, the average estimation error are kept below the quality parameter ε by many orders of magnitude. For instance, even for $\varepsilon = 0.1$ (the largest in our experiments), the average estimation error is in the order of 10^{-11}, and maximum estimation error of any one of the centrality measure is in the order of 10^{-9}. We also compare the running time and the accuracy of our progressive sampling algorithm with the fixed-size sample approach. Our results show that the impacts of the improvements on the sample size obtained by the former approach are better observed in particular for smaller values of ε.

3.5.1 Real-world graphs

We perform our experimental evaluation on publicly available real-world graph datasets from Stanford Large Network Dataset Collection (Leskovec and Krevl, 2014) and Network Repository (Rossi and Ahmed, 2015), described in Tables 3.1 and 3.3. These graphs spam from social, peer-to-peer, and citations networks. Our implementation uses Python 3.7 language, the NetworkX library for graph manipulation, and the NLOpt library for computing the minimization function in Theorem 11. The NetworkX library provides an exact algorithm for the percolation centrality which we use to compare with our approximation algorithms. The experiments were performed on a 2.6 MHz Intel Xeon E5-2650v2 octa core with 48GB of RAM and Ubuntu 14.04 64-bit operating system.

Graph	Type	|V|	|E|	Approximated $\text{diam}_V(G)$	Exact Algorithm Running Time (in secs.)
wiki–Vote	Directed	7115	103689	11	214.2481
p2p–Gnutella08	Directed	6301	20777	14	111.894
oregon1–010331	Undirected	10670	22002	14	1050.589
ca–CondMat	Undirected	23133	93497	21	6256.0498
asas20000102	Undirected	6474	13895	14	338.276

Table 3.1: Datasets details for the real-world graphs of the experiments of the fixed-size sample algorithm.

exact/approx. time average ratio				
Graph	$\varepsilon = 0.04$	$\varepsilon = 0.06$	$\varepsilon = 0.08$	$\varepsilon = 0.1$
wiki–Vote	10.36	22.17	36.93	52.71
p2p–Gnutella	9.92	19.59	33.91	53.21
oregon1–010331	17.66	39.7	70.14	108.71
ca–CondMat	30.25	67.32	120.98	191.1
as20000102	9.61	20.67	38.81	55.4

Table 3.2: Ratio for average running time after five runs of the percolation centrality estimation algorithm and the exact algorithm.

Graph	Type	|V|	|E|	Approximated $\text{diam}_V(G)$	Exact Algorithm Running Time (in secs.)
p2p–Gnutella04	Directed	10876	39994	39	257.8975
Cit–HepPh	Directed	34546	421578	42	4455.429
p2p–Gnutella31	Directed	62586	147892	42	9692.5041
socfb–Berkeley13	Undirected	22900	852419	33	15271.9312

Table 3.3: Dataset details for the real-world graphs of the experiments of the progressive sampling algorithm

In all experiments in graphs of Tables 3.1 and 3.3, we set the percolation state x_v, for each $v \in V$, as a random number between 0 and 1 and the weights of each edge $e \in E$ as an integer random number between 1 and 100. In both fixed-size and progressive sampling algorithms, we set the parameters $\delta = 0.1$ and $c = 0.5$ (as suggested by Löffler and Phillips (2009)), and the accuracy parameter as $\varepsilon = \{0.1, 0.08, 0.06, 0.04\}$. In the progressive sampling algorithm, we also set $\varepsilon = 0.01$, and the constant for the geometric sampling schedule $g = \{1.2, 1.5, 2.0\}$.

We run the fixed-size sample algorithm five times for the graphs in Table 3.1 for each value of ε. Them maximum error is taken among all five runs. These results are shown in Figures 3.2, 3.3, and 3.4. In Table 3.2 we show how many times our algorithm is faster than the exact method in the average of the five runs. We compute this factor by dividing the running time of the exact method by the running time of our algorithm. Note that for the largest graph our algorithm is around 30 times faster, on average, than the exact method.

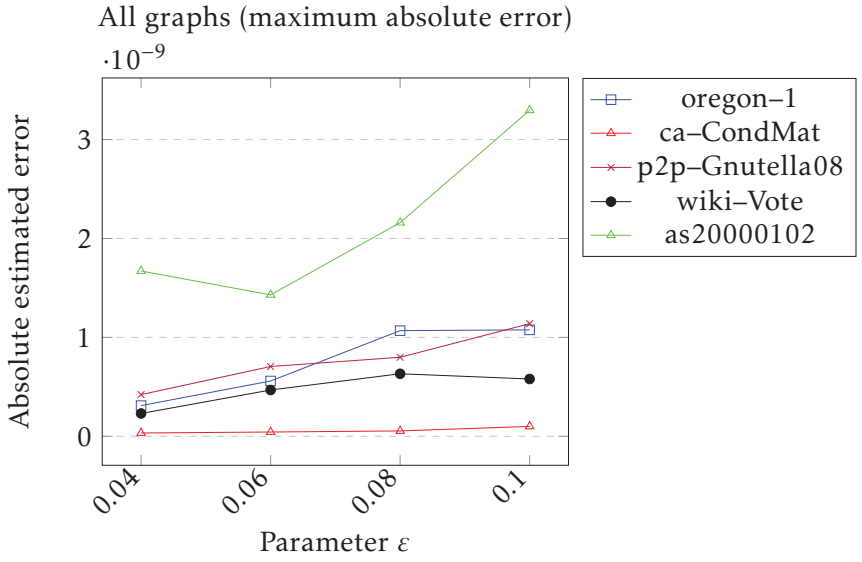


Figure 3.2: Percolation centrality absolute maximum error estimation.

In the progressive sampling algorithm, we report results for the graphs in Table 3.3, for five experiments in the combination of parameters ε and g. We report the results of the running time and the size of the sample attained by the progressive sampling schedule in Tables 3.4, 3.5, 3.6, and 3.7, where each table is associated to one of the real–world graphs described in Table 3.3.

The highlighted entries in the tables are the cases where the progressive sampling schedule achieved smaller samples in comparison with the fixed-size sample approach. In all graphs and all values of ε, the improvement on the sample size were

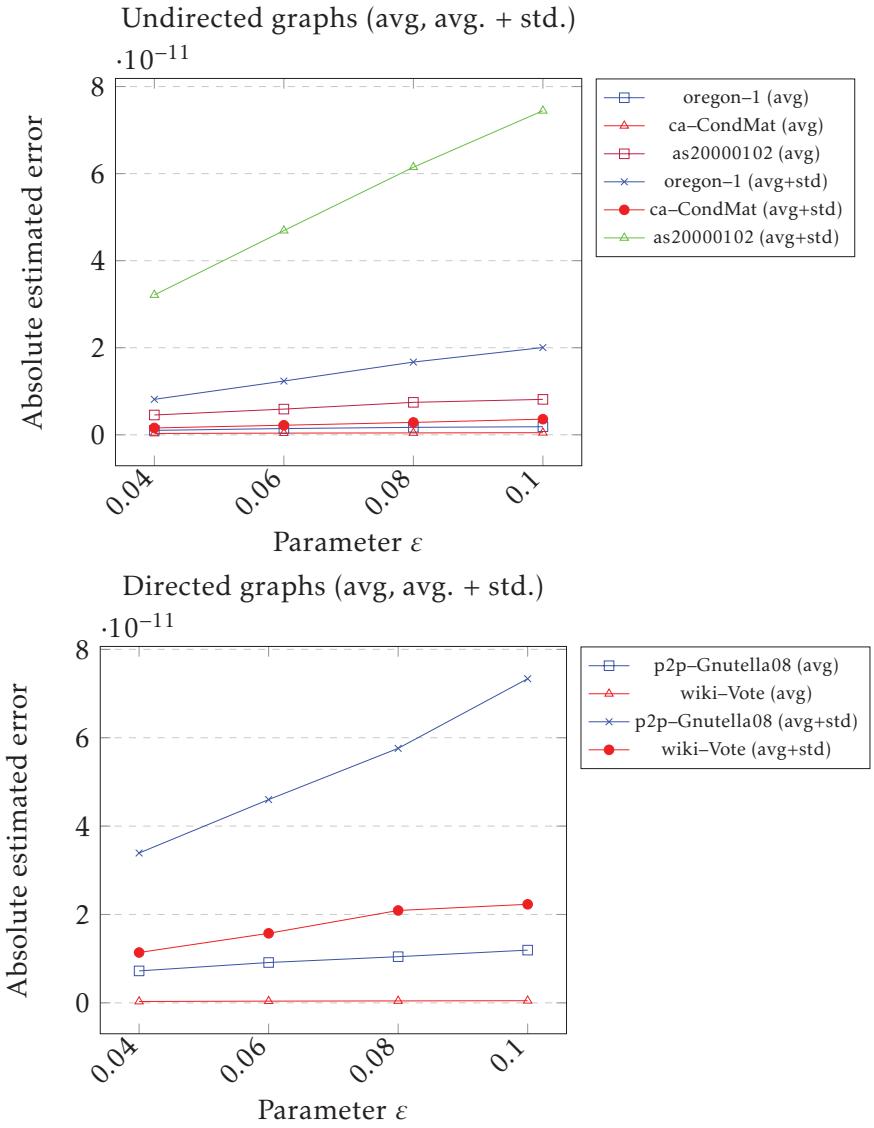


Figure 3.3: Percolation centrality absolute error estimation.

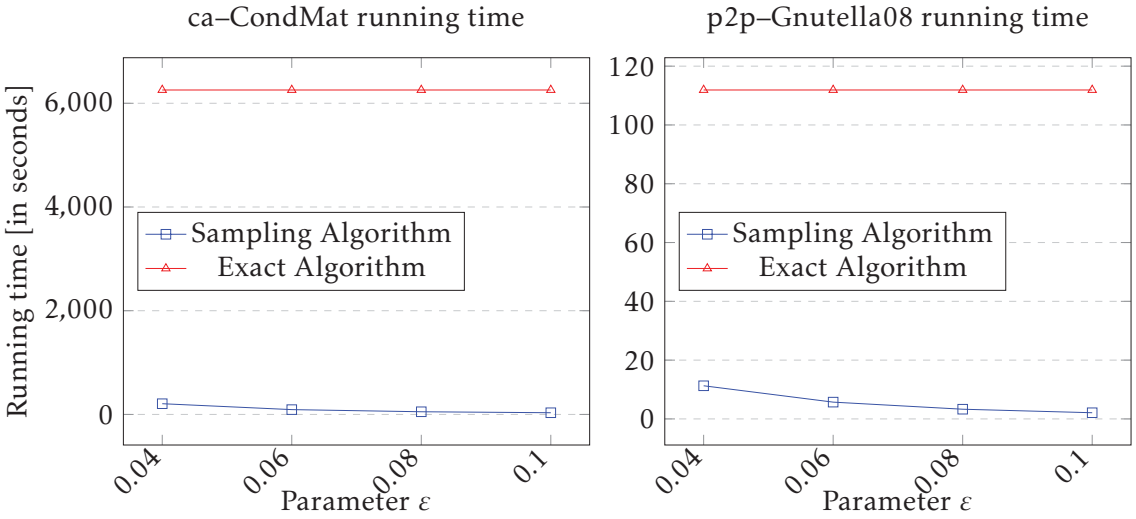


Figure 3.4: Average running time for 5 runs of Algorithm 2

ε	Schedule constant	Sample Size (Initial Sample Final Sample)	$\varepsilon - \eta$	Final η	Progressive Sampling Running Time (in secs.)	Fixed Sample Running Time (in secs.)	Fixed Sample Size
0.1	1.2	350 420	-0.00001166 0.001573909	0.098426091	11.76664	10.62395	416
	1.5	350 524	-0.000012202 0.0141203486	0.0858796514	13.38731		
	2	350 699	-0.00000943 0.027747531	0.0722524692	17.53773		
0.08	**1.2**	504 **605**	-0.000009637 0.001227134	0.078772866	**15.04645**	16.61353	649
	1.5	504 756	-0.00000974 0.011037825	0.068962175	21.94075		
	2	504 1008	-0.000009834 0.021745709	0.058254291	25.03958		
0.06	**1.2**	819 **983**	-0.000009936 0.000889796	0.059110204	**27.16701**	29.54515	1154
	1.5	819 1229	-0.000009997 0.008049222	0.051950778	31.63912		
	2	819 1628	-0.000010057 0.015912334	0.044087666	43.98850		
0.04	**1.2**	1664 **1997**	-0.000010195 0.000567447	0.039432553	**53.29527**	65.21671	2595
	1.5	1664 **2496**	-0.000010281 0.00518215	0.034817851	**61.23428**		
	2	1664 3328	-0.00001037 0.010290545	0.029709455	82.76411		
0.02	**1.2**	5909 **7091**	-0.000010685 0.000264471	0.019735529	**176.70750**	257.03656	10379
	1.5	5909 **8863**	-0.000010748 0.002474338	0.017525662	**219.01537**		
	2	5909 11817	-0.000010807 0.004946187	0.015053813	300.53190		

Table 3.4: p2p–Gnutella04 Graph

ε	Schedule constant	Sample Size (Initial Sample Final Sample)	$\varepsilon - \eta$	Final η	Progressive Sampling Running Time (in secs.)	Fixed Sample Running Time (in secs.)	Fixed Sample Size
0.1	1.2	350 420	-0.000011661 0.001573908	0.098426093	77.43216	80.69897	416
	1.5	350 524	-0.000012194 0.014123353	0.0858766471	91.91335		
	2	350 699	-0.000009476 0.027747498	0.072252502	125.93146		
0.08	**1.2**	504 **605**	-0.000009677 0.001227094	0.078772906	**111.36629**	129.91781	649
	1.5	504 756	-0.000009769 0.011037804	0.068962197	159.96246		
	2	504 1008	-0.000009847 0.021745687	0.058254314	188.53688		
0.06	**1.2**	819 **983**	-0.000009956 0.000889777	0.059110223	**183.27423**	222.75616	1154
	1.5	819 1229	-0.000010018 0.0080492	0.0519508002	245.29033		
	2	819 1628	-0.000010088 0.015912295	0.044087705	302.89113		
0.04	**1.2**	1664 **1997**	-0.000010235 0.000567401	0.039432599	**360.51697**	493.08546	2595
	1.5	1664 **2496**	-0.000010337 0.005182085	0.034817915	**486.70175**		
	2	1664 3328	-0.000010431 0.010290473	0.029709528	616.52572		
0.02	**1.2**	5909 **7091**	-0.000010808 0.000264345	0.019735656	**1327.98216**	2029.62927	10379
	1.5	5909 **8863**	-0.000010883 0.002474196	0.017525804	**1734.40681**		
	2	5909 11817	-0.000010954 0.004946029	0.015053971	2192.27154		

Table 3.5: Cit–HepPh graph

ε	Schedule constant	Sample Size (Initial Sample Final Sample)	ε - η	Final η	Progressive Sampling Running Time (in secs.)	Fixed Sample Running Time (in secs.)	Fixed Sample Size
0.1	1.2	350 420	-0.000011326 0.001574200	0.098425800	35.22300	40.36586	416
	1.5	350 524	-0.000012007 0.014120532	0.085879468	44.06515		
	2	350 699	-0.000012293 0.027747634	0.072252366	55.88123		
0.08	**1.2**	504 **605**	-0.000009545 0.001227216	0.078772784	**54.85581**	55.63046	649
	1.5	504 756	-0.000009653 0.011037899	0.068962101	63.78716		
	2	504 1008	-0.000009766 0.021745775	0.058254225	86.09895		
0.06	**1.2**	819 **983**	-0.000009873 0.000889857	0.059110143	**2.28515**	101.97880	1154
	1.5	819 1229	-0.000009939 0.008049265	0.051950735	100.22565		
	2	819 1628	-0.000010012 0.015912386	0.044087614	141.78238		
0.04	**1.2**	1664 **1997**	-0.000010155 0.000567474	0.039432526	**182.54564**	225.51281	2595
	1.5	1664 **2496**	-0.000010273 0.005182146	0.034817854	**209.05493**		
	2	1664 3328	-0.000010379 0.010290510	0.029709490	288.22010		
0.02	**1.2**	5909 **7091**	-0.000010795 0.000264357	0.019735643	**614.76785**	903.45608	10379
	1.5	5909 **8863**	-0.000010875 0.002474201	0.017525799	**772.12881**		
	2	5909 11817	-0.000010956 0.004946023	0.015053977	1037.35445		

Table 3.6: p2p–Gnutella31 Graph

ε	Schedule constant	Sample Size (Initial Sample Final Sample)	$\varepsilon - \eta$	Final η	Progressive Sampling Running Time (in secs.)	Fixed Sample Running Time (in secs.)	Fixed Sample Size
0.1	1.2	350 420	-0.000012282 0.001576370	0.098423630	790.94315	695.12026	366
	1.5	350 524	-0.000009636 0.014122944	0.085877056	979.45032		
	2	350 699	-0.000009850 0.027747135	0.072252865	1315.06880		
0.08	1.2	504 605	-0.000010015 0.001226759	0.078773241	1150.53222	1064.03995	571
	1.5	504 756	-0.000010129 0.011037425	0.068962575	1429.07482		
	2	504 1008	-0.000010262 0.021745247	0.058254753	1854.15828		
0.06	**1.2**	819 **983**	-0.000010430 0.000889299	0.059110701	**1832.42449**	1876.60577	1015
	1.5	819 1229	-0.000010541 0.008048669	0.051951331	2279.52619		
	2	819 1628	-0.000010637 0.015911753	0.044088247	3063.31744		
0.04	**1.2**	1664 **1997**	-0.000010767 0.000566874	0.039433126	**3668.25732**	4222.63659	2283
	1.5	1664 2496	-0.000010851 0.005181580	0.034818420	4573.80160		
	2	1664 3328	-0.000010931 0.010289983	0.029710017	6170.42000		
0.02	**1.2**	5909 **7091**	-0.000011230 0.000263926	0.019736074	**13529.57006**	16781.54561	9129
	1.5	5909 8863	-0.000011290 0.002473796	0.017526204	16329.16631		
	2	5909 11817	-0.000011347 0.004945649	0.015054351	21674.77987		

Table 3.7: socfb–Berkeley13 Graph

obtained by the smallest geometric schedule constant, which is $g = 1.2$. For $\varepsilon = 0.02$ and $\varepsilon = 0.04$, the progressive sampling algorithm also acquired smaller samples for $g = 1.5$, except for the socfb-Berkeley13 graph, which is the most dense graph among the dataset. The main reason that the fixed-size sample algorithm outperforms the progressive sampling one when $g = 2.0$ is that much more pairs of vertices are sampled than necessary, leading to a value of η that is much smaller than the desired ε.

In both approaches, the error of the estimation is within ε for all vertices of every graph even though this guarantee could possibly fail with probability $\delta = 0.1$. However, in addition to the better confidence results than the theoretical guarantee, the most surprising fact is that for all graphs used in the experiments the maximum error among the estimation error of each vertex is around 10^{-10} and the average error among all vertices is around 10^{-11}, even when we set the quality guarantee to $\varepsilon = 0.1$. We observe that the differences between the error obtained in the progressive sampling approach and the fixed-size sample approach are within 10^{-14}. These results are shown in Figures 3.5 and 3.6.

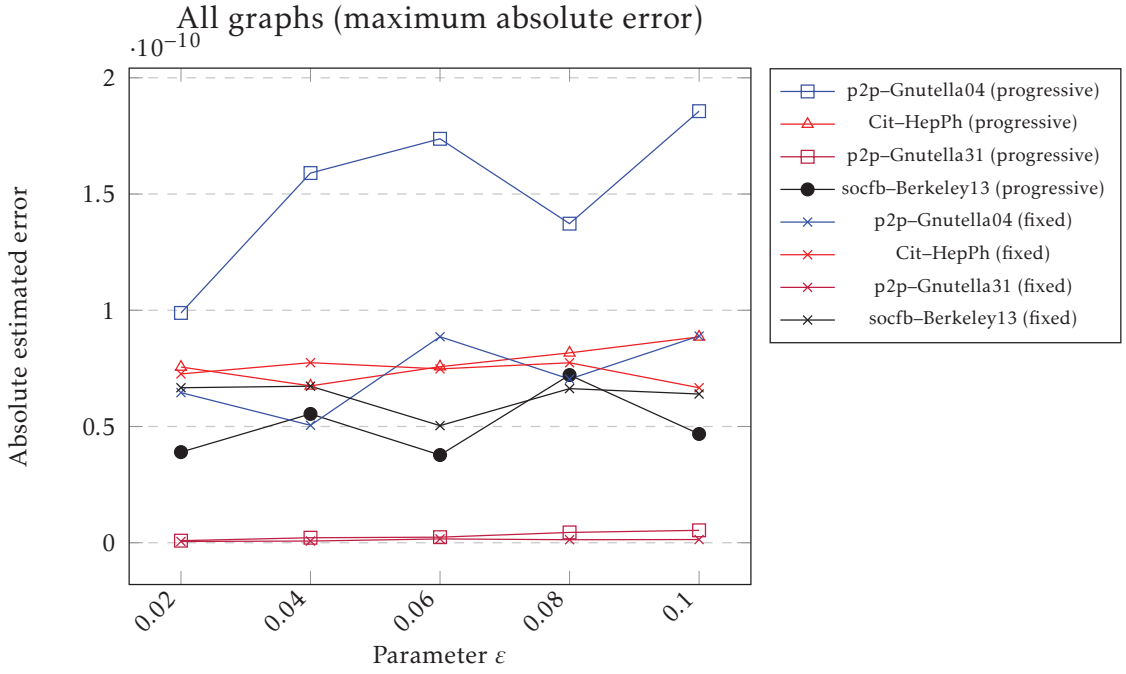


Figure 3.5: Percolation centrality absolute maximum error estimation for $g = 1.2$.

3.5.2 Synthetic graphs

We run experiments using synthetic graphs in order to validate the scalability of our algorithm, since for this task we need a battery of similar graphs of increasing size. We use power-law graphs generated by the Barabási-Albert model (Barabási and Albert, 1999) for such experiments, with each vertex creating two edges, obtaining undirected unweighted power-law graphs with average degree of 2. We use a sequence of synthetic graphs increasing in size and compared the execution time of our progressive sampling algorithm with the exact algorithm provided by NetworkX library and with the fixed-size sample algorithm. We set the percolation states of each vertex as a random number between 0 and 1.

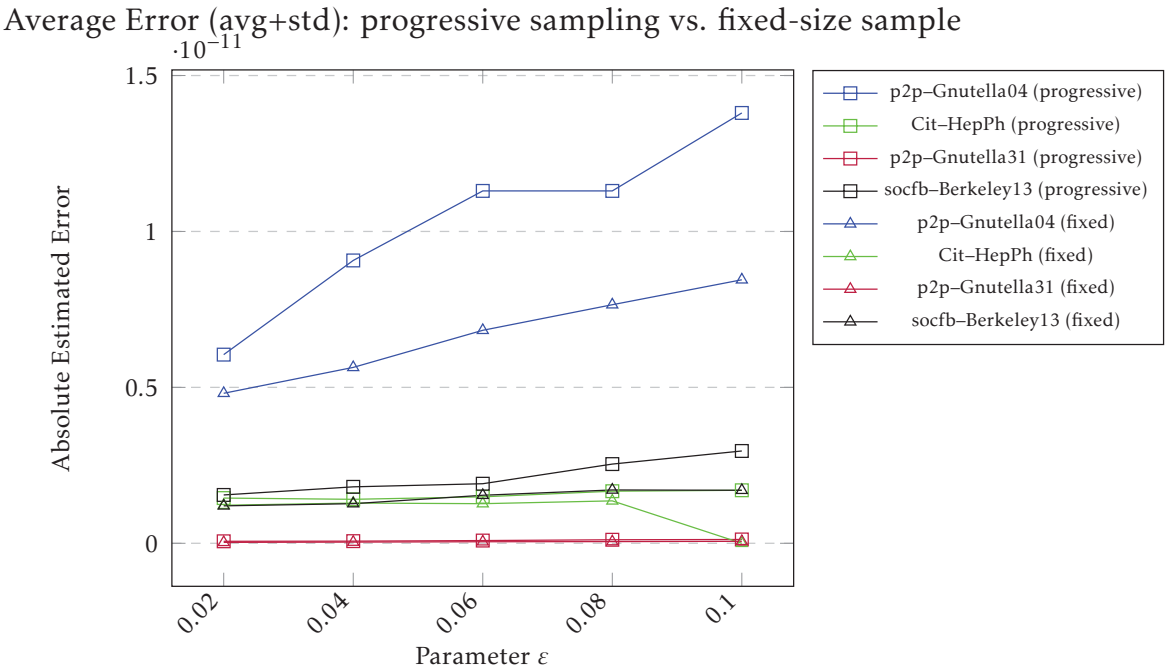


Figure 3.6: Percolation centrality average error estimation for $g = 1.2$

In the experiments we use graphs with the number of vertices n in $\{2000, 4000, 8000, 16000, 32000, 64000\}$. The values of ε and g are fixed at 0.03 and 1.2, respectively. The results are shown in Figure 3.7.

3.6 CONCLUDING REMARKS

We presented a sampling-based algorithm to accurately estimate the percolation centrality of all vertices of a directed weighted graph with high probability. The proposed algorithm has expected running time $\mathcal{O}(m \log^2 n)$, and the estimation is within ε of the exact value with probability $1 - \delta$, for *fixed* constants $0 < \varepsilon, \delta < 1$. The running time of the algorithm is reduced to $\mathcal{O}((m+n)\log n)$ if the input graph is unweighted. Since many large scale graphs, in practical applications, are sparse and have small vertex-diameter (typically of size $\log n$), our algorithm provides a fast approximation for such graphs (more precisely running in $\mathcal{O}(n \log n \log\log n)$ time).

Our results indicate that the proposed approach is practical in real-world graphs, as validated by our experimental evaluation. The returned estimation errors are many orders of magnitude smaller than the theoretical worst case guarantee for graphs of a variety of sizes. As expected, the fixed-size sample and the progressive sampling algorithms are much faster than the exact algorithm. The progressive sampling approach is around 14 times faster for the largest real-world graph and 42 times faster for the largest synthetic graph. Furthermore, the progressive sampling algorithm returned good quality estimations using a smaller sample set in comparison to the one obtained by the fixed-size sample algorithm, leading to improvements on the running time.

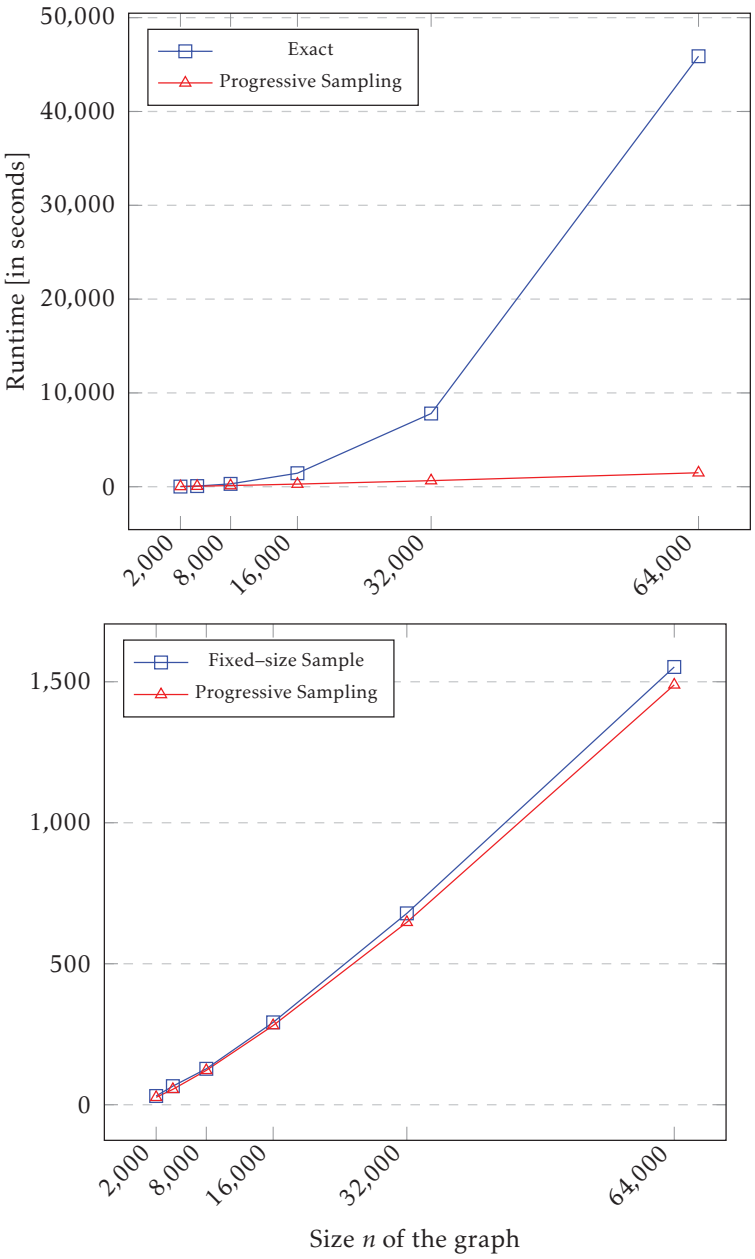


Figure 3.7: Scalability experiments, with $\varepsilon = 0.03$ and $g = 1.2$.

4 ALL-PAIRS SHORTEST PATH (APSP) PROBLEM

The All-Pairs Shortest Path (APSP) is the problem of computing the distance between every pair of vertices in a weighted graph. The APSP problem is very well studied and there has been recent results for a variety of assumptions for the input graph (directed/undirected, integer/real edge weights, etc) (Williams, 2014; Chan, 2012; Eirinakis et al., 2017; Brodnik and Grgurovič, 2017). In this chapter we assume that the input is an undirected graph G with n vertices and m edges with non-negative weights.

In our scenario, the fastest known exact algorithms are the algorithm proposed by Williams (2014), which runs in $\mathcal{O}\left(\frac{n^3}{2^c\sqrt{\log n}}\right)$ time, for some constant $c > 0$, and by Pettie and Ramachandran (2002) for the case of sparse graphs, which runs in $\mathcal{O}(nm \log \alpha(m,n))$ time, where $\alpha(m,n)$ is the Tarjan's inverse-Ackermann function. If no assumption is taken about the sparsity of the graph, then it is an open question whether the APSP problem can be solved in *strictly* subcubic time, i.e. $\mathcal{O}(n^{3-c})$, for some $c > 0$, even when the edge weights are natural numbers.

Recent results in fine-grained complexity indicate that the cubic-time complexity for the APSP is tight (Roditty and Vassilevska Williams, 2013; Abboud and Williams, 2014; Abboud et al., 2018), reinforcing the hypothesis that there is no strictly subcubic algorithm for such task (Vassilevska Williams, 2015). Since the exact computation of this version is expensive for large graphs, especially the dense ones, it is natural dealing with alternative versions of the problem, whether they are approximate (Dor et al., 2000; Roditty and Shapira, 2011) or applied to restricted scenarios (Shoshan and Zwick, 1999). In this chapter, we follow this line of work, dealing with a relaxation of the problem in the sense that the classical APSP is a special case for a given adjustable parameter. More specifically, we aim to compute, with high probability, all the lengths of shortest paths that meet a certain *centrality* requirement. The idea is that the centrality of a shortest path P is higher when a large number of shortest paths has P as a subpath. The precise definition of this centrality measure is given in Section 4.1.

In this relaxed version of the APSP, given constant parameters $0 < \varepsilon, \delta < 1$, we propose a sampling algorithm that outputs, with probability at least $1 - \delta$, the (exact) distance between every pair of vertices that admits a shortest path with centrality at least ε. The central idea of the algorithm is to sample roots of shortest paths trees. In order to give a bound for the sample size that is sufficient to meet the input parameters, we use sample complexity tools, namely, Vapnik–Chervonenkis (VC) dimension theory and the ε-net theorem. We define a range space associated with a set of *canonical shortest paths* in G between every pair of distinct vertices. One of the main results that we prove is that the VC dimension of such range space is 2 and that the bound for the sample size is $r = \left\lceil \frac{c}{\varepsilon}\left(2\ln(\frac{1}{\varepsilon}) + \ln\frac{1}{\delta}\right)\right\rceil$, where c is a constant around $\frac{1}{2}$ (Löffler and Phillips, 2009). This result is interesting, since it does not depend neither on the size of the input n, which is the case if one uses standard techniques based on union-bound, nor on the topological structure of the graph that may vary with n in many cases. As a consequence of this bound for the sample size, we obtain a sampling algorithm for our problem with running time $\mathcal{O}(m + n\log n + (\mathrm{diam}_V(G))^2)$, where $\mathrm{diam}_V(G)$ is the vertex-diameter of the input graph.

If one sets ε as a function of n, in the limit case, when $\varepsilon(n) = \frac{1}{n(n-1)}$, the algorithm solves – with high probability – the classical APSP problem, but with time complexity exceeding the running time of the exact algorithms from the literature (Williams, 2018; Pettie and Ramachandran, 2002). However, it is still an interesting problem to know for which functions $\varepsilon(n)$ we still have a strictly subcubic sampling algorithm. We show that to obtain a $\mathcal{O}(n^{3-c})$ time algorithm using our strategy, $\varepsilon(n)$ must be $\Omega\left(\frac{W_0(n')}{n'}\right)$, where $n' = n^{1-c}$ (for a constant $c > 0$) and $W_0(n')$ is the branch 0 of the Lambert-W function defined for $n' \geq 0$, a non-algebraic value such that $W_0(n') = \ln n' - \ln\ln n' + \Theta\left(\frac{\ln\ln n'}{\ln n'}\right)$, which holds for $n' \geq e$.

This chapter is organized as follows: Section 4.1 states the definitions and notations; in Section 4.1.1 we present the main results of this work regarding canonical paths; in Section 4.2 we present the range space that models our problem and then we compute the bound for its VC dimension; in Section 4.3 we present the algorithm for the version of the APSP that we are tackling and the corresponding modifications on Dijkstra's algorithm that we consider in our algorithm; in Section 4.4 we present the adaptation of the main algorithm for the problem of estimating the centrality of each vertex in a graph, and Section 4.5 contains the conclusion and final remarks.

4.1 SHORTEST PATHS, CANONICAL PATHS, AND SHORTEST PATHS TREES

Let $G = (V, E)$ be an undirected graph, with $n = |V|$ and $m = |E|$. Even though G is undirected, for convenience we use the notation (u, v) for an edge of G.

Let $\lambda : V \to \{1, \ldots, n\}$ be an arbitrary vertex ordering of G. Consider the set of shortest paths $\mathcal{L}_{uv} = \{P \in \mathcal{C}_{uv} : \lambda(\text{pred}_P(v)) \text{ is minimum}\}$, recalling that $\mathcal{C}_{uv}$ is the set of all shortest paths from u to v and $\text{pred}_P(v)$ is the predecessor of v in a shortest path P. Note that there is only one vertex w that satisfies the property "$\lambda(\text{pred}_P(v))$ is minimum", so even if there are several paths in $\mathcal{L}_{uv}$, the last edge (w, v) is the same for all of them. Next, we introduce the definition of a *canonical path* with respect to λ.

Definition 12 (Canonical path (CP))**.** *Consider a pair of vertices $(u, v) \in V^2$ in G. The* canonical path *(CP) from u to v, denoted P, is recursively defined as the shortest path in $\mathcal{C}_{uv}$ such that*

case 1: $|\mathcal{L}_{uv}| = 1$**.** *Then $P \in \mathcal{L}_{uv}$ is the canonical path from u to v.*

case 2: $|\mathcal{L}_{uv}| > 1$**.** *Let w be the (unique) predecessor of v in the shortest paths of $\mathcal{L}_{uv}$. Then, the canonical path P from u to v corresponds to the canonical path from u to w plus the edge (w, v).*

Proposition 1. *Given a pair of vertices $(u, v) \in V^2$, the CP from u to v exists and it is unique.* □

To see that Proposition 1 holds, note that at each recursive step, there is only one vertex w satisfying the property that defines $\mathcal{L}_{uv}$, and there is only one canonical path from u to w. Besides, the recursion presented above always stop in the base case, since the distance between a pair of vertices in a recursive step is smaller than the distance of a pair of vertices analyzed in the previous step. The base is the one where there is only one (u, u')-subpath which is the shortest path from u to u', for $u' \in \text{Inn}(P)$.

Another important observation about canonical paths is that the canonical path from u to v is not necessarily the same as the canonical path from v to u.

A *shortest paths tree (SPT)* of a vertex u is a spanning tree of G such that the path from u to every other vertex of this tree is a shortest path in G. There might be many SPTs for a given vertex. In this chapter we are interested in fixing one canonical SPT T_u, for every vertex u of G. More precisely, for the given (arbitrary) vertex ordering λ, each shortest path in T_u is a canonical path. In Section 4.3.1 we give more details on the computation of T_u by a modification on Dijkstra's algorithm where, briefly speaking, λ is used as a tie-breaking criterion. We also call T_u the *Dijkstra tree* of u.

A shortest path that starts at the root of a Dijkstra tree is called a *branch* of G. More formally, given T_u, for every $v \neq u$, the shortest path from u to v is a branch, denoted $\mathcal{B}_{uv}$.In addition, every subpath of $\mathcal{B}_{uv}$ is also a shortest path in G, and we denote such set of subpaths (including $\mathcal{B}_{uv}$) as $S(\mathcal{B}_{uv})$.

We introduce the *shortest path centrality* of a pair of vertices (u, v). The idea, intuitively, is that a shortest path is *central* if several other shortest paths pass through it. This same idea is used in the well-known betweenness metric for a vertex (Freeman, 1977), where a vertex has high betweenness if many shortest paths pass through it.

In order to formally define the shortest path centrality we first need the following. Let t_{uv} be the number of canonical paths that contain a shortest path from u to v as subpath, defined as

$$t_{uv} = \sum_{(a,b)\in V^2 : a \neq b} \mathbb{1}_{uv}(\mathcal{B}_{ab}),$$

where $\mathbb{1}_{uv}(\mathcal{B}_{ab})$ is the indicator function that returns 1 if there is some shortest path from u to v as subpath of the branch $\mathcal{B}_{ab}$ (and 0 otherwise).

Definition 13 (Shortest Path Centrality)**.** *Given a pair* $(u, v) \in V^2$*, the* shortest path centrality *of* (u, v) *is defined as*

$$c(u, v) = \frac{t_{uv}}{n(n-1)}, \quad \textit{where } n = |V|.$$

4.1.1 Key Results on Canonical Paths

Before we present the main results of this chapter in Section 4.2, we need first a key technical result concerning canonical paths. We show in Theorem 17 that any subpath of a canonical path is also a canonical path.

Lemma 3. *Given a pair of vertices* $(u, v) \in V^2$*, let* P *be the CP from* u *to* v *in* G*. If* $|\mathcal{L}_{uv}| = 1$*, then every subpath of* P *is also a CP.*

Proof. Let P' be a (u', v')-subpath of P. Suppose by contradiction that P' is not a CP. Let $Q' \neq P'$ be the shortest path $Q' = (u', \ldots, v')$ in G which is the CP from u' to v'.

Case 1: $v' \neq v$. Let S_1 be a (u, u')-subpath and S_2 be a (v', v)-subpath, both from P. Let Q be the concatenation of S_1, Q', and S_2. Note that P' and Q' have the same length (since both are shortest paths), and so does P and Q. Since P and Q have the same vertices from v' to v, then the predecessor of v in both paths is the same. Hence, P and Q are in $\mathcal{L}_{uv}$. But then $|\mathcal{L}_{uv}| > 1$, a contradiction.

Case 2: $v' = v$. Let w and w' be the predecessors of v in P' and Q', respectively. Note that $w \neq w'$. Thus, since (w', v) is the last edge of the Q', by the definition of CP,

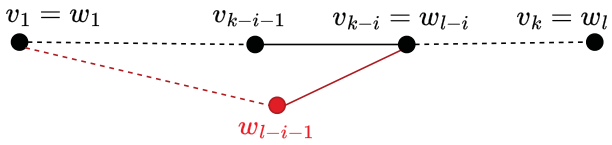


Figure 4.1: Illustration of vertex v_{k-i} in the shortest paths P (depicted in black color) and Z (depicted in red color).

$\lambda(w') < \lambda(w)$. But then in the edge (w, v) of P, vertex w does not have the minimum index among all possible predecessors of v, contradicting the fact that P is a CP. □

Lemma 4. *Given a pair of vertices $(u, v) \in V^2$, let P be the CP from u to v in G. Let w be the predecessor of v in P. Then the (u, w)-subpath of P is the CP from u to w.*

Proof. Let P' be the (u, w)-subpath of P. In the case of $|\mathcal{L}_{uv}| = 1$, then from Lemma 3 we have P' is the CP from u to w. Otherwise, by Definition 12 (case 2) applied to P, it must hold that P' is the CP from u to w. □

Lemma 5. *Given a pair of vertices $(u, v) \in V^2$, let P be the CP from u to v in G. Then for each $z \in Inn(P)$, the (u, z)-subpath of P is the CP from u to z.*

Proof. Let P' be the (u, z)-subpath of P and w be the predecessor of v in P. We prove our claim by induction on the number of edges from z to v. The base case is the one where $z = w$ (i.e. P' is the (u, w)-subpath of P). This holds from Lemma 4.

Let z' be the predecessor of z in P and let P'' be the (u, z')-subpath of P. For the induction step, we show that if P' is the CP from u to z, then P'' is the CP from u to z'.

By Definition 12 applied to P', there are two cases to consider: $|\mathcal{L}_{uz}| = 1$ (case 1) and $|\mathcal{L}_{uz}| > 1$ (case 2). In case 1, by Lemma 3 applied to P', the shortest path P'' must be the CP from u to z'. In case 2, by Definition 12 (case 2) applied to P', the CP from u to z' is P''. □

Lemma 6. *Given a pair of vertices $(u, v) \in V^2$, let P be the CP from u to v in G. Then for each $z \in Inn(P)$, the (z, v)-subpath of P is the CP from z to v.*

Proof. Let Q be the (z, v)-subpath of P. We prove by contradiction supposing that Q is not the CP from z to v in G. Then there is a shortest path Y which is the CP from z to v in G. Consider the subpath of P from u to z concatenated with Y, and denote such concatenation as Z. Note that, even though the number of vertices of Q and Y may be different, the length of Q and Y is the same, since both are shortest paths. The same applies to P and Z.

Denote the vertices in P and Z as $P = (u = v_1, \ldots, v = v_k)$ and $Z = (u = w_1, \ldots, v = w_l)$. Let v_{k-i} be the vertex of P such that i is maximum, $0 \leq i < k$, and such that the following holds: for all $1 \leq j \leq i$, the vertex v_{k-j} in P is the same as the vertex w_{l-j} in Z (Figure 4.1). For simplicity, denote v_{k-i} as q, v_{k-i-1} as q', and w_{l-i-1} as y'. Note that the edges in the (q, v)-subpaths of P and Z are the same, but (q', q) and (y', q) is not the same edge.

Let Q' and Y' be the (z, q)-subpaths of P and Y, respectively (Figure 4.2). Since we are assuming that Y is the CP from z to v in G, then by Lemma 5, Y' is the CP from z to q in G. Note that $Q' \neq Y'$ (since $Q \neq Y$), and hence, Q' is not the CP from z to q in G. Thus, $\lambda(q') > \lambda(y')$.

From Lemma 5 applied to P, the (u, q)-subpath of P is a CP. But this path is a shortest path such that q' is not the vertex with minimum index among all possible predecessors of q (recall that $\lambda(q') > \lambda(y')$), a contradiction. □

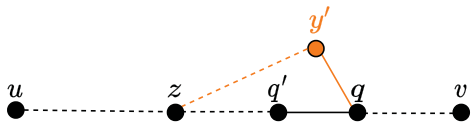


Figure 4.2: Illustration of the shortest path from z to q (in orange), denoted Q', in the proof of Lemma 6.

Theorem 17. *Given a pair of vertices $(u,v) \in V^2$, let P be the CP from u to v in G. Then for each $(u',v') \in V^2$, the (u',v')-subpath of P is the CP from u' to v' in G.*

Proof. Let P' be the (u',v')-subpath of P. From Lemma 6, the (u',v)-subpath of P, denoted Q, is a CP. From Lemma 5, since Q is the CP from u' to v in G, then P' is the CP from u' to v' in G. □

4.2 RANGE SPACE AND VC DIMENSION RESULTS

In this section, we first define the problem in terms of a range space, and then we show that its VC dimension is constant, which directly impacts in the size of the sample to be used by our algorithms. In fact, we show that this sample size only depends on the parameters of quality and confidence, ε and δ, respectively.

Let $n = |V|$ and $\mathcal{T}$ be the set of n Dijkstra trees of G. Recall that such trees are, by definition, composed by canonical paths. The universe X is defined for the set of all branches of Dijkstra trees, i.e.

$$X = \bigcup_{(a,b)\in V^2 : b \neq a} \mathcal{B}_{ab}.$$

For each pair $(u,v) \in V^2$, let p_{uv} be the canonical path from u to v, according to Definition 12. Each range τ_{uv} is defined as $\tau_{uv} = \{\mathcal{B}_{ab} \in X : p_{uv} \in S(\mathcal{B}_{ab})\}$. In other words, we can say that $\mathcal{B}_{ab}$ is in the range of (u,v) if $\mathcal{B}_{ab}$ *passes* through a canonical path between u and v. Let $\mathcal{I} = \{\tau_{uv} : (u,v) \in V^2\}$ be the range set. So, $\mathcal{R} = (X,\mathcal{I})$ is the range space defined for our problem.

Now we show how to plug our range space $\mathcal{R}$ with Definition 4 so we can use Theorem 7 to bound the sample size that is tight enough for the task that we are tackling. We first show in Theorem 18 that $c(u,v) = \Pr_\pi(\tau_{uv})$. For this result, we have that each tree $T_a \in \mathcal{T}$ is sampled with probability $\pi(T_a) = \frac{1}{n}$ and each branch $\mathcal{B}_{ab} \in T_a$ is sampled with probability $\frac{1}{n-1}$, leading to the probability distribution $\pi(\mathcal{B}_{ab}) = \frac{1}{n(n-1)}$ (which is a proper distribution as the sum is equal to 1). Let $\mathbb{1}_{uv}(\mathcal{B}_{ab})$ be the indicator function that returns 1 if there is some canonical path from u to v as subpath of $\mathcal{B}_{ab}$, i.e. $\mathcal{B}_{ab} \in \tau_{uv}$, and 0 otherwise.

Theorem 18. *For $(u,v) \in V^2$, $\Pr_\pi(\tau_{uv}) = c(u,v)$.*

Proof. For fixed $(u,v) \in V^2$ and considering that a branch $\mathcal{B}_{ab} \in X$ is sampled with probability $\pi(\mathcal{B}_{ab}) = \frac{1}{n(n-1)}$, we have

$$\begin{aligned}
\Pr_{\pi}(\tau_{uv}) &= \sum_{T_a \in \mathcal{T}} \sum_{\mathcal{B}_{ab} \in T_a} \pi(\mathcal{B}_{ab}) \mathbb{1}_{uv}(\mathcal{B}_{ab}) \\
&= \frac{1}{n(n-1)} \sum_{T_a \in \mathcal{T}} \sum_{\mathcal{B}_{ab} \in T_a} \mathbb{1}_{uv}(\mathcal{B}_{ab}) \\
&= \frac{1}{n(n-1)} \sum_{a \in V} \sum_{b \in V : b \neq a} \mathbb{1}_{uv}(\mathcal{B}_{ab}) \\
&= \frac{t_{uv}}{n(n-1)} = c(u,v).
\end{aligned}$$

The first equality follows from the fact that the probability that a branch lies on the range τ_{uv} is equal to counting the individual probabilities of each branch that is in τ_{uv}. □

For problems involving shortest paths, such as the ones presented in the work of Riondato and Kornaropoulos (2016) and Lima et al. (2022a), a bound for the sample size using VC dimension can be obtained. The referred work typically apply the same proof structure, having a bound based on the vertex-diameter of a graph G, denoted $\mathrm{diam}_V(G)$, as in Theorem 19 (we present such proof for the sake of completeness). Even though $\mathrm{diam}_V(G)$ might be as large as n, in particular, this bound is exponentially smaller for graphs with logarithmic vertex-diameter, which are common in practice.

Although the bound presented in Theorem 19 depends on a combinatorial structure of G, in this chapter we present in Theorems 20 and 21 a bound that depends only on the desired quality and confidence parameters of the solution. More specifically, for these two theorems we have that $\mathrm{VCDim}(G) = 2$ for a given graph G with respect to a fixed vertex ordering λ, where $\mathrm{VCDim}(G)$ denotes the VC dimension of the range space $\mathcal{R} = (X, \mathcal{I})$ related to a graph G.

Theorem 19. *For a given graph $G = (V,E)$,*

$$VCDim(G) \leq \lfloor 2 \lg diam_V(G) + 1 \rfloor.$$

Proof. Let $\mathrm{VCDim}(G) = k$, where $k \in \mathbb{N}$, because $\mathrm{VCDim}(G)$ is finite. Then, there is $S \subseteq U$ such that $|S| = k$ and S is shattered by $\mathcal{I}$. Each $\mathcal{B}_{ab} \in S$ must appear in 2^{k-1} different ranges in $\mathcal{I}$, from the definition of shattering. On the other hand, $\mathcal{B}_{ab}$ has length at most $\mathrm{diam}_V(G)$. Then the maximum number of subpaths of $\mathcal{B}_{ab}$, denoted $|S(\mathcal{B}_{ab})|$, is $\mathrm{diam}_V(G) \cdot (\mathrm{diam}_V(G) - 1)$. Thus, the branch $\mathcal{B}_{ab}$ lies in at most $|S(\mathcal{B}_{ab})|$ ranges, and therefore,

$$2^{k-1} \leq |S(\mathcal{B}_{ab})| \leq \mathrm{diam}_V(G) \cdot (\mathrm{diam}_V(G) - 1) \leq (\mathrm{diam}_V(G))^2.$$

Solving for k, $\mathrm{VCDim}(G) = k \leq \lfloor 2 \lg \mathrm{diam}_V(G) + 1 \rfloor$. □

For Theorems 20 and 21, we introduce the definition of *meeting path* between two canonical paths P_1 and P_2, and in Lemma 7 we prove that there is only one such path between P_1 and P_2. We use this fact to prove that $\mathrm{VCDim}(G) \leq 2$ in Theorem 20.

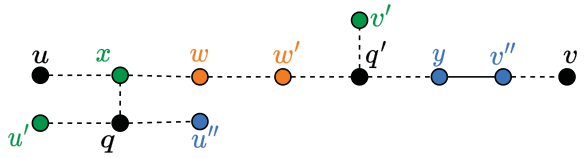


Figure 4.3: Case where $\tau_{xw} \cap S = \{P_1, P_2, P_3\}$, for $P_1 = (u, \ldots, v)$, $P_2 = (u', \ldots, v')$, and $P_3 = (u'', \ldots, v'')$.

Definition 14. *Consider two different canonical paths P_1 and P_2. We say that a canonical path $Z = (z, \ldots, z')$ is a* meeting path *between P_1 and P_2 if Z is a maximal (z, z')-subpath of P_1 and P_2.*

Lemma 7. *Consider two different canonical paths P_1 and P_2 having a meeting path $Z = (z, \ldots, z')$. Then Z is the only meeting path between P_1 and P_2 in G.*

Proof. Let $P_1 = (x, \ldots, x')$ and $P_2 = (y, \ldots, y')$. Suppose by contradiction that P_1 and P_2 have two disjoint meeting paths $A' = (a, \ldots, a')$ and $B' = (b, \ldots, b')$ in G. Without losing of generality, A' is contained in the (x, a') and (y, a')-subpaths of P_1 and P_2, respectively. Correspondingly, B' is contained in the (b, x') and (b, y')-subpaths of P_1 and P_2, respectively.

Suppose that the (a', b)-subpaths of P_1 and P_2 are not the same. Denote such paths as C' and C'', respectively. By Theorem 17, we have that C' and C'' are CPs, since P_1 and P_2 are CPs. But there is only one CP for each pair of vertices of V. Denote the CP from a' to b in G as D. Thus, C' and C'' must correspond to D. Denote by Z the concatenation of A', D, and B'. Then it is a contradiction that A' and B' are maximal, since both paths are contained in Z. Thus, Z is the only meeting path between P_1 and P_2 and it is maximal. □

Theorem 20. *Consider a graph $G = (V, E)$. For any fixed ordering λ over V,*

$$VCDim(G) \leq 2.$$

Proof. Suppose that $\text{VCDim}(G) > 2$. Then there is a set of canonical paths $\mathcal{S} = \{P_1, P_2, P_3\}$ that is shattered by $\mathcal{I}$. These paths are described as $P_1 = \{u, \ldots, v\}$, $P_2 = \{u', \ldots, v'\}$, and $P_3 = \{u'', \ldots, v''\}$. Let W be the (w, w')-subpath of P_1 that is also contained in P_2 and P_3. From the definition of shattering, this path must exist so that $\tau_{ww'} \cap S = \{P_1, P_2, P_3\}$. Let x be the farthest predecessor of w in P_1 such that, w.l.o.g. , the (x, w)-subpath of P_1, denoted X, is also contained in P_2 (but not in P_3). Let y be the farthest successor of P_1 such that a (q', y)-subpath of P_1, denoted Y, is also contained in P_3 (but not in P_2). Note that X and Y must exist so that $\tau_{xw} \cap S = \{P_1, P_2\}$ and $\tau_{q'y} \cap S = \{P_1, P_3\}$.

Suppose that there is a (q, x)-subpath of P_2 that is contained in P_3 but not in P_1, as depicted in Figure 4.3. Since the CP from u' to v' is not the same as the one from v' to u' (and correspondingly for u'' and v''), and P_2 and P_3 must pass through W, then q is not contained in X. From Lemma 7, all the vertices from q to w' must be the same in P_2 and P_3. Hence, P_3 goes through x, and from our initial assumption, P_2 does not have any intersection with a vertex that comes before x in P_1. Besides, P_3 goes through q' and Y. Therefore, any subpath of P_2 starting in q is also a subpath of P_3. This contradicts that $\tau_{xw} \cap S = \{P_1, P_2\}$ since $\tau_{xw} \cap S = \{P_1, P_2, P_3\}$.

Consider now the (q', v')-subpath of P_2, denoted P_2'. Suppose that P_3 has an intersection with a (r, r')-subpath of P_2' (Figure 4.4). From our initial assumption, P_3 goes through W and Y, so it passes through q', and q' reaches r. Hence, from Lemma

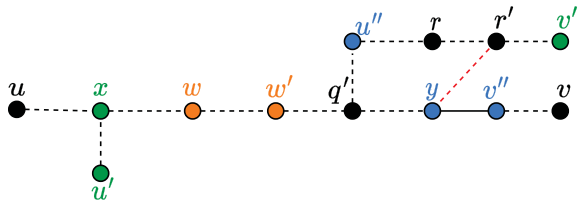


Figure 4.4: Case where $\tau_{q'y} \cap S = \{P_1\}$, for $P_1 = (u,\ldots,v)$, $P_2 = (u',\ldots,v')$, and $P_3 = (u'',\ldots,v'')$. The red dashed path correspond to a shortest path that cannot happen.

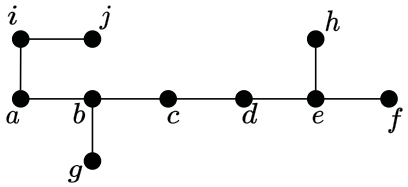


Figure 4.5: Graph with $\text{VCdim}(G) \geq 2$.

7, all the vertices from q' to r' must be the same in P_2 and P_3. In this case, P_3 does not contain a (r', w)-subpath, otherwise P_1 and P_3 would form a cycle starting and ending in r'. Besides, P_3 does not have a (r', y)-subpath or a (r', y')-subpath, for any $y' \in \text{Inn}(Y)$, otherwise that would be two different CPs from r' to y'. Hence, P_3 does not pass through the (q', y)-subpath of P_1, contradicting that $\tau_{q'y} \cap S = \{P_1, P_3\}$ since $\tau_{q'y} \cap S = \{P_1\}$. □

Theorem 21. *Consider a graph $G = (V, E)$. For any fixed ordering λ over V,*

$$VCDim(G) \geq 2.$$

Proof. Consider the graph as in Figure 4.5. Then, for $P_1 = (a, b, c, d, e, f)$, $P_2 = (g, b, c, d, e, h)$, and $S = \{P_1, P_2\}$, we have: $\tau_{ac} = \{P_1\}$, $\tau_{gc} = \{P_2\}$, $\tau_{cd} = \{P_1, P_2\}$, and $\tau_{aj} = \emptyset$. □

4.3 ALGORITHMS

For an undirected graph $G = (V, E)$ with non-negative edges weights, with $n = |V|$ and $m = |E|$, we first present in Section 4.3.1 a modified version of Dijkstra's algorithm which takes into consideration a given vertex ordering λ, and then we show that this algorithm correctly computes the canonical paths in the corresponding Dijkstra tree it returns. Then, in Section 4.3.2 we present the relaxed APSP algorithm that returns, with probability at least $1 - \delta$, the computation of shortest paths with centrality least ε.

4.3.1 Modified Dijkstra

We describe the modified Dijkstra's algorithm with respect to a given vertex ordering λ, and then we show the correctness of this modification.

The algorithm returns canonical paths that are uniquely defined for each pair of vertices of a graph $G = (V, E)$. Let s be the root of the Dijkstra tree T_s returned by the algorithm. Then s is the source of every shortest path in T_s. The main difference between the modified Dijkstra's algorithm and the classical one is the tie-breaking criterion on the selection of edges to be added in a shortest path. More specifically, recall that Dijkstra's algorithm maintains in every step a set S of vertices that have its estimated distance from s determined. The vertex v in $V \setminus S$ with minimum distance from s is

selected to be added in S. An edge $(u, w) \in E$ is relaxed if the minimum distance from s to u plus the weight of (u, w) improves the minimum distance from s to w.

In a given step of the modified Dijkstra, if more than one vertex in $V \setminus S$ have the same value of the minimum distance estimation, then the one with minimum index in λ is chosen to be added in S. In the case of edges relaxation, consider a vertex $y \in V \setminus S$ and its predecessor y' in some shortest path from s to y. An edge $(u, y) \in E$ is relaxed also considering the case where the minimum distance from s to u plus the weight of (u, y) has the same value as the minimum distance estimation from s to y' plus the weight of (y', y). In this case, the tie-breaking depends on which vertex between u and y' has the minimum index in λ.

Theorem 22 shows that the modified Dijkstra's algorithm correctly computes all the canonical paths from a source s to any other vertex in V with respect to λ. Note that S is a priority queue that is also modified to give higher priority to vertices with lowest indexes in λ in the case of ties in the vertices minimum distance estimations. We observe, however, that these modifications do not increase the running time of the priority queue operations.

Theorem 22. *All shortest paths computed by a modified Dijkstra's algorithm with respect to a given vertex ordering λ are CP.*

Proof (Sketch). Similar to the proof of correctness of the original Dijkstra's algorithm presented in Theorem 22.6 of Cormen et al. (2022), the proof is by induction on the size of S.

Let s be the source vertex. For each $u \in V$, let $\tilde{d}(s, u)$ the estimated minimum distance from s to u in a given step of the algorithm, and let $d(s, u)$ be the exact minimum distance from s to u. For $|S| = 0$, the set S is empty and then this base is trivially true. For the base where $|S| = 1$, we have $S = \{s\}$, and then $\tilde{d}(s, s) = d(s, s) = 0$. Besides, s does not have a predecessor, since it is the source, so the base is also true for this case. For the inductive step, we have the following hypothesis: for all $v \in S$, we have that $\tilde{d}(s, v) = d(s, v)$ and the predecessor of v in the Dijkstra tree of s is the one with minimum index in λ. Proving that $\tilde{d}(s, v) = d(s, v)$ follow the same arguments of the proof of correctness presented by Cormen et al. (2022) for the original Dijkstra's algorithm.

To prove that the predecessor of v in the Dijkstra tree of s, denoted v', is the one with minimum index in λ among all possible predecessors of v, we prove that all edges (z, v) where $\tilde{d}(s, v) = \tilde{d}(s, z) + \omega(z, v)$ were examined when the edge (v', v) were relaxed. Consider, by contradiction, that there is some vertex u' that has the minimum index in λ among all possible predecessors of v, but that the edge (u', v) was not examined before vertex v is added to S. If the edge (u', v) was not examined, then v was added in S before u'. In this case, this happened either because $\tilde{d}(s, v) < \tilde{d}(s, u')$ or because $\tilde{d}(s, v) = \tilde{d}(s, u')$ and $\lambda(v) < \lambda(u')$. However, in both cases, then u' could not be the predecessor of v, since $\tilde{d}(s, u')$ should be strictly smaller than $\tilde{d}(s, v)$ to be considered as a possible predecessor of v. Hence, all $y \in S$ with $\tilde{d}(s, y) < \tilde{d}(s, v)$ should have been examined before v, and hence, v' is the predecessor of v with minimum index in λ among all such vertices. This value never changes again once v is added in S. □

4.3.2 Computing Shortest Paths with High Centrality

Given constant $0 < \varepsilon, \delta < 1$, Algorithm 5 computes, with probability $1 - \delta$, the distances between pair of vertices with centrality at least ε. We also briefly describe the necessary

modifications on the algorithm so that the shortest path associated to such distances be also computed.

Algorithm 5 ProbabilisticAllPairsShortestPaths(G,ε,δ)

Input: weighted graph $G = (V, E)$ with $n = |V|$, constant parameters $0 < \varepsilon, \delta < 1$.
Output: distance d_{uv}, for each $(u, v) \in V^2$ s.t. $c(u, v) > \epsilon$, with probability $1 - \delta$.
1: **for** $i \leftarrow 1$ **to** $\left\lceil \frac{c}{\varepsilon}\left(2\ln\frac{1}{\varepsilon} + \ln\frac{1}{\delta}\right)\right\rceil$ **do**
2: sample $a \in V$ with probability $1/n$
3: $T_a \leftarrow$ singleSourceShortestPaths(a) // modified Dijkstra
4: sample $b \in V \setminus \{a\}$ with probability $1/(n-1)$
5: $\mathcal{B}_{ab} \leftarrow$ shortest path from a to b in T_a
6: **for each** $(u, v) \in \mathcal{B}_{ab} \times \mathcal{B}_{ab}$ **do** // u closer to a, v closer to b
7: $d_{uv} \leftarrow d_{av} - d_{au}$ // d_{au} and d_{av} come from T_a
8: **return** each d_{uv} in the distances table

Theorem 23. *Consider a (u, v)-path such that $c(u, v) \geq \varepsilon$. Algorithm 5 computes the exact distance between u and v with probability $1 - \delta$.*

Proof. Algorithm 5 samples several branches and we first assume that such samples are an ε-net (we show later that this is indeed true). Recalling the range space modeling (Section 4.3.2), the sample of branches is denoted by S, and the (u, v)-path is related to a range τ_{uv}.

As, by lines 2 and 4, the branch is sampled with probability $1/n(n-1)$, then by Theorem 18, we have that $c(u, v) = \Pr(\tau_{uv})$. Thus, as $c(u, v) \geq \varepsilon$, so $\Pr(\tau_{uv}) \geq \varepsilon$. As we are assuming that the sample is an ε-net, by Definition 4, then $|\tau_{uv} \cap S| \geq 1$ for all τ_{uv} such that $\Pr(\tau_{uv}) \geq \varepsilon$. That is, since $c(u, v) \geq \varepsilon$ then at least one branch of the sample S contains the (u, v)-path. If a branch $\mathcal{B}_{ab}$ in S contains the (u, v)-path, then in line 3 the exact distance between u and v is computed, since the (u, v)-path which is a subpath of the shortest path from a to b is also minimal, so its distance d_{uv} can be computed as $d_{av} - d_{au}$.

Now it remains to prove that the sample S is indeed an ε-net. Note that in lines 1–7, the loop is executed $r = \left\lceil \frac{c}{\varepsilon}\left(2\ln\frac{1}{\varepsilon} + \ln\frac{1}{\delta}\right)\right\rceil$ times, so our sample has at least size r. By Theorems 7, 20, and 21, this sample size is sufficient for it to be an ε-net with probability at least $1 - \delta$. □

Theorem 24. *Algorithm 5 runs in $\mathcal{O}(m + n\log n + (diam_V(G))^2)$ time.*

Proof. Lines 2, 4, and 5 takes linear time. Line 3 (the modified Dijkstra) runs in $\mathcal{O}(m + n\log n)$, as the modifications do not change the running time of the original Dijkstra's algorithm. Each iteration of the loop in line 6 takes time $\mathcal{O}((\mathrm{diam}_V(G))^2)$ since the length of $\mathcal{B}_{ab}$ cannot be greater than the vertex diameter of the graph. The distances returned by Dijkstra's algorithm in line 3 are stored in a table d. Since operations of insertion, deletion, and search on this data structure take time $\mathcal{O}(1)$, then updating table d takes $\mathcal{O}(1)$ time. Assuming constant ε and δ, the number of loop iterations in lines 1–7 is constant, and the result follows. □

As it is common to APSP and search algorithms, Algorithm 5 also constructs a data structure from which, for all vertices (u, w), a shortest path from u to w can be retrieved. We can store the predecessors of each vertex that is in $\mathcal{B}_{ab}$ so that a

(u,v)-subpath of $\mathcal{B}_{ab}$ can be retrieved by a backward traversing from v to u on these predecessors. This modification does not change the execution time of the original algorithm.

It is interesting to consider what is the smallest value of ε for which our algorithm would still perform faster than the best conjectured time in literature for any algorithm for the APSP problem. To analyze this, the ε value must reflect on the execution time, and can no longer be a constant. Therefore, we now consider that ε is a function of n, denoted $\varepsilon(n)$.

Let k be the sample size (which impacts on the number of times line 1 of Algorithm 5 is executed). Since ε is no longer considered constant, so $k = \mathcal{O}\left(\frac{1}{\varepsilon(n)} \ln \frac{1}{\varepsilon(n)}\right)$. Thus, our algorithm performs in $\mathcal{O}(k \cdot (m + n \log n + (\text{diam}_V(G))^2))$. To simplify the analysis, we assume the worst case of our algorithm ($m = \mathcal{O}(n^2)$), whose time is $\mathcal{O}(k \cdot n^2)$.

As the best conjectured time is $\mathcal{O}(n^{3-c})$, for a constant $c > 0$ (Williams, 2018), then we are looking for the value of $\varepsilon(n)$ such that the time of our algorithm is upper bounded by $\mathcal{O}(n^{3-c})$, i.e. $\mathcal{O}(k \cdot n^2) = \mathcal{O}(n^{3-c})$. Thus, $k = n^{1-c}$, and then

$$\frac{1}{\varepsilon(n)} \ln \frac{1}{\varepsilon(n)} = n^{1-c}.$$

Solving for $\varepsilon(n)$, we have $\varepsilon(n) = \frac{W_0(n^{1-c})}{n^{1-c}}$, where $W_0(n^{1-c})$ is the branch 0 of the Lambert-W function (Weisstein, 2013). To simplify the notation, let $n' = n^{1-c}$. If $n' \geq e$, then a known bound (Hoorfar and Hassani, 2008) for $W_0(n')$ is $W_0(n') = \ln n' - \ln \ln n' + \Theta\left(\frac{\ln \ln n'}{\ln n'}\right)$. Therefore $\varepsilon(n) = \frac{\ln n' - \ln \ln n' + \Theta\left(\frac{\ln \ln n'}{\ln n'}\right)}{n'}$.

The smallest value for the centrality is $1/n(n-1)$, which is the case for a path in which only itself is contained in it. So, to compute the distance of paths with such small centrality, we have to use ε so small that the execution time exceeds that of the best existing algorithms (Williams, 2018; Pettie and Ramachandran, 2002). On the other hand, by the reasoning above, we can use ε as small as $\frac{\ln n}{n}$. Intuitively and crudely, this means that almost all distances are computed, and the execution time is still kept below the best conjectured time. Finally, it is worth noting that the few non-computed paths are not arbitrary. In fact, they have low centrality, i.e. they are neither central nor considered important.

4.4 ESTIMATING THE SHORTEST PATH CENTRALITY

The main objective of this chapter is the computation of the distance of shortest paths that have high centrality. However, one might be interested in computing the value of the centrality of such shortest paths. In this section we give the outline of how to adapt our algorithm so that the centrality of each $(u,v) \in V^2$ can be estimated within ε error, with probability at least $1-\delta$, for $0 < \varepsilon, \delta < 1$. For this task we can use the more general result of Theorem 7 applied to the notion of *ε-sample*, which states that a collection of elements $S \subseteq U$ sampled with respect to π with $|S| = \frac{c}{\varepsilon^2}\left(k + \ln \frac{1}{\delta}\right)$ is an ε-sample with probability at least $1-\delta$, where k is the VC dimension of the range space that models the problem. More precisely, an ε-sample generalizes an ε-net in the sense that it not only intersects ranges of a sufficiently large size but it also guarantees the right relative frequency of each range in $\mathcal{I}$ within the sample S.

The idea is that after building a sample of size $r = \lceil \frac{c}{\varepsilon^2}(2 + \ln \frac{1}{\delta}) \rceil$, we build a counting table $\tilde{t}$ to estimate the number of branches that contain a certain canonical path as a subpath. More specifically, the entry $\tilde{t}_{uv}$ estimates the value t_{uv} (recall Definition 13 in Section 4.1) for the pair of vertices (u, v). For this, the value $\tilde{t}_{uv}$ is incremented by $1/r$ in lines 6 and 7 if the branch $\mathcal{B}_{ab}$ contains the canonical path between u and v as subpath. At the end of the algorithm, if a pair of vertices (u, v) is not included in the table, the estimation for the centrality is assumed to be zero. This modification does not change the asymptotic running time of Algorithm 5. We state that in Corollary 2.

Corollary 2. *Given an undirected graph $G = (V, E)$ with non-negative edge weights, with $n = |V|$, and a sample of size $r = \lceil \frac{c}{\varepsilon^2}(2 + \ln \frac{1}{\delta}) \rceil$, Algorithm 5 has running time $\mathcal{O}(m + n \log n + (diam_V(G))^2)$ for computing a table from which the centrality estimation of each $u, v \in V^2$ can be retrieved.* □

4.5 CONCLUDING REMARKS

In this chapter we presented a range space having the domain composed by the shortest paths of a graph G where there is one shortest path for each pair of vertices in G. This shortest path is canonical, i.e. it is fixed according to a given vertex ordering. We show that the VC dimension of such range space is 2. We show that this result can be applied to bound the sample size required for an approximation algorithm for a relaxed version of the All-Pairs Shortest Path problem (APSP). In this version, we compute, with probability at least $1 - \delta$, the distance of shortest paths of G having centrality at least ε, for $0 < \varepsilon, \delta < 1$. We present a $\mathcal{O}(m + n \log n + (\text{diam}_V(G))^2)$ running time algorithm for this task. We show that a sample of shortest paths of size $\lceil \frac{c}{\varepsilon}\left(2 \ln \frac{1}{\varepsilon} + \ln \frac{1}{\delta}\right) \rceil$ is sufficient for achieving the desired result. So, in an application where one might be interested only in computing *central* shortest paths the algorithm is rather efficient and it depends only on the parameters ε and δ (classical approaches in literature based in union bound, for example, typical require sample sizes that depend on the size of the input).

An open question that we are particularly interested is the connection between ε and n or $\text{diam}_V(G)$ for specific input distributions. For the general case, trivially setting $\varepsilon = \frac{1}{n(n-1)}$, by Theorem 7 (ii), we have a guarantee that distance of every shortest path in G is computed with probability $1 - \delta$, but that would increase the algorithm complexity to $\tilde{O}(n^3)$. This may not be a surprise since APSP may not admit a strictly subcubic algorithm. In fact, we show that ε must be at least $\frac{\ln n' - \ln \ln n' + \Theta\left(\frac{\ln \ln n'}{\ln n'}\right)}{n'}$, where $n' = 1 - c$, so that the running time of our algorithm be $\mathcal{O}(n^{3-c})$, for some $c > 0$.

5 LOCAL CLUSTERING COEFFICIENT

The occurrence of *clusters* in networks is a central field of investigation in the area of network theory (Holland and Leinhardt, 1971; Watts and Strogatz, 1998). The existence of such phenomena motivated the creation of a variety of measures in order to quantify its prevalence; the *clustering coefficient* (Easley and Kleinberg, 2010; Barabási and Pósfai, 2016; Newman, 2010) is one of the most popular of these measures.

There are global and local versions of the clustering coefficient. Given a graph, its *global clustering coefficient* is a value that quantifies the overall clustering of the graph in terms of the number of existing triangles. If the objective, however, is to analyze features of complex networks such as modularity, community structure, assortativity, and hierarchical structure, then the concept of *local clustering coefficient* is a better fit. This measure quantifies the degree in which a vertex is a part of a cluster in a graph. Simply speaking, the measure is related to the ratio of the number of triangles existing in the neighborhood of the target vertex to the total number of pair of nodes in the neighborhood (Figure 5.1). A precise definition for this measure is provided in Definition 15.

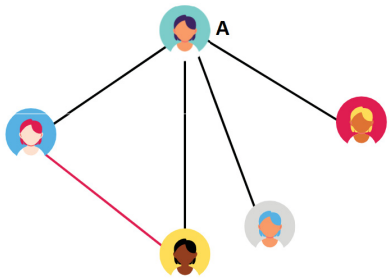


Figure 5.1: Example of a graph representing a social network (each vertex is a person and each edge represents that two people are friends). On this example, the local clustering coefficient of person A is equal to 1/6.

The local clustering coefficient was originally proposed by Watts and Strogatz (1998) in order to determine if a graph has the property of being *small-world*. Intuitively, this coefficient measures how close the neighborhood of a vertex is to being a clique. Over the years, many variants of this measure have been proposed, making it somewhat difficult to provide a unified comparison between all these approaches under the light of algorithmic complexity.

One of these variations is the study of Soffer and Vazquez (2005) on the influence of the degree of a vertex on the local clustering computation, a modification on the original measure where the *degree-correlation* is filtered out. The work of Li et al. (2018) provides a measure combining the local clustering coefficient and the local degree sum of a vertex, but focused on a specific application of influence spreading. Other extensions of the measure and their applications in particular scenarios include link prediction (Gupta and Sardana, 2015; Wu et al., 2016) and community detection (Nascimento, 2014; Zhang et al., 2014; Pan et al., 2019; Ji et al., 2020; Liu and Xia, 2020). In the theoretical front, working on random graphs, Kartun-Giles and Bianconi (2019) gives a statistical analysis of the topology of nodes in networks from different application scenarios. There are also many recent bounds for the average clustering of *power-law*

graphs (Fronczak et al., 2003; Bloznelis, 2013; Krot and Ostroumova Prokhorenkova, 2015; Iskhakov et al., 2020), a graph model that represents many social and natural phenomena.

The algorithmic complexity of the exact computation of the local clustering coefficient of each vertex of a graph is typically associated with the local triangle counting problem, since the second problem can be translated to the first (Becchetti et al., 2010). A brute-force algorithm that lists all vertex triples to check how many of them are triangles runs in cubic time. These running time can be improved to $\mathcal{O}(m^{2w/(w+1)})$ (Alon et al., 1997), where w is 2.373 is the exponent of matrix multiplication, or to $\mathcal{O}(\alpha(G) \cdot m)$ (Chiba and Nishizeki, 1985), where $\alpha(G)$ is the arboricity of graph G, without using matrix multiplication in this case. Since $\alpha(G) = \mathcal{O}(\sqrt{m})$, then this later algorithm runs in $\mathcal{O}(m^{3/2})$ in the worst case.

In this chapter, however, we are interested in faster approximation algorithms for obtaining good quality estimations. Let $n = |V|$ and $m = |E|$ in a graph G. In the work of Kutzkov and Pagh (2013), the authors show an ε-approximation streaming algorithm to estimate, with probability $1-\delta$, the local clustering coefficient of each vertex of degree at least d in expected time $\mathcal{O}(\frac{m}{\alpha\varepsilon^2} \log \frac{1}{\varepsilon} \log \frac{n}{\delta})$, where α is the local clustering coefficient of such vertex. In this case, ε and δ are not treated as constants. The work of Buriol et al. (2007) also proposed a streaming algorithm to obtain a $(1 \pm \varepsilon)$-approximation for the global clustering coefficient of a graph. Their algorithm requires one pass over the stream of edges of such graph. In the work of Zhang et al. (2017), the authors propose an ε-approximation *MapReduce*-based algorithm for the local clustering coefficient, and empirically compare its performance with other approximation algorithms designed using this type of approach (Kolda et al., 2014; Seshadhri et al., 2013).

For the local triangle counting problem, the work of Becchetti et al. (2010) presents semi-streaming algorithms to compute a relative-error approximation to the number of triangles of each vertex of a graph in time $\mathcal{O}(m \cdot k)$, where k is the number of passes over the data in the algorithms. In fact, approximation algorithms for triangle counting are usually based on random-walk, streaming-based methods, graph sparsification, triple sampling, vertex/edge sampling, and linear-algebra based methods. Al Hasan and Dave (2018) provide a nice survey on such methods, including exact and approximation algorithms for the global triangle counting problem.

Results for computing the top k vertices with the highest local clustering coefficient were also proposed (Brautbar and Kearns, 2010; Zhang et al., 2015; Li et al., 2017). In particular, Zhang et al. (2015) use VC dimension and the ε-sample theorem on their algorithm analysis, but in a different sample space than the one that we are facing here, and for a scenario which is not exactly the one that we are tackling. In fact, sample complexity analysis has been shown to be an effective tool in the design of some graph algorithms, e.g. the computation of betweenness (Riondato and Kornaropoulos, 2016; Riondato and Upfal, 2018) and percolation centralities (Lima et al., 2020, 2022a).

In this chapter we present an algorithm that samples edges from an input graph G and, for fixed constants $0 < \varepsilon, \delta, p < 1$, outputs an estimation $\tilde{l}(v)$ for the exact value $l(v)$ of the local clustering coefficient of each vertex $v \in V$. The results respect $|l(v) - \tilde{l}(v)| \leq \varepsilon l(v)$, with probability at least $1-\delta$ whenever $l(v)$ is at least $pm/\binom{\delta_v}{2}$, where δ_v is the degree of v. The main theme in this chapter is that, by using Vapnik–Chervonenkis (VC) dimension theory, we can obtain an upper bound for the sample size that is tighter than the ones given by standard Hoeffding and union-bound sampling

techniques. In particular, we show that the sample size does not depend on the size of G, but on a specific property of it, more precisely, its maximum degree Δ.

In Section 5.1 we give a definition for the VC dimension of the range space related to a graph and show in Theorem 25 that, for any graph, the VC dimension is at most $\lfloor \lg(\Delta - 1) \rfloor + 1$. The sample size used in the algorithm depends, roughly speaking, on this value. In Corollary 3, we show that our analysis is tight by presenting an explicit construction of a class of graphs for which the VC dimension reaches this upper bound. Even so, we also provide a tighter analysis for the case in which the input graph belongs to certain graph classes. In the class of *bounded-degree graphs* the VC dimension is bounded by a constant. In the case of *planar graphs*, we show, in Corollary 4, that the VC dimension is at most 2. We also describe, in this section, the proposed algorithm that estimates the local clustering coefficient of each vertex using VC dimension theory (Algorithm 6).

In Section 5.2, we show that the running time for the general case of our algorithm is $\mathcal{O}(\Delta \lg \Delta + m)$. In Corollaries 5 and 6 we present an analysis for planar graphs and for bounded-degree graphs, cases where the running time drops to, possibly, sublinear time. In the case of planar graphs, the Algorithm 6 has running time $\mathcal{O}(\Delta)$. In the case of bounded-degree graphs the running time is $\mathcal{O}(1)$ if a bound for the value of Δ is given as a part of the input, and $\mathcal{O}(n)$ otherwise.

5.1 ESTIMATION FOR THE LOCAL CLUSTERING COEFFICIENT

In this section, we first formally present the definition of local clustering coefficient, and then we define the range space associated to a graph G and its corresponding VC dimension. At the end of this section, we describe the proposed approximation algorithm.

5.1.1 Preliminaries

Let $G = (V, E)$ be a graph where V is the set of vertices and E the set of edges. For each vertex $v \in V$, let δ_v be the degree of v, and $\Delta_G = \max_{v \in V}\{\delta_v\}$ the maximum degree of the graph G. When the context is clear, we simply use Δ instead of Δ_G. We refer to a triangle as being a complete graph with three vertices. Given $v \in V$, we let T_v be the number of triangles that contain v.

Definition 15. *(Local Clustering Coefficient) Given a graph $G = (V, E)$, the* local clustering coefficient *of a vertex $v \in V$ is*

$$l(v) = \frac{2T_v}{\delta_v(\delta_v - 1)}.$$

5.1.2 Range Space and VC dimension

Let $G = (V, E)$ be a graph. The range space $\mathcal{R} = (X, \mathcal{I})$ associated with G is defined as follows. The universe X is defined to be the set of edges E. We define a range τ_v, for each $v \in V$, as $\tau_v = \{e \in E :$ both endpoints of e are neighbors of v in $G\}$, and the range set corresponds to $\mathcal{I} = \{\tau_v : v \in V\}$. For the sake of simplicity, we often use VCDim(G) – instead of VCDim$(\mathcal{R})$ – to denote the VC dimension of the range space $\mathcal{R}$ associated with G.

Theorem 25 shows an upper bound for VCDim(G).

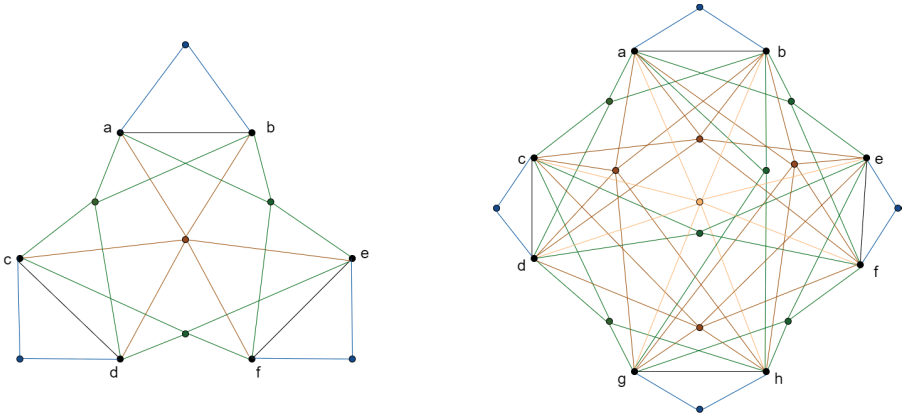


Figure 5.2: The first two graphs of the construction of the family $\mathcal{G}$. In the case of G_3 (left), the edges of S are $e_1 = \{a,b\}$, $e_2 = \{c,d\}$, and $e_3 = \{e,f\}$. In the case of G_4 (right), the edges of S are $e_1 = \{a,b\}$. $e_2 = \{c,d\}$, $e_3 = \{e,f\}$, and $e_4 = \{(g,h\}$. Non-indexed vertices are labeled and depicted in black. We depict the indexed vertices in different colors, depending on the size of its neighborhood in S.

Theorem 25. *$VCDim(G) \leq \lfloor \lg(\Delta - 1) \rfloor + 1$.*

Proof. By definition, an edge $e \in E$ belongs to a range τ_v if both endpoints of e, say, a and b, are neighbors of v. That is, the number of ranges that contain e corresponds to the common neighbors of a and b. The maximum number of common neighbors a pair of vertices may have is Δ. Therefore, e is contained in at most $\Delta - 1$ ranges. Assuming that $\text{VCDim}(\mathcal{R}) = d$, then from Definition 2, the edge e must appear in 2^{d-1} ranges. We have

$$2^{d-1} \leq \Delta - 1 \Longrightarrow d - 1 \leq \lg(\Delta - 1) \Longrightarrow d \leq \lfloor \lg(\Delta - 1) \rfloor + 1 .$$ □

One may ask when the bound given in Theorem 25 is tight. We now present an explicit construction of a family of graphs $\mathcal{G} = (G_d)_{d \geq 3}$ in order to show that this bound is tight with relation to Δ. A graph G_d, for $d \geq 3$, of this family is constructed as follows. Initially, we create d disjoint edges $e_1, \ldots, e_d$. The endpoints of these edges are called *non-indexed vertices*. For every non-empty subset of k edges $e_{i_1}, \ldots, e_{i_k}$, for $1 \leq i_1 < i_2 < \ldots < i_k \leq d$, we create a vertex $v_{(i_1, i_2, \ldots, i_k)}$ and connect it to both endpoints of each edge in the subset. These vertices are called *indexed vertices*. Figure 5.2 illustrates G_3 and G_4.

Claim 1. $\Delta_{G_d} = 2^{d-1} + 1$.

Proof. A vertex v in a graph G_d can be either indexed or non-indexed. We analyze each case separately.

Let v be a non-indexed vertex that is an endpoint of an edge e_j. W.l.o.g., we may assume that $j = 1$. The vertex v is adjacent to every indexed vertex with indices of the form $(1, i_1, \ldots, i_k)$. The first index is fixed, so there are 2^{d-1} indices of this form. So v is adjacent to 2^{d-1} indexed vertices. Also, v is adjacent to the other endpoint of e_1. Therefore, the degree of any non-indexed vertex is $2^{d-1} + 1$.

The degree of an indexed vertex cannot be larger than $2d$, since such vertex is adjacent to, at most, both endpoints of each edge $e_1, \ldots, e_d$. Since $2^{d-1} + 1 \geq 2d$, the result follows. □

Theorem 26. *For every $d \geq 3$, $VCDim(G_d) \geq \lfloor \lg(\Delta_{G_d} - 1) \rfloor + 1$.*

Proof. Remember, $\mathcal{R} = (X, \mathcal{I})$, where $X = E$ and $\mathcal{I} = \{\tau_v : v \in V\}$ where $\tau_v = \{e \in E :$ the endpoints of e are neighbors of v in $G\}$. First, we present a sample $S \subseteq X$, $|S| = d$, which is shattered, i.e. $|\mathcal{I}_S| = 2^d$, concluding that the VC dimension is at least d. After that, we show that $d = \lfloor \lg(\Delta_{G_d} - 1) \rfloor + 1$, which proves the theorem.

Let $S = \{e_1, \ldots, e_d\}$. Consider an indexed vertex $v' = v_{(i_1, i_2, \ldots, i_k)}$. By the construction of the graph, we have that $S \cap \tau_{v'} = \{e_{i_1}, \ldots, e_{i_k}\}$, for all $\tau_{v'}$, i.e. there is a one-to-one mapping of each v' to each $S \cap \tau_{v'}$. Since there are $2^d - 1$ indexed vertices v' (there is an indexed vertex for every subset except for the empty set), then there are $2^d - 1$ different intersections. Finally, the intersection that generates the empty set can be obtained by $S \cap \tau_{v''}$, where v'' is any non-indexed vertex. In other words,

$$|\{S \cap \tau_v : \tau_v \in \mathcal{I}\}| = |\mathcal{I}_S| = 2^d,$$

i.e. $\text{VCDim}(G_d) \geq d$. Now, using Claim 1, we have that

$$\lfloor \lg(\Delta_{G_d} - 1) \rfloor + 1 = \lfloor \lg(2^{d-1} + 1 - 1) \rfloor + 1 = \lfloor d - 1 \rfloor + 1 = d. \qquad \square$$

Combining Theorems 25 and 26, we conclude that the VC dimension of the range space is tight, as stated by Corollary 3.

Corollary 3. *For every $d \geq 3$, there is a graph G such that*

$$VCDim(G) = d = \lfloor \lg(\Delta - 1) \rfloor + 1. \quad \square$$

Next we define a more general property that holds for a graph G_d.

Property P We say that a graph $G = (V, E)$ has the *Property P* if exists $S \subseteq E$, $|S| \geq 3$, such that:

(i) For each $e = \{u, v\} \in S$, e has at most one endpoint that is also an endpoint of other edge in S, i.e. $|e \cap (S \setminus \{e\})| \leq 1$.

(ii) For each subset $S' \subseteq S$, there is at least one vertex $v_{S'}$ that is adjacent to both endpoints of each edge of S'.

For every $d \geq 3$, Theorem 27 gives conditions based on Property P that a graph must obey in order to have VC dimension at least d.

Theorem 27. *Let G be a graph. If $VCdim(G) \geq 3$, then G has Property P.*

Proof. We prove the contrapositive of the statement, i.e. we show that if G does not have Property P, then $\text{VCdim}(G) < 3$. Note that if we assume that G does not have Property P, then for all $S \subseteq E, |S| \geq 3$, we have that either condition (i) or condition (ii) is false.

If it is the case that (ii) is false, then for all $S \subseteq E, |S| \geq 3$, there is a set $S' \subseteq S$ such that there is no $v_{S'} \in V$ which is adjacent to both endpoints of each edge in S'. We have that the number of subsets of S is $2^{|S|}$, so G must have at least $2^{|S|}$ vertices so that $\mathcal{I}_S = 2^{|S|}$. From the definition of shattering, if $\mathcal{I}_S < 2^{|S|}$, then it is not possible that $\text{VCdim}(G) \geq |S|$. Since $|S| \geq 3$, it cannot be the case that $\text{VCdim}(G) \geq 3$.

Now consider the case where (i) is false. In this case, for all $S \subseteq E, |S| \geq 3$, there is an edge $e = \{u, v\} \in S$ where both u and v are endpoints of other edges in S (i.e. $|e \cap (S \setminus \{e\})| = 2$). We name such edge $e_2 = \{b, c\}$. Suppose w.l.o.g. that e_2 shares its endpoints with the edges $e_1 = \{a, b\}$ and $e_3 = \{c, d\}$. Then every triangle containing e_1

and e_3 necessarily contains e_2. Denote by z the vertex which forms triangles with e_1 and e_2. Then z also forms a triangle with e_2, since it is adjacent to both b and c, which are the endpoints of e_2. Hence, the subset $\{e_1, e_3\}$ cannot be generated from the intersection of $\mathcal{I}$ with e_1, e_2, and e_3. Therefore it cannot be the case that $\text{VCdim}(G) \geq 3$. □

Although Theorem 25 gives a tight bound for the VC dimension, if we have more information about the type of graph that we are working, we can prove better results. In Corollary 4, we show that if G is a graph from the class of planar graphs, then the VC dimension of G is at most 2. Another very common class of graphs where we can achieve a constant bound for the VC dimension is the class of *bounded-degree graphs*, i.e. graphs where Δ is bounded by a constant. For this class, the upper bound comes immediately from Theorem 25.

Note that, even though planar graphs and bounded-degree graphs are both classes of sparse graphs, such improved bounds for the VC dimension for these classes do not come directly from the sparsity of these graphs, since we can construct a (somewhat arbitrary) class of sparse graphs $\mathcal{G}'$ where the VC dimension is as high as the one given by Theorem 25. The idea is that $\mathcal{G}' = (G'_d)_{d\geq 3}$, where each graph G'_d is the union of G_d with a sufficiently large sparse graph. In the other direction, one should note that dense graphs can have small VC dimension as well, since complete graphs have VC dimension at most 2. This comes from the fact that complete graphs do not have the Property P. In fact, for a K_q, $q \geq 4$, the VC dimension is exactly 2, since any set of two edges that have one endpoint in common can be shattered in this graph.

Corollary 4. *If G is a planar graph, then $VCDim(G) \leq 2$.*

Proof. We prove that the VC dimension of the range space of a planar graph is at most 2 by demonstrating the contrapositive statement. More precisely, from Theorem 27, we have that if $\text{VCDim}(G) \geq 3$, then G has *Property P*. In this case we show that G must contain a subdivision of a $K_{3,3}$, concluding that G cannot be planar, according to the Theorem of Kuratowski (West, 2000).

From Theorem 27, G has a subset of edges $\{e_1, e_2, e_3\}$ respecting conditions (i) and (ii) of *Property P*. Let $e_1 = \{a, b\}$, $e_2 = \{c, d\}$, and $e_3 = \{e, f\}$. Note that these three edges may have endpoints in common. By *condition (i)*, we may assume that $a \neq c \neq e$. By symmetry, w.l.o.g. , there are three possibilities for the vertices b, d, and f: (1) they are all distinct vertices, (2) we have $d = f$, but $b \neq f$, and (3) they are the same vertex, i.e. $b = d = f$. In Figure 5.3 we show three graphs, one for each of these three possible configurations for the arrangement of edges e_1, e_2, and e_3. By *condition (ii)* there are at least four vertices, say, u, v, w, and x respecting the following:

- u is adjacent to all vertices of $\{a, b, c, d, e, f\}$;
- v is adjacent to all vertices of $\{a, b, c, d\}$ and not adjacent to both e and f;
- w is adjacent to all vertices of $\{a, b, e, f\}$ and not adjacent to both c and d;
- x is adjacent to all vertices of $\{c, d, e, f\}$ and not adjacent to both a and b.

Note that, even though every edge depicted in Figure 5.3 is mandatory in G, there may be other edges in G that are not shown in the picture.

Since all of $\{v, c\}, \{c, x\}$, and $\{x, e\}$ are edges in G, then there is a path from v to e in G. Let $E(P)$ be the edges of this path. Consider $A = \{a, b, e\}$ and $B = \{u, v, w\}$, and let

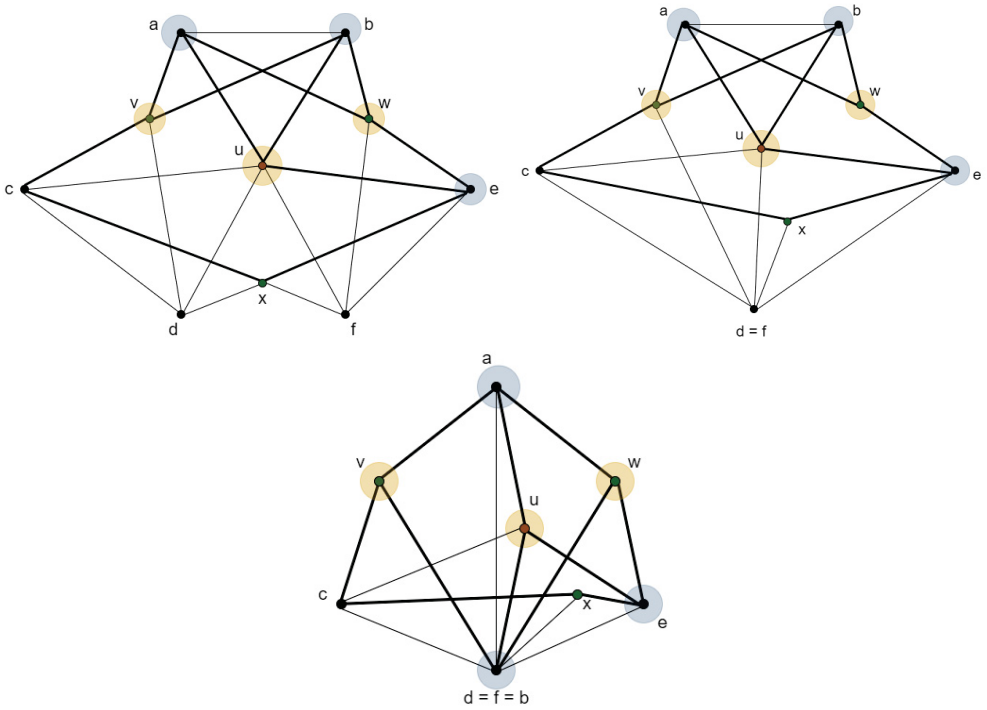


Figure 5.3: Three possible arrangements for the edges $e_1 = \{a, b\}$, $e_2 = \{c, d\}$, and $e_3 = \{e, f\}$ from the proof of Corollary 4. The case where b, d, and f are distinct vertices is depicted above. The case where $d = f$, but $b \neq f$ is shown below in the left. Below in the right, we show the case where $b = d = f$. In all three cases, these edges are part of a subgraph H of G that contains a subdivision of a $K_{3,3}$.

X be the set of edges with one endpoint in A and one endpoint in B. We can obtain a subgraph H of G that contains a subdivision of a bipartite graph $K_{3,3}$ with bipartition (A, B) in the following way. The vertex set of H is $A \cup B \cup \{c, x\}$ and the edge set of H is $X \cup E(P)$. Therefore G cannot be a planar graph. □

5.1.3 Algorithm

The algorithm takes as input a graph $G = (V, E)$ and the constant parameters $0 < \varepsilon, \delta, p < 1$. It outputs the estimation $\tilde{l}(v)$ for the exact value $l(v)$ of the local clustering coefficient for each vertex $v \in V$, such that

$$|l(v) - \tilde{l}(v)| \leq \varepsilon l(v), \text{ with probability at least } 1 - \delta \text{ whenever } l(v) \geq \sigma_v(p),$$

where $\sigma_v(p) = pm/\binom{\delta_v}{2}$ is an adjustable function, depending on p. The idea, roughly speaking, is that $l(v) \geq \sigma_v(p)$ holds if the neighborhood of v is not too small.

Next we present Algorithm 6. At the beginning all $\tilde{T}_v$ are set to zero.

Algorithm 6 localClusteringEstimation(G,ε,δ,p)

Input: Graph $G=(V,E)$ with m edges, constant parameters $0<\varepsilon,\delta,p<1$.
Output: Local clustering coefficient estimation $\tilde{l}(v)$, $\forall v\in V$ such that $l(v)\geq\sigma_v(p)$.

1: $r\leftarrow\left\lceil\frac{c'}{\varepsilon^2 p}\left((\lfloor\lg\Delta-1\rfloor+1)\log\frac{1}{p}+\log\frac{1}{\delta}\right)\right\rceil$
2: **for** $i\leftarrow 1$ **to** r **do**
3: sample an edge $e=\{a,b\}\in E$ uniformly at random
4: **for all** $v\in N_a$ **do**
5: **if** $v\in N_b$ **then**
6: $\tilde{T}_v\leftarrow\tilde{T}_v+\frac{m}{r}$
7: **return** $\tilde{l}(v)\leftarrow\frac{2\tilde{T}_v}{\delta_v(\delta_v-1)}$, for each $v\in V$.

5.2 CORRECTNESS AND RUNNING TIME

In Theorems 28 and 29 we prove the correctness and running time of Algorithm 6.

Theorem 28. *Given a graph $G=(V,E)$, let $S\subseteq E$ be a sample of size*

$$r=\left\lceil\frac{c}{\varepsilon^2 p}\left((\lfloor\lg\Delta-1\rfloor+1)\log\frac{1}{p}+\log\frac{1}{\delta}\right)\right\rceil,$$

for given constants $0<p,\varepsilon,\delta<1$ and $c>0$. Algorithm 6 returns with probability at least $1-\delta$ an approximation $\tilde{l}(v)$ to $l(v)$ within ε relative error, for each $v\in V$ such that $l(v)\geq\sigma_v(p)$.

Proof. For each $v\in V$, let $\mathbb{1}_v(e)$ be the function that returns 1 if $e\in\tau_v$ (and 0 otherwise). Thus, $T_v=\sum_{e\in E}\mathbb{1}_v(e)$. The estimated value $\tilde{T}_v$, computed by Algorithm 6, is incremented by m/r whenever an edge $e\in S$ belongs to τ_v, i.e.

$$\tilde{T}_v=\sum_{e\in S}\frac{m}{r}\mathbb{1}_v(e).$$

Note that

$$\tilde{T}_v=\sum_{e\in S}\frac{m}{r}\mathbb{1}_v(e)=\frac{m}{r}\sum_{e\in S}\mathbb{1}_v(e)=m\cdot\frac{|S\cap\tau_v|}{|S|}.$$

Thus, assuming that we have a relative (p,ε)-approximation (Definition 9),

$$\frac{|T_v-\tilde{T}_v|}{T_v}=\frac{\left|m\cdot\Pr_\pi(\tau_v)-m\cdot\frac{|S\cap\tau_v|}{|S|}\right|}{m\cdot\Pr_\pi(\tau_v)}=\frac{\left|\Pr_\pi(\tau_v)-\frac{|S\cap\tau_v|}{|S|}\right|}{\Pr_\pi(\tau_v)}\leq\varepsilon.$$

Or, simply put, $|T_v-\tilde{T}_v|\leq\varepsilon T_v$. Therefore,

$$|l(v)-\tilde{l}(v)|=\frac{2|T_v-\tilde{T}_v|}{\delta_v(\delta_v-1)}\leq\frac{2\varepsilon T_v}{\delta_v(\delta_v-1)}=\varepsilon l(v).$$

Combining this with Theorems 8 and 25, and using a sample S with size

$$r=\left\lceil\frac{c}{\varepsilon^2 p}\left((\lfloor\lg\Delta-1\rfloor+1)\log\frac{1}{p}+\log\frac{1}{\delta}\right)\right\rceil,$$

we have that Algorithm 6 provides an ε-error estimation for $l(v)$ with probability $1-\delta$ for all $v \in V$ such that $\Pr(\tau_v) \geq p$. But $\Pr(\tau_v) \geq p$ if and only if $l(v) \geq \sigma_v(p)$ since

$$l(v) = \frac{T_v}{\binom{\delta_v}{2}} = \frac{m \Pr(\tau_v)}{\binom{\delta_v}{2}}.$$

□

We remark that $\tilde{T}_v$ is an unbiased estimator for T_v, since

$$\mathbb{E}[\tilde{T}_v] = \mathbb{E}\left[\sum_{e \in S} \frac{m}{r} \mathbb{1}_v(e)\right] = \frac{m}{r} \sum_{e \in S} \Pr(e \in \tau_v) = \frac{m}{r} \sum_{e \in S} \frac{|\tau_v|}{m} = T_v.$$

Theorem 29. *Given a graph $G = (V, E)$ and a sample of size*

$$r = \left\lceil \frac{c}{\varepsilon^2 p} \left((\lfloor \lg \Delta - 1 \rfloor + 1) \log \frac{1}{p} + \log \frac{1}{\delta} \right) \right\rceil,$$

Algorithm 6 has running time $\mathcal{O}(\Delta \lg \Delta + m)$.

Proof. In line 1, the value of Δ can be computed in time $\Theta(m)$. Given an edge $\{a, b\}$ we first store the neighbors of b in a directed address table. Then, lines 4, 5, and 6 take time $\mathcal{O}(\Delta)$ by checking, for each $v \in N_a$, if v is in the table. Hence, the total running time of Algorithm 6 is $\mathcal{O}(r \cdot \Delta + m) = \mathcal{O}(\Delta \lg \Delta + m)$. □

As mentioned before, for specific graph classes, the running time proved in Theorem 29 can be reduced. We can achieve this either by proving that graphs in such classes have a smaller VC dimension, or by looking more carefully at the algorithm analysis for such classes. In Corollaries 5 and 6 we present results for two such classes.

Corollary 5. *If G is a planar graph, then Algorithm 6 has running time $\mathcal{O}(\Delta)$.*

Proof. By Corollary 4, $\mathrm{VCDim}(G) \leq 2$. So, the sample size in the Algorithm 6 changes from a function of Δ to a constant. Note that, in particular, since we do not need to find the value of Δ, line 1 can be computed in time $\mathcal{O}(1)$. As with the proof of Theorem 29, lines 4, 5, and 6 still take time $\mathcal{O}(\Delta)$. Since r is constant, line 2 takes constant time. So, the total running time of Algorithm 6 is $\mathcal{O}(r \cdot \Delta) = \mathcal{O}(\Delta)$. □

Another case where we can provide a better running time for the algorithm is the case for *bounded-degree graphs*, i.e. the case where the maximum degree of any graph in the class is bounded by a constant.

Corollary 6. *Let G be a bounded-degree graph, where d is such bound. Algorithm 6 has running time $\mathcal{O}(1)$ or $\mathcal{O}(n)$, respectively, depending on whether d is part of the input or not.*

Proof. If d is part of the input, then the number of samples r in line 1 can be computed in time $\mathcal{O}(1)$. Line 2 is executed $\mathcal{O}(1)$ times, and the remaining of the algorithm, in lines 4, 5, and 6, takes $\mathcal{O}(1)$ time, since the size of the neighborhood of every vertex is bounded by a constant.

On the other hand, if d is not part of the input, then Δ must be computed for executing line 1. In this case we check the degree of every vertex by traversing its adjacency list. All these adjacency lists have constant size. Performing this for all vertices takes time $\mathcal{O}(n)$. The other steps of the algorithm take constant time. □

5.3 CONCLUDING REMARKS

We present a sampling algorithm for the local clustering problem. In our analysis we define a range space associated to the input graph, and show how the sample size of the algorithm relates to the VC dimension of this range space. This kind of analysis takes into consideration the combinatorial structure of the graph, so the size of the sample of edges used by the algorithm depends on the maximum degree of the input graph.

Our algorithm executes in time $\mathcal{O}(\Delta \lg \Delta + m)$ in the general case and guarantees, for given parameters ε, δ, and p, that the approximation value has relative error ε with probability at least $1 - \delta$, for every node whose clustering coefficient is greater than a certain function adjusted by the parameter p. For planar graphs we show that the sample size can be bounded by a constant, and the running time in this case is $\mathcal{O}(\Delta)$. In the case of bounded-degree graphs, where there is also a constant bound on the sample size, the running time drops to $\mathcal{O}(1)$ or $\mathcal{O}(n)$, depending on whether the bound on the degree is part of the input or not.

6 THE DOMINATING SET AND THE VERTEX COVER PROBLEMS

Empirical studies from the late 1990's and early 2000's (Faloutsos et al., 2011; Barabási and Albert, 1999; Kleinberg et al., 1999; Broder et al., 2011; Kumar et al., 2000; Kleinberg and Lawrence, 2001; Guelzim et al., 2002; Siganos et al., 2003; Eubank et al., 2004) pointed out that a number of large real-world networks — also commonly called *complex networks* — from social, biological, and technological applications follow a *power law* on their vertex degree distribution. We can informally describe a power law as a function that decreases in the vertex degree i as i grows large for a fixed exponent $\beta > 0$ and a proportionality constant α, i.e. $f(i) = \alpha i^{-\beta}$. Random graph models for such complex networks are referred as *power-law graphs*. There is evidence that optimization problems might be easier for power-law graphs than for graphs in general (Park and Lee, 2001; Gkantsidis et al., 2003; Eubank et al., 2004; Da Silva et al., 2013; Demaine et al., 2019). More precisely, if one assumes that the input graph is drawn from a distribution where the expected degree distribution follows a power law, then several problems admit approximation algorithms with expected factors that may not be achievable for general graphs (Gast et al., 2012; Gast and Hauptmann, 2014; Gast et al., 2015; Vignatti and Silva, 2016).

Random graph models with arbitrary degree distributions have been studied since at least the late 1970's (Bender and Canfield, 1978; Wormald, 1980; Bollobás, 1998; Molloy and Reed, 1995, 1998; Chung and Lu, 2002, 2004; Britton et al., 2006). In this chapter we use the *generalized random graph* (GRG) model, introduced by Britton et al. (2006), which is a generalization of the well-known Erdős–Rényi random graph model with weights assigned to the vertices of the graph. These weights are used for obtaining an arbitrary expected distribution for the vertex degrees. One advantage of this model is that the edges of the graph are created independently. In order to have an expected power-law distribution, we use the sequence of weights given by the formula described in the work of Aiello et al. (2001). The authors propose a random graph model known as ACL(α, β), which is also a model for power-law graphs, but it does not have the convenience of having independent edge probabilities.

We refer to the random graph model used in this chapter as GRG(α, β) (the precise definitions are given in Section 6.1). We note that the well-known Chung–Lu model (Chung and Lu, 2002, 2004) also uses a sequence of weights for the vertices, so that the expected degree of each vertex corresponds to its weight. In the work of Vignatti and Silva (2016), the authors show that the edge probabilities of the Chung–Lu model and the GRG(α, β) are asymptotically the same for the particular degree sequence that we are using considering. As a consequence, every result presented in this chapter also holds for the Chung–Lu model.

The main result we prove is a lower bound for the expected size of the neighborhood of vertices of degree one. As a consequence, we obtain tighter bounds for the approximability of both the minimum dominating set and the vertex cover problems, improving the previous results from Gast et al. (2015) and Vignatti and Silva (2016), respectively. The minimum dominating set (MDS) problem consists of finding the minimum set of vertices $D \subseteq V$ in a graph $G = (V, E)$ such that each $v \in V$ is either in D or has at least one neighbor in D. The minimum vertex cover (MVC) problem corresponds to finding the minimum set $C \subseteq V$ such that each $e \in E$ has at least one

endpoint in C (Garey and Johnson, 1979). Both problems are $\mathcal{NP}$-Hard (Garey and Johnson, 1979) and have applications in a variety of contexts and scenarios (Wang et al., 2009; Wu et al., 2006; Xu and Zhou, 2016; Nacher and Akutsu, 2016; Gusev, 2020; Javad-Kalbasi et al., 2019; Miao et al., 2019). In fact, Ferrante et al. (2008) showed that these problems remain $\mathcal{NP}$-Hard for the (deterministic) class of graphs respecting the degree distribution given by the formula described in the ACL(α, β) model (Aiello et al., 2001).

The minimum dominating set problem is conjectured not to admit a polynomial time approximation algorithm with a strictly sublogarithmic factor unless $\mathcal{P} = \mathcal{NP}$ (Raz and Safra, 1997). Similarly, the vertex cover problem is conjectured not to admit a polynomial time approximation algorithm with a factor smaller than 2 (Khot and Regev, 2008). However, when restricted to power-law graphs, both barriers can be overtaken (Gast and Hauptmann, 2014; Gast et al., 2015; Vignatti and Silva, 2016). An approximation factor of $\mathcal{O}(\log n)$ can be achieved for the MDS problem using an approximation algorithm for graphs in general. Gast et al. (2015) showed that the expected factor of approximation for this algorithm is constant when the input graph is a random sample from the ACL(α, β) model. In this chapter we use the GRG(α, β) to show that for $2 < \beta \leq 2.52$ and $2.729 < \beta < 2.85$ the expected approximation factor is significantly smaller than the one obtained by Gast et al. (2015). We note that, in many power-law graphs that model practical applications, β falls between 2 and 3 (Broder et al., 2011; Jeong et al., 2001; Liljeros et al., 2001; Redner, 1998). Additionally we show that our results also imply a significantly better expected approximation factor for the MVC for graphs in the GRG(α, β) model, for $2 < \beta < 4$, where this factor is near 1 as β gets closer to 4. It is important to highlight, though, that our bounds for the MDS cannot be directly compared with the ones of Gast and Hauptmann (2014) and Gast et al. (2015) since the random graph models are not exactly the same.

At the center of our analysis for both the MDS and MVC problems there is a proof of a lower bound for the expected size of the neighborhood of the vertices with degree one. We use this lower bound to estimate the optimal solution obtained by an approximation algorithm together with a simple preprocessing step. Following the previous approaches of Gast and Hauptmann (2014), Gast et al. (2015) and Vignatti and Silva (2016), the idea is that the neighborhood of degree one vertices is included in the optimal solution — this corresponds to a large portion of the vertices —- and an approximation algorithm is used in the remaining part of the graph. The expected approximation factors for the MDS and MVC problems, respectively denoted by $\phi(\beta)$ and $\psi(\beta)$, correspond to

$$\phi(\beta) \lesssim \frac{\zeta(\beta) + \mathrm{Li}_{\beta-1}(1/e)\left(\frac{\mathrm{Li}_{\beta-1}(1/e)}{2\zeta(\beta-1)} - 1\right)}{\zeta(\beta)\rho(\beta) - \frac{(\mathrm{Li}_{\beta-1}(1/e))^2}{2\zeta(\beta-1)}}$$

and

$$\psi(\beta) \lesssim 2 - \left(\frac{\rho(\beta)}{1 - \frac{\mathrm{Li}_{\beta}(1/e)}{\zeta(\beta)} - \frac{\mathrm{Li}_{\beta-1}(1/e)}{\zeta(\beta)}}\right),$$

where $\rho(\beta) \approx 1 - \frac{\mathrm{Li}_\beta\left(\left(\frac{1}{e}\right)^{\frac{\mathrm{Li}_{\beta-1}(1/e)}{\zeta(\beta-1)}}\right)}{\zeta(\beta)}$, for $2 < \beta < 4$. The symbols "$\approx$" and "$\lesssim$" denote asymptotic approximations for, respectively, equality and upper bound, and the $\zeta(\beta)$ and $\mathrm{Li}_\beta(z)$ denote the Riemman zeta and the polylogarithmic functions, respectively (see Section 6.1). The upper bounds of $\phi(\beta)$ and $\psi(\beta)$ can be better understood from Figures 6.1 and 6.2. As far as we know, the expected approximation factors obtained for both problems are the best for power-law graphs.

This chapter is organized as follows: in Section 6.1 we provide the definitions of our random graph model; in Section 6.2 we present the crux of our analysis, which is a lower bound for the neighborhood of the degree one vertices; in Section 6.3 we show our strategy for dealing with the approximability of the MDS problem; Section 6.4 describes our new results for approximability of the MVC problem, and Section 6.5 presents the concluding remarks and directions for future work.

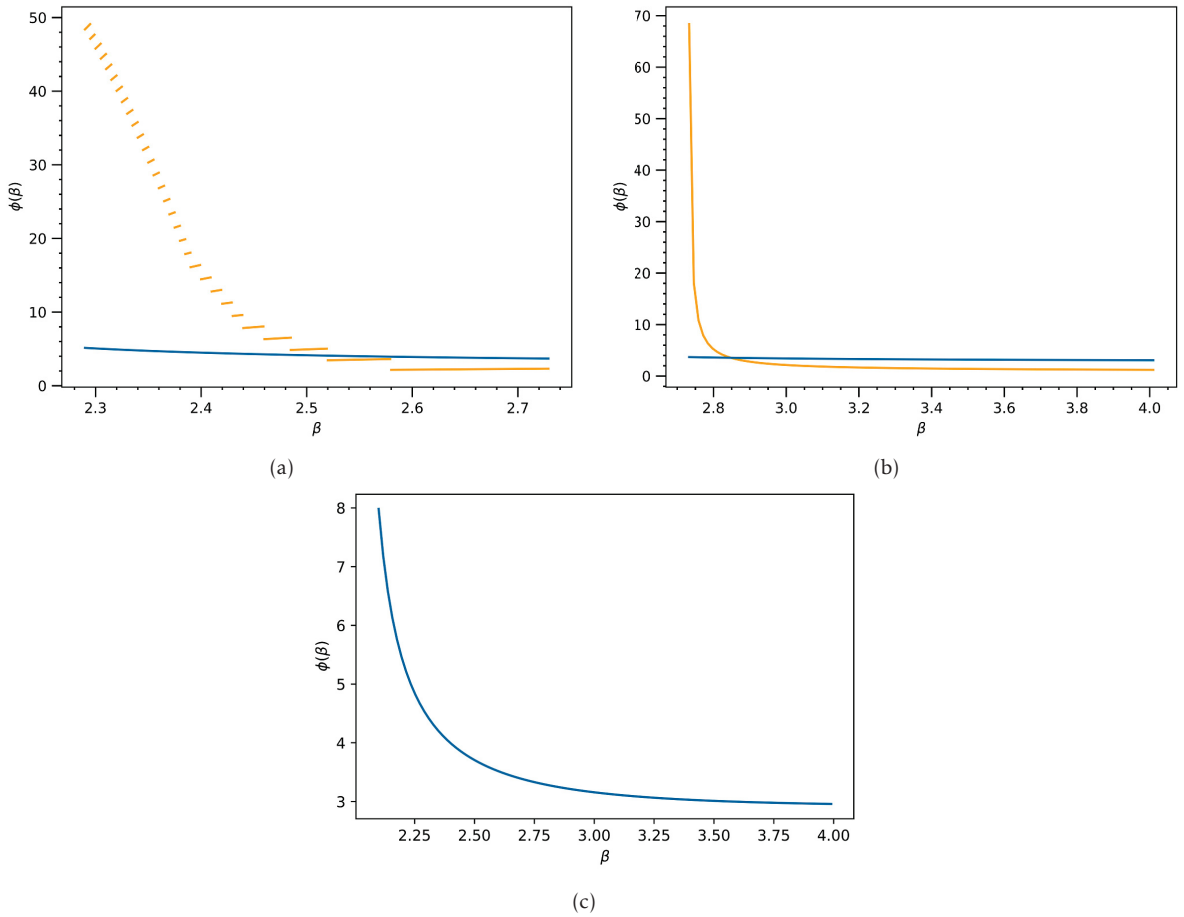


Figure 6.1: In (c), the graph of our approximation factor for the minimum dominating set problem $2 < \beta < 4$. In (a) and (b), we compare our bound (darker blue line) with the results of Gast et al. (2015) (lighter orange line).

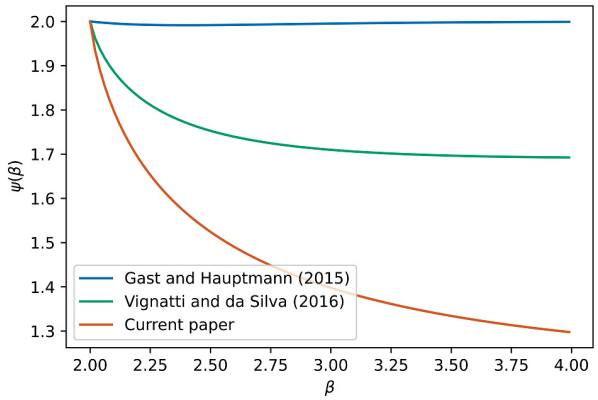


Figure 6.2: Comparison of the expected approximation factor between our work and the results of Gast and Hauptmann (2014) and Vignatti and Silva (2016), for $2 < \beta < 4$.

6.1 PRELIMINARIES

Throughout this chapter, we use $\approx$ to denote an *asymptotic approximation*, i.e. given functions $f(n)$ and $g(n)$, then $f(n) \approx g(n)$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$. We also use $\lesssim$ and $\gtrsim$, respectively, to denote an *asymptotic upper* and *lower bound approximation*. Formally, we have that $f(n) \lesssim g(n)$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} \leq 1$, and $f(n) \gtrsim g(n)$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} \geq 1$. It is worth mentioning that the lower and upper asymptotic approximations that are used here are stronger than the Ω and $\mathcal{O}$ asymptotic notations.

For the next definitions and throughout the results of this chapter, we denote $\zeta(\beta) = \sum_{j=1}^{\infty} \frac{1}{j^\beta}$ the Riemann zeta function and $\mathrm{Li}_\beta(z) = \sum_{j=1}^{\infty} \frac{z^j}{j^\beta}$ the polylogarithmic function.

Let $G = (V, E)$ be a random graph with $n = |V|$ and $m = |E|$. Consider the vertex set $V = \{1, 2, \ldots, |V|\}$. In this work we use the GRG model proposed by Britton et al. (2006), where there is a weight w_v associated to each vertex $v \in V$. We denote W_k the set of vertices having weight k, i.e. $W_k = \{v \in V \mid w_v = k\}$. Let w be a vector with entries $w_1, \ldots, w_{|V|}$. In the GRG model, every edge (i, j) is created independently at random with probability $\Pr((i, j) \in E) = \frac{w_i w_j}{\ell_n + w_i w_j}$, where $\ell_n = \sum_{v \in V} w_v$. In the literature p_{ij} usually refers to the probability of an edge connecting vertex i and vertex j. For the sake of convenience, however, we refer to p_{ij} as the probability of a vertex having weight i connects to a vertex having weight j.

Naturally, the vertex degrees depends on w, so we set the weights in such vector using similar principles of the ones from the work of Aiello et al. (2001) to create a power-law random graph with exponent $\beta > 2$. Consider $y_j = \left\lfloor \frac{e^\alpha}{j^\beta} \right\rfloor$, for each $j = 1, \ldots, \Delta$, where $\Delta = \lfloor e^{\alpha/\beta} \rfloor$ and $\alpha = \ln\left(\frac{|V|}{\zeta(\beta)}\right)$. On the ACL$(\alpha, \beta)$ model, there are y_j vertices of fixed degree j. Similarly, in our model, we assign weight j to y_j vertices. We denote by GRG(α, β) a GRG random graph having such distribution on its vertex degrees.

Note that, from the definition of α, we have $|V| = e^\alpha \zeta(\beta)$. Aiello et al. (2001) observe that we can ignore rounding in the values of y_j and Δ. However, some extra care has to be taken in the values of y_j in the ACL(α, β) model, since the vertex degrees sequence must be a graphic sequence. In the GRG(α, β) model we do not need such restriction since y_j is associated to the weights and not to the degrees.

Using the y's values defined above, note that

$$\ell_n = \sum_{v \in V} w_v = \sum_{j=1}^{\Delta} j \cdot y_j \approx \sum_{j=1}^{\Delta} j \cdot \frac{e^\alpha}{j^\beta} \approx e^\alpha \zeta(\beta - 1),$$

and hence, and edge connecting a vertex of degree i with a vertex of degree j is created independently at random with probability $p_{ij} = \frac{ij}{e^\alpha \zeta(\beta-1)+ij}$. In Lemma 2.1 (Vignatti and Silva, 2016), the authors show that $p_{ij} \approx \frac{ij}{e^\alpha \zeta(\beta-1)}$. On the other hand, using the y's values on the Chung–Lu model (Chung and Lu, 2002, 2004), we have $p_{ij} = \frac{ij}{e^\alpha \zeta(\beta-1)}$. Thus, we conclude that GRG(α, β) and Chung–Lu models are asymptotically equivalent for the power-law weight distribution that we use here, and all results in this chapter hold in the Chung–Lu model.

We use the notation $u \to v$ to refer to the event where the vertex u is adjacent to v in the resulting graph G. The degree of $v \in V$ is denoted by δ_v and we denote V_k the set of vertices of degree k.

Let $V^- = V \setminus (V_0 \cup V_1)$. For $S \subseteq V$, denote $G[S]$ the graph induced by S and denote $N(S)$ the neighborhood of S in G, i.e. the set of vertices that are adjacent to a vertex of S. The set $N(V_1)$ denotes the neighborhood of V_1 in G and it can be expressed as $N(V_1) = N(V_1)^- \cup N(V_1)^{(1)}$, where $N(V_1)^-$ corresponds to the set of vertices in $N(V_1)$ that have degree greater than one and $N(V_1)^{(1)}$ are vertices of $N(V_1)$ that have degree equal to one.

Lemma 8 (Vignatti and Silva (2016), Lemma 3.1)**.** *Let $q_{ik} = 1 - p_{ik}$. Then*

$$\prod_{k=1}^{\Delta} q_{ik}^{|W_k|} \approx \frac{1}{e^i}.$$

□

Lemma 9 (Vignatti and Silva (2016), Lemma 3.2)**.**

$$\Pr(v \in W_i) = \frac{(e^\alpha / i^\beta)}{e^\alpha \zeta(\beta)} = \frac{1}{i^\beta \zeta(\beta)}.$$

□

Lemma 10 (Vignatti and Silva (2016), Lemma 3.3)**.**

$$\Pr(v \in V_0 \mid v \in W_i) \approx \frac{1}{e^i}.$$

□

Lemma 11 (Vignatti and Silva (2016), Lemmas 3.5 and 3.6)**.**

$$\Pr(v \in V_0) \approx \frac{Li_\beta(1/e)}{\zeta(\beta)} \quad \textit{and} \quad \Pr(v \in V_1) \approx \frac{Li_{\beta-1}(1/e)}{\zeta(\beta)}.$$

□

Lemma 12. *Let* $q_{jk} = 1 - p_{jk}$, *where* $p_{jk} = \frac{jk}{e^{\alpha}\zeta(\beta-1)}$. *Then*

$$\prod_{l=1}^{\Delta}(q_{kl}q_{il})^{|W_l|} \approx \frac{1}{e^{i+k}}.$$

Proof. Trivially from Lemma 8. □

6.2 TECHNICAL LEMMAS

The main result of this section is the expected value of $|N(V_1)|$ and its corresponding parts, i.e. $|N(V_1)^{-}|$ and $|N(V_1)^{(1)}|$. We show these results in Lemmas 16, 20, and 21. The size of these sets are crucial for the approximation algorithms presented in Sections 6.3 and 6.4. For both algorithms we can run a preprocessing step in the set of vertices in $N(V_1)^{-}$ and $N(V_1)^{(1)}$. We observe that the vertices in $N(V_1)^{(1)}$ are all in V_1 and each edge between vertices from this set corresponds to an isolated edge.

A first observation is that we are interested in estimating the size of large sets, such as V_1 and $N(V_1)$. These sets grow asymptotically with the size of the graph. On the other hand, for large graphs, probabilities of events related to one particular vertex or one particular edge are asymptotically negligible, as shown in Lemma 13. We combine these two facts in Lemmas 14 and 15 in order to show that for a given vertex v, adjacent to a given vertex w, the asymptotic probability of the event $\delta_v = 1$ is the same of the event $\delta_v = 0$ in the graph induced by $V \setminus \{w\}$.

Lemma 13. *Consider* $j,k \in \{1,\ldots,e^{\alpha/\beta}\}$ *and* $q_{jk} = 1 - p_{jk}$, *where*

$$p_{jk} = \frac{jk}{e^{\alpha}\zeta(\beta-1)}.$$

Then, $q_{jk} \approx 1$.

Proof. Using the fact $\beta > 2$,

$$\lim_{\alpha\to\infty}\frac{jk}{e^{\alpha}\zeta(\beta-1)} \leq \frac{1}{\zeta(\beta-1)}\lim_{\alpha\to\infty}\frac{e^{\frac{\alpha}{\beta}}e^{\frac{\alpha}{\beta}}}{e^{\alpha}} = \frac{1}{\zeta(\beta-1)}\lim_{\alpha\to\infty}e^{\alpha\left(\frac{2}{\beta}-1\right)} = 0.$$ □

Lemma 14. *Consider* $(u,w) \in V^2$ *such that* $u \in W_j$ *and* $w \in W_i$. *Then*

$$\Pr(u \in V_1 \mid u \in W_j \text{ and } w \in W_i \text{ and } w \to u) \approx \Pr(u \in V_0 \mid u \in W_j).$$

Proof. Let X_v be the binary random variable associated to vertex u such that $X_v = 1$ if $u \to v$, $X_v = 0$ otherwise. Note that these binary random variables are mutually independent, since edges are independently generated in our random graph model. We now compute the probability of u not being adjacent to any other vertex in V except w. That is,

$$\Pr(u \in V_1 \mid u \in W_j \text{ and } w \in W_i \text{ and } w \to u)$$

$$= \Pr\left(\bigcap_{\substack{v \in V \\ v \neq u \neq w}} X_v = 0 \,\middle|\, u \in W_j \text{ and } w \in W_i \text{ and } w \to u \right)$$
$$= \prod_{\substack{v \in V \\ v \neq u \neq w}} \Pr(X_v = 0 \mid u \in W_j \text{ and } w \in W_i \text{ and } w \to u)$$
$$= \frac{1}{q_{ij} q_{jj}} \prod_{k=1}^{\Delta} \prod_{v \in W_k} q_{jk} = \frac{1}{q_{ij} q_{jj}} \prod_{k=1}^{\Delta} q_{jk}^{|W_k|} \approx \frac{1}{e^j} \frac{1}{q_{ij} q_{jj}} \approx \frac{1}{e^j} \approx \Pr(u \in V_0 \mid u \in W_j),$$

where the approximations follow from Lemmas 8, 10, and 13. □

Lemma 15. *Consider $(u, v) \in V^2$ such that $u \in W_i$ and $v \in W_j$. Then*

$$\begin{aligned} &\Pr(u \in V_1 \text{ and } v \in V_1 \mid u \to v \text{ and } u \in W_i \text{ and } v \in W_j) \\ \approx &\Pr(u \in V_0 \mid u \in W_i) \cdot \Pr(v \in V_0 \mid v \in W_j). \end{aligned}$$

Proof. Consider the random variable X_{zu} with respect to u, defined for each $z \in V$, such that $X_{zu} = 1$ if $z \to u$ (and $X_{zu} = 0$ otherwise). The random variable X_{zv} is defined analogously to X_{zu}.

Note that each X_{zu} (and X_{zv}) are mutually independent, since edges are independently generated in our random graph model. We now compute the probability of u not being adjacent to any other vertex in V except v (and vice-versa for v). Then

$$\begin{aligned} &\Pr(u \in V_1 \text{ and } v \in V_1 \mid u \to v \text{ and } u \in W_i \text{ and } v \in W_j) \\ =&\Pr\left(\bigcap_{\substack{z \in V \\ z \neq u \neq v}} (X_{zu} = 0) \text{ and } \bigcap_{\substack{z \in V \\ z \neq u \neq v}} (X_{zv} = 0) \,\middle|\, u \in W_i \text{ and } v \in W_j \text{ and } u \to v \right) \\ =&\Pr\left(\bigcap_{\substack{z \in V \\ z \neq u \neq v}} (X_{zu} = 0) \,\middle|\, u \in W_i \text{ and } v \in W_j \text{ and } u \to v \right) \\ &\cdot \Pr\left(\bigcap_{\substack{z \in V \\ z \neq u \neq v}} (X_{zv} = 0) \,\middle|\, u \in W_i \text{ and } v \in W_j \text{ and } u \to v \right) \\ =&\prod_{\substack{z \in V \\ z \neq u \neq v}} \Pr(X_{zu} = 0 \mid u \in W_i \text{ and } v \in W_j \text{ and } u \to v) \\ &\cdot \prod_{\substack{z \in V \\ z \neq u \neq v}} \Pr(X_{zv} = 0 \mid u \in W_i \text{ and } v \in W_j \text{ and } u \to v) \\ \approx &\frac{1}{e^i} \frac{1}{q_{ii} q_{ij}} \frac{1}{e^j} \frac{1}{q_{ij} q_{jj}} \approx \frac{1}{e^j} \frac{1}{e^i} \approx \Pr(u \in V_0 \mid u \in W_i) \Pr(v \in V_0 \mid v \in W_j), \end{aligned}$$

where the second and third equations follow since the events are mutually independent, and the approximations follow from Lemmas 8, 10, and 13. □

Lemma 16.

$$\mathrm{E}[|N(V_1)^{(1)}|] \approx \frac{e^{\alpha}(Li_{\beta-1}(1/e))^2}{\zeta(\beta-1)}.$$

Proof. Consider the binary random variable X_{uv} defined as follows:

$$X_{uv} = \begin{cases} 1, \text{ if } u \in V_1 \text{ and } v \in V_1 \text{ and } u \to v \\ 0, \text{ otherwise.} \end{cases}$$

Then we have

$$\begin{aligned}
&\Pr(u \in V_1 \text{ and } v \in V_1 \text{ and } u \to v) \\
&= \sum_{i=1}^{\Delta} \Pr(u \in V_1 \text{ and } v \in V_1 \text{ and } u \to v \mid u \in W_i) \Pr(u \in W_i) \\
&= \sum_{i=1}^{\Delta} \sum_{j=1}^{\Delta} \Pr(u \in V_1 \text{ and } v \in V_1 \text{ and } u \to v \mid u \in W_i \text{ and } v \in W_j) \Pr(v \in W_j) \Pr(u \in W_i).
\end{aligned}$$

For given events A, B, and C, by the definition of conditional probability we have that

$$\Pr(A \text{ and } B \mid C) = \frac{\Pr(A \mid B \text{ and } C)\Pr(B \mid C)\Pr(C)}{\Pr(C)} = \Pr(A \mid B \text{ and } C)\Pr(B \mid C).$$

Therefore,

$$\begin{aligned}
&\Pr(u \in V_1 \text{ and } v \in V_1 \text{ and } u \to v \mid u \in W_i \text{ and } v \in W_j) \\
=&\Pr(u \in V_1 \text{ and } v \in V_1 \mid u \to v \text{ and } u \in W_i \text{ and } v \in W_j) \Pr(u \to v \mid u \in W_i \text{ and } v \in W_j) \\
\approx&\Pr(u \in V_0 \mid u \in W_i) \cdot \Pr(v \in V_0 \mid v \in W_j) \Pr(u \to v \mid u \in W_i \text{ and } v \in W_j)
\end{aligned}$$

where the approximation is given by Lemma 15. By Lemmas 9 and 10, we have

$$\begin{aligned}
\mathrm{E}[|N(V_1)^{(1)}|] &= \sum_{(u,v)\in V^2} \Pr(X_{uv} = 1) \\
&= \sum_{(u,v)\in V^2} \sum_{i=1}^{\Delta} \sum_{j=1}^{\Delta} (\Pr(X_{uv} = 1 \mid u \in W_i \text{ and } v \in W_j) \cdot \Pr(u \in W_i) \cdot \Pr(v \in W_j)) \\
&\approx \sum_{(u,v)\in V^2} \sum_{i=1}^{\Delta} \sum_{j=1}^{\Delta} \Big(\Pr(u \in V_0 \mid u \in W_i) \cdot \Pr(v \in V_0 \mid v \in W_j) \\
&\quad \cdot \Pr(u \to v \mid u \in W_i \text{ and } v \in W_j) \Pr(u \in W_i) \Pr(v \in W_j) \Big) \\
&\approx \sum_{(u,v)\in V^2} \sum_{i=1}^{\Delta} \sum_{j=1}^{\Delta} \frac{1}{e^{i+j}} \frac{ij}{e^{\alpha}\zeta(\beta-1)} \frac{1}{(ij)^{\beta}\zeta(\beta)^2}
\end{aligned}$$

$$= \frac{1}{e^{\alpha}\zeta(\beta-1)\zeta(\beta)^2} \sum_{(u,v)\in V^2} \sum_{i=1}^{\Delta} \frac{i}{e^i i^{\beta}} \sum_{j=1}^{\Delta} \frac{j}{e^j j^{\beta}} \approx \frac{e^{2\alpha}\zeta(\beta)^2(\mathrm{Li}_{\beta-1}(1/e))^2}{e^{\alpha}\zeta(\beta-1)\zeta(\beta)^2}$$

$$= \frac{e^{\alpha}(\mathrm{Li}_{\beta-1}(1/e))^2}{\zeta(\beta-1)}. \qquad \square$$

Given a fixed vertex v of weight j and a set of vertices $Y \subseteq V$ adjacent to v, we show in Lemma 17 that all events of the type "y is adjacent only to v", $y \in Y$, are approximately mutually independent.

Lemma 17. *For fixed $v \in V$ with weight j, for any $u \in V$ with weight i, and for a subset $S \subseteq V$, such that $u \notin S$,*

$$\Pr\left(v \to u \text{ and } u \in V_1 \,\middle|\, \bigcap_{y\in S}(v \to y \text{ and } y \in V_1)\right) \approx \Pr(v \to u \text{ and } u \in V_1).$$

Proof. We have that

$$\Pr\left(u \in V_1 \text{ and } v \to u \,\middle|\, \bigcap_{y\in S}(v \to y \text{ and } y \in V_1)\right)$$

$$= \frac{\Pr\left(u \in V_1 \text{ and } v \to u \text{ and } \bigcap_{y\in S} v \to y \text{ and } \bigcap_{y\in S} y \in V_1\right)}{\Pr\left(\bigcap_{y\in S} v \to y \text{ and } \bigcap_{y\in S} y \in V_1\right)}$$

$$= \frac{\Pr\left(u \in V_1 \text{ and } \bigcap_{y\in S} y \in V_1 \,\middle|\, v \to u \text{ and } \bigcap_{y\in S} v \to y\right)\Pr\left(v \to u \,\middle|\, \bigcap_{y\in S} v \to y\right)}{\Pr\left(\bigcap_{y\in S} y \in V_1 \,\middle|\, \bigcap_{y\in S} v \to y\right)}$$

by the fact that

$$\Pr(A \text{ and } B \mid C \text{ and } D) = \frac{\Pr(A \text{ and } B \text{ and } C \text{ and } D)}{\Pr(C \text{ and } D)}$$

$$= \frac{\Pr(A \text{ and } C \mid B \text{ and } D)\cdot\Pr(B \mid D)\Pr(D)}{\Pr(C \mid D)\cdot\Pr(D)} = \frac{\Pr(A \text{ and } C \mid B \text{ and } D)\cdot\Pr(B \mid D)}{\Pr(C \mid D)}.$$

Consider a vertex w with weight k. Let X_{uw} be the binary random variable having $X_{uw} = 1$ if $w \to u$ (and $X_{uw} = 0$ otherwise). The set of events $w \to u$, for each $w \in V$, is mutually independent. Then

$$\Pr\left(u \in V_1 \text{ and } \bigcap_{y\in S} y \in V_1 \,\middle|\, v \to u \text{ and } \bigcap_{y\in S} v \to y\right)$$

$$= \Pr\left(\bigcap_{y\in S} X_{yu} = 0 \text{ and } \bigcap_{y\in S} X_{uy} = 0 \bigcap_{y\in S} \bigcap_{\substack{y'\in S \\ y\neq y'}} (X_{yy'} = 0 \text{ and } X_{y'y} = 0) \right.$$

$$\left. \text{and} \bigcap_{\substack{w\in\{V\setminus S\} \\ w\neq u\neq v}} X_{wu} = 0 \text{ and } \bigcap_{y\in S} \bigcap_{\substack{w\in\{V\setminus S\} \\ w\neq u\neq v}} X_{wy} = 0 \right)$$

$$= \prod_{y\in S} \Pr(X_{yu} = 0 \text{ and } X_{uy} = 0) \prod_{y\in S} \prod_{\substack{y'\in S \\ y\neq y'}} \Pr(X_{yy'} = 0 \text{ and } X_{y'y} = 0)$$

$$\cdot \prod_{\substack{w\in V\setminus S \\ w\neq u\neq v}} \Pr(X_{wu} = 0) \prod_{y\in S} \prod_{\substack{w\in V\setminus S \\ w\neq u\neq v}} \Pr(X_{wy} = 0).$$

We have that

$$\Pr\left(\bigcap_{y\in S} y \in V_1 \,\middle|\, \bigcap_{y\in S} v \to y \right)$$

$$= \Pr\left(\bigcap_{y\in S} \bigcap_{\substack{y'\in S \\ y\neq y'}} (X_{yy'} = 0 \text{ and } X_{y'y} = 0) \text{ and } \bigcap_{y\in S} \bigcap_{\substack{w\in\{V\setminus S\} \\ w\neq u\neq v}} X_{wy} = 0 \text{ and } \bigcap_{y\in S} X_{uy} = 0 \right)$$

$$= \prod_{y\in S} \prod_{\substack{y'\in S \\ y\neq y'}} \Pr(X_{yy'} = 0 \text{ and } X_{y'y} = 0) \prod_{y\in S} \prod_{\substack{w\in V\setminus S \\ w\neq u\neq v}} \Pr(X_{wy} = 0) \prod_{y\in S} \Pr(X_{uy} = 0)$$

$$= \prod_{y\in S} \prod_{\substack{y'\in S \\ y\neq y'}} \Pr(X_{yy'} = 0 \text{ and } X_{y'y} = 0) \prod_{y\in S} \prod_{\substack{w\in V\setminus S \\ w\neq u\neq v}} \Pr(X_{wy} = 0)$$

$$\cdot \prod_{y\in S} \Pr(X_{uy} = 0) \prod_{y\in S} \Pr(X_{uy} = 0 \text{ and } X_{yu} = 0).$$

The last equality comes from the fact that

$$\prod_{y\in S} \Pr(X_{uy} = 0 \text{ and } X_{yu} = 0) = \prod_{y\in S} \Pr(X_{yu} = 0 \mid X_{uy} = 0) \Pr(X_{uy} = 0)$$

$$= \prod_{y\in S} \Pr(X_{yu} = 0).$$

In addition, the event $v \to u$ is independent from $\bigcap_{y\in S} v \to y$, and hence,

$$\Pr\left(v \to u \,\middle|\, \bigcap_{y\in S} v \to y \right) = \Pr(v \to u) \approx \frac{ij}{e^{\alpha}\zeta(\beta - 1)}.$$

Therefore,

$$\begin{aligned}
&\Pr\left(v \to u \text{ and } u \in V_1 \,\middle|\, \bigcap_{y\in S}(v \to y \text{ and } y \in V_1)\right)\\
=&\frac{ij}{e^{\alpha}\zeta(\beta-1)}\prod_{y\in S}\Pr(X_{yu}=0)\prod_{\substack{w\in V\setminus S\\ w\neq u\neq v}}\Pr(X_{wu}=0)\\
=&\frac{ij}{e^{\alpha}\zeta(\beta-1)}\prod_{\substack{w\in V\\ w\neq u\neq v}}\Pr(X_{wu}=0)\\
\approx&\frac{ij}{e^{\alpha}\zeta(\beta-1)}\frac{1}{e^{i}}\frac{1}{q_{ii}q_{ij}}\approx\frac{ij}{e^{i}e^{\alpha}\zeta(\beta-1)}
\end{aligned}$$

where the approximations in the last line come from Lemmas 12 and 13. By Lemma 14, this corresponds to

$$\Pr(u \in V_1 \text{ and } v \to u) = \Pr(u \in V_1 \mid u \to v)\Pr(u \to v).$$

This concludes the proof. □

Corollary 7. *For fixed $v \in V$ with weight j and for any $u \in V$ with weight i, the events "$v \to u$ and $u \in V_1$" are approximately mutually independent.* □

For the lemmas and theorems below, we denote by $v \longrightarrow S$ the event of the vertex v be connected to the set $S \subseteq V$.

Lemma 18.

$$\Pr(v \longrightarrow V_1 \mid v \in W_j) \gtrsim 1 - \left(\frac{1}{e}\right)^{\frac{jLi_{\beta-1}(1/e)}{\zeta(\beta-1)}}.$$

Proof. Let X_u be the binary random variable associated to $u \in W_i$, for $1 \leq i \leq \Delta$, such that $X_u = 1$ if $v \to u$ and $u \in V_1$ (and $X_u = 0$ otherwise). From De Morgan's law, from Corollary 7 and Lemma 17, and the fact that $\left(1 - \frac{a}{x}\right)^x \leq \left(\frac{1}{e}\right)^a$, for all $a \in \mathbb{R}$, we have

$$\begin{aligned}
\Pr(v \longrightarrow V_1 \mid v \in W_j) &= \Pr\left(\bigcup_{u\in V}(v \to u \text{ and } u \in V_1) \;\middle|\; v \in W_j\right)\\
&= 1 - \Pr\left(\bigcap_{u\in V}(v \nrightarrow u \text{ or } u \notin V_1) \;\middle|\; v \in W_j\right)\\
&= 1 - \Pr\left(\bigcap_{i=1}^{\Delta}\bigcap_{u\in W_i}(X_u = 0) \;\middle|\; v \in W_j\right)\\
&\approx 1 - \prod_{i=1}^{\Delta}\prod_{u\in W_i}\left(1 - \frac{ij}{e^{i}e^{\alpha}\zeta(\beta-1)}\right)
\end{aligned}$$

$$= 1 - \prod_{i=1}^{\Delta} \left(1 - \frac{ij}{e^i e^\alpha \zeta(\beta-1)}\right)^{e^\alpha/i^\beta}$$
$$\gtrsim 1 - \prod_{i=1}^{\Delta} \left(\frac{1}{e}\right)^{\frac{ij}{e^i \zeta(\beta-1) i^\beta}} = 1 - \left(\frac{1}{e}\right)^{\sum_{i=1}^{\Delta} \frac{ij}{e^i \zeta(\beta-1) i^\beta}}$$
$$\approx 1 - \left(\frac{1}{e}\right)^{\frac{j \mathrm{Li}_{\beta-1}(1/e)}{\zeta(\beta-1)}}.$$

□

Let $\rho(\beta) \approx 1 - \frac{\mathrm{Li}_\beta\left(\left(\frac{1}{e}\right)^{\frac{\mathrm{Li}_{\beta-1}(1/e)}{\zeta(\beta-1)}}\right)}{\zeta(\beta)}$.

Lemma 19.

$$\Pr(v \longrightarrow V_1) \gtrsim \rho(\beta).$$

Proof. By Lemmas 9 and 18,

$$\Pr(v \longrightarrow V_1) = \sum_{j=1}^{\Delta} \Pr(v \longrightarrow V_1 \mid v \in W_j) \Pr(v \in W_j)$$
$$\gtrsim \sum_{j=1}^{\Delta} \left(1 - \left(\frac{1}{e}\right)^{\frac{j \mathrm{Li}_{\beta-1}(1/e)}{\zeta(\beta-1)}}\right) \frac{1}{j^\beta \zeta(\beta)} \approx 1 - \frac{\mathrm{Li}_\beta\left(\left(\frac{1}{e}\right)^{\frac{\mathrm{Li}_{\beta-1}(1/e)}{\zeta(\beta-1)}}\right)}{\zeta(\beta)}.$$

□

Lemma 20.

$$\mathrm{E}[|N(V_1)|] \gtrsim e^\alpha \zeta(\beta) \rho(\beta).$$

Proof. Let X_v be the binary random variable associated to $v \in V$ such that $X_v = 1$ if $v \longrightarrow V_1$ (and $X_v = 0$ otherwise). Then by Lemma 19,

$$\mathrm{E}[|N(V_1)|] = \sum_{v \in V} \Pr(v \longrightarrow V_1) \gtrsim e^\alpha \zeta(\beta) \rho(\beta).$$

□

Lemma 21.

$$\mathrm{E}[|N(V_1)^-|] \gtrsim e^\alpha \left(\zeta(\beta)\rho(\beta) - \frac{(Li_{\beta-1}(1/e))^2}{\zeta(\beta-1)}\right).$$

Proof. Directly from $\mathrm{E}[|N(V_1)|] = \mathrm{E}[|N(V_1)^-|] + \mathrm{E}[|N(V_1)^{(1)}|]$, and Lemmas 16 and 20. □

6.3 APPROXIMATION FACTOR FOR THE MINIMUM DOMINATING SET PROBLEM

The strategy that we use for finding an approximation is similar to the one of Gast et al. (2015). We start with a preprocessing step where we include every vertex of $N(V_1)^-$ and half of the vertices of $N(V_1)^{(1)}$ in the solution. Then we apply an approximation algorithm in the graph induced by $V \setminus (N(V_1) \cup V_1)$. Consider the set $N(V_1)^{(1)'} \subseteq N(V_1)$, where $|N(V_1)^{(1)'}| = |N(V_1)^{(1)}|/2$, and denote by R the set $R = V \setminus (N(V_1)^- \cup$

V_1). In Lemma 22 we prove that the approximation factor $\phi(\beta)$ for the minimum dominating set problem corresponds to $\frac{r|\text{OPT}(R)|+|N(V_1)^-|+|N(V_1)^{(1)'}|}{|\text{OPT}(R)|+|N(V_1)^-|+|N(V_1)^{(1)'}|}$, where OPT($R$) is the optimal dominating set in R. We observe that, as in Lemma 4.1 (Vignatti and Silva, 2016), this holds for any graph G (i.e. no probabilistic argument is used in the proof). In the next results in this section, with the exception of Lemma 22, we treat the sizes of OPT(R), $N(V_1)$, and R as expected values of random variables. The bounds for the approximation factor given by Theorem 30 and Corollary 9 are illustrated in Figures 6.1 and 6.3, respectively, where we compare our results with the bounds of Theorem 4 from the work of Gast et al. (2015). Due to the nature of the random graphs the authors use, they obtained two functions for the approximation factor $\phi(\beta)$, defining the appropriated ranges for β in each case.

Lemma 22. *The approximation factor $\phi(\beta)$ for the minimum dominating set problem is at most*

$$\frac{r|OPT(R)| + |N(V_1)^-| + |N(V_1)^{(1)'}|}{|OPT(R)| + |N(V_1)^-| + |N(V_1)^{(1)'}|},$$

where r is the approximation factor of the algorithm applied to set R.

Proof. Consider $V^* = V_1 \cup N(V_1)$. We first prove that the following two conditions hold:

(i) G contains a minimum dominating set D such that $(N(V_1)^- \cup N(V_1)^{(1)'}) \subseteq D$, and

(ii) $\text{OPT}(V^*) = |N(V_1)^-| + |N(V_1)^{(1)'}|$.

For each edge $(x, y) \in E$ such that $x \in V_1$ and $y \in V^-$, either x or y (but not both) must belong to D (otherwise D is not minimum). If $x \in V_1$, then $(D \setminus \{x\}) \cup \{y\}$ is also a minimum dominating set, then, using the same exchange argument, there is a minimum dominating set containing every vertex of $N(V_1)^-$. For each pair of vertices $(x, y) \in V_1$ where $x \to y$, then either x or y (but not both) must belong to D, therefore, half of the vertices from $N(V_1)^{(1)}$ are in D. We denote such set by $N(V_1)^{(1)'}$. So, (i) holds.

From (i), we have that the graph induced by $N(V_1)^- \cup N(V_1)^{(1)'}$ is an optimal solution for $G[V^*]$. Besides, sets $N(V_1)^-$ and $N(V_1)^{(1)}$ are disjoint, and hence, (ii) holds. Now let OPT(V) denote the size of the optimal solution such that condition (i) holds. From (ii), we have that

$$\begin{aligned}\text{OPT}(V) &\leq |\text{OPT}(R) \cup N(V_1)^-| + |N(V_1)^{(1)'}| \\ &= |\text{OPT}(R)| + |N(V_1)^-| + |N(V_1)^{(1)'}|,\end{aligned}$$

where the last equality comes from the fact that $R \cap N(V_1)^- = \emptyset$.

Let OPT(V)$'$ be the size of the solution obtained by the approximation strategy. Then

$$\phi(\beta) \leq \frac{\text{OPT}(V)'}{\text{OPT}(V)} \leq \frac{r|\text{OPT}(R)| + |N(V_1)^-| + |N(V_1)^{(1)'}|}{|\text{OPT}(R)| + |N(V_1)^-| + |N(V_1)^{(1)'}|},$$

where the last inequality comes from the fact that $\frac{za+b}{a+b} \leq \frac{zc+b}{c+b}$ for $z, a, b, c \in \mathbb{R}$, where $z > 1$ and $a \leq c$. □

Corollary 8.

$$\phi(\beta) \leq \frac{r|OPT(R)| + |N(V_1)^-| + |N(V_1)^{(1)'}| + |V_0|}{|OPT(R)| + |N(V_1)^-| + |N(V_1)^{(1)'}| + |V_0|}$$

where r is the approximation factor of the algorithm applied to set R, and V_0 is the set of vertices that have degree 0. □

In Theorem 30 we give a constant upper bound for the expected value of $\phi(\beta)$. In the proof of our upper bound we use the next result by Gast et al. (2015), adapted to the random graph model we use. The approximation algorithm has an approximation factor given by $\mathcal{O}(\log \Delta)$, where $\Delta = e^{\alpha/\beta}$ is the maximum degree of a vertex in $G[R]$.

Lemma 23 (Gast et al. (2015), Section 8))**.** *For $2 < \beta < 4$,*

$$\phi(\beta) = \max\left\{ \frac{r|OPT(R)| + |N(V_1)^-| + |N(V_1)^{(1)}|/2}{|OPT(R)| + |N(V_1)^-| + |N(V_1)^{(1)}|/2} \;\middle|\; |OPT(R)| \leq |R|, \right.$$
$$\left. r = \min\left\{ \frac{\alpha}{\beta}, \frac{|R|}{|OPT(R)|} \right\} \right\} \leq \frac{|R| + |N(V_1)^-| + |N(V_1)^{(1)}|/2}{\frac{\beta}{\alpha}|R| + |N(V_1)^-| + |N(V_1)^{(1)}|/2}. \quad \square$$

Theorem 30.

$$\phi(\beta) \lesssim \frac{\zeta(\beta) + Li_{\beta-1}(1/e)\left(\frac{Li_{\beta-1}(1/e)}{2\zeta(\beta-1)} - 1\right)}{\zeta(\beta)\rho(\beta) - \frac{(Li_{\beta-1}(1/e))^2}{2\zeta(\beta-1)}},$$

for non-empty $N(V_1)^{(1)}$, for $2 < \beta < 4$.

Proof. From Lemmas 22 and 23, we have that the upper bound for the approximation factor $\phi(\beta)$ corresponds to

$$\phi(\beta) \leq \frac{\mathrm{E}[|R|] + \mathrm{E}[|N(V_1)^-|] + \mathrm{E}[|N(V_1)^{(1)}|]/2}{\frac{\beta}{\alpha}\mathrm{E}[|R|] + \mathrm{E}[|N(V_1)^-|] + \mathrm{E}[|N(V_1)^{(1)}|]/2}.$$

By linearity of expectation, $\mathrm{E}[|R|] = |V| - \mathrm{E}[|N(V_1)^-|] - \mathrm{E}[|V_1|]$, since $N(V_1)^-$ and V_1 are disjoint.

Writing $\mathrm{E}[|N(V_1)^-|] \gtrsim e^\alpha a$, $\mathrm{E}[|V_1|] \approx e^\alpha \zeta(\beta) b$, and $\mathrm{E}[|N(V_1)^{(1)}|] \approx e^\alpha c$, where a, b, and c are the constant parts on the expected size of each set, then

$$\phi(\beta) \leq \frac{e^\alpha(\zeta(\beta) - a - b) + e^\alpha(a + \frac{c}{2})}{\frac{\beta}{\alpha}e^\alpha(\zeta(\beta) - a - b) + e^\alpha(a + \frac{c}{2})} \leq \frac{\zeta(\beta) - b + \frac{c}{2}}{a + \frac{c}{2}}$$

since $\frac{\beta}{\alpha}(\zeta(\beta) - a - b) \geq 0$. The result follows from Lemmas 11, 16, and 21. □

In our analysis, following the same criteria applied by Gast et al. (2015), we did not include vertices of degree 0 in the solution. For the more general case, the approximation factor follows from Corollary 8 and Theorem 30.

Corollary 9. *For non-empty sets $N(V_1)^{(1)}$ and V_0 (set of isolated vertices in G), with $2 < \beta < 4$,*

$$\phi(\beta) \lesssim \frac{\zeta(\beta) + Li_{\beta-1}(1/e)\left(\frac{Li_{\beta-1}(1/e)}{2\zeta(\beta-1)} - 1\right) + Li_{\beta}(1/e)}{\zeta(\beta)\rho(\beta) - \frac{(Li_{\beta-1}(1/e))^2}{2\zeta(\beta-1)} + Li_{\beta}(1/e)}. \quad \square$$

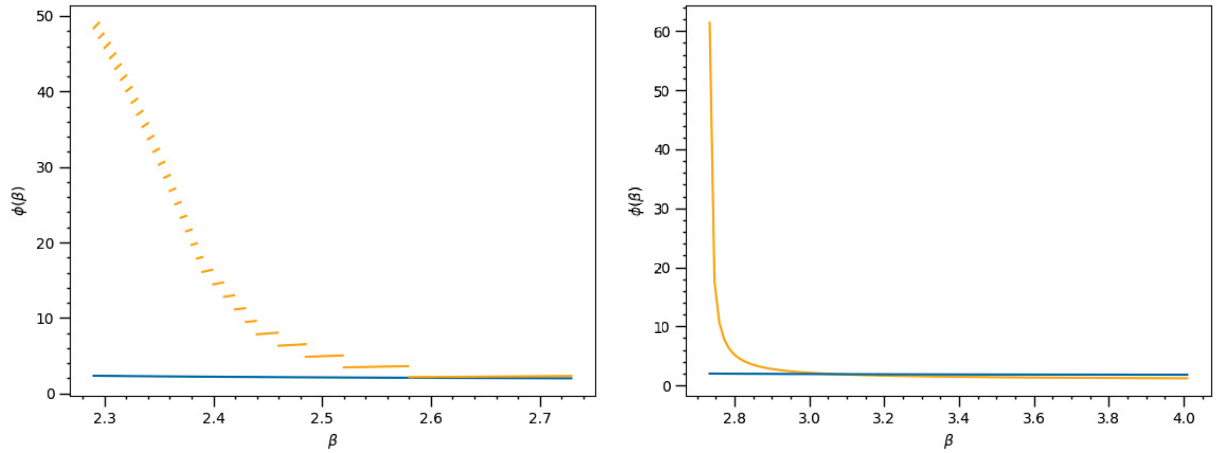


Figure 6.3: Expected approximation factor given by Corollary 9 for $2 < \beta \leq 2.729$ (graph in the left) and $2.729 < \beta < 4$ (graph in the right). The darker line (blue) corresponds to values obtained by our bounds, and the lighter line (orange) corresponds to the expected approximation factor described in Theorem 4 in (Gast et al., 2015). In the graph on the left, the function from Gast et al. (2015) is not continuous.

6.4 APPROXIMATION FACTOR FOR THE VERTEX COVER PROBLEM

In this section we show a better factor of approximation for the algorithm for the MVC problem described by Vignatti and Silva (2016). The algorithm has an approximation factor strictly smaller than 2 for power-law graphs, what may not be achievable for graphs in general (Khot and Regev, 2008). The approximation factor from Vignatti and Silva (2016) is an improvement of a previous result of Gast and Hauptmann (2014) (although some care should be taken in comparing both results, since the random graph models are not exactly the same, as we have discussed in the beginning of this chapter). In this section we show that the results obtained in Section 6.2 imply a better guarantee for the approximation factor for the algorithm of Vignatti and Silva (2016). We illustrate such differences in Figures 6.2 and 6.4.

The idea is similar, but not identical to the strategy described in Section 6.3. For the MVC problem we include all vertices of $N(V_1)$ in the solution and then run a 2-approximation algorithm in $V \setminus (N(V_1) \cup V_1)$. We state Lemma 24 ((Vignatti and Silva, 2016)) and give the proofs for Lemma 25, Corollary 10, and Theorem 31, although the proofs are similar to the referred section, for the sake of completeness. Similarly to Section 6.3, OPT(V), $|N(V_1)|$, and $|V^-|$ are treated as expected values of random variables, except in Lemma 24. For the next lemma, recall that $N(V_1)^{(1)'}$ is the set composed by half of the vertices from $N(V_1)^{(1)}$.

Lemma 24 (Vignatti and Silva (2016)), Lemma 4.1)**.** *The following three conditions hold:*

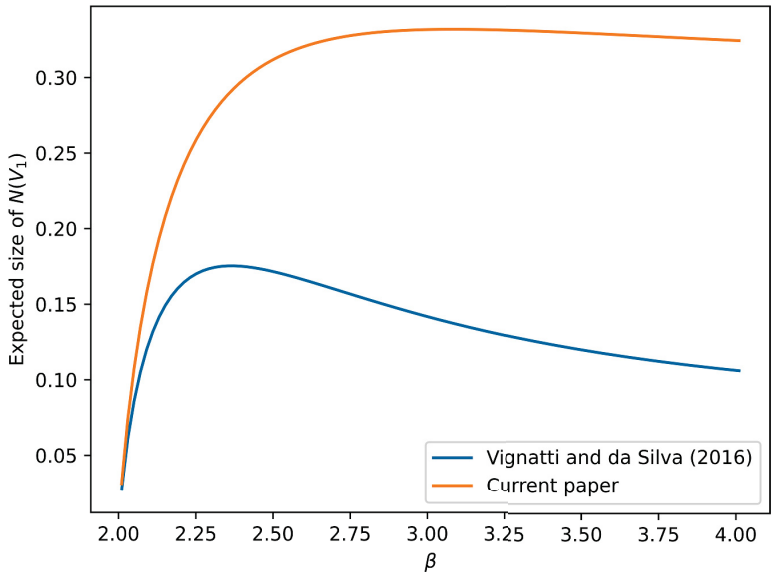


Figure 6.4: Expected values of $\mathrm{E}[|N(V_1)|]$ as a fraction of V.

(i) *G contains a minimum vertex cover C such that* $N(V_1)^- \cup N(V_1)^{(1)'} \subseteq C$,

(ii) $OPT(V^*) = |N(V_1)^- \cup N(V_1)^{(1)'}|$, *and*

(iii) $OPT(V) = OPT(V^*) + OPT(V \setminus V^*)$,

where $V^* = V_1 \cup N(V_1)$. □

We observe that Lemma 24 (i) is originally stated as "$N(V_1) \subseteq C$ and that there is no vertex of V_1 in C". However, the proof also holds by noting that $N(V_1) = N(V_1)^- \cup N(V_1)^{(1)}$ and that $N(V_1)^- \cup N(V_1)^{(1)'} \subseteq N(V_1)^- \cup N(V_1)^{(1)}$.

Lemma 25. *Let* $\rho(\beta) \approx 1 - \dfrac{Li_\beta\left(\left(\frac{1}{e}\right)^{\frac{Li_{\beta-1}(1/e)}{\zeta(\beta-1)}}\right)}{\zeta(\beta)}$. *Then*

$$\frac{OPT(V^*)}{OPT(V)} \gtrsim \left(\frac{\rho(\beta)}{1 - \frac{Li_\beta(1/e)}{\zeta(\beta)} - \frac{Li_{\beta-1}(1/e)}{\zeta(\beta)} + \frac{(Li_{\beta-1}(1/e))^2}{2\zeta(\beta-1)}} \right).$$

Proof. By Lemma 24 (i), $\mathrm{OPT}(V) \leq |V^-| + |N(V_1)^{(1)}|/2$. From Lemma 11,

$$|V^-| \leq |V| \left(1 - \frac{\mathrm{Li}_\beta(1/e)}{\zeta(\beta)} - \frac{\mathrm{Li}_{\beta-1}(1/e)}{\zeta(\beta)}\right).$$

From Lemma 16,

$$|N(V_1)^{(1)}| \approx \frac{e^\alpha (\mathrm{Li}_{\beta-1}(1/e))^2}{\zeta(\beta-1)} \leq \frac{e^\alpha \zeta(\beta)(\mathrm{Li}_{\beta-1}(1/e))^2}{\zeta(\beta-1)}.$$

By Lemmas 24 (ii) and 20, $\mathrm{OPT}(V^*) = |N(V_1)| \geq |V|\rho(\beta)$. Combining the two bounds, we have

$$\frac{\mathrm{OPT}(V^*)}{\mathrm{OPT}(V)} \gtrsim \left(\frac{\rho(\beta)}{1 - \frac{\mathrm{Li}_\beta(1/e)}{\zeta(\beta)} - \frac{\mathrm{Li}_{\beta-1}(1/e)}{\zeta(\beta)} + \frac{(\mathrm{Li}_{\beta-1}(1/e))^2}{2\zeta(\beta-1)}} \right). \qquad \square$$

Corollary 10.

$$\frac{OPT(V \setminus V^*)}{OPT(V)} \lesssim 1 - \left(\frac{\rho(\beta)}{1 - \frac{Li_\beta(1/e)}{\zeta(\beta)} - \frac{Li_{\beta-1}(1/e)}{\zeta(\beta)} + \frac{(Li_{\beta-1}(1/e))^2}{2\zeta(\beta-1)}} \right). \qquad \square$$

Proof. By Lemma 24 (iii), $\frac{\mathrm{OPT}(V^*)+\mathrm{OPT}(V\setminus V^*)}{\mathrm{OPT}(V)} = 1$. The result holds from Lemma 25. $\square$

Theorem 31. *The expected approximation factor $\psi(\beta)$ for the vertex cover problem corresponds to*

$$\psi(\beta) \lesssim 2 - \left(\frac{\rho(\beta)}{1 - \frac{Li_\beta(1/e)}{\zeta(\beta)} - \frac{Li_{\beta-1}(1/e)}{\zeta(\beta)} + \frac{(Li_{\beta-1}(1/e))^2}{2\zeta(\beta-1)}} \right).$$

Proof. From Lemma 24 we have that an optimal solution has the set $N(V_1)$. Hence, we apply a 2-approximation algorithm in $G[V \setminus V^*]$ and return $C \cup N(V_1)$ as solution, where C is the solution given by the 2-approximation algorithm. Since C and $N(V_1)$ are disjoint, by Lemma 24 ((ii) and (iii)) and Corollary 10,

$$\begin{aligned}
|C \cup N(V_1)| &= |C| + |N(V_1)| \leq 2\mathrm{OPT}(V \setminus V^*) + \mathrm{OPT}(V^*) \\
&= 2\mathrm{OPT}(V \setminus V^*) + \mathrm{OPT}(V) - \mathrm{OPT}(V \setminus V^*) \\
&= \mathrm{OPT}(V \setminus V^*) + \mathrm{OPT}(V) \\
&\lesssim \mathrm{OPT}(V) + \left(1 - \frac{\rho(\beta)}{1 - \frac{\mathrm{Li}_\beta(1/e)}{\zeta(\beta)} - \frac{\mathrm{Li}_{\beta-1}(1/e)}{\zeta(\beta)} + \frac{(\mathrm{Li}_{\beta-1}(1/e))^2}{2\zeta(\beta-1)}} \right) \mathrm{OPT}(V) \\
&= \left(2 - \frac{\rho(\beta)}{1 - \frac{\mathrm{Li}_\beta(1/e)}{\zeta(\beta)} - \frac{\mathrm{Li}_{\beta-1}(1/e)}{\zeta(\beta)} + \frac{(\mathrm{Li}_{\beta-1}(1/e))^2}{2\zeta(\beta-1)}} \right) \mathrm{OPT}(V). \qquad \square
\end{aligned}$$

6.5 CONCLUDING REMARKS

In this chapter we present an upper bound $\phi(\beta)$ for the expected approximation factor to the minimum dominating set problem in power-law graphs with $2 < \beta < 4$. We use the generalized random graph model of Britton et al. (2006) with expected power-law degree distribution. We show that for $2 < \beta \leq 2.52$ and $2.729 < \beta < 2.85$ the bound is tighter than the one of Gast et al. (2015). We show that the same techniques can also be applied to the vertex cover problem, improving the previous bound of Vignatti and Silva (2016) for the minimum vertex cover problem. As far as we know, the approximation factors obtained for both problems are the best known factors for power-law graphs.

7 CONCLUSION AND FUTURE DIRECTIONS

We have proposed sampling based approximation algorithms for graph problems using sample complexity tools on their design. We have showed that our algorithms output high quality solutions, with high probability, and that outperform in running time their exact approaches counterparts by a large margin. More specifically, we showed approximation algorithms for the problems of percolation centrality, of a relaxed version of APSP, and of local clustering coefficient. We also have proposed approximation algorithms that are not based on sample complexity theory for the problems of the minimum dominating set and the minimum set cover, both with a particular focus on power-law graphs. We have proven tight bounds for the approximation factors of such problems which, as far as we know, are the best known factors for power-law graphs.

The research questions for further work that we enumerate next are possible lines of research to follow, which can be divided in two branches: in the first, one may work to obtain algorithms for other problems in graphs using sample complexity analysis (question Q1); in the second, one can investigate the possibility of using the techniques that we have studied to obtain results in related areas, such as the field of fine-grained complexity and the connection between sampling and optimization (questions Q2, Q3, and Q4).

- Q1: For what other centrality measures can we obtain similar results to the previous ones of this work using randomized algorithms designed using sample complexity theory, VC-dimension, and Rademacher averages? The closeness centrality, for instance, is also exactly computed in $\mathcal{O}(n^3)$ by an APSP based algorithm and the best known randomized algorithm that estimates the centralities within ε to the original value with probability at least $1 - \delta$, from Wang and Eppstein (2006), uses standard Hoeffding and union bounds.

- Q2: In Chapter 4 we pointed out that there may be a connection between the values of ε and the hardness assumption of the APSP problem. An interesting question that might be worth of investigation is the possibility of computing all shortest paths in a graph, with high probability, achieving a strictly subcubic algorithm for some specific input distributions.

- Q3: The computation of the diameter of general unweighted graphs cannot be computed in truly subquadratic time under the Strong Exponential Time Hypothesis (Roditty and Vassilevska Williams, 2013). The work of Ducoffe et al. (2020) prove that for some classes of graphs, however, it is possible to compute the diameter in truly subquadratic time. They generalize these classes as the graphs with constant distance VC-dimension. Can we obtain similar results for other graph problems, like detecting a triangle, detecting negative triangles in a graph, and finding minimum weight cycles on a graph with non-negative weights?

- Q4: Is it possible to extract theoretical conclusions from sampling algorithms such as the ones in this work, typically related to counting problems, that shed light into corresponding optimization problems?

REFERENCES

Abboud, A., Grandoni, F., and Williams, V. V. (2014). Subcubic Equivalences Between Graph Centrality Problems, APSP and Diameter. In *Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1681–1697. SIAM.

Abboud, A. and Williams, V. (2014). Popular Conjectures Imply Strong Lower Bounds for Dynamic Problems. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 434–443.

Abboud, A., Williams, V., and Yu, H. (2018). Matching Triangles and Basing Hardness on an Extremely Popular Conjecture. *SIAM Journal on Computing*, 47(3):1098–1122.

Abraham, I., Delling, D., Fiat, A., Goldberg, A. V., and Werneck, R. F. (2011). VC-dimension and Shortest Path Algorithms. In *International Colloquium on Automata, Languages, and Programming*, pages 690–699. Springer.

Aiello, W., Chung, F., and Lu, L. (2001). A Random Graph Model for Power-law Graphs. *Experimental Mathematics*, 10(1):53–66.

Aingworth, D., Chekuri, C., and Motwani, R. (1996). Fast Estimation of Diameter and Shortest Paths (Without Matrix Multiplication). In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '96, pages 547–553.

Al Hasan, M. and Dave, V. S. (2018). Triangle counting in large networks: a review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(2):e1226.

Alon, N., Yuster, R., and Zwick, U. (1997). Finding and counting given length cycles. *Algorithmica*, 17(3):209–223.

Anthonisse, J. M. (1971). The rush in a directed graph. *Stichting Mathematisch Centrum. Mathematische Besliskunde*, (BN 9/71).

Anthony, M. and Bartlett, P. L. (2009). *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, New York, NY, USA, 1st edition.

Barabási, A.-L. and Albert, R. (1999). Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512.

Barabási, A.-L. and Pósfai, M. (2016). *Network science*. Cambridge University Press.

Becchetti, L., Boldi, P., Castillo, C., and Gionis, A. (2010). Efficient algorithms for large-scale local triangle counting. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(3):1–28.

Bender, E. A. and Canfield, E. R. (1978). The Asymptotic Number of Labeled Graphs with given Degree Sequences. *Journal of Combinatorial Theory, Series A*, 24(3):296–307.

Benedikt, M. and Libkin, L. (2002). Aggregate Operators in Constraint Query Languages. *Journal of Computer and System Sciences*, 64(3):628–654.

Bhattacharya, A., Chakraborty, S., Ghosh, A., Mishra, G., and Paraashar, M. (2022). Disjointness through the lens of Vapnik–Chervonenkis dimension: Sparsity and beyond. *computational complexity*, 31(2):1–31.

Bloznelis, M. (2013). Degree and clustering coefficient in sparse random intersection graphs. *The Annals of Applied Probability*, 23(3):1254–1289.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1989). Learnability and the Vapnik-Chervonenkis Dimension. *J. ACM*, 36(4):929–965.

Bollobás, B. (1998). Random Graphs. In *Modern graph theory*, pages 215–252. Springer.

Bondy, J. A. and Murty, U. S. R. (1976). *Graph Theory with Applications*. Elsevier, New York.

Boucheron, S., Bousquet, O., and Lugosi, G. (2005). Theory of Classification: A Survey of Some Recent Advances. *ESAIM: Probability and Statistics*, 9:323–375.

Bousquet, N., Lagoutte, A., Li, Z., Parreau, A., and Thomassé, S. (2015). Identifying codes in hereditary classes of graphs and VC-dimension. *SIAM Journal on Discrete Mathematics*, 29(4):2047–2064.

Brandes, U. (2001). A Faster Algorithm for Betweenness Centrality. *Journal of Mathematical Sociology*, 25(2):163–177.

Brautbar, M. and Kearns, M. (2010). Local algorithms for finding interesting individuals in large networks. In *ICS*.

Britton, T., Deijfen, M., and Martin-Löf, A. (2006). Generating simple random graphs with prescribed degree distribution. *Journal of statistical physics*, 124(6):1377–1397.

Broadbent, S. R. and Hammersley, J. M. (1957). Percolation Processes: I. Crystals and Mazes. *Math. Proc. of the Cambridge Philosophical Society*, 53(3):629–641.

Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., and Wiener, J. (2011). Graph structure in the web. In *The Structure and Dynamics of Networks*, pages 183–194. Princeton University Press.

Brodnik, A. and Grgurovič, M. (2017). Solving All-pairs Shortest Path by Single-source Computations: Theory and Practice. *Discrete Applied Mathematics*, 231:119–130.

Brönnimann, H. and Goodrich, M. T. (1995). Almost Optimal Set Covers in Finite VC-dimension. *Discrete & Computational Geometry*, 14(4):463–479.

Buriol, L. S., Frahling, G., Leonardi, S., and Sohler, C. (2007). Estimating clustering indexes in data streams. In *European Symposium on Algorithms*, pages 618–632. Springer.

Chan, T. M. (2012). All-Pairs Shortest Paths for Unweighted Undirected Graphs in o(mn) Time. *ACM Trans. Algorithms*, 8(4).

Chiba, N. and Nishizeki, T. (1985). Arboricity and subgraph listing algorithms. *SIAM Journal on computing*, 14(1):210–223.

Chung, F. (2010). Graph theory in the information age. *Notices of the AMS*, 57(6):726–732.

Chung, F. and Lu, L. (2002). Connected Components in Random Graphs with given Expected Degree Sequences. *Annals of Combinatorics*, 6(2):125–145.

Chung, F. and Lu, L. (2004). The Average Distance in a Random Graph with given Expected Degrees. *Internet Mathematics*, 1(1):91–113.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2022). *Introduction to algorithms*. MIT press.

Da Silva, M. O., Gimenez-Lugo, G. A., and Da Silva, M. V. G. (2013). Vertex Cover in Complex Networks. *International Journal of Modern Physics C*, 24(11):1350078.

Demaine, E. D., Reidl, F., Rossmanith, P., Villaamil, F. S., Sikdar, S., and Sullivan, B. D. (2019). Structural Sparsity of Complex Networks: Bounded Expansion in Random Models and Real-world Graphs. *Journal of Computer and System Sciences*, 105:199–241.

Dor, D., Halperin, S., and Zwick, U. (2000). All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759.

Ducoffe, G., Habib, M., and Viennot, L. (2020). Diameter computation on h-minor free graphs and graphs of bounded (distance) vc-dimension. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1905–1922. SIAM.

Easley, D. A. and Kleinberg, J. M. (2010). *Networks, Crowds, and Markets - Reasoning About a Highly Connected World*. Cambridge University Press.

Eirinakis, P., Williamson, M. D., and Subramani, K. (2017). On the Shoshan-Zwick Algorithm for the All-Pairs Shortest Path Problem. *J. Graph Algorithms Appl.*, 21(2):177–181.

Eubank, S., Kumar, V. A., Marathe, M. V., Srinivasan, A., and Wang, N. (2004). Structural and algorithmic aspects of massive social networks. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 718–727.

Faloutsos, M., Faloutsos, P., and Faloutsos, C. (2011). On power-law relationships of the internet topology. In *The Structure and Dynamics of Networks*, pages 195–206. Princeton University Press.

Ferrante, A., Pandurangan, G., and Park, K. (2008). On the Hardness of Optimization in Power-law Graphs. *Theoretical Computer Science*, 393(1-3):220–230.

Freeman, L. C. (1977). A Set of Measures of Centrality Based on Betweenness. *Sociometry*, pages 35–41.

Fronczak, A., Fronczak, P., and Hołyst, J. A. (2003). Mean-field theory for clustering coefficients in Barabási-Albert networks. *Physical Review E*, 68(4):046126.

Gandhi, S., Suri, S., and Welzl, E. (2010). Catching Elephants with Mice: Sparse Sampling for Monitoring Sensor Networks. *ACM Transactions on Sensor Networks (TOSN)*, 6(1):1–27.

Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability*, volume 174. freeman San Francisco.

Gast, M. and Hauptmann, M. (2014). Approximability of the vertex cover problem in power-law graphs. *Theoretical Computer Science*, 516:60–70.

Gast, M., Hauptmann, M., and Karpinski, M. (2012). Improved approximation lower bounds for vertex cover on power law graphs and some generalizations. *arXiv preprint arXiv:1210.2698*.

Gast, M., Hauptmann, M., and Karpinski, M. (2015). Inapproximability of dominating set on power law graphs. *Theoretical Computer Science*, 562:436–452.

Gkantsidis, C., Mihail, M., and Saberi, A. (2003). Conductance and Congestion in Power Law Graphs. In *Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 148–159.

Gross-Amblard, D. (2003). Query-Preserving Watermarking of Relational Databases and XML Documents. In *Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 191–201.

Guelzim, N., Bottani, S., Bourgine, P., and Képès, F. (2002). Topological and causal structure of the yeast transcriptional regulatory network. *Nature genetics*, 31(1):60–63.

Gupta, A. K. and Sardana, N. (2015). Significance of clustering coefficient over Jaccard index. In *2015 Eighth International Conference on Contemporary Computing (IC3)*, pages 463–466. IEEE.

Gusev, V. V. (2020). The vertex cover game: Application to transport networks. *Omega*, 97:102102.

Har-Peled, S. and Sharir, M. (2011). Relative (p, ε)-approximations in geometry. *Discrete & Computational Geometry*, 45(3):462–496.

Haussler, D. and Welzl, E. (1986). Epsilon-Nets and Simplex Range Queries. In *Proceedings of the Second Annual Symposium on Computational Geometry*, SCG '86, page 61–71, New York, NY, USA. Association for Computing Machinery.

Holland, P. W. and Leinhardt, S. (1971). Transitivity in structural models of small groups. *Comparative Group Studies*, 2(2):107–124.

Hoorfar, A. and Hassani, M. (2008). Inequalities on the Lambert-W function and hyperpower function. *J. Inequal. Pure and Appl. Math*, 9(2):5–9.

Iskhakov, L., Kamiński, B., Mironov, M., Prałat, P., and Prokhorenkova, L. (2020). Local clustering coefficient of spatial preferential attachment model. *Journal of Complex Networks*, 8(1):cnz019.

Javad-Kalbasi, M., Dabiri, K., Valaee, S., and Sheikholeslami, A. (2019). Digitally annealed solution for the vertex cover problem with application in cyber security. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2642–2646. IEEE.

Jeong, H., Mason, S. P., Barabási, A.-L., and Oltvai, Z. N. (2001). Lethality and centrality in protein networks. *Nature*, 411(6833):41–42.

Ji, Q., Li, D., and Jin, Z. (2020). Divisive algorithm based on node clustering coefficient for community detection. *IEEE Access*, 8:142337–142347.

Johnson, S. G. (2014). The NLopt Nonlinear-optimization Package. `http://github.com/stevengj/nlopt`.

Kartun-Giles, A. P. and Bianconi, G. (2019). Beyond the clustering coefficient: A topological analysis of node neighbourhoods in complex networks. *Chaos, Solitons & Fractals: X*, 1:100004.

Kearns, M. J., Vazirani, U. V., and Vazirani, U. (1994). *An introduction to computational learning theory*. MIT press.

Khot, S. and Regev, O. (2008). Vertex cover might be hard to approximate to within $2-\varepsilon$. *Journal of Computer and System Sciences*, 74(3):335–349.

Kleinberg, J. (2004). Detecting a network failure. *Internet Mathematics*, 1(1):37–55.

Kleinberg, J. and Lawrence, S. (2001). The structure of the web. *Science*, 294(5548):1849–1850.

Kleinberg, J., Sandler, M., and Slivkins, A. (2004). Network Failure Detection and Graph Connectivity. In *SODA*, volume 4, pages 76–85.

Kleinberg, J. M., Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A. S. (1999). The web as a graph: Measurements, models, and methods. In *International Computing and Combinatorics Conference*, pages 1–17. Springer.

Kolda, T. G., Pinar, A., Plantenga, T., Seshadhri, C., and Task, C. (2014). Counting triangles in massive graphs with mapreduce. *SIAM Journal on Scientific Computing*, 36(5):S48–S77.

Kranakis, E., Krizanc, D., Ruf, B., Urrutia, J., and Woeginger, G. (1997). The VC-dimension of Set Systems defined by Graphs. *Discrete Applied Mathematics*, 77(3):237–257.

Krot, A. and Ostroumova Prokhorenkova, L. (2015). Local clustering coefficient in generalized preferential attachment models. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 15–28. Springer.

Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tomkins, A., and Upfal, E. (2000). Stochastic models for the web graph. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 57–65. IEEE.

Kutzkov, K. and Pagh, R. (2013). On the streaming complexity of computing local clustering coefficients. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 677–686.

Leskovec, J. and Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`.

Li, M., Zhang, R., Hu, R., Yang, F., Yao, Y., and Yuan, Y. (2018). Identifying and ranking influential spreaders in complex networks by combining a local-degree sum and the clustering coefficient. *International Journal of Modern Physics B*, 32(06):1–14.

Li, X., Chang, L., Zheng, K., Huang, Z., and Zhou, X. (2017). Ranking weighted clustering coefficient in large dynamic graphs. *World Wide Web*, 20(5):855–883.

Liljeros, F., Edling, C. R., Amaral, L. A. N., Stanley, H. E., and Åberg, Y. (2001). The web of human sexual contacts. *Nature*, 411(6840):907–908.

Lima, A. M., da Silva, M. V., and Vignatti, A. L. (2022a). Percolation centrality via rademacher complexity. *Discrete Applied Mathematics*, 323:201–216.

Lima, A. M., da Silva, M. V. G., and Vignatti, A. L. (2020). Estimating the percolation centrality of large networks through pseudo-dimension theory. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, page 1839–1847, New York, NY, USA. Association for Computing Machinery.

Lima, A. M., Vignatti, A. L., and da Silva, M. V. G. (2019). Recognizing power-law graphs by machine learning algorithms using a reduced set of structural features. In *Anais do XVI Encontro Nacional de Inteligência Artificial e Computacional*, pages 611–621. SBC.

Lima, A. M. d., da Silva, M. V., and Vignatti, A. L. (2021a). A deductive-formal derivation for the preferential attachment metric for link prediction. In *Anais do IV Workshop de Pesquisa em Computação dos Campos Gerais*, pages 31–36.

Lima, A. M. d., da Silva, M. V. G., and Vignatti, A. L. (2022b). Estimating the clustering coefficient using sample complexity analysis. In *LATIN 2022: Theoretical Informatics: 15th Latin American Symposium, Guanajuato, Mexico, November 7–11, 2022, Proceedings*, page 328–341, Berlin, Heidelberg. Springer-Verlag.

Lima, A. M. d., Vignatti, A. L., and da Silva, M. V. (2021b). Problema APSP via Dimensão-VC e médias de rademacher. In *Anais do VI Encontro de Teoria da Computação*, pages 13–16. SBC.

Liu, S. and Xia, Z. (2020). A two-stage BFS local community detection algorithm based on node transfer similarity and local clustering coefficient. *Physica A: Statistical Mechanics and its Applications*, 537:122717.

Löffler, M. and Phillips, J. M. (2009). Shape Fitting on Point Sets with Probability Distributions. In *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, pages 313–324. Springer Berlin Heidelberg.

Miao, D., Liu, X., Li, Y., and Li, J. (2019). Vertex cover in conflict graphs. *Theoretical Computer Science*, 774:103–112.

Mitzenmacher, M. and Upfal, E. (2017). *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, 2nd edition.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of Machine Learning*. The MIT Press.

Molloy, M. and Reed, B. (1995). A Critical Point for Random Graphs with a given Degree Sequence. *Random structures & algorithms*, 6(2-3):161–180.

Molloy, M. and Reed, B. (1998). The Size of the Giant Component of a Random Graph with a given Degree Sequence. *Combinatorics, Probability and Computing*, 7(3):295–305.

Nacher, J. C. and Akutsu, T. (2016). Minimum dominating set-based methods for analyzing biological networks. *Methods*, 102:57–63.

Nascimento, M. C. (2014). Community detection in networks via a spectral heuristic based on the clustering coefficient. *Discrete Applied Mathematics*, 176:89–99.

Newman, M. E. J. (2010). *Networks: an introduction*. Oxford University Press.

Oneto, L., Ghio, A., Anguita, D., and Ridella, S. (2013). An Improved Analysis of the Rademacher Data-dependent Bound using its self Bounding Property. *Neural Networks*, 44:107 – 111.

Pan, X., Xu, G., Wang, B., and Zhang, T. (2019). A novel community detection algorithm based on local similarity of clustering coefficient in social networks. *IEEE Access*, 7:121586–121598.

Park, K. and Lee, H. (2001). On the Effectiveness of Route-based Packet Filtering for Distributed DoS Attack Prevention in Power-law Internets. *ACM SIGCOMM computer communication review*, 31(4):15–26.

Pettie, S. and Ramachandran, V. (2002). Computing Shortest Paths with Comparisons and Additions. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, pages 267–276, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Piraveenan, M., Prokopenko, M., and Hossain, L. (2013). Percolation Centrality: Quantifying Graph-Theoretic Impact of Nodes during Percolation in Networks. *PLOS ONE*, 8(1):1–14.

Provost, F., Jensen, D., and Oates, T. (1999). Efficient Progressive Sampling. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, page 23–32, New York, NY, USA. Association for Computing Machinery.

Radon, J. (1921). Mengen Konvexer Körper, die Einen Gemeinsamen Punkt Enthalten. *Mathematische Annalen*, 83(1-2):113–115.

Raz, R. and Safra, S. (1997). A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 475–484.

Redner, S. (1998). How popular is your paper? an empirical study of the citation distribution. *The European Physical Journal B-Condensed Matter and Complex Systems*, 4(2):131–134.

Riondato, M., Akdere, M., Çetintemel, U., Zdonik, S. B., and Upfal, E. (2011). The VC-Dimension of SQL Queries and Selectivity Estimation through Sampling. In Gunopulos, D., Hofmann, T., Malerba, D., and Vazirgiannis, M., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 661–676, Berlin, Heidelberg. Springer Berlin Heidelberg.

Riondato, M. and Kornaropoulos, E. M. (2016). Fast Approximation of Betweenness Centrality through Sampling. *Data Mining and Knowledge Discovery*, 30(2):438–475.

Riondato, M. and Upfal, E. (2014). Efficient Discovery of Association Rules and Frequent Itemsets through Sampling with Tight Performance Guarantees. *ACM Trans. Knowl. Discov. Data*, 8(4).

Riondato, M. and Upfal, E. (2015). Mining Frequent Itemsets through Progressive Sampling with Rademacher Averages. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 1005–1014, New York, NY, USA. Association for Computing Machinery.

Riondato, M. and Upfal, E. (2018). ABRA: Approximating Betweenness Centrality in Static and Dynamic Graphs with Rademacher Averages. *ACM Trans. Knowl. Discov. Data*, 12(5):61:1–61:38.

Riondato, M. and Vandin, F. (2020). MiSoSouP: Mining Interesting Subgroups with Sampling and Pseudodimension. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 14(5):1–31.

Roditty, L. and Shapira, A. (2011). All-pairs shortest paths with a sublinear additive error. *ACM Transactions on Algorithms (TALG)*, 7(4):1–12.

Roditty, L. and Vassilevska Williams, V. (2013). Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 515–524.

Rossi, R. A. and Ahmed, N. K. (2015). The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Santoro, D., Tonon, A., and Vandin, F. (2020). Mining Sequential Patterns with VC-Dimension and Rademacher Complexity. *Algorithms*, 13(5):123.

Schaefer, M. (1999). Deciding the Vapnik–Červonenkis Dimension is Σ_3^P-complete. *Journal of Computer and System Sciences*, 58(1):177–182.

Seidel, R. (1995). On the All-Pairs-Shortest-Path Problem in Unweighted Undirected Graphs. *Journal of Computer and System Sciences*, 51(3):400 – 403.

Seshadhri, C., Pinar, A., and Kolda, T. G. (2013). Fast triangle counting through wedge sampling. In *Proceedings of the SIAM Conference on Data Mining*, volume 4, page 5. Citeseer.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press.

Shoshan, A. and Zwick, U. (1999). All pairs shortest paths in undirected graphs with integer weights. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 605–614. IEEE.

Siganos, G., Faloutsos, M., Faloutsos, P., and Faloutsos, C. (2003). Power laws and the as-level internet topology. *IEEE/ACM Transactions on networking*, 11(4):514–524.

Soffer, S. N. and Vazquez, A. (2005). Network clustering coefficient without degree-correlation biases. *Physical Review E*, 71(5):057101.

Sontag, E. D. (1998). VC dimension of Neural Networks. *NATO ASI Series F Computer and Systems Sciences*, 168:69–96.

Vapnik, V. N. and Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280.

Vassilevska Williams, V. (2015). Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *10th International Symposium on Parameterized and Exact Computation (IPEC 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Vignatti, A. L. and Silva, M. V. G. (2016). Minimum vertex cover in generalized random graphs with power law degree distribution. *Theoretical Computer Science*, 647:101–111.

Wang, F., Camacho, E., and Xu, K. (2009). Positive influence dominating set in online social networks. In *International Conference on Combinatorial Optimization and Applications*, pages 313–321. Springer.

Wang, J. and Eppstein, D. (2006). Fast Approximation of Centrality. *Graph Algorithms and Applications*, 5(5):39.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world'networks. *Nature*, 393(6684):440–442.

Weisstein, E. (2013). Lambert W-function, from Wolfram Math-World.

West, D. B. (2000). *Introduction to Graph Theory*. Prentice Hall, 2 edition.

Williams, R. (2014). Faster All-pairs Shortest Paths via Circuit Complexity. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 664–673, New York, NY, USA. ACM.

Williams, R. (2018). Faster All-Pairs Shortest Paths via Circuit Complexity. *SIAM Journal on Computing*, 47(5):1965–1985.

Williams, V. V. and Williams, R. (2010). Subcubic Equivalences Between Path, Matrix and Triangle Problems. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 645–654. IEEE.

Wormald, N. C. (1980). Some Problems in the Enumeration of Labelled Graphs. *Bulletin of the Australian Mathematical Society*, 21(1):159–160.

Wu, J., Cardei, M., Dai, F., and Yang, S. (2006). Extended dominating set and its applications in ad hoc networks using cooperative communication. *IEEE Transactions on Parallel and Distributed Systems*, 17(8):851–864.

Wu, Z., Lin, Y., Wang, J., and Gregory, S. (2016). Link prediction with node clustering coefficient. *Physica A: Statistical Mechanics and its Applications*, 452:1–8.

Xu, Y.-Z. and Zhou, H.-J. (2016). Generalized minimum dominating set and application in automatic text summarization. *Journal of Physics: Conference Series*, 699:012014.

Zhang, H., Zhu, Y., Qin, L., Cheng, H., and Yu, J. X. (2017). Efficient local clustering coefficient estimation in massive graphs. In *International Conference on Database Systems for Advanced Applications*, pages 371–386. Springer.

Zhang, J., Tang, J., Ma, C., Tong, H., Jing, Y., and Li, J. (2015). Panther: Fast top-k similarity search on large networks. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1445–1454.

Zhang, R., Li, L., Bao, C., Zhou, L., and Kong, B. (2014). The community detection algorithm based on the node clustering coefficient and the edge clustering coefficient. In *Proceeding of the 11th World Congress on Intelligent Control and Automation*, pages 3240–3245. IEEE.