IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Algorithms for Metric Properties of Large Real-World Networks

From Theory to Practice and Back

Ph.D. Program:	**Computer, Decision, and System Science**
Curriculum:	**Computer Science**
Cycle:	**XXIX**

Ph.D. Student
Michele Borassi

Advisor
Prof. Pierluigi Crescenzi
Università di Firenze

Co-advisor
Prof. Rocco De Nicola
IMT School for Advanced Studies
Lucca

October 29, 2016

The dissertation of Michele Borassi is approved.

Program Coordinator:	Prof. Rocco De Nicola IMT School for Advanced Studies Lucca
Advisor:	Prof. Pierluigi Crescenzi Università di Firenze
Co-advisor:	Prof. Rocco De Nicola IMT School for Advanced Studies Lucca

The dissertation of Michele Borassi has been reviewed by:

First reviewer:	Prof. Tim Roughgarden Stanford University
Second reviewer:	Prof. Ulrik Brandes University of Konstanz
Third reviewer:	Prof. Kurt Mehlhorn Max Planck Institute for Informatics (Saarbrücken)

IMT School for Advanced Studies Lucca

October 29, 2016

To my family:
especially, the one
that my fiancée and I
will start with our wedding!

Contents

Technical sections are marked with (∗).

Vita

May 18, 1989	Born in Milan
2008-2011	Bachelor degree in Mathematics University of Pisa 110/110 *cum laude*
2008-2011	First Level Diploma Scuola Normale Superiore
2011-2013	Master degree in Mathematics University of Pisa 110/110 *cum laude*
2011-2013	Second Level Diploma Scuola Normale Superiore 70/70 *cum laude*
2013-2016	Ph.D. Student IMT School for Advanced Studies Lucca

Publications

Michele Borassi, Pierluigi Crescenzi, Michel Habib, Walter A. Kosters, Andrea Marino, and Frank W. Takes. On the solvability of the Six Degrees of Kevin Bacon game - A faster graph diameter and radius computation method. In *Proceedings of the 7th International Conference on Fun with Algorithms (FUN)*, pages 57–68, 2014

Michele Borassi, Pierluigi Crescenzi, Michel Habib, Walter A. Kosters, Andrea Marino, and Frank W. Takes. Fast diameter and radius BFS-based computation in (weakly connected) real-world graphs - With an application to the Six Degrees of Separation games. *Theoretical Computer Science*, 586:59–80, 2014

Michele Borassi, David Coudert, Pierluigi Crescenzi, and Andrea Marino. On computing the hyperbolicity of real-world graphs. In *Proceedings of the 23rd European Symposium on Algorithms (ESA)*, pages 215–226. Springer, 2015

Michele Borassi, Alessandro Chessa, and Guido Caldarelli. Hyperbolicity measures democracy in real-world networks. *Physical Review E*, 92(3):032812, 2015

Michele Borassi, Pierluigi Crescenzi, and Andrea Marino. Fast and simple computation of top-k closeness centralities. *arXiv preprint 1507.01490*, 2015

Michele Borassi, Pierluigi Crescenzi, and Michel Habib. Into the square - On the complexity of some quadratic-time solvable problems. *Electronic Notes in Computer Science*, 322:51–67, 2016

Michele Borassi and Emanuele Natale. KADABRA is an adaptive algorithm for betweenness via random approximation. In *Proceedings of the 24th European Symposium on Algorithms*, 2016

Michele Borassi. A Note on the Complexity of Computing the Number of Reachable Vertices in a Digraph. *Information Processing Letters*, 116(10):628–630, 2016

Elisabetta Bergamini, Michele Borassi, Pierluigi Crescenzi, Andrea Marino, and Henning Meyerhenke. Computing top-k closeness centrality faster in unweighted graphs. In *Proceedings of the Meeting on Algorithm Engineering and Experiments (ALENEX)*, pages 68–80, 2016

Michele Borassi, Pierluigi Crescenzi, and Luca Trevisan. An axiomatic and an average-case analysis of algorithms and heuristics for metric properties of graphs. In *Proceedings of the 27th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2017. Accepted

List of Figures

List of Tables

List of Notations

Symbols

$\mathbb{E}[\boldsymbol{X}]$ The expected value of the random variable $\boldsymbol{X}$.

$\mathbb{P}(\boldsymbol{E})$ The probability of the event $\boldsymbol{E}$.

$\sqcup$ Disjoint union.

Greek uppercase

$\Gamma(s)$ Set of neighbors of s, that is $\boldsymbol{\Gamma}^1(s)$.

$\boldsymbol{\Gamma}^\ell(s)$ The set of nodes at distance exactly ℓ from s.

$\boldsymbol{\Delta}^\ell(s)$ A set whose cardinality should be close to the number of nodes at distance $\ell+1$ from s.

$\boldsymbol{\Theta}^\ell(s)$ $\cup_{i=0}^{\ell}\boldsymbol{\Delta}^i(s)$.

Θ For two functions f, g, $f = \Theta(g)$ if $f = \mathcal{O}(g)$ and $g = \mathcal{O}(f)$.

Ω For two functions f, g, $f = \omega(g)$ if $g = \mathcal{O}(f)$.

Greek lowercase

α An integer.

$\alpha(s)$ Lower bound on the number $r(s)$ of nodes reachable from a given node s (see Section 5.6.4).

β The exponent of a power law degree distribution (that is, the probability that a node has degree d is proportional to $d^{-\beta}$).

$\boldsymbol{\gamma}^\ell(s)$ The number of nodes at distance exactly ℓ from s.

$\tilde{\gamma}^\ell(s)$ Upper bound on $\boldsymbol{\gamma}^\ell(s)$ (see Section 5.3).

δ The hyperbolicity of a graph, that is, the maximum of $\delta(s,t,u,v)$ over all quadruples of nodes (s,t,u,v).

$\delta(s,t,u,v)$ The hyperbolicity of the quadruple of nodes (s,t,u,v), that is, half the difference between the two biggest sums among $\operatorname{dist}(s,t) + \operatorname{dist}(u,v), \operatorname{dist}(s,u) + \operatorname{dist}(t,v), \operatorname{dist}(s,v) + \operatorname{dist}(t,u)$.

δ_{avg} The average hyperbolicity of a quadruple of nodes

δ_L A lower bound on the hyperbolicity.

$\boldsymbol{\delta}^\ell(s)$ The cardinality of $\boldsymbol{\Delta}^\ell(s)$.

ε A (small) constant, which usually does not depend on the number of nodes n in the graph.

η The distribution of a branching process conditioned on survival.

$\eta(1)$ The probability that the distribution η assigns to the outcome 1 (see [76]).

$\eta_L(v)$ The probability of error in a probabilistic lower bound on the betweenness of the node v.

$\eta_U(v)$ The probability of error in a probabilistic upper bound on the betweenness of the node v.

$\boldsymbol{\vartheta}^\ell(s)$ $\sum_{i=0}^{\ell} \boldsymbol{\delta}^i(s)$.

λ The additive error in the betweenness centrality approximation.

λ The degree distribution of the graph.

μ The residual distribution (see Definition 8.2).

$\boldsymbol{\pi}$ A shortest path.

ρ_v The weight of the node v (which should be close to the degree of v in random graphs).

σ_{st} The number of shortest paths from s to t.

$\sigma_{st}(v)$ The number of shortest paths from s to t passing through v.

τ A fixed time (usually used in a martingale).

$\boldsymbol{\tau}$ A stopping time for a martingale.

$\boldsymbol{\tau}_s\left(n^x\right)$ $\min\{\ell \in \mathbb{N} : \boldsymbol{\gamma}^\ell(s) > n^x\}$.

ω For two functions f, g, $f = \omega(g)$ if $g = o(f)$.

ω The minimum real number such that it is possible to multiply two $n \times n$ matrices in time $\mathcal{O}(n^\omega)$ (currently, it is known that $2 \leq \omega < 2.3728639$ [81]).

$\omega(s)$ Upper bound on the number $r(s)$ of nodes reachable from a given node s (see Section 5.6.4).

Roman uppercase

$\mathcal{C}$ A collection of subsets of a ground set X.

C A strongly connected component.

$\tilde{D}$ An approximation of the diameter D of a graph.

D The diameter of a graph.

D_L A lower bound on the diameter of a graph.

$\mathcal{E}$ The set of edges in the strong component graph.

E The set of edges.

$\mathcal{F}$ A σ-algebra, used for martingales.

$F\left(d \to S\right)$ The function defined in Definition A.8.

$\mathcal{G}$ The strong component graph.

G A graph.

L Lower bound on some quantity.

$L(s,r)$ Generic lower bound on $f(s)$, if $r(s)=r$.

$L^B(t)$ A lower bound on the backward eccentricity $\mathrm{ecc}^B(t)$ of the node t.

$L_\ell^{\mathrm{CUT}}(s,r)$ Lower bound on $f(s)$, if $r(s)=r$, defined as $(n-1)\frac{S_\ell^{\mathrm{CUT}}(s,r)}{(r-1)^2}$.

$L^F(s)$ A lower bound on the forward eccentricity $\mathrm{ecc}^F(s)$ of the node s.

$L_v^{\mathrm{LB}}(s,r)$ Lower bound on $f(s)$, if $r(s)=r$, defined as $(n-1)\frac{S_v^{\mathrm{LB}}(s,r)}{(r-1)^2}$.

$L^{\mathrm{NB}}(s,r)$ Lower bound on $f(s)$, if $r(s)=r$, defined as $(n-1)\frac{S^{\mathrm{NB}}(s,r)}{(r-1)^2}$.

M The sum of the weights of all nodes in the graph.

$M_i(\lambda)$ The ith moment of a distribution λ, that is, the expected value of $\boldsymbol{X}^i$, where $\boldsymbol{X}$ is a λ-distributed random variable.

$N_h(s)$ The set made by the h nodes closest to s (including s).

$\boldsymbol{N}^\ell(s)$ The set of nodes at distance at most ℓ from s.

$R(s)$ Set of nodes reachable from s (by definition, $s \in R(s)$).

$R^B(t)$ The set of nodes from which it is possible to reach a given node t.

$R^F(s)$ The set of nodes reachable from a given node s.

R_U An upper bound on the radius of a graph.

$\boldsymbol{S}$ A random variable (usually, a sum of other random variables).

$S(s)$ Total distance of node s, that is $\sum_{t\in R(s)} \mathrm{dist}(s,t)$.

$S_\ell^{\mathrm{CUT}}(s,r)$ Lower bound on $S(s)$ if $r(s)=r$, used in the `updateBoundsBFSCut` function (see Lemma 5.4).

$S_v^{\mathrm{LB}}(s,r)$ Lower bound on $S(s)$ if $r(s)=r$, used in the `updateBoundsLB` function (see Equations (5.3) and (5.4)).

$S^{\mathrm{NB}}(s,r)$ Lower bound on $S(s)$ if $r(s)=r$, used in the `computeBoundsNB` function (see Proposition 5.1).

$T\left(d \to n^x\right)$ The average number of steps for a node of degree d to obtain a neighborhood of n^x nodes.

U Upper bound on some quantity.

$U^B(t)$ An upper bound on the backward eccentricity $\mathrm{ecc}^B(t)$ of the node t.

$U^F(s)$ An upper bound on the forward eccentricity $\mathrm{ecc}^F(s)$ of the node s.

$\mathcal{V}$ The set of nodes in the strong component graph.

V The set of nodes.

V' A set of "canditate nodes" to be radial nodes (in Chapter 4, $V'=V_1'\cup V_2'$, where V_1' is the set of nodes in a component of maximum size, and V_2' is the set of nodes that are able to reach a node in V_1').

$\mathrm{Var}(\boldsymbol{X})$ The variance of the random variable $\boldsymbol{X}$.

VD The vertex diameter of a graph, that is, the maximum number of nodes in a shortest path (on unweigthed graphs, $\mathrm{VD} = D + 1$).

$\boldsymbol{X}$ A random variable.

X A set (usually, a ground set of some collection $\mathcal{C}$).

$\boldsymbol{Y}$ A random variable.

$\tilde{\boldsymbol{Z}}^{\ell}$ A branching process conditioned on survival.

$\boldsymbol{Z}^{\ell}$ A branching process.

$\boldsymbol{Z}^{\tau}$ A martingale.

Roman lowercase

a.a.s. Asymptotically almost surely (that is, with probability $1 - o(1)$).

$\mathrm{bc}(v)$ The betweenness centrality of a node v.

c The constant appearing in Axiom 1 (the letter c is also used to denote other constants throughout the thesis).

c A node which is usually assumed to be central.

$c(s)$ The closeness centrality of the node s, that is, $\frac{(r(s)-1)^2}{(n-1)\sum_{t \in R(s)} \mathrm{dist}(s,t)}$.

d An integer (usually, the degree of a node).

$\deg(s)$ The degree of a node s.

$\mathrm{dist}(s, t)$ The distance between two nodes s and t (unless explicitly stated, every edge has length 1).

$\mathrm{dist}_{\mathrm{avg}}$ The average distance between two nodes.

$\mathrm{dist}_{\mathrm{avg}}(n)$ A function of the number n of nodes and of the degree distribution of a random graph, which should be close to the average distance (see Axiom 1).

$\mathrm{ecc}(s)$ The eccentricity of a node s, that is, $\max_{t \in V} \mathrm{dist}(s, t)$ (see Definition 8.7).

$\mathrm{ecc}^B(t)$ The backward eccentricity of a node t.

$\mathrm{ecc}^B_{scc}(t)$ The backward eccentricity of a node t on the graph induced by the strongly connected component of t.

$\mathrm{ecc}^F(s)$ The forward eccentricity of a node s.

$\mathrm{ecc}^F_{scc}(s)$ The forward eccentricity of a node s on the graph induced by the strongly connected component of s.

f A function; the probability that there is an edge (v, w) in a random graph is denoted by $f(\frac{\rho_v \rho_w}{M})$).

$f(\tilde{\boldsymbol{b}}(v), \eta_L(v), \omega, \tau)$ A function used to provide a probabilistic lower bound in betweenness centrality approximation (see Theorem 6.5).

$f(s)$ Farness of node s, that is, $\frac{(n-1)\sum_{t \in R(s)} \mathrm{dist}(s,t)}{(r(s)-1)^2}$.

$g(\tilde{\boldsymbol{b}}(v), \eta_U(v), \omega, \tau)$ A function used to provide a probabilistic upper bound in betweenness centrality approximation (see Theorem 6.5).

h An integer.

k An integer.

ℓ An integer, usually used to denote levels of a BFS tree, or distances.

m The number of edges in the graph.

m_{avg} The average number of edges visited by an algorithm.

m_{vis} The number of edges visited by an algorithm

n The number of nodes in the graph.

$\boldsymbol{n}^{\ell}(s)$ The number of nodes at distance at most ℓ from s.

$o, \mathcal{O}$ The standard asymptotic notations.

$\text{outdeg}(v)$ Out-degree of a node in a directed graph.

p A polynomial.

$r(s)$ Number of nodes reachable from s, including s (in other words, $\big|R(s)\big|$).

s A node (usually, the starting node of a BFS, or of a path).

t A node (usually, the last node of a path).

u A node.

v A node.

w A node.

w.h.p. With high probability (that is, with probability $1 - o\left(n^{-k}\right)$ for each $k \in \mathbb{N}$).

x A real number, usually between 0 and 1.

y A real number, usually between 0 and 1.

Acknowledgments

In this section, I would like to thank everyone who helped me writing this thesis: without them, I would never be able to complete this work!

First and foremost, my supervisor Pierluigi Crescenzi: even before starting my PhD, he taught me how to program, we worked together on my Master thesis, and on two papers. Shortly after I started my PhD, he pointed me to interesting and challenging problems, where theory and practice interact with each other. Even during my PhD, he was more than a supervisor: he really cared for my results, he introduced me to several people who helped me in this work, and he listened to me mumbling about probabilistic stuff for hours and hours! Thank you very much!

A special thanks goes to IMT, the institution that hosted me for these three fantastic years: in particular, many thanks to my internal advisor Rocco De Nicola. I would really have liked to write a paper together, and during my first year I tried to work with him, but unfortunately our research interests are very far. I hope that, in the future, we will be able to find a common ground!

In IMT, I would also like to thank Guido Caldarelli and Alessandro Chessa, with whom I collaborated in writing a paper: this paper dealt more with physics than computer science, and it opened me a whole new research area. It was a great adventure, but unfortunately I did not have time to continue in this direction: I hope that, in the future, we will be able to continue working together on these topics!

Furthermore, during my PhD, I spent four months as a guest of the Simons Institute for the Theory of Computing, UC Berkeley. I would like to thank the institute, and in particular Luca Trevisan, who invited me and helped me a lot in writing the paper presented in Chapter 8. I also thank him for suggesting me to publish this work at the SODA conference: I am really, really happy that this paper got accepted!

Many thanks also to all my co-authors, which I list here in the order in which I met them. First, Andrea Marino: we published seven papers together, five of which are in this thesis! In terms of number of papers, he is only after my supervisor Pierluigi (ten papers, eight of which are in this thesis).

Furthermore, I would like to thank Frank Takes and Walter Kosters, with whom I published the first two papers of this PhD!

Then, many thanks to Michel Habib, who invited me to Paris on my first year (I spent an awesome week there), who helped me a lot with the paper from which Chapter 3 is taken, and for continuing working with me for my whole PhD (we were co-authors of two more papers).

Among my other co-authors, a special thanks goes to David Coudert and Nathann Cohen, with whom I worked in Google Summer of Code 2015 - a project financed by Google, in which I was asked to implement new algorithms and speed-up existing algorithms in the SageMath graph library. I am really grateful to them: if Pierluigi taught me how to program, David and Nathann taught me how to write good code, and how to cooperate in an open-source project. Furthermore, I collaborated with David on our paper on the hyperbolicity. Thank you very much!

And thanks to Elisabetta Bergamini and Henning Meyerhenke, for teaching me how to work with NetworKit and for the nice and funny story of the conference ALENEX 2016:

we submitted two very similar papers, but luckily we realized the problem shortly after the deadline. We merged the two papers, and we started a collaboration!

I also want to thank Paolo Boldi and Sebastiano Vigna: even if we did not publish a paper together, we worked together on the implementation of my algorithms in the great WebGraph library. Furthermore, thank you very much for referencing my application for a Google internship, which might now be transformed in a full-time position: you don't know how much this means to me!

Among my co-workers, I also want to thank the reviewers of this thesis: Tim Roughgarden, Ulrik Brandes, and Kurt Mehlhorn. Their thorough reviews and their suggestions helped me improving significantly the quality of this thesis! Thank you very much!

However, the acknowledgments section is not only for co-workers: at the beginning, I said that "I would like to thank everyone who helped me writing this thesis". By this, of course, I also meant my family and friends!

First and foremost, I would really like to thank my fiancée Sara, who always supported me when I had problems, and who partied with me when I received good news! Thanks to her, I have spent three wonderful years of PhD: not only I had a lot of fun with my work, but I was even happier when I could spend some time at home! And thank you for preventing me from focusing only on the work, for always being there, and for sharing everything during these years! From a practical point of view, thank you for teaching me English, by forcing me to watch movies in the original language, thank you for teaching me how to cook and keep a house, thank you for dancing with me every Friday, thank you for always taking me out when I was lazy, and for a thousand other things! I love you!

During these three years, I also received a lot of support from my family: my mum and dad were always there when I needed them, and they helped me a lot with my work! Thank you very much! Many thanks also the rest of my family: even if we did not meet very often, I believe that we spent some quality time together!

Moreover, I would like to thank the only person in the intersection between the set of my co-authors and the set of my friends: Emanuele Natale. I decided to thank him among my friends, because spending four great months in Berkeley in a 15 squared meters room beats a paper together! Anyway, I am very happy that we shared this adventure, and that we received the Best Student Paper Award together at ESA! And thanks to all the other people who shared this adventure with us: among many others, thanks to Marc Roth, Cornelius Brand, Holger Dell, Judith Abecassis, John Lapinskas, Heng Guo, Marylou Gabrié, and Marine Le Morvan.

Furthermore, many thanks to all my friends in the Nineteenth Century Dance School, with whom I spent wonderful evenings!

Last, but not least, I would like to thank all my friends from Scuola Normale Superiore (Pisa), who are now spread all over the world. Thank you for our dinners, for the holidays in Ravascletto, for running together, for playing nerd games with me, and for the nights spent playing Age of Empires II! In particular, many thanks to Fabrizio Bianchi (who will soon be my best man), Marco Marengon, Giovanni Mascellani, Irene Regini, Giovanni Paolini, Aleksandra Baranova, Davide Lombardo, Alessandra Caraceni, Gennady Uraltsev, Davide Orsucci, Francesco Guatieri, Marco Fantini, Enrico Pracucci, Alice Leone, Sara Boezio, and all the other people that, unfortunately, I cannot mention for space constraints!

Abstract

Motivated by complex networks analysis, we study algorithms that compute metric properties of real-world graphs. In the worst-case, we prove that, under reasonable assumptions, the trivial algorithms based on computing the distance between all pairs of nodes are almost optimal.

Then, we try to overcome these bottlenecks by designing new algorithms that work surprisingly well in practice, even if they are not efficient in the worst-case. We propose new algorithms for the computation of the diameter, the radius, the closeness centrality, the betweenness centrality, and and the hyperbolicity: these algorithms are much faster than the *textbook* algorithms, when tested on real-world complex networks, and they also outperform similar approaches that were published in the literature. For example, to solve several problems, our algorithms are thousands, and even billions of times faster than the textbook algorithm, on standard inputs.

However, the experimental results are not completely satisfactory from a theoretical point of view. In order to fill this gap, we develop an axiomatic framework where these algorithms can be evaluated and compared: we define some axioms, we show that real-world networks satisfy these axioms, and we prove that our algorithms are efficient if the input satisfies these axioms. This way, we obtain results that do not depend on the specific dataset used, and we highlight the main properties of the input that are exploited. A further confirmation of the validity of this approach is that the results obtained mirror very well the empirical results.

Finally, we prove that the axioms are verified on realistic models of random graphs, such as the Configuration Model, the Chung-Lu model, and the Norros-Reittu model. This way, our axiomatic analyses can be turned into average-case analyses on these models, with no modification. This modular approach to average-case complexity has two advantages: we can prove results in several models with a single worst-case analysis, and we can validate the choice of the model by showing that the axioms used are verified in practice.

Chapter 1

Introduction

In many different areas of science, scientists use graphs to model several phenomena, such as interactions between individuals, protein reactions in a cell, links between electronic devices, citations between papers, co-occurrences of words in texts, etc. Recently, thanks to the development of social networks, to the overwhelming power of modern computers, and to several other factors, the size of the data available to scientists has drastically increased: now, it is quite common to deal with graphs of millions, and even billions of nodes and edges.

On the one hand, thanks to the availability of these data, we can perform new and more precise analyses, that were unthinkable in the past. On the other hand, we have to face new challenges, because it is quite hard to process all these data, even with modern computers: polynomial-time algorithms, and even quadratic-time algorithms, might not terminate in a reasonable amount of time on these large inputs. For this reason, several works have tried to develop faster and faster algorithms.

Among the properties of interest, a significant role is played by metric properties: basically, we turn the graph into a metric space by defining the distance between two nodes s and t as the number of edges in a shortest path from s to t. Then, we can analyze the properties of this metric space: the diameter, the radius, the average distance, the hyperbolicity, and so on. Furthermore, we can use this metric to define centrality measures, such as eccentricity, closeness centrality, and betweenness centrality. All these quantities are precisely defined in Chapter 2, where we also provide some basic terminology and we state the most important problems considered in this thesis.

Unfortunately, in the worst-case, the best way to address these problems is through trivial approaches, based on computing the distance between all pairs of nodes: indeed, the existence of significantly faster algorithms would falsify widely believed conjectures, such as the Strong Exponential Time Hypothesis [92], the Orthogonal Vector conjecture [3], and the Hitting Set conjecture [3]. These hardness results are the topic of Chapter 3, where we collect the main existing hardness results, and we prove new ones.

Knowing these results, people tried to find a way to overcome them. A first possibility is to use approximation algorithms [138, 47], or parametrized algorithms [3]. However, in practice, these approaches are not widespread: indeed, these algorithms are quite complicated, and there is no implementation available. For these reasons, the most common approach used is empirical: in some cases, researchers use heuristics with no guarantee at all on the quality of the solution obtained, while in other cases they use exact algorithms that have the same performances as the trivial approach, in the worst-case. However, despite the lack of theoretical guarantees of these algorithms, they are surprisingly very good when they are tested on real-world inputs.

In this thesis, we develop new such algorithms, and we experimentally show that they improve over existing ones, by testing them on large datasets of real-world networks. These algorithms are not only theoretical: their implementation is publicly available in the Webgraph library [23], in the Sagemath graph library [152], or in the NetworKit library [151].

More specifically, in Chapter 4, we study the computation of the diameter, defined as the maximum distance between two nodes, and the radius, defined as $\min_{s \in V} \max_{t \in V} \operatorname{dist}(s,t)$. We provide a new heuristic to lower bound the diameter and upper bound the radius, which improves over the existing heuristics, and we turn this heuristic into an exact algorithm, by adding a verification phase. The latter algorithm outperforms the best existing algorithms, and it makes it possible to compute the radius and the diameter of several graphs with millions of nodes and hundreds of millions of edges.

In Chapter 5, we provide a new algorithm for computing the k most central nodes according to *closeness centrality.* The new algorithm outperforms both the trivial algorithm and previous approaches, making it possible to compute the 100 most central nodes in very large networks.

In Chapter 6, we provide a new algorithm that approximates the *betweenness centrality* of a node. This algorithm exploits a new adaptive sampling technique, which works on general graphs, and the concept of balanced bidirectional breadth-first search, which has extremely good performances on real-world graphs. Indeed, in several networks, the time needed to compute a shortest path with this technique is sublinear in the input size, and in some cases it is even close to $\sqrt{m}$, where m is the number of edges in the graph.

In Chapter 7, we consider the problem of computing the hyperbolicity of a graph. This problem is quite different from the other problems considered in this work: indeed, it is much harder than the previous ones in the worst-case, because the textbook algorithm needs time $\mathcal{O}(n^4)$. We describe a new algorithm that outperforms the fastest existing algorithm, when tested on practical inputs, and it makes it possible to compute the hyperbolicity of graphs with up to 70 000 nodes.

However, these results are not completely satisfactory from a theoretical point of view, because the validation of the efficiency of these algorithms has always been empirical: by running them on practical instances, one can show that they outperform the textbook algorithm, or similar heuristic approaches [23, 83, 141, 50, 114, 62, 154, 165, 63, 155, 57, 61, 66, 67, 8, 129, 58]. This kind of analysis has significant disadvantages: it might be biased by the choice of the specific dataset used for the evaluation, and it provides no insight into the reasons why these algorithms are so efficient.

Obviously, the efficiency of these algorithms is linked to some properties of the typical input [155, 153, 116]: indeed, although the networks analyzed come from different research fields, they have some characteristics in common. For example, they are all quite sparse, their degree distribution is power law, they show high clustering coefficient, and the distances are usually very small (*small-world* phenomenon). Usually, networks showing these properties are called *complex networks*.

However, little is known about the link between the properties of complex networks and the efficiency of these heuristics and algorithms: the major result of this thesis is to fill this gap, by developing a theoretical framework in which these algorithms can be evaluated and compared. Our framework is axiomatic: we define some axioms, we experimentally show that these axioms hold in many complex networks, and we perform a worst-case analysis on the class of graphs satisfying these axioms (Chapter 8). This analysis improves the standard empirical analysis for three main reasons: we formally validate the efficiency of the algorithms considered, we highlight the properties exploited, and we perform a comparison that does not depend on the specific dataset used for the evaluation. A further confirmation of the validity of this approach comes from the results obtained, that are very similar to existing empirical results.

Furthermore, we show that these axioms are satisfied by some models of random graphs, asymptotically almost surely (a.a.s.), that is, with probability that tends to 1 as the number of nodes n goes to infinity: as a consequence, all these results can be turned into average-case analyses on these models, with no modification. This modular approach to average-case complexity has two advantages: since our axioms are satisfied by different models, we can prove results in all these models with a single worst-case analysis. Furthermore, we clearly highlight which properties of random graphs we are using: this way, we can experimentally validate the choice of the probabilistic model, by showing that these properties are reflected by real-world networks. Since the proof is rather technical, Chapter 8 only contains a sketch, and the full proof is available in Appendix A.

Most of the work presented has already been published in [33, 32, 30, 29, 34, 31, 35, 36, 28, 20]. For more information, at the beginning of each chapter, we cite all the papers that contributed to the chapter itself; moreover, in a small literature review at the end of each chapter, we explain in detail the contributions of the single papers, and we acknowledge other people's contributions.

All the chapters are almost independent and self-contained, assuming the reader knows the content of Chapter 2, which defines the basic notions used in this thesis. To help the reader, we have also collected all notations used throughout the thesis in the Table of Notations at the beginning.

Finally, in order to improve readability, we marked all technical sections with $(*)$ in the Table of Contents.

1.1 Foundations and Related Work

This thesis combines results in several research fields: worst-case polynomial reductions, the design of heuristics and algorithms that are efficient on real-world graphs, metric properties of real-world and random graphs, probabilistic analysis of algorithms. Since it is impossible to provide a comprehensive account of the state of the art in all these areas, we just point the reader to the most recent and comprehensive surveys.

1.1.1 Worst-Case Polynomial Reductions

In the first chapter of this thesis, we prove that, under reasonable complexity assumptions, the problems considered in this thesis are hard in the worst-case. This chapter takes inspiration from a large amount of research on reductions between polynomial-time solvable problems: usually, these reductions show that a problem does not admit a fast algorithm, unless widely believed conjectures prove to be false.

One of the most famous conjectures considered in the literature is the 3Sum conjecture given three sets A, B, and C of n integers, it is not possible to decide whether there exists $a \in A, b \in B$, and $c \in C$ such that $a+b+c=0$, in time $\mathcal{O}(n^{2-\varepsilon})$. This problem has been widely studied, but the best algorithms so far are only *mildly subquadratic*, that is, their time complexity is $o(n^2)$, but not $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$. The first 3Sum-based reductions appear in [80], and since then several problems were proved to be 3Sum-hard, especially in computational geometry (for a comprehensive account, we refer to the surveys [99, 91]).

In the context of graph theory, very different results hold if we consider dense graphs (where $m = \mathcal{O}(n^2)$, where n is the number of nodes and m is the number of edges), and sparse graphs (where $m = \mathcal{O}(n)$). Let us first consider dense graphs: some works have proved hardness results if the graph is weighted, assuming that the All Pairs Shortest Paths problem is not solvable in $\mathcal{O}(n^{3-\varepsilon})$ [171, 1]. In the case of unweighted graph, few results are available, because, in this setting, several problems are linked to matrix multiplication, a problem whose complexity is not completely understood [170, 81].

Another line of research has considered sparse graphs, where $m = \mathcal{O}(n)$, which should better model real-world complex networks [125]. In this setting, many hardness results follow from the Strong Exponential Time Hypothesis (SETH), which says that there is no algorithm for solving the k-Sat problem in time $\mathcal{O}((2-\varepsilon)^n)$, where $\varepsilon > 0$ does not depend on k [92]. The first reduction dealing with this hypothesis was proved in [168], and it shows the hardness of finding two disjoint sets in a collection of n sets. Following the same line, many other papers proved similar results for specific problems [131, 138, 47, 169, 44, 4, 2]. Finally, in [3], the authors prove results based on two new conjectures: the Orthogonal Vector conjecture and the Hitting Set conjecture. The Orthogonal Vector conjecture simply says that the problem of finding two disjoint sets in a collection of n sets is hard, and it implies SETH, as already shown in [168]. The Hitting Set conjecture is a sort of dual of the Orthogonal Vector conjecture, and it says that it is hard to find a hitting set in a collection of n sets. Until now, very little is known on the latter conjecture, apart from the fact that it implies the Orthogonal Vector conjecture [3].

Our contribution in this field is a collection of all the results on sparse graphs in a unified work, and the proof of some new reductions, dealing with the problems considered in this thesis. These results further confirm the interest in the probabilistic analyses performed in previous chapters: indeed, they show a clear separation between what can be proved in the worst-case and what can be obtained in our setting.

1.1.2 Efficient Algorithms on Complex Networks

In the past, researchers published several algorithms that achieve good performances in practice, even if there is no guarantee in the worst-case [23, 83, 141, 50, 114, 62, 154, 165, 63, 155, 57, 61, 66, 67, 8, 129, 58]. All these works follow the same principle: they describe an algorithm, they prove that the result of the algorithm is correct, but they do not provide theoretical guarantees on the efficiency of the algorithm. Then, they collect a dataset of practical instances, and they show that the running time of the algorithm on these instances improves a baseline, that can be the *textbook* algorithm, or other similar approaches. The underlying assumption is that the dataset considered encompasses all kinds of inputs that arise in practice.

This approach was very successful in two cases: road networks and complex networks. For the former setting, we refer the interested reader to [5] and the references therein; for the latter setting, we refer the reader to the literature reviews at the end of Chapters 4 to 7, which contain a complete account of the state of the art in the computation of the quantities considered in each single chapter.

Our contribution in this field is the design of new algorithms that outperform all existing approaches, both in our experiments and in the probabilistic analyses in our framework, and the implementation of these new algorithms in well-known graph libraries: WebGraph [23], and Sagemath [152].

1.1.3 Random Graphs and Probabilistic Analyses

In the literature, there are several paper that study metric properties of random graphs: most of these results are summarized in [160, 159]. Furthermore, the proofs of our main theorems are generalizations and adaptations of the proofs in [76, 127, 27], if the average degree is finite. Our main contribution in this area is a unified analysis of the different models, and the generalization to infinite mean degrees. As far as we know, the only work that addresses the latter case is [158], which only computes the typical distance between two nodes

Furthermore, our work relies on several works that outline the main properties of complex networks, and that develop models that satisfy such properties: for example, the choice of the power law degree distribution is validated by extensive empirical work (for a survey, we refer to [126]).

Despite this large amount of research on models of real-world graphs, few works have tried to address the problem of evaluating heuristics and algorithms on realistic models. Among these works, one of the most successful approaches deals with heuristic approaches in the computation of shortest paths in road networks [83, 141, 67, 66]. In [5], the authors provide an explanation of the efficiency of these approaches, based on the concept of *highway dimension*. Another algorithm achieving very good performances in practice is the graph compression technique in [23], implemented in the WebGraph library. In [53], the authors prove that in most existing models no algorithm can achieve good compression ratio, and they provide a new model where the algorithm in [23] works well.

Another similar approach is provided in [38]: assuming only that the degree distribution is power law, the authors manage to analyze some algorithms, and to prove that the axiomatic analysis improves the worst-case analysis. However, their axioms apply to local properties, such as patterns in subgraphs, but not to the metric properties that we study in this thesis.

Moreover, in [85], the authors define the notion of *triangle dense graphs*, that captures the high number of triangles present in most complex networks (intuitively, a friend of my friend is likely to be my friend). Under this assumption, they prove that it is possible to find dense subgraphs with small radius, in a way that they contain many of the triangles in the graph. In the future, this decomposition might be used to speed-up algorithms on triangle dense graphs, as suggested by one of the open problems in the conclusion. This work sets forth the research agenda of defining worst-case conditions on graphs, that are meant to generalize all of the popular generative models for complex networks: it discusses the main features of this approach, and it provides a practical application. This thesis belongs to the same research area: indeed, most of the advantages and disadvantages of our approach are already described in [85]. However, with respect to [85], our assumptions are more specific, and they are tailored to the computation of metric properties. On the one hand, this restricts the domain of application to metric algorithms, and it makes it harder to prove these properties on many graph models: for this reason, we restrict our attention to simple models, that can be analyzed mathematically (probably, our assumptions hold in many other models, but it might be very difficult to prove it). On the other hand, thanks to our tailored assumptions, we can carry on the analysis of practical algorithms, used all over the world, concluding the bridge between theory and practice that was mentioned in [85] as an open problem.

Other papers have tried to follow a similar approach with metric properties: the most notable attempt [50] deals with the *Gromov hyperbolicity* of the input graph [84]. For example, the 2-Sweep heuristic [114] provides good approximations of the diameter of hyperbolic graphs. However, this approach cannot be applied to some algorithms, like the iFub, and when it can be applied, the theoretical guarantees are still far from the empirical results, because real-world graphs are usually not hyperbolic according to Gromov's definition [29]. Another attempt was to consider the correlation between the efficiency of the algorithm and specific characteristics of the input graph [153]: however, also in this case, no definitive explanation of the efficiency of these algorithm was found.

In this work, we give the first *fullproof* argument that shows the efficiency of algorithm for metric properties on complex networks. Differently from previous attempts, this analysis works with several different algorithms, and the results are in line with practical experiments. Furthermore, as a side result of this analysis, we obtain a new point of view in the analysis of random graphs, that provides simpler proofs of existing results, and some *new* results, such as the asymptotic value of the diameter of a random power law graph where the average degree is infinite.

Chapter 2

Preliminaries

Abstract

In this chapter, we introduce some basic terminology that is repeatedly used throughout this work, and we define the main problems addressed. Then, we briefly sketch the results obtained.

2.1 Basic Graph Definitions

In this section, we define the graph quantities that we study in this thesis, together with some basic terminology. For a more detailed treatment, we refer to any introductory book on graph theory, such as [59].

First of all, a graph G is a pair (V, E), where V is the set of nodes, and $E \subseteq V^2$ is a set of edges (which can also be seen as a binary relation on V). If two nodes v, w are linked by an edge, we say that $E(v, w)$ holds, using a relational notation. We denote $n = |V|$ and $m = |E|$; furthermore, many times, we use integers from 0 to $n - 1$ to denote the nodes, and consequently pairs of integers to denote the edges. Finally, for each node $v \in V$ in an undirected (resp., directed) graph, we denote by neighbors of v the set of nodes $w \in V$ such that $E(v, w)$ holds.

If the edge relation E is symmetric (that is, $E(u, v)$ holds if and only if $E(v, u)$ holds), then we say that the graph is *undirected*, and with abuse of notation we define m as the number of undirected edges (that is, we count the pair (u, v) and the pair (v, u) only once). Otherwise, we say that the graph is *directed* (we often use the term digraph to abbreviate directed graph). In undirected graphs, we draw edges as lines, while, in directed graphs, we use arrows to keep track of the direction of an edge. Examples are provided in Figures 2.1 and 2.2.

In general, we say that a node t is reachable from a node s if there is a path from s to t, that is, a sequence $s = v_0, \ldots, v_k = t$ of nodes such that $E(v_i, v_i + 1)$ holds for each i between 0 and $k - 1$. The set of nodes reachable from a given node s is denoted by $R^F(s)$, and the set of nodes from which it is possible to reach a node t is denoted by $R^B(t)$. Note that, if G is undirected, then $R^F(s) = R^B(s)$ for any node s: hence, we drop the superscript and we simply use $R(s)$). We also use the notation $r(s)$ to denote $|R(s)|$.

An undirected (resp., directed) graph is *connected* (resp., *strongly connected*) if, for any node s, $R^F(s) = V$ (equivalently, $R^B(s) = V$). A digraph is said to be *weakly connected* if the undirected graph resulting from removing the orientation of the edges is connected.

A *connected component* (resp., *strongly connected component* or SCC) of an undirected (resp., directed) graph is a subgraph that is connected (resp., strongly connected), and is maximal with respect to this property. The *strong component graph* of a digraph G is the directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{C_1, \ldots, C_k\}$ is the set of SCCs of G and an

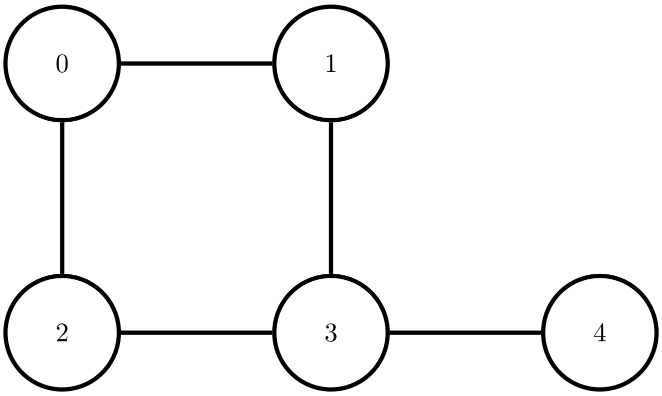


Figure 2.1. An example of undirected graph, where $n = 5$ and $m = 5$.

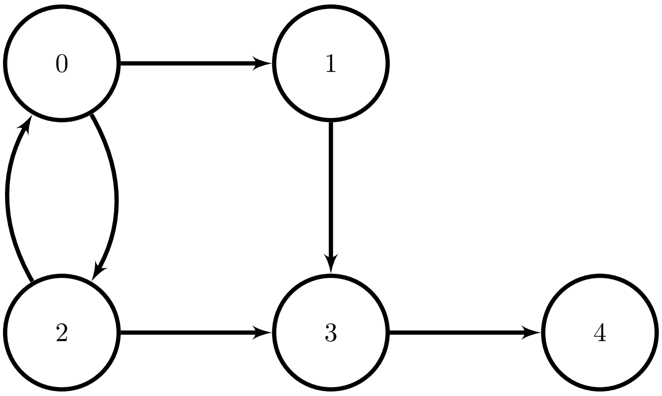


Figure 2.2. An example of directed graph, where $n = 5$ and $m = 6$.

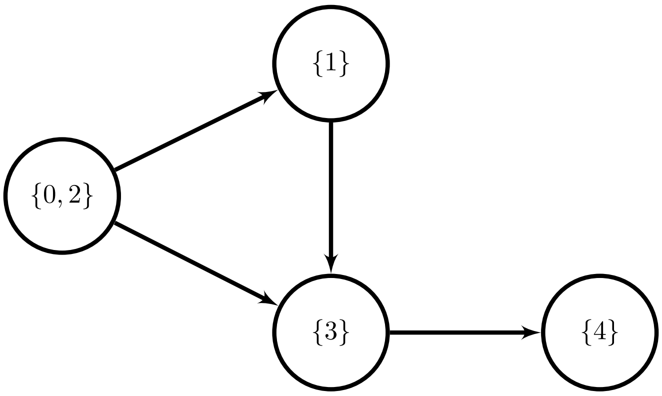


Figure 2.3. The strong component graph corresponding to the graph in Figure 2.2.

edge (C_i, C_j) exists if there is at least one edge in G from a node in the component C_i to a node in the component C_j. Observe that $\mathcal{G}$ is an acyclic digraph: hence, we may assume that a *topological order* is specified for its nodes, that is, $\mathcal{V}$ is ordered such that, for each $(C_i, C_j) \in \mathcal{E}$, $i < j$. For more background on these concepts, we refer to [14].

For example, the graph in Figure 2.1 is connected, and consequently it has one connected component, while the graph in Figure 2.2 is not connected, and its strongly connected components are the sets of vertices $\{0, 2\}, \{1\}, \{3\}, \{4\}$. The corresponding strong component graph is plotted in Figure 2.3.

To conclude this section, let us make some assumptions that simplify our notations, without affecting the generality of our results. First of all, a graph might have isolated nodes, that is, nodes that do not belong to any edge. Since these nodes have no effect on distances, and since they can be easily removed from a graph, we assume that they do not exist. Furthermore, in some cases, researchers have studied *multigraphs*, where two nodes can be connected by multiple edges, and graphs with self-loops, where there can be an edge (s, s), for the same vertex s. However, removing multiple edges and self loops has no effect on distances: for this reason, from now on we assume that all our graphs have no multiple edge and no self-loop.

2.2 Metric Properties of Graphs

This thesis is focused on metric properties: basically, we transform the graph into a metric space by defining the distance $\mathrm{dist}(s,t)$ between two nodes s, t as the minimum number of edges in a path from s to t, or $+\infty$ if there is no such path (that is, s and t are not connected). In other words, a node s is at distance 1 from all its neighbors, at distance 2 from the neighbors of its neighbors, and so on. For example, both in Figure 2.1 and in Figure 2.2, $\mathrm{dist}(0,1) = 1$, $\mathrm{dist}(0,3) = 2$ (there are two different shortest paths of length 2, namely $(0,1,3)$ and $(0,2,3)$), and $\mathrm{dist}(0,4) = 3$. Furthermore, in Figure 2.2, $\mathrm{dist}(1,0) = +\infty$, because there is no path from 1 to 0.[1]

It is easy to prove that this definition satisfies the following properties:

- non-negativity, that is, $\mathrm{dist}(s,t) \geq 0$;
- identity of indiscernibles, that is, $\mathrm{dist}(s,t) = 0$ if and only if $s = t$;
- triangle inequality, that is, $\mathrm{dist}(s,u) \leq \mathrm{dist}(s,t) + \mathrm{dist}(t,u)$.

In order to prove that the quantity dist is a distance in the mathematical sense, we also need the symmetric property, that is, $\mathrm{dist}(s,t) = \mathrm{dist}(t,s)$: however, this property is satisfied only if the graph is undirected (for example, in Figure 2.2, $\mathrm{dist}(0,1) = 1$ and $\mathrm{dist}(1,0) = +\infty$). In any case, following the standard convention, we improperly use the term distance even in the directed case.

In the following, we denote by $\boldsymbol{\Gamma}^\ell(s)$ the set of nodes at distance exactly ℓ from s, by $\boldsymbol{N}^\ell(s)$ the set of nodes at distance at most ℓ from s. Furthermore, we let $\boldsymbol{\gamma}^\ell(s) = |\boldsymbol{\Gamma}^\ell(s)|$, and $\boldsymbol{n}^\ell(s) = |\boldsymbol{N}^\ell(s)|$. For example, in the graph in Figure 2.1, $\boldsymbol{\Gamma}^0(0) = \{0\}$, $\boldsymbol{\Gamma}^1(0) = \{1,2\}$, $\boldsymbol{\Gamma}^2(0) = \{3\}$, and $\boldsymbol{\Gamma}^3(0) = \{4\}$; $\boldsymbol{N}^0(0) = \{0\}$, $\boldsymbol{N}^1(0) = \{0,1,2\}$, $\boldsymbol{N}^2(0) = \{0,1,2,3\}$, and $\boldsymbol{N}^3(0) = V$.

The diameter of a graph is the maximum distance between two connected nodes, that is,

$$D = \max_{s,t \in V, \mathrm{dist}(s,t) < +\infty} \mathrm{dist}(s,t).$$

In other works, D is simply defined as $\max_{s,t \in V} \mathrm{dist}(s,t)$, so that each disconnected graph has diameter $+\infty$: however, since most real-world graphs are not (strongly) connected, in this thesis we focus on the former definition, which is also more informative (for more information, we refer to Chapter 4).

The eccentricity of a node s is

$$\mathrm{ecc}(s) = \max_{t \in V, \mathrm{dist}(s,t) < +\infty} \mathrm{dist}(s,t).$$

By definition, $D = \max_{s \in V} \mathrm{ecc}(s)$. Moreover, the radius of a (strongly) connected graph is

[1] In general, one might be interested in giving a weight to edges and define the distance based on the sum of the weights of all the edges in a shortest path. For simplicity, this thesis only deals with unweighted graphs, but most of the ideas can be applied to the weighted case, as well.

defined as[2]

$$R = \min_{s \in V} \mathrm{ecc}(s).$$

For example, in Figure 2.1, $\mathrm{ecc}(0) = \mathrm{ecc}(4) = 3$, and $\mathrm{ecc}(1) = \mathrm{ecc}(2) = \mathrm{ecc}(3) = 2$: the radius is 2.

Diameter, radius, and eccentricities are very important measures in graphs, and they are widely used to describe real-world data. For example, a very large amount of research has focused on the analysis of *small-world* networks, by observing that in most real-world networks the diameter is much smaller than one would expect. In many networks with millions of nodes, the diameter can be as small as 10!

Other approaches have considered the average distance instead of the maximum distance. For example, it is possible to define the *average distance* in a graph as

$$\mathrm{dist}_{\mathrm{avg}} = \mathrm{avg}_{s,t \in V, 0 < \mathrm{dist}(s,t) < +\infty} \mathrm{dist}(s,t).$$

Moreover, the *farness* of a node s is

$$f(s) = \mathrm{avg}_{t \in R(s) - \{s\}} \mathrm{dist}(s,t).$$

In a (strongly) connected graph, from the farness, it is possible to define the *closeness centrality* measure as

$$c(s) = \frac{1}{f(s)}.$$

The intuitive idea is to consider a node s central (that is, $c(s)$ is large) if it can reach all the other nodes in a small number of steps. The closeness centrality is a widely used centrality measure, with several applications across different areas of science.

In the literature, in order to generalize the closeness centrality to disconnected graphs, researchers have proposed various approaches. The simplest approach is simply to keep $\frac{1}{f(s)}$, but this would lead to weird behaviors: for example, in Figure 2.2, the closeness of 4 would be undefined, and 3 would be very central, because its closeness is 1, the maximum value achievable by any vertex, in any graph. However, intuitively, 3 should not be central, because it can only reach a single node. For this reason, a different generalization was proposed in the literature [110, 166, 24, 25, 129], named Lin's index (in this work, for simplicity, we refer to Lin's index as closeness centrality):

$$f(s) = \frac{\sum_{t \in R(s)} \mathrm{dist}(s,t)}{r(s) - 1} \cdot \frac{n-1}{r(s)-1} \qquad c(s) = \frac{1}{f(s)}$$

If a node s has (out)degree 0, the previous fraction becomes $\frac{0}{0}$: in this case, the closeness of s is arbitrarily set to 0.

In the example in Figure 2.2, with this definition, $c(4) = 0$, $c(3) = \frac{1}{4}$, $c(1) = \frac{1}{3}$, $c(0) = \frac{4}{7}$, and $c(2) = \frac{2}{3}$, so that the nodes that reach the largest amount of other nodes are more central.

Other similar alternatives are the harmonic centrality, defined by

$$c(s) = \sum_{t \in V} \frac{1}{\mathrm{dist}(s,t)}$$

, and the exponential centrality, defined by

$$c(s) = \sum_{t \in V} C^{\mathrm{dist}(s,t)}$$

[2] Currently, there is no widely accepted definition of radius of a graph which is not connected: for instance, a node with no outgoing edge has eccentricity 0, but it does not make much sense to define the radius of the corresponding graph to be 0. In Chapter 4, we propose a more sensible definition of radius of a directed graph.

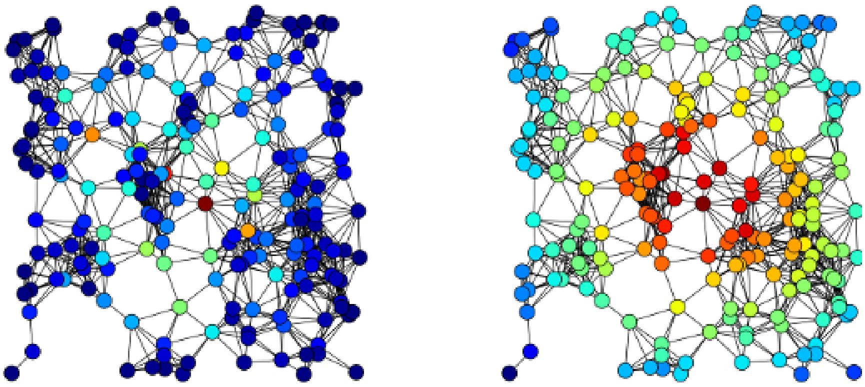


Figure 2.4. A heat map that highlights central vertices according to betweenness and closeness centrality.

for some constant $C < 1$ (the convention is that, if $\mathrm{dist}(s,t) = +\infty$, $\frac{1}{\mathrm{dist}(s,t)} = C^{\mathrm{dist}(s,t)} = 0$).

A different approach in centrality measures considers the number of shortest paths passing through a node, instead of the distance needed to travel from a note to any other node. The resulting measure is called *betweenness centrality*: a probabilistic way to define it is the following. Consider two randomly chosen nodes s, t, and choose a random shortest path between s and t. The betweenness centrality $\mathrm{bc}(v)$ of a node v is the probability that the shortest path chosen passes through v. Note that the generalization to disconnected graphs is very natural: indeed, if s and t are not connected, we simply do not choose any shortest path.

If the closeness centrality selects as central the nodes that are "close to many other nodes", the betweenness centrality select "bridges", from which many shortest paths should pass. For example, if the graph has some communities containing most edges, and few edges across different communities, we expect that the closeness central nodes are in the largest community, while the betweenness central nodes are close to many edges across different communities. A visualization of this feature is provided in Figure 2.4.

Finally, this thesis studies the hyperbolicity of a graph. In the literature, there are evidences that it is possible to embed real-world graphs in hyperbolic spaces, preserving interesting properties: following this evidence, one might be interested in trying to prove that the metric underlying most real-world graphs is hyperbolic, in some sense. The most widespread definition of hyperbolicity of a metric space is provided by Gromov [84] and, intuitively, it measures how far the graph is from a tree (since trees have hyperbolicity 0). Since the definition is quite complicated, we refer to Chapter 7 for more information.

2.3 Problems Studied in This Thesis

The goal of this thesis is to provide fast, practical algorithms that compute metric quantities (in particular, the metric quantities defined in the previous section). First, one might be interested in simply deciding if a graph is connected or computing its connected components: however, we ignore these two problems because they can be solved in subquadratic time in the worst case (actually, in time $\mathcal{O}(m)$).

Then, one might be interested in computing the distance between two nodes: for a single distance, again there are subquadratic algorithms, but if one is interested in computing many distances, this might not be the case. This problem can be formalized as the design of a *distance oracle*: such oracle has a preprocessing phase, where auxiliary data structures are computed, and a query phase, where the oracle tries to answer distance queries as fast as possible. The goal is to obtain a good tradeoff between preprocessing time and query time.

Other problems deal with the computation of the radius, of the diameter, of the eccentricity of all the nodes, and the hyperbolicity of a graph. For centrality measures such as betweenness and closeness centrality, we are interested in computing the k most central nodes, along with their centrality, and in the case of betweenness centrality also in approximating the centrality of single nodes (note that the closeness centrality of a node can be computed in linear time).

More formally, we are interested in studying all the problems in the following list.

Problem: AllEccentricities.
Input: a graph $G = (V, E)$.
Output: the eccentricity of each node $s \in V$, that is, $\max_{t \in V, \mathrm{dist}(s,t) < +\infty} \mathrm{dist}(s, t)$.

Problem: BetweennessCentrality.
Input: a graph $G = (V, E)$.
Output: the betweenness centrality of each node v of G, that is,

$$\frac{1}{n(n-1)} \sum_{v \neq s \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}.$$

where σ_{st} is the number of shortest paths from s to t, and $\sigma_{st}(v)$ is the number of shortest paths from s to t passing through v.

Problem: BetweennessCentralityTopK.
Input: a graph G and an integer $k < n$.
Output: the k nodes with highest closeness centrality in the graph G. The betweenness centrality of a node v is defined as

$$\frac{1}{n(n-1)} \sum_{v \neq s \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}},$$

where σ_{st} is the number of shortest paths from s to t, and $\sigma_{st}(v)$ is the number of shortest paths from s to t passing through v (if $\sigma_{st} = 0$, we just sum 0).

Problem: ClosenessCentralityTopK.
Input: a graph G and an integer $k < n$.
Output: the k nodes with highest closeness centrality in the graph G. The closeness centrality of a node s in a connected graph is defined as

$$c(s) = \frac{n-1}{\sum_{t \in V} \mathrm{dist}(s,t)}.$$

Problem: Diameter.
Input: a graph G.
Output: the diameter of G, that is, $\max_{s,t \in V, \mathrm{dist}(s,t) < +\infty} \mathrm{dist}(s, t)$.

Problem: DistanceOracle.
Input: a graph G.
Output: a distance oracle, that is, an algorithm that performs a preprocessing phase after which it can answer distance queries of the form “compute $\mathrm{dist}(s, t)$”.

Problem: Hyperbolicity.

Input: a graph $G = (V, E)$.
Output: the maximum over each quadruple of nodes s, t, u, v of $\delta(s, t, u, v)$.

Problem: NumReachableNodes.
Input: a directed graph $G = (V, E)$.
Output: for each node $v \in V$, the number $r(v)$ of nodes reachable from v.

Problem: Radius.
Input: a graph $G = (V, E)$.
Output: the radius of G, which, in a (strongly) connected graph, is defined as

$$\max_{s \in V} \min_{t \in V} \operatorname{dist}(s, t).$$

2.4 Complexity of Computing Metric Quantities

First of all, all the problems defined in the previous section can be solved in time and space polynomial in the input size: an approach that almost always works starts by computing the *distance matrix*, which contains $\operatorname{dist}(s, t)$ in position (s, t), and it uses this matrix to compute the value we are interested in. For example, if we want to compute the diameter, we simply output the maximum finite value in this matrix, if we want the average distance we compute the average of the entries, and so on.

The running-time of these algorithms is the time to compute the distance matrix, plus (usually) $\mathcal{O}(n^2)$ to compute the quantity we are interested in. The main bottleneck of this approach is the computation of the distance matrix, which can be done in the following ways.

- We repeatedly multiply the adjacency matrix by itself: this approach takes time $\mathcal{O}(n^\omega \log n)$, where ω is the minimum value such that matrix multiplication can be performed in n^ω. Currently, we know that $2 \leq \omega \leq 2.3728639$ [81]. More refined bounds that work also for small edge weights are proved in [174].
- We perform a breadth-first search (BFS) from each node: this approach takes time $\mathcal{O}(mn)$.

Asymptotically, the first approach is faster if $m > n^{\omega-1}$; however, in practice, the second approach is almost always preferred, because most real-world networks are sparse (that is, m is not much bigger than n), and because the first approach contains large constants hidden in the $\mathcal{O}$ notation. For this reason, in this thesis, we consider the BFS approach as the *textbook* approach, which has $\mathcal{O}(mn)$ running-time.

There are two notable exceptions to this framework, namely betweenness centrality and hyperbolicity: in both cases, it is not easy to compute these quantities starting from the distance matrix. For betweenness centrality, the problem can be solved by adapting the BFS approach: in a seminal paper [39], Brandes shows that the betweenness centrality of all the nodes in a graph can be computed in $\mathcal{O}(mn)$ time (assuming all arithmetic operations can be performed in $\mathcal{O}(1)$ time). For uniformity with the previous cases, we consider this algorithm as the baseline, even if it is much harder and more interesting than the other textbook approaches. The hyperbolicity problem is even harder: the direct approach takes time $\mathcal{O}(n^4)$, and our goal is to improve over this (even if there are faster, but not practical, algorithms based on fast matrix multiplication).

Although polynomial, the $\mathcal{O}(mn)$ running time can be unfeasible on large real-world networks. For example, in standard networks, n might be 1 million and m might be 100 millions: assuming we perform mn operations and our processor performs 1 billion operation

Table 2.1. A summary of the results in this thesis. White pieces indicate results already found before this thesis, while black pieces indicate new results. Kings denote complete results, such as hardness results matching the running time of the best algorithms, practical algorithms that guarantee the correctness of the result, complete probabilistic analyses. Pawns indicate partial results, such as hardness results not matching existing bounds, practical algorithms that do not guarantee the correctness of the results, probabilistic analyses of only some parts of the algorithm.

Problem	Hardness Result	Practical Algorithm	Probabilistic Analysis
AllEccentricities	♔	♟	♟
BetweennessCentrality (approx)		♚	♟
BetweennessCentrality (exact)	♚		
BetweennessCentralityTopK (approx)		♚	♟
BetweennessCentralityTopK (exact)	♚		
ClosenessCentralityTopK	♔ ♟	♚	♚
Diameter	♔	♚	♚
DistanceOracle	♙	♔	♚
Hyperbolicity	♟	♚	
NumReachableNodes	♚		
Radius	♔	♚	♚

Table 2.2. Implementations of the algorithms in this thesis.

Algorithm	Quantity	Library	Class
SumSweep	Diameter, radius, eccentricities (undirected)	WebGraph	algo.SumSweepUndirectedDiameterRadius
SumSweep	Diameter, radius, eccentricities (directed)	WebGraph	algo.SumSweepDirectedDiameterRadius
BCM	Top-k closeness / harmonic / exponential	NetworKit	centrality.ClosenessTopK
BCM (simplified)	Top-k closeness / harmonic / exponential	WebGraph	algo.TopKGeometricCentrality
BCM (simplified)	Top-k closeness	SageMath	graphs.centrality.centrality_closeness_top_k
KADABRA	Betweenness (all/top-k)	-	Available online
KADABRA	Distance between two nodes	-	Available online
KADABRA (in preparation)	Betweenness (all/top-k)	SageMath	
KADABRA (in preparation)	Distance between two nodes	SageMath	
hyp	Hyperbolicity	SageMath	graphs.hyperbolicity
hyp (in preparation)	Hyperbolicity	WebGraph	algo.Hyperbolicity

per second, we would still need more than 1 day to conclude the computation (note that, in reality, the constant in the $\mathcal{O}$ is much larger than 1, making this estimate very optimistic, even considering faster processors and parallelization). For this reason, the main goal of this thesis is to improve over this baseline.

First, since we aim to practical applications, we ignore all logarithmic factors: indeed, in practice, the (constant) overhead due to a more complicated algorithm can be comparable, or even higher than the gain. Our goal becomes to improve over the $\mathcal{O}(mn)$ running-time by a polynomial factor.

Furthermore, since our focus in on real-world graphs, where m is not much bigger than n, we only use m to parametrize our algorithms: for this reason, we say that an algorithm is *subquadratic* if it runs in time $\mathcal{O}(m^{2-\varepsilon})$ (we recall that m is at least $\frac{n}{2}$, because we removed all isolated nodes).[3] Our goal is to find subquadratic algorithms for all the aforementioned problems.

In the first part of the thesis (Chapter 3), we show that, in the worst-case, it is probably impossible to achieve such goal. More formally, we restrict our attention to sparse graphs, where $m = \mathcal{O}(n)$, and we prove that the existence of an algorithm that runs in $\mathcal{O}(n^{2-\varepsilon})$ on such graphs would falsify widely believed assumptions, such as the Strong Exponential Time Hypothesis, the Orthogonal Vector conjecture, and the Hitting Set conjecture. Note that this statement is *stronger* than simply saying that subquadratic algorithms cannot exist, or that the $\mathcal{O}(mn)$ running-time cannot be improved: indeed, any algorithm running in $\mathcal{O}(m^{1-\varepsilon}n)$, $\mathcal{O}(mn^{1-\varepsilon})$, $\mathcal{O}(m^{2-\varepsilon})$ on general graphs would be subquadratic when restricted to sparse graphs. This means that, in the worst-case, there is very little room to improve over existing algorithms.

In the second part of the thesis (Chapters 4 to 7), we provide new algorithms that outperform the textbook algorithm on real-world networks, and we define the *improvement factor* as the ratio between mn, which is the number of edges visited by the textbook algorithm, and the number of edges actually visited by our newly designed algorithm. The improvement factors obtained with our new algorithms are usually surprisingly large, achieving speed-ups of several orders of magnitude.

In the last part of the thesis (Chapter 8), we develop a theoretical framework in which these algorithms can be evaluated and compared: we prove that they are efficient in the worst under certain conditions, and we show that some widely used graph models satisfy these conditions. All the models we consider in this thesis generate sparse graphs, or graphs where $m = \mathcal{O}(n \log n)$: for this reason, in that chapter we can ignore the issue of m vs n, because all running time bounds have the form $\mathcal{O}(n^{c-o(1)})$ for some c: this way, we can simply switch between m and n by incorporating the logarithmic factor in the $o(1)$ part.

A summary of all the results available in this thesis is available in Table 2.1. Furthermore, in Table 2.2, we provide a list of all the publicly available implementations of our algorithms.

[3] Classically, the term *subquadratic* denotes $o(m^2)$, and *truly subquadratic* denotes $\mathcal{O}(m^{2-\varepsilon})$. However, in the field of polynomial reductions, our definition is becoming more and more popular, because it is more natural [3].

Chapter 3

Lower Bounds in the Worst-Case

Abstract

For all the problems defined in the previous chapter, it is unlikely that subquadratic algorithms exist: indeed, such algorithms would falsify widely believed conjectures, such as the Strong Exponential Time Hypothesis, the Orthogonal Vector conjecture, and the Hitting Set conjecture.

In this chapter, we survey existing hardness results for these problems, and we collect them in a web of reduction. Then, we prove some new reductions, dealing with the computation of the number of reachable nodes, of closeness centrality, of betweenness centrality, of Lin's index, and of the subset graph of a given collection of sets.

As we already said in the first two chapter, despite a long line of research, for most of the problems considered in this thesis the trivial approaches are still optimal, or almost optimal in the worst case. In other words, there is no *subquadratic* algorithm which is able to solve them.

This lack of improvement is probably due to intrinsic theoretic bottlenecks: indeed, it was proved that a faster solution for any of these problems would imply unexpected breakthrough for other important problems. In the literature, the main tool used to prove this kind of results is a particular kind of Karp reduction. Usually, a Karp reduction from a problem Π to a problem Π' is a function Φ that maps instances of Π to instances of Π', such that the result is preserved, and the size of $\Phi(I)$ is polynomial in the size of I, for each instance I of Π. In our framework, since all our problems are polynomial-time solvable, we need a more refined definition: instead of asking that the size of $\Phi(I)$ is polynomial in the size of I, we ask that the size of $\Phi(I)$ satisfies $|\Phi(I)| \leq |I|p(\log(|I|))$ for some polynomial p. This way, if we have a subquadratic algorithm that solves Π' and a Karp reduction from Π to Π', we can easily design a subquadratic algorithm for Π.

However, this tool is only able to prove that a problem is harder than another, and it does not say anything on the absolute complexity of a problem. For this reason, we need a "starting point", that is, a problem that is assumed to be hard. Unfortunately, in this context, the standard approach of using complete problems does not work, because there is no notion of completeness linked to the class of quadratic-time solvable problem. For this reason, researchers have simply started from specific problems, that have been widely studied, relying on the assumption that if an algorithm existed, then it would have been found before.

In the literature, the first such problem considered is the 3Sum problem [80, 99, 91], from which it is possible to prove the hardness of several problems in computational geometry, and some problems dealing with graphs [2]. In the context of dense, weighted graphs, other results are based on the hardness of computing the All Pairs Shortest Paths, assuming that this task cannot be performed in time $\mathcal{O}(n^{3-\varepsilon})$ for any $\varepsilon > 0$ [171, 1].

However, as discussed in Chapter 2, in this chapter we focus on sparse graphs, where $m = \mathcal{O}(n)$: indeed, if we prove that a problem is hard on sparse graphs, we obtain "for free" that there is no algorithm running in $\mathcal{O}(m^{2-\varepsilon})$, $\mathcal{O}(mn^{1-\varepsilon})$, $\mathcal{O}(m^{1-\varepsilon}n)$ on general graphs.

In this context, the most used assumption is the Strong Exponential Time Hypothesis (SETH), that says that there is no algorithm that solves the k-Sat problem in time $\mathcal{O}((2-\varepsilon)^n)$, where $\varepsilon > 0$ does not depend on k [92]. The first SETH-based reduction was proved in [168]: the authors show that it is impossible to find two disjoint sets in a given collection of sets, unless SETH is false (we name this problem TwoDisjointSets). Starting from this result, dozens of reductions were proved in the literature, for several different problems [131, 138, 47, 44, 4, 2].

A further step was performed in [169, 31, 3]: almost all the reductions proved until now can be rephrased as reductions that start from the TwoDisjointSets problem. Hence, we can start from a stronger conjecture, assuming that the TwoDisjointSets is not solvable in subquadratic time. This conjecture was named Orthogonal Vector conjecture, because it can be restated in terms of finding two orthogonal vectors in a collection [3].

Finally, in [31, 3], the authors observe that most of the problems that are proved to be hard assuming the Orthogonal Vector conjecture have the form $\exists x, \exists y, \varphi(x,y)$ for some formula φ: for example, the TwoDisjointSets problem asks whether there exist two sets x, y which are disjoint (note that, by the contrapositive, we can restate these problem using $\forall$ instead of $\exists$). However, almost no hardness result was proved for problems in the form $\exists x, \forall y$, even if several problems can be easily stated in this form, such as computing the radius of a graph, that is, $\min_{s \in V} \max_{t \in V} \text{dist}(s,t)$ (note that the radius is smaller than r if $\exists s \in V, \forall t \in V, \text{dist}(s,t) \leq r$).

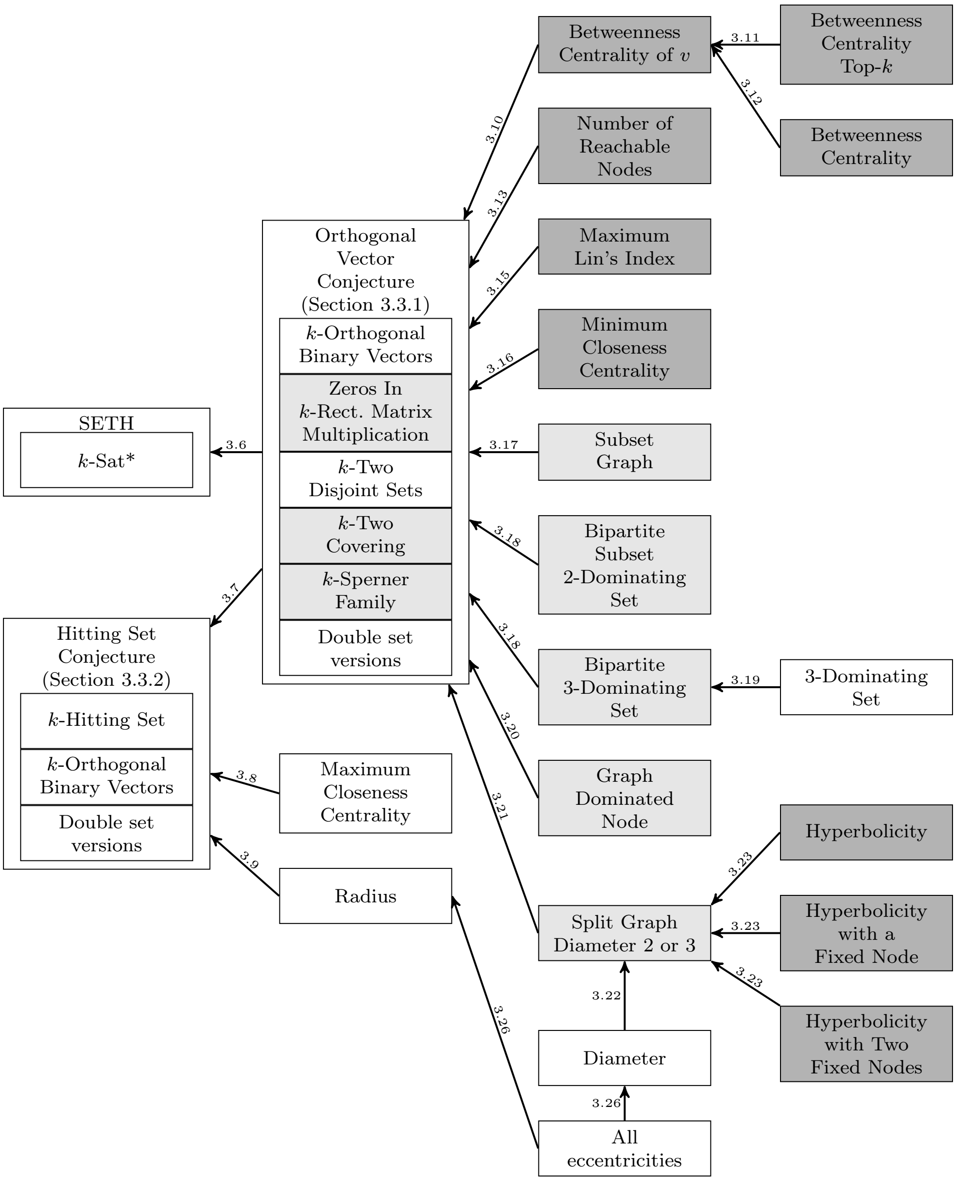
For this reason, in [3], the authors state another conjecture, named Hitting Set conjecture, which is a "dual" of the Orthogonal Vector conjecture, because it is based on the hardness of the HittingSet problem, that is, given a collection $\mathcal{C}$ of subsets of a given ground set X, finding whether there is a set $C \in \mathcal{C}$ that has nonempty intersection to all other sets in $\mathcal{C}$ (hence, we start from a problem of the form $\exists C, \forall C', C \cap C' \neq \emptyset$).[1] Unfortunately, there is no evidence of the hardness of the latter problem, apart from the fact that no subquadratic algorithm is known, and the similarity to the Orthogonal Vector conjecture. However, it is possible to prove that the Orthogonal Vector conjecture is implied by the Hitting Set conjecture, and the HittingSet problem can be reduced to some problems that have the form $\exists\forall$, such as computing the radius of a graph or the maximum closeness centrality.

3.1 Our Contribution

In this chapter, we collect the most important reductions proved until now, with a specific focus on the problems defined in Chapter 2, we put them into a single web of reductions, and, most importantly, we use this web in order to prove some *new* reductions (see Figure 3.1). The new reductions deal with the following problems.

BetweennessCentrality: despite numerous works, such as [39, 13], no truly subquadratic algorithm computing the betweenness centrality is known, even of a single node (BetweennessCentralityNode), or for finding the most central node (BetweennessCentralityTopK with $k = 1$). Moreover, in [13], it is said that finding better results for approximating the betweenness centrality of all nodes is a "challenging open problem". The only hardness result was proved in the weighted case [1]: computing the betweenness centrality is at least as hard as computing the All Pairs Shortest Paths, and providing a relative-error approximation is at least as hard as computing the diameter. Conversely, our result deals with the sparse, unweighted case, and it does not only show that computing the betweenness centrality of all nodes in

[1] In the original version of the Hitting Set conjecture [3], the authors use an equivalent form where C and C' are taken from two different collections of sets.



Betweenness Centrality of v
Betweenness Centrality Top-k
Betweenness Centrality
Number of Reachable Nodes
Maximum Lin's Index
Minimum Closeness Centrality
Subset Graph
Bipartite Subset 2-Dominating Set
Bipartite 3-Dominating Set
3-Dominating Set
Graph Dominated Node
Hyperbolicity
Hyperbolicity with a Fixed Node
Hyperbolicity with Two Fixed Nodes
Split Graph Diameter 2 or 3
Diameter
All eccentricities
Orthogonal Vector Conjecture (Section 3.3.1)
k-Orthogonal Binary Vectors
Zeros In k-Rect. Matrix Multiplication
k-Two Disjoint Sets
k-Two Covering
k-Sperner Family
Double set versions
SETH
k-Sat*
Hitting Set Conjecture (Section 3.3.2)
k-Hitting Set
k-Orthogonal Binary Vectors
Double set versions
Maximum Closeness Centrality
Radius
3.6
3.7
3.8
3.9
3.10
3.11
3.12
3.13
3.15
3.16
3.17
3.18
3.18
3.19
3.20
3.21
3.22
3.23
3.23
3.23
3.26
3.26

Figure 3.1. The web of reductions proved in this chapter. Gray problems indicate original results, white problems indicate known results, and light-gray problems indicate intermediate steps, which nevertheless can be interesting in their own rights, such as the DiameterSplitGraph2Or3 problem. By "double set versions", we mean equivalent versions of the problems considered, where instead of having one collection $\mathcal{C}$ of sets, we have two collections $\mathcal{C}_1, \mathcal{C}_2$, and we ask whether there are sets $C_1 \in \mathcal{C}_1$, $C_2 \in \mathcal{C}_2$ with specific properties. The boxes corresponding to the conjectures collect all the problems whose hardness is equivalent to the conjecture itself.

subquadratic time is against SETH or the Orthogonal Vector conjecture, but it also presents the same result for computing the betweenness of a single node. We can also suppose without loss of generality that this node is the most central node in the graph.

ClosenessCentralityMinimum: another fundamental quantity in graph analysis is closeness centrality, defined for the first time in 1950 [17] and recently reconsidered when analyzing real-world networks (for the interested reader, we refer to [105] and the references therein). This parameter has also raised algorithmic interest, and the most recent result is a very fast algorithm to approximate the closeness centrality of all nodes [55].

There are also hardness results: computing the most central node in subquadratic time falsifies the Hitting Set conjecture. In this chapter, use the Orthogonal Vector conjecture to prove the hardness of finding the “least central” node with respect to this measure. Simple consequences of this results are the hardness of computing the closeness centrality of all nodes, or of extracting a “small enough” set containing all peripheral nodes.

LinIndexMaximum: efficient algorithms for computing Lin’s index were developed in the literature [129, 20], but none of them is subquadratic in the worst-case. Here, we show that it is hard to compute the most central node according to Lin’s index, assuming the Orthogonal Vector conjecture. A similar result was deduced in [3], because that paper proves that computing the node with maximum closeness centrality in subquadratic time refutes the Hitting Set conjecture (and in connected graphs Lin’s index coincides with closeness centrality). However, our reduction starts from the stronger Orthogonal Vector conjecture, which is much more established than the Hitting Set conjecture.

NumReachableNodes: a very basic quantity in the analysis of a graph is the number of nodes reachable from a given node s. In undirected graphs, this quantity can be computed in linear time, and one might be tempted to assert the same result for directed graphs, since strongly connected components are computable in linear time. A subquadratic algorithm to perform this task could have many applications: to name a few, there are algorithms that need to compute (or estimate) these values [32], the number of reachable nodes is used in the definition of other measures, such as Lin’s index [110, 166, 129], and it can be useful in the analysis of the transitive closure of a graph (indeed, the out-degree of a node v in the transitive closure is the number of nodes reachable from v). Here, we prove that this quantity is hard to compute: indeed, an algorithm that computes the number of reachable nodes in subquadratic time would falsify the Orthogonal Vector conjecture, and the same result holds if we restrict our attention to acyclic graphs with small diameter.

Hyperbolicity: the Gromov hyperbolicity of a graph [84] recently got the attention of researchers in the field of network analysis [50, 75, 172, 58] (see Chapter 7. The trivial algorithm to compute the hyperbolicity needs time $\mathcal{O}(n^4)$; however, this trivial approach can be improved, as shown in [77]. In particular, the authors describe an algorithm with time complexity $\mathcal{O}(n^{3.69})$, which is a consequence of an algorithm that computes in time $\mathcal{O}(n^{2.69})$ the maximum hyperbolicity of a quadruple where one of the nodes is fixed. The latter work also provides a hardness result for this problem, namely, that an algorithm running in time smaller than $\mathcal{O}(n^{3.05})$ would imply faster algorithms for (max,min) matrix product. Here, we use another approach: we prove that recognizing graphs of hyperbolicity at most 1 is not solvable in subquadratic time, unless the Orthogonal Vector conjecture is false. Furthermore, the same result holds if we fix one node s, and even if we fix two nodes s, t in the quadruple (note that the latter problem is quadratic-time solvable). Although this bound is quite far from the $\mathcal{O}(n^{3.69})$ upper bound, it might be useful in the design of new algorithms, especially in the case where there are fixed nodes.

SubsetGraph: given a collection $\mathcal{C}$ of subsets of a given ground set X, this problem asks to compute the subset graph of $\mathcal{C}$, which is defined as a graph whose nodes are the elements of $\mathcal{C}$, and which contains an edge (C, C') if $C \subseteq C'$. For this problem, there are algorithms running in $o(n^2)$ [173], but only by small logarithmic factors. In [135, 71], the authors prove matching lower bounds, based on the number of edges in the subset graph, which might be quadratic with respect to the input size. Our results show that the complexity of computing the subset graph is not due to the output size only, but it is intrinsic: in particular, we prove that even deciding whether the subset graph has no edge is hard. This excludes the existence of a subquadratic algorithm to check if a solution is correct, or a subquadratic algorithm for instances where the output is sparse.

Moreover, we include in our web of reductions several "intermediate" problems. Among them, we have some variations of finding dominating sets in a graph (which is one of the 21 Karp's NP-complete problems [97]), detecting subsets with particular characteristics in a given collection, and distinguishing split graphs of diameter 2 and 3. The latter result uses a different reduction from the one in [138, 47], and it is significant because it implies the hardness of computing the diameter of graphs in a class that contains split graphs: for instance, chordal graphs, where it is possible to approximate the diameter with an additive error of at most 1 [50]. The hardness proof of these results are already known, either as "hidden" steps in proofs already appeared in the literature, or as simple corollaries of previous work. However, we think that it is important to highlight these intermediate results, both because they might be interesting in their own right, and because they might be useful in the simplification of existing proofs and in the design of new proofs.

In Section 3.2, we define all the problems considered in this chapter, while in Section 3.3 we precisely define SETH, the Orthogonal Vector conjecture, and the Hitting Set conjecture, we prove that several definitions are equivalent, and we show that the Orthogonal Vector conjecture is implied by the other two conjectures (no other implication among these conjectures is known). Then, in Section 3.4 we prove all the reductions in Figure 3.1.

3.2 Problem Definitions

Although some of the problems analyzed were already defined in Chapter 2, here we define many other problems, used as "intermediate steps" in the reductions.

Problem: Bipartite3DominatingSet.
Input: a bipartite graph $G = (V, E)$.
Output: a triple u, v, w such that $V = N(u) \cup N(v) \cup N(w) \cup \{u, v, w\}$, if it exists (we denote by $N(u)$ the set of neighborx of u).

Problem: BipartiteSubset2DominatingSet.
Input: a graph $G = (V, E)$ and a subset $V' \subseteq V$.
Output: a pair v, w such that $V' = N(v) \cup N(w) \cup \{v, w\}$, if it exists (we denote by $N(u)$ the set of neighborx of u).

Problem: ClosenessCentralityMaximum.
Input: a graph $G = (V, E)$ and a threshold σ.
Output: **True** if there exists a node with closeness centrality larger than σ, **False** otherwise. The closeness centrality of a node s is defined as

$$c(s) = \frac{n-1}{\sum_{t \in V} \text{dist}(s,t)}.$$

Problem: DiameterSplitGraph2Or3.
Input: a split graph G.
Output: **True** if G has diameter 2, **False** otherwise.

Problem: DominatedNode.
Input: a graph (V, E).

Output: **True** if there are nodes v, w such that $N(v) \supseteq N(w)$, **False** otherwise.

Problem: DominatingSet3.
Input: a graph $G = (V, E)$.
Output: a triple v, w, x such that $V = N(v) \cup N(w) \cup N(x) \cup \{v, w, x\}$, if it exists.

Problem: HyperWith1FixedNode.
Input: a graph $G = (V, E)$ and a node s.
Output: the maximum over each triple of nodes t, u, v of $\delta(s, t, u, v) = S_1 - S_2$, where S_1 is the maximum sum among $\mathrm{dist}(s, t) + \mathrm{dist}(u, v)$, $\mathrm{dist}(s, u) + \mathrm{dist}(t, v)$, $\mathrm{dist}(s, v) + \mathrm{dist}(t, u)$, and S_2 the second maximum sum.

Problem: HyperWith2FixedNodes.
Input: a graph $G = (V, E)$ and two nodes s, t.
Output: the maximum over each pair of nodes u, v of $\delta(s, t, u, v) = S_1 - S_2$, where S_1 is the maximum sum among $\mathrm{dist}(s, t) + \mathrm{dist}(u, v)$, $\mathrm{dist}(s, u) + \mathrm{dist}(t, v)$, $\mathrm{dist}(s, v) + \mathrm{dist}(t, u)$, and S_2 the second maximum sum.

Problem: k-HittingSet.
Input: a set X and a collection $\mathcal{C}$ of subsets of X such that $|X| < k \log(|\mathcal{C}|)$.
Output: **True** if there is a hitting set C, that is, $C \cap C' \neq \emptyset$ for each $C' \in \mathcal{C}$, **False** otherwise.

Problem: k-OrthogonalBinaryVectors.
Input: a collection $\mathcal{C}$ of binary vectors on a space of size at most $k \log(|\mathcal{C}|)$.
Output: **True**, if there are two orthogonal vectors, **False** otherwise.

Problem: k-OrthogonalToAllVectors.
Input: a collection $\mathcal{C}$ of binary vectors on a space of size at most $k \log(|\mathcal{C}|)$.
Output: **True**, if there is a vector which is not orthogonal to any other vector, **False** otherwise.

Problem: k-Sat*.
Input: two sets of variables $\{x_i\}$, $\{y_j\}$ of the same size, a set C of clauses over these variables, such that each clause has at most size k, the set of possible evaluations of $\{x_i\}$ and the set of possible evaluations of $\{y_j\}$.
Output: **True** if there is an evaluation of all variables that satisfies all clauses, **False** otherwise.

Problem: k-SpernerFamily.
Input: a set X and a collection $\mathcal{C}$ of subsets of X such that $|X| < k \log(|\mathcal{C}|)$.
Output: **True** if there are two sets $C, C' \in \mathcal{C}$ such that $C \subseteq C'$, **False** otherwise.

Problem: k-TwoCovering.

Input: a set X and a collection $\mathcal{C}$ of subsets of X such that $|X| < k\log(|\mathcal{C}|)$.
Output: **True** if there are two sets $C, C' \in \mathcal{C}$ such that $X = C \cup C'$, **False** otherwise.

Problem: k-TwoDisjointSets.
Input: a set X and a collection $\mathcal{C}$ of subsets of X such that $|X| < k\log(|\mathcal{C}|)$.
Output: **True** if there are two disjoint sets $C, C' \in \mathcal{C}$, **False** otherwise.

Problem: k-ZerosMatrixMultiplication.
Input: a $(0-1)$-matrix M of size $(k\log n) \times n$.
Output: **True** if M^TM contains a 0, **False** otherwise.

Problem: LinIndexMaximum.
Input: a graph $G = (V, E)$ and a threshold σ.
Output: **True** if there exists a node with Lin's index bigger than σ, **False** otherwise. The Lin's index of a node s is defined as

$$\frac{(r(v)-1)^2}{(n-1)\sum_{t\in R(v)} \mathrm{dist}(s,t)},$$

where $R(v)$ is the set of nodes reachable from v, $r(v) = |R(v)|$.

Problem: SubsetGraph.
Input: a set X and a collection $\mathcal{C}$ of subsets of X.
Output: the graph $G = (\mathcal{C}, E)$, where, for each $C, C' \in \mathcal{C}$, $(C, C') \in E$ if and only if $C \subseteq C'$.

3.3 The Orthogonal Vector Conjecture and the Hitting Set Conjecture

In this section, we prove the equivalence of the different versions of the Orthogonal Vector conjecture and the Hitting Set conjecture, and we show the implications between these two, and SETH. We decided to distinguish theorems, that correspond to technical reductions, and remarks, that correspond to simple reductions, or sometimes to restatement of the original problem.

3.3.1 The Orthogonal Vector Conjecture

Let us first define the Orthogonal Vector conjecture as follows: there is no $\varepsilon > 0$ such that, for all $k \geq 1$, there is an algorithm that can determine in $\mathcal{O}(n^{2-\varepsilon})$ time if there is an orthogonal pair in a list of n vectors $\mathcal{C} \subseteq \{0,1\}^{k\log n}$.[2] In other words, we are saying that the k-OrthogonalBinaryVectors problem cannot be solved in subquadratic time.

First, we can restate this problem in terms of the existence of zeros in M^TM, where M is a rectangular matrix $k\log n \times n$ whose lines are the vectors in $\mathcal{C}$.

Remark 3.1. If there is a subquadratic algorithm for k-OrthogonalBinaryVectors, then there is a subquadratic algorithm for k-ZerosMatrixMultiplication, and viceversa.

[2] All the proofs would still work if we redefine the k-OrthogonalBinaryVectors problem by setting the dimension of the ground space as $(\log n)^k$ instead of $k\log n$. In this case, also the proof that SETH implies the Orthogonal Vector conjecture would be simpler. However, we chose to use $k\log n$ because it is more established in the literature (see for instance [3]).

Proof. Each instance of the k-OrthogonalBinaryVectors problem can be associated to an instance of the k-ZerosMatrixMultiplication problem, by mapping the vectors of the k-OrthogonalBinaryVectors problem to the columns of the input matrix M of the k-ZerosMatrixMultiplication problem. Clearly, there are two orthogonal vectors if and only if M has two orthogonal columns, if and only if the matrix $M^T M$ contains a zero. □

Moreover, the k-OrthogonalBinaryVectors problem can be easily linked to the hardness of various set problems: the simplest one is the k-TwoDisjointSets problem (also named CooperativeSubsetQuery in [168]), which asks to find two disjoint sets in a collection $\mathcal{C}$ of subsets of a given ground set X, assuming $|X| < k\log(|\mathcal{C}|)$.

Remark 3.2 ([168]). If there is a subquadratic algorithm for k-TwoDisjointSets, then there is a subquadratic algorithm for k-OrthogonalBinaryVectors, and viceversa.

Sketch of proof. Each instance of the k-OrthogonalBinaryVectors problem can be associated to an instance of the k-TwoDisjointSets problem, and viceversa: the ground set corresponds to the set of dimensions of the space, and a value 1 in a vector corresponds to an element in the set. In this construction, two disjoint sets correspond to two orthogonal vectors. □

An equivalent formulation can be obtained by taking the complement of the two sets in the k-TwoDisjointSets problem: we obtain the k-TwoCovering problem. This problem has as input a collection $\mathcal{C}$ of subsets of a ground set X, and it asks whether there are two sets $C_1, C_2 \in C$ such that $C_1 \cup C_2 = X$.

Remark 3.3. If there is a subquadratic algorithm for k-TwoDisjointSets, then there is a subquadratic algorithm for k-TwoCovering, and viceversa.

Sketch of proof. We use the fact that two sets $C_1, C_2 \in \mathcal{C}$ are disjoint if and only if $X - C_1, X - C_2$ cover X (indeed, $C_1 \cap C_2 = \emptyset$ if and only if $X - (C_1 \cap C_2) = X$, if and only if $(X - C_1) \cup (X - C_2) = X$). Following this idea, given an instance of the k-TwoDisjointSets problem, we obtain a corresponding instance of the k-TwoCovering problem by complementing all sets. Note that this operation cannot increase the input size by more than a logarithmic factor, because the size of X is at most $k\log(|\mathcal{C}|)$. □

A more complicated result links the complexity of the k-TwoDisjointSets to the complexity of the k-SpernerFamily, which, given a collection $\mathcal{C}$ of sets, asks whether there are $C_1, C_2 \in \mathcal{C}$ such that $C_1 \subseteq C_2$.

Theorem 3.4. *If there is a subquadratic algorithm for k-*TwoDisjointSets*, then there is a subquadratic algorithm for k-*SpernerFamily*, and viceversa.*

*Reduction from k-*TwoDisjointSets *to k-*SpernerFamily. Consider an instance $(X, \mathcal{C})$ of k-TwoDisjointSets. To start, let us define the reduction as $(X, \mathcal{C}')$, where $\mathcal{C}' = \mathcal{C} \cup \bar{\mathcal{C}}$ (we denote $\bar{\mathcal{C}} := \{X - C : C \in \mathcal{C}\}$). This is not the correct reduction, but we see later how to adapt it. If we find two sets $C_1 \in \mathcal{C}, C_2 \in \bar{\mathcal{C}}$ such that $C_1 \subseteq C_2$, we know that C_1 and $X - C_2$ are in $\mathcal{C}$ and they are disjoint, so we have found a solution. However, when we solve the k-SpernerFamily problem, we are not sure that we obtain two sets $C_1 \in \mathcal{C}$ and $C_2 \in \bar{\mathcal{C}}$. We solve this issue by slightly modifying the set X and $\mathcal{C}'$, to rule out all other possibilities. In order to avoid the existence of $C_1, C_2 \in \mathcal{C}$ such that $C_1 \subseteq C_2$, we define $h := \lceil \log_2(|\mathcal{C}|) \rceil$, and we add two sets $Y = \{y_1, \dots, y_h\}$ and $Z = \{z_1, \dots, z_h\}$ to X. Then, we add Y and Z to each set in $\bar{\mathcal{C}}$ and we add to each element $C \in \mathcal{C}$ some y_i and some z_j, so that no element of $\mathcal{C}$ can dominate another element in $\mathcal{C}$ (for example, we may associate each set C with a unique binary number with h bits, and code this number using y_i as zeros and z_j as ones). This way, we preserve dominations between sets in $\bar{\mathcal{C}}$ and sets in $\mathcal{C}$, and we also avoid that a set in $\mathcal{C}$ dominates another set. Finally, we need to avoid that two sets in $\bar{\mathcal{C}}$ dominate each other: it is enough to make the same construction adding new sets Y' and Z' of logarithmic size, and use them to uniquely code any element in $\bar{\mathcal{C}}$. In order to preserve dominations between $\mathcal{C}$ and $\bar{\mathcal{C}}$, none of the elements in Y' and Z' is added to subsets in $\mathcal{C}$. □

Reduction from k-SpernerFamily to k-TwoDisjointSets. Let us consider an instance $(X, \mathcal{C})$ of k-SpernerFamily and let us map it to an instance of k-TwoDisjointSets $(X \cup \{x_1, x_2\}, \mathcal{C}')$, where $\mathcal{C}' := \mathcal{C}_1 \cup \mathcal{C}_2$, $\mathcal{C}_1 := \{C \cup x_1 : C \in \mathcal{C}\}$ and $\mathcal{C}_2 := \{(X - C) \cup x_2 : C \in \mathcal{C}\}$.

If C_1 and C_2 are disjoint sets in $\mathcal{C}'$, one of them must be in $\mathcal{C}_1$ and the other in $\mathcal{C}_2$ (because of x_1 and x_2). Hence, there are two disjoint sets in $\mathcal{C}'$ if and only if there is a set $C_1 \in \mathcal{C}_1$, $C_2 \in \mathcal{C}_2$ such that $C_1 \cap C_2 = \emptyset$, if and only if there are two sets $C_1', C_2' \in \mathcal{C}$ such that $C_1' \cap (X - C_2') = \emptyset$, if and only if $C_1' \subseteq C_2'$. We have proved that two disjoint sets in $\mathcal{C}'$ correspond to two sets in $\mathcal{C}$ such that one set dominates the other. □

Finally, we observe that all the aforementioned set problems deal with a collection $\mathcal{C}$ of sets. However, in some cases, we might want to work with two collections $\mathcal{C}_1$ and $\mathcal{C}_2$: for instance, in the k-TwoDisjointSets problem, we might want to check if there are two disjoint sets $C_1 \in \mathcal{C}_1, C_2 \in \mathcal{C}_2$. It turns out that the two versions of the problem are equivalent: indeed, the version with a single collection can be easily reduced to the version with two collections by choosing $\mathcal{C}_1 = \mathcal{C}_2$. The converse is a bit harder: in the k-TwoDisjointSets problem it is enough to add a new element x_1 to each set in $\mathcal{C}_1$, and a new element x_2 to each set in $\mathcal{C}_2$, and define $\mathcal{C}$ as the union of these two collections. Then, the proofs of equivalence used in this section can be modified in order to deal with the case with two collections $\mathcal{C}_1$ and $\mathcal{C}_2$.

3.3.2 The Hitting Set Conjecture

Let us start by defining the Hitting Set conjecture as follows: there is no $\varepsilon > 0$ such that, for all $k \geq 1$, there is an algorithm that can determine in $\mathcal{O}(n^{2-\varepsilon})$ time if there is a hitting set in a collection of n subsets of a given ground set X, where $|X| \leq k \log n$.[3] In other words, we are saying that the k-HittingSet problem cannot be solved in subquadratic time (differently from the Orthogonal Vector conjecture, for historical reasons, we start from the version dealing with sets).

As we did with the Orthogonal Vector conjecture, we can prove that this conjecture is equivalent to the hardness of several problems, that can be obtained from the k-HittingSet by taking complements. However, since all the reductions we prove deal with the k-HittingSet, and since the problems obtained are less natural than their counterparts, we only consider two variation, and we refer the interested reader to [3] for more details.

In particular, let us restate this conjecture in terms of orthogonal vectors.

Remark 3.5 ([168]). If there is a subquadratic algorithm for k-HittingSet, then there is a subquadratic algorithm for k-OrthogonalToAllVectors, and viceversa.

Sketch of proof. Each instance of the k-OrthogonalToAllVectors problem can be associated to an instance of the k-HittingSet problem, and viceversa: the ground set corresponds to the set of dimensions of the space, and a value 1 in a vector corresponds to an element in the set. In this construction, two disjoint sets correspond to two orthogonal vectors, and consequently a hitting set corresponds to a vector that is not orthogonal to any other vector in the collection. □

Finally, also in this case we can consider versions with two collections of sets $\mathcal{C}_1, \mathcal{C}_2$: in this version, we are looking for a set $C_1 \in \mathcal{C}_1$ that hits all sets $C_2 \in \mathcal{C}_2$. Again, proving the equivalence of this version with the version with just one collection is quite simple: clearly, the version with one collection is easier, because we can always choose $\mathcal{C}_1 = \mathcal{C}_2$. To reduce the version with two collections, we add a new element x to the set X, and we define

$$\mathcal{C} = \{\{x\}\} \cup \{C \cup \{x\} : C \in \mathcal{C}_1\} \cup \mathcal{C}_2.$$

[3] Also in this case, all the proofs would still work if we redefine the k-HittingSet problem by setting the dimension of X to $(\log n)^k$ instead of $k \log n$.

Then, if there is a hitting set in $\mathcal{C}$, then it must be of the form $C \cup \{x\}$ with $C \in \mathcal{C}_1$, because no set in $\mathcal{C}_2$ hits the set $\{x\}$, and $\{x\}$ does not hit any set in $\mathcal{C}_2$. Hence, there is a hitting set in $\mathcal{C}$ if and only if there is a set of the form $C \cup \{x\}$ with $C \in \mathcal{C}_1$ that hits all sets in $\mathcal{C}$, if and only if C hits all sets in $\mathcal{C}_2$. This concludes the reduction between the two versions on the k-HittingSet.

3.3.3 Implications Between Conjectures

In this section, we show that the Orthogonal Vector conjecture is implied both by SETH and by the Hitting Set Conjecture. Currently, no other implication between these conjectures is known.

Theorem 3.6 ([168]). *The Strong Exponential Time Hypothesis implies the Orthogonal Vector conjecture.*

Proof. Let k-Sat* be the problem defined as follows: we are given a propositional formula in conjunctive normal form on n variables, such that all clauses have length at most k, and let us assume that we also input all possible evaluations of the variables in V_1 and in V_2, where V_1 and V_2 are two sets of $\frac{n}{2}$ variables (the latter part of the input is used to increase the input size, so that we are able to solve k-Sat* in quadratic time by checking all possible evaluations of the variables). Clearly, if we can solve k-Sat* in subquadratic time, then we can solve k-Sat in time

$$\mathcal{O}\left(\left(2^{\frac{n}{2}}\right)^{2-\varepsilon}\right) = \mathcal{O}\left(\left(2^{\frac{2-\varepsilon}{2}}\right)^{n}\right) = \mathcal{O}\left((2-\delta)^n\right),$$

against SETH. So, we only need a reduction from the k-Sat* problem to a problem whose hardness is equivalent to the Orthogonal Vector conjecture, namely the k-TwoCovering problem, in the variation with two collections of sets. Our ground set is the set of clauses: the set $\mathcal{C}_1$ (resp., $\mathcal{C}_2$) contains all the sets of clauses that are satisfied by a given evaluation of the variables in V_1 (resp., V_2).

More formally, $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$, $\mathcal{C}_1 := \{\{c \in C : v \models c\} : v \in V_1\}$ and similarly $\mathcal{C}_2 := \{\{c \in C : v \models c\} : v \in V_2\}$ (note that $v \models c$ means that the evaluation v is sufficient to make the clause c true, independently on the evaluation of the remaining variables). It is easy to prove that this construction can be performed in time $2^{\frac{n}{2}}p(n)$ for some polynomial $p(n)$, and the number of sets is at most $2^{\frac{n}{2}}$. Furthermore, each assignment corresponds naturally to two partial assignments in $\mathcal{C}_1$ and in $\mathcal{C}_2$, and the assignment satisfies the formula if and only if all clauses are satisfied, if and only if the two corresponding sets $C_1 \in \mathcal{C}_1, C_2 \in \mathcal{C}_2$ cover the set of clauses. Furthermore, since we want a covering to correspond to a set in $\mathcal{C}_1$ and a set in $\mathcal{C}_2$, we add two more elements x_1, x_2, we add x_1 to all sets in $\mathcal{C}_1$ and x_2 to all sets in $\mathcal{C}_2$.

The only problem in this construction is that the ground set has size at most $2n^k = \log_2^k(|\mathcal{C}|)$, which in general can be bigger than $\log(|\mathcal{C}|)$. To avoid this issue, a possibility would be to restate all our problems considering a ground set of size $\log^k |\mathcal{C}|$ instead of $k\log(|\mathcal{C}|)$. However, following most of the literature, we use Corollary 1 in [92], which says that, for all $\delta > 0$ and positive k, there is a constant C so that any k-Sat formula Φ with n variables can be expressed as $\Phi = \vee_{i=1}^{t}\Psi_i$, where $t \leq 2^{\delta n}$ and each Ψ_i is a k-Sat formula with at most Cn clauses. Moreover, this disjunction can be computed by an algorithm running in time $p(n)2^{\delta n}$ for some polynomial $p(n)$.

This way, instead of performing the reduction starting from the original problem, we perform the reduction for each of the t formulas Ψ_i. A simple computation yields that, if we can solve the C-SpernerFamily problem in $\mathcal{O}(n^{2-\varepsilon})$ for some $\varepsilon > 0$, then we can solve the k-Sat problem in $\mathcal{O}(p(n)t(2^{\frac{n}{2}})^{2-\varepsilon}) = \mathcal{O}((2-\delta)^n)$ for a suitable choice of δ. The latter result is against SETH. □

Theorem 3.7 ([3]). *The Hitting Set conjecture implies the Orthogonal Vector conjecture.*

Algorithm 1: reduction from the k-HITTINGSET problem to the k-TWODISJOINTSETS problem.

```
1  $h \leftarrow \lfloor \sqrt{n} \rfloor$;
2  for $i$=1,...,$h$ do
3      for $j$=1,...,$h$ do
4          while $\exists C_1 \in \mathcal{C}_{1i}, C_2 \in \mathcal{C}_{2j} : C_1 \cap C_2 = \emptyset$ do
5              remove $C_1$ from $\mathcal{C}_{1i}$;
6          end
7      end
8      if $\mathcal{C}_{1i} \neq \emptyset$ then return an element in $\mathcal{C}_{1i}$;
9  end
10 return "There is no hitting set."
```

Proof. Let us prove the contrapositive: we assume that we can solve the k-TWODISJOINTSETS problem in $\mathcal{O}(n^{2-\varepsilon})$, and we prove that we can solve the k-HITTINGSET problem in $\mathcal{O}(n^{2-\delta})$ for $\delta = \frac{\varepsilon}{2}$ (actually, we use the version of the k-TWODISJOINTSETS problem and the HITTINGSET problem with two collections of sets).

Let us consider an instance of the k-HITTINGSET problem: we have to find a set $C_1 \in \mathcal{C}_1$ that is not disjoint to any set $C_2 \in \mathcal{C}_2$, if it exists. To solve this instance, we divide $\mathcal{C}_1$ and $\mathcal{C}_2$ into at most $h = \lfloor \sqrt{n} \rfloor$ sets of size $\mathcal{O}(\sqrt{n})$, obtaining $\mathcal{C}_{11}, \dots, \mathcal{C}_{1h}, \mathcal{C}_{21}, \dots, \mathcal{C}_{2h}$. Then, we iteratively check if there are $C_1 \in \mathcal{C}_{1i}, C_2 \in \mathcal{C}_{2j}$ such that $C_1 \cap C_2 = \emptyset$: if this is the case, we can safely remove C_1 from $\mathcal{C}_{1i}$, because C_1 is not a hitting set. Otherwise, we continue the procedure. If, at some point, we cannot remove any element in $\mathcal{C}_{1i}$, it means that we have found a hitting set, because $\mathcal{C}_{1i}$ is not disjoint to any set in $\mathcal{C}_2$. Otherwise, at some point, all sets $\mathcal{C}_{1i}$ are empty, and consequently we know that there is no hitting set. The pseudocode is available in Algorithm 1.

Let us analyze the running time of this algorithm. Every time we perform Line 5, either we remove an element from C_{1i}, or the value of j is increased: at most, these two actions can be performed $\mathcal{O}(n)$ times, and consequently Line 5 is performed $\mathcal{O}(n)$ times. Since all other operations cost $\mathcal{O}(n)$ at most, the total running time is $\mathcal{O}(n)$, multiplied by the time needed to find if there are two disjoint sets in a collection of $\mathcal{O}(\sqrt{n})$ sets. If the Orthogonal Vector conjecture is false, then the latter task can be performed in $\mathcal{O}(\sqrt{n}^{2-\varepsilon})$, and consequently the total running time is $\mathcal{O}(n^{2-\frac{\varepsilon}{2}})$, which is truly subquadratic. Consequently, if the Orthogonal Vector conjecture is false, then the Hitting Set conjecture is false as well. □

3.4 Proof of the Other Reductions

In this section, we prove all the reductions in Figure 3.1. For simplicity, since in this thesis we do not analyze approximation algorithms, we do not try to optimize the reductions with respect to the approximation factor obtained, and we refer to the original papers for more information.

Theorem 3.8 ([3]). *The* MAXIMUMCLOSENESSCENTRALITY *problem is not solvable in subquadratic time unless the Hitting Set conjecture is false.*

Proof. We prove this theorem through a reduction from the HITTINGSET problem to the problem of computing the maximum closeness centrality. For simplicity, we consider the *farness*, which is the inverse of the closeness centrality: the farness of a node s in a connected graph is defined as $f(s) = \frac{1}{n-1} \sum_{t \in V} \operatorname{dist}(s,t)$. Given an instance $(X, \mathcal{C})$ of the HITTINGSET problem, we construct a graph G as follows (see Figure 3.2).

- We define $V = V_1 \cup V_2 \cup \{u\} \cup \{v\} \cup \mathcal{C}_0 \cup X_1 \cup X_2 \cup \mathcal{C}_1$, where V_1, V_2 are sets of (large)

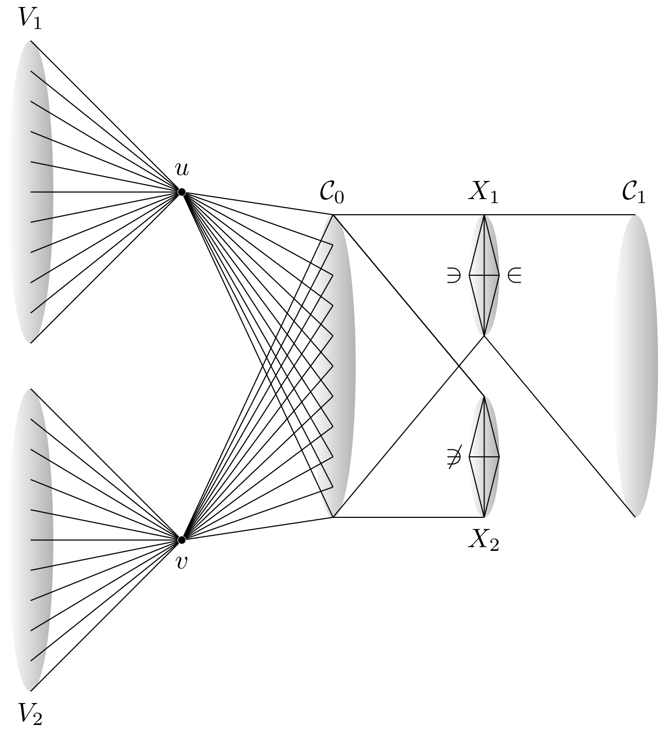


Figure 3.2. The reduction used to prove the hardness of the MaximumClosenessCentrality problem.

cardinality $N = 3(|\mathcal{C}| + |X|)$, $\mathcal{C}_0, \mathcal{C}_1$ are copies of $\mathcal{C}$, X_1, X_2 are copies of X, and u, v are extra nodes.

- The node u is connected to any node in $\mathcal{C}_2$ and any node in $\mathcal{C}_0$, and similarly the node v is connected to any node in $\mathcal{C}_3$ and any node in $\mathcal{C}_0$.
- A node $C \in \mathcal{C}_0$ is connected to a node $x \in X_1$ if $x \in C$, and similarly a node $C \in \mathcal{C}_1$ is connected to a node $x \in X_1$ if $x \in C$.
- Each pair of nodes in X_1 is connected, and each pair of nodes in X_2 is connected.
- A node $x \in X_2$ is connected to a node $C \in \mathcal{C}_0$ if $x \notin C$.
- No other pair of nodes is connected.

First, let us compute the farness of a node $C \in \mathcal{C}_0$:

$$\begin{aligned} f(C) &= \sum_{w \in V} \mathrm{dist}(C, w) \\ &= \sum_{w \in V_1 \cup V_2} 2 + \sum_{w \in \{u,v\}} 1 + \sum_{w \in \mathcal{C}_0} \mathrm{dist}(C, w) + \sum_{w \in X_1 \cup X_2} \mathrm{dist}(C, w) + \sum_{w \in \mathcal{C}_1} \mathrm{dist}(C, w) \\ &= 4N + 2 + 2(|\mathcal{C}_0| - 1) + |C| + 2|X - C| + |X - C| + 2|C| + \sum_{w \in \mathcal{C}_1} \mathrm{dist}(C, w) \\ &= 4N + 2|\mathcal{C}| + 3|X| + 2|\mathcal{C}_1| + |\{C_1 \in \mathcal{C}_1 : C_1 \cap C = \emptyset\}| \\ &= 4N + 4|\mathcal{C}| + 3|X| + |\{C_1 \in \mathcal{C}_1 : C_1 \cap C = \emptyset\}| \\ &= 4N + 4|\mathcal{C}| + 3|X| + |\{C' \in \mathcal{C} : C \cap C' = \emptyset\}| \end{aligned}$$

(we used that the distance between C and a node $C_1 \in \mathcal{C}_1$ is 3 if $C \cap C_1 = \emptyset$, 2 otherwise).

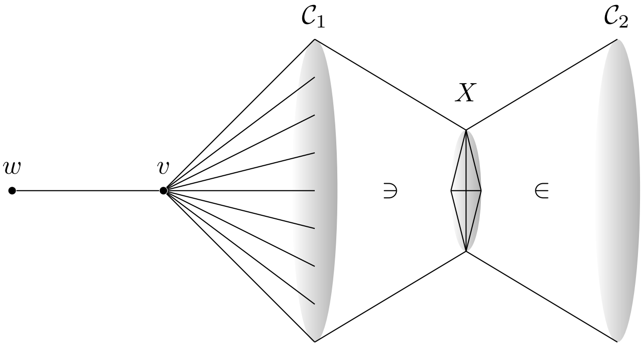


Figure 3.3. The reduction used to prove the hardness of the Radius problem.

We claim that the minimum farnessre is $4N+4|\mathcal{C}|+3|X|$ if and only if there is a hitting set $C \in \mathcal{C}$. One direction is easy: if there is a hitting set $C \in \mathcal{C}$, then $|\{C' \in \mathcal{C} : C \cap C' = \emptyset\}| = 0$, and $f(C) = 4N+4|\mathcal{C}|+3|X|$. For the other direction, if there is no hitting set, then all nodes in $\mathcal{C}_0$ have farness at least $4N + 4|\mathcal{C}| + 3|X| + 1$, because $|\{C' \in \mathcal{C} : C \cap C' = \emptyset\}| \geq 1$ for each C. Let us prove that all the other nodes in the graph have bigger farness: first, $f(w) \geq 6N$ for each node $w \in X_1 \cup X_2 \cup \mathcal{C}_1$, because $\text{dist}(w,t) \geq 3$ for each $t \in V_1 \cup V_2$. If we choose $6N > 4N+4|\mathcal{C}|+3|X|$ (for instance, $N = 2(|\mathcal{C}|+|X|)$), then $f(w) \geq 6N > 4N+4|\mathcal{C}|+3|X|$. It remains to prove that $f(w) > 4N+4|\mathcal{C}|+3|X|$ for each $w \in V_1 \cup V_2 \cup \{u,v\}$ (by symmetry, we only consider $w \in V_1 \cup \{u\}$). If $w \in V_1$, $f(w) \geq f(u)$, because the only neighbor of w is u; moreover, $f(u) \geq 4N + 4|\mathcal{C}| + 4|X| > 4N + 4|\mathcal{C}| + 3|X|$, because u is at distance 1 from each node in V_1, at distance 3 from each node in V_2, at distance 1 from each node in $\mathcal{C}_0$, at distance 3 from each node in $\mathcal{C}_1$, and at distance 2 from each node in $X_1 \cup X_2$. This concludes the reduction.

□

Theorem 3.9 ([3])**.** *The* Radius *problem is not solvable in subquadratic time unless the Hitting Set conjecture is false.*

Proof. We prove this theorem through a reduction from the HittingSet problem to the Radius problem. Given an instance $(X, \mathcal{C})$ of the HittingSet problem, we construct a graph G as follows (see Figure 3.3).

- The set of nodes is made by five parts, and each part is only connected to the previous or the next one:
  1. the first part is made by a single node v;
  2. the second part is made by a single node w;
  3. the third part is a copy $\mathcal{C}_1$ of $\mathcal{C}$;
  4. the fourth part is a copy of X;
  5. the fifth part is a copy $\mathcal{C}_2$ of $\mathcal{C}$.
- There is an edge from v to w.
- There is an edge from w to each node in $\mathcal{C}_1$.
- For each element $x \in C$, where $x \in X$ and $C \in \mathcal{C}$, there is an edge (x, C_1) and (x, C_2), where C_1 and C_2 are the two copies of C in $\mathcal{C}_1$ and $\mathcal{C}_2$.
- All the elements in the X are connected.

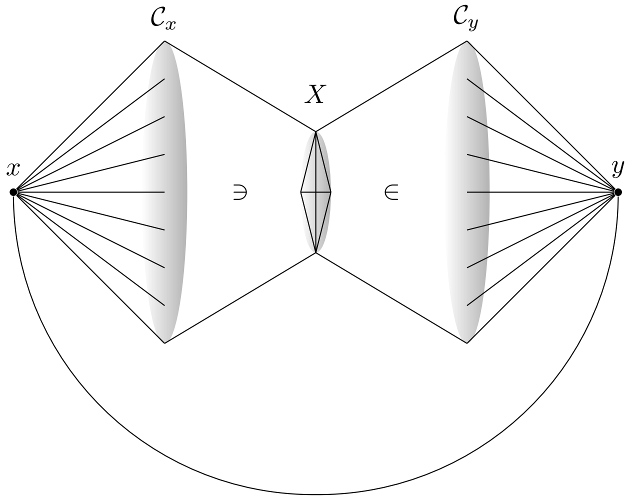


Figure 3.4. The reduction used to prove the hardness of the BetweennessCentralityNode problem.

First, it is easy to prove that this reduction yields a graph where $m, n = \mathcal{O}(|X||\mathcal{C}|) = \mathcal{O}(|\mathcal{C}|\log(|\mathcal{C}|))$. Furthermore, the radius is at least 2, because each part of the graph is only connected to the previous and to the next one, and if the radius is 2, then the radial node must be in $\mathcal{C}_1$. Moreover, all the nodes in $\mathcal{C}_1$ can reach v, w, all the other nodes in $\mathcal{C}_1$, and all the nodes in X in at most 2 steps. Hence, the radius is 2 if and only if there is a node in $\mathcal{C}_1$ that can reach every node in $\mathcal{C}_2$ in 2 steps, if and only if there is a hitting set, because a node $C_1 \in \mathcal{C}_1$ can reach a node $C_2 \in \mathcal{C}_2$ in 2 steps if and only if the two sets C_1 and C_2 have a common element. This means that, if we can find the radius in time $\mathcal{O}(m^{2-\varepsilon})$, then we can find if there are two disjoint sets in $\mathcal{O}(|\mathcal{C}|^{2-\varepsilon}\log^{2-\varepsilon}(|\mathcal{C}|)) = \mathcal{O}(|\mathcal{C}|^{2-\frac{\varepsilon}{2}})$. □

Theorem 3.10. *The* BetweennessCentralityNode *problem is not solvable in subquadratic time unless the Orthogonal Vector conjecture is false.*

Proof. We prove the theorem by reduction from the k-TwoDisjointSets problem. Let us consider an instance $(X, \mathcal{C})$ of k-TwoDisjointSets, and let us construct a graph $G = (V, E)$ as follows:

- $V := \{x\} \cup \mathcal{C}_x \cup X \cup \mathcal{C}_y \cup \{y\}$, where X is the ground set, and $\mathcal{C}_x, \mathcal{C}_y$ are two identical copies of $\mathcal{C}$ (in the graph, $y, x, \mathcal{C}_x, X, \mathcal{C}_y$ somehow resemble a cycle);
- all pairs of nodes in X are connected;
- node x is connected to y and to each node in $\mathcal{C}_x$;
- node y is connected to x and to each node in $\mathcal{C}_y$;
- connections between $\mathcal{C}_x$ and X and connections between $\mathcal{C}_y$ and X are made according to the $\in$-relation.

The input node of our problem is x. We observe that the graph is a big "cycle" made by five "parts": $y, x, \mathcal{C}_x, X, \mathcal{C}_y$. Moreover, it is possible to move from one part to any node in the next part in one step (except if we start or arrive in X, and in that case we need at most two steps). As a consequence, no shortest path can be longer than 3. This proves that any shortest path in the sum passing through x must be of one of the following forms:

- a path from y to $\mathcal{C}_x$;
- a path from $\mathcal{C}_x$ to $\mathcal{C}_y$;

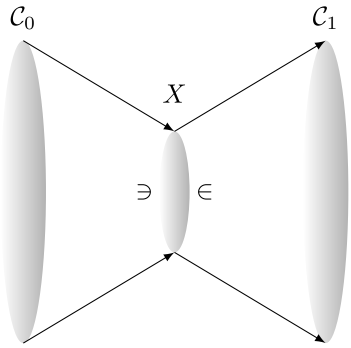


Figure 3.5. The reduction used to prove the hardness of the NumReachableNodes problem.

- a path from y to X

We note that the third case never occurs, because there exists a path of length 2 from y to any node in X. The first case occurs for each node in $\mathcal{C}_x$, and no other shortest path exists from y to nodes in $\mathcal{C}_x$: these nodes contribute to the sum by $|\mathcal{C}|$. Finally, the second case occurs if and only if there is a pair of nodes in $\mathcal{C}_y$ and $\mathcal{C}_x$ having no path of length 2, that is, two disjoint sets in $\mathcal{C}$. This proves that the betweenness of x is bigger than $|\mathcal{C}|$ if and only if there are two disjoint sets in $\mathcal{C}$. □

Remark 3.11. We can also assume that x is the most central node, by adding new nodes that are only connected to x.

Remark 3.12. The BetweennessCentrality problem is not solvable in subquadratic time unless the Orthogonal Vector conjecture is false.

Proof. If we cannot compute in subquadratic time the centrality of a single node, clearly we cannot compute the centrality of all nodes. □

Theorem 3.13. *The* NumReachableNodes *problem is not solvable in subquadratic time unless the Orthogonal Vector conjecture is false.*

Proof. Let us consider an instance of the TwoDisjointSets problem $(X, \mathcal{C})$, and let us construct a digraph $G = (V, E)$ as follows. We let $\mathcal{C}_0$, $\mathcal{C}_1$ be two copies of $\mathcal{C}$, and we define $V = \mathcal{C}_0 \sqcup \mathcal{C}_1 \sqcup X$, where $\sqcup$ denotes the disjoint union. Then, we add a directed edge from a node $C_0 \in \mathcal{C}_0$ to a node $x \in X$ if $x \in C_0$, and similarly we add a directed edge from $x \in X$ to $C_1 \in \mathcal{C}_1$ if $x \in C$. A representation of the reduction is shown in Figure 3.5.

It is clear that, in the graph constructed, $m, n = \mathcal{O}(|X||\mathcal{C}|) = \mathcal{O}(|\mathcal{C}|\log(|\mathcal{C}|))$. Assume that we can compute in subquadratic time the number $r(C_0)$ of nodes reachable from each node $C_0 \in \mathcal{C}_0$. From this, we can also compute in time $\mathcal{O}(1)$ the number of nodes in $\mathcal{C}_1$ that a node $C_0 \in \mathcal{C}_0$ can reach, because this number is

$$\begin{aligned}|\{C_1 \in \mathcal{C}_1 : C_0 \text{ can reach } C_1\}| &= r(C_0) - |\{C_0\}| - |\{x \in X : C_0 \text{ can reach } x\}| \\ &= r(C_0) - 1 - \operatorname{outdeg}(C_0).\end{aligned}$$

It is easy to see that a node $C_0 \in \mathcal{C}_0$ can reach a node $C_1 \in \mathcal{C}_1$ if and only if the sets corresponding to C_0 and C_1 have a common element: hence, we can check if there are two disjoint sets by checking if

$$\forall C_0, |\{C_1 \in \mathcal{C}_1 : C_0 \text{ can reach } C_1\}| = |\mathcal{C}_1|.$$

The time needed to perform this task is $\mathcal{O}(m^{2-\varepsilon} + n)$, because we assumed that we can compute the number of reachable nodes in subquadratic time, and all the other operations need time $\mathcal{O}(n)$. Overall, the running time is

$$\mathcal{O}(m^{2-\varepsilon}) = \mathcal{O}\left(|\mathcal{C}|^{2-\varepsilon}\log(|\mathcal{C}|)^{2-\varepsilon}\right) = \mathcal{O}\left(|\mathcal{C}|^{2-\frac{\varepsilon}{2}}\right).$$

We proved that a subquadratic algorithm for NumReachableNodes yields a subquadratic algorithm for the TwoDisjointSets problem, which is against the Orthogonal Vector conjecture. □

Remark 3.14. This reduction still works if we assume the input graph to be acyclic, and to have diameter at most 2.

Theorem 3.15. *The* LinIndexMaximum *problem is not solvable in subquadratic time unless the Orthogonal Vector conjecture is false.*

Proof. As done in previous cases, we construct a reduction from the TwoDisjointSets problem.

Given an instance $(X, \mathcal{C})$ of the TwoDisjointSets problem, and given a set $C \in \mathcal{C}$, let R_C be $|\{C' \in \mathcal{C} : C \cap C' \neq \emptyset\}|$. The TwoDisjointSet problem has no solutions if and only if $R_C = |\mathcal{C}|$ for all $C \in \mathcal{C}$; indeed, $R_C = |\mathcal{C}|$ means that C intersects all the sets in $\mathcal{C}$. We construct a directed graph $G = (V, E)$, where $|V|, |E| = \mathcal{O}(|\mathcal{C}||X|) = \mathcal{O}(|\mathcal{C}|\log|\mathcal{C}|)$, such that:

1. V contains a set of nodes $\mathcal{C}_0$ representing the sets in $\mathcal{C}$ (from now on, if $C \in \mathcal{C}$, we denote by C_0 the corresponding node in $\mathcal{C}_0$);
2. the centrality of C_0 is a function $c(R_C)$, depending only on R_C (that is, if $R_C = R_{C'}$ then $c(C_0) = c(C'_0)$);
3. the function $c(R_C)$ is decreasing with respect to R_C;
4. the most central node is in $\mathcal{C}_0$.

In such a graph, the node with maximum closeness corresponds to the set C minimizing R_C: indeed, it is in $\mathcal{C}_0$ by Condition 4, and it minimizes R_C by Condition 2-3. Hence, assuming we can find C_0 in time $\mathcal{O}(n^{2-\varepsilon})$, we can easily check if the closeness of C_0 is $c(|\mathcal{C}|)$: if it is not, it means that the corresponding TwoDisjointSet instance has a solution of the form (C, C') because $R_C \neq \mathcal{C}$. Otherwise, for each C', $R_{C'} \geq R_C = |\mathcal{C}|$, because $c(C'_0) \leq c(C_0) = c(|\mathcal{C}|)$, and $c()$ is decreasing with respect to R_C. This means that $R_C = |\mathcal{C}|$ for each C, and there are no two disjoints sets. This way, we can solve the TwoDisjointSets problem in $\mathcal{O}(n^{2-\varepsilon}) = \mathcal{O}((|\mathcal{C}|\log|\mathcal{C}|)^{2-\varepsilon}) = \mathcal{O}(|\mathcal{C}|^{2-\frac{\varepsilon}{2}})$, against the Orthogonal Vector conjecture.

To construct this graph (see Figure 3.6), we start by the construction in the previous proof: we add to V the copy $\mathcal{C}_0$ of $\mathcal{C}$, another copy $\mathcal{C}_1$ of $\mathcal{C}$ and a copy X_1 of X. These nodes are connected as follows: for each element $x \in X$ and set $C \in \mathcal{C}$, we add an edge (C_0, x) and (x, C_1), where C_0 is the copy of C in $\mathcal{C}_0$, and C_1 is the copy of C in $\mathcal{C}_1$. Moreover, we add a copy X_2 of X and we connect all pairs (C_0, x) with $C \in \mathcal{C}$, $x \in X$ and $x \notin C$. This way, the Lin's index of a node $C_0 \in \mathcal{C}_0$ is $\frac{(|X|+R_C)^2}{(n-1)(|X|+2R_C)}$ (which only depends on R_C).

To enforce Items 3 and 4, we add a path of length p leaving each node in $\mathcal{C}_1$, and q nodes linked to each node in $\mathcal{C}_0$, each of which has out-degree $|\mathcal{C}|$: we show that by setting $p = 7$ and $q = 36$, all required conditions are satisfied.

More formally, we have constructed the following graph $G = (V, E)$:

- $V = Z \cup Y \cup \mathcal{C}_0 \cup X_1 \cup X_2 \cup \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_p$, where Z is a set of cardinality $q|\mathcal{C}|$, Y a set of cardinality q, the $\mathcal{C}_i$s are copies of $\mathcal{C}$ and the X_is are copies of X;
- each node in Y has $|\mathcal{C}|$ neighbors in Z, and these neighbors are disjoint;
- for each $x \in C$, there are edges from $C_0 \in \mathcal{C}_0$ to $x \in X_1$, and from $x \in X_1$ to $C_1 \in \mathcal{C}_1$;
- for each $x \notin C$, there is an edge from $C_0 \in \mathcal{C}_0$ to $x \in X_2$;

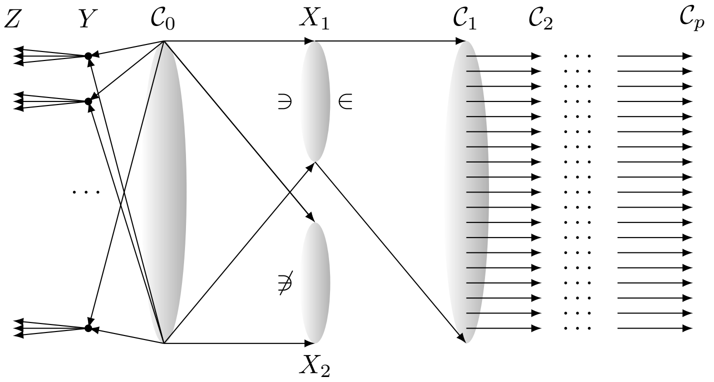


Figure 3.6. The reduction used to prove the hardness of the LinIndexMaximum problem.

- each $C_i \in \mathcal{C}_i$, $1 \leq i \leq p$, is connected to the same set $C_{i+1} \in \mathcal{C}_{i+1}$;
- no other edge is present in the graph.

Note that the number of edges in this graph is $\mathcal{O}(|\mathcal{C}||X|) = \mathcal{O}(|\mathcal{C}| \log^l(|\mathcal{C}|))$, because $|X| < \log^l(|\mathcal{C}|)$,

Claim 3.1. *Assuming $|\mathcal{C}| > 1$, all nodes outside $\mathcal{C}_0$ have Lin's index at most $\frac{2|\mathcal{C}|}{n-1}$, where n is the number of nodes.*

Proof of claim. If a node is in Z, X_2, or $\mathcal{C}_p$, its Lin's index is not defined, because it has out-degree 0.

A node $y \in Y$ reaches $|\mathcal{C}|$ nodes in 1 step, and hence its Lin's index is $\frac{|\mathcal{C}|^2}{|\mathcal{C}|(n-1)} = \frac{|\mathcal{C}|}{n-1}$.

A node in $\mathcal{C}_i$ reaches $p - i$ other nodes, and their distance is $1, \dots, p - i$: consequently, its Lin's index is $\frac{(p-i)^2}{\frac{(p-i)(p-i+1)}{2}(n-1)} = \frac{2(p-i)}{(n-1)(p-i+1)} \leq \frac{2}{n-1}$.

Finally, for a node $x \in X_1$ contained in N_x sets, for each $1 \leq i \leq p$, x reaches N_x nodes in $\mathcal{C}_i$, and these nodes are at distance i. Hence, the Lin's index of x is $\frac{(pN_x)^2}{\frac{p(p+1)}{2}N_x(n-1)} = \frac{2pN_x}{(n-1)(p+1)} \leq \frac{2N_x}{n-1} \leq \frac{2|\mathcal{C}|}{n-1}$. This concludes the proof. □

Let us now compute the Lin's index of a node $C \in \mathcal{C}_0$. The reachable nodes are:

- all q nodes in Y, at distance 1;
- all $|\mathcal{C}|q$ nodes in Z, at distance 2;
- $|X|$ nodes in X_1 or X_2, at distance 1;
- R_C nodes in $\mathcal{C}_i$ for each i, at distance $i + 1$ (the sum of the distances of these nodes is $\sum_{i=1}^{p} i + 1 = -1 + \sum_{i=1}^{p+1} i = \frac{(p+2)(p+1)}{2} - 1$).

Hence, the Lin's index of C is:

$$c(R_C) = \frac{(q(1 + |\mathcal{C}|) + |X| + pR_C)^2}{\left(q(1 + 2|\mathcal{C}|) + |X| + \left(\frac{(p+1)(p+2)}{2} - 1\right) R_C\right)(n-1)}$$
$$= \frac{(q(1 + |\mathcal{C}|) + |X| + pR_C)^2}{\left(q(1 + 2|\mathcal{C}|) + |X| + g(p)R_C\right)(n-1)}$$

where $g(p) = \frac{(p+1)(p+2)}{2} - 1$. We want to choose p and q satisfying:

a. the Lin's index of nodes in $\mathcal{C}_0$ is bigger than $\frac{2|\mathcal{C}|}{n-1}$ (and hence bigger than the Lin's index of all other nodes);

b. $c(R_C)$ is a decreasing function of R_C for $0 \leq R_C \leq |\mathcal{C}|$.

In order to satisfy Condition b., the derivative $c'(R_C)$ of c is

$$\frac{2p(q(1+|\mathcal{C}|)+|X|+pR_C)\left(q(1+2|\mathcal{C}|)+|X|+g(p)R_C\right)-g(p)(q(1+|\mathcal{C}|)+|X|+pR_C)^2}{\left(q(1+2|\mathcal{C}|)+|X|+g(p)R_C\right)^2(n-1)}$$
$$=(q(1+|\mathcal{C}|)+|X|+pR_C)\frac{[pg(p)R_c+2p\left(q(1+2|\mathcal{C}|)+|X|\right)-g(p)(q(1+|\mathcal{C}|)+|X|)]}{\left(q(1+2|\mathcal{C}|)+|X|+g(p)R_C\right)^2(n-1)}.$$

This latter value is negative if and only if $pg(p)R_c+2p\left(q(1+2|\mathcal{C}|)+|X|\right)-g(p)(q(1+|\mathcal{C}|)+|X|)<0$. Assuming $g(p)\geq 5p$ and $R_C<|\mathcal{C}|$, this value is:

$$\begin{aligned}
&pg(p)R_C+2p\left(q(1+2|\mathcal{C}|)+|X|\right)-g(p)(q(1+|\mathcal{C}|)+|X|)\\
&\leq pg(p)|\mathcal{C}|+2pq+4pq|\mathcal{C}|+2p|X|-g(p)(q-|C|-|X|)\\
&\leq pg(p)|\mathcal{C}|+4pq|\mathcal{C}|-g(p)q|\mathcal{C}|\\
&\leq pg(p)|\mathcal{C}|-pq|\mathcal{C}|.
\end{aligned}$$

Assuming $q>g(p)$, we conclude that $c'(R_C)<0$ for $0\leq R_C\leq|\mathcal{C}|$, and we satisfy Condition b.. In order to satisfy Condition a., we want $c(R_C)\geq\frac{2|\mathcal{C}|}{n+1}$ (since $c(R_C)$ is decreasing, it is enough $c(|\mathcal{C}|)\geq\frac{2|\mathcal{C}|}{n+1}$). Under the assumptions $q>g(p)$, $0<|X|\leq|\mathcal{C}|$ (which trivially holds for $|\mathcal{C}|$ big enough, because $|X|\leq\log^p|\mathcal{C}|$),

$$\begin{aligned}
c(|\mathcal{C}|)&=\frac{(q(1+|\mathcal{C}|)+|X|+pR_C)^2}{\left(q(1+2|\mathcal{C}|)+|X|+g(p)R_C\right)(n-1)}\\
&\geq\frac{q^2|\mathcal{C}|^2}{(q(3|\mathcal{C}|)+|\mathcal{C}|+|\mathcal{C}|)(n-1)}\\
&\geq\frac{q|\mathcal{C}|}{5(n-1)}>\frac{2|\mathcal{C}|}{n-1}
\end{aligned}$$

if $q>10$.

To fulfill all required conditions, it is enough to choose $p=7$, $g(p)=35$, and $q=36$.

□

Theorem 3.16. *The* ClosenessCentralityMinimum *problem is not solvable in subquadratic time unless the Orthogonal Vector conjecture is false.*

Proof. Instead of minimizing the closeness centrality, we try to maximize the *farness*, which is the inverse of the closeness centrality, and it is defined by

$$f(s)=\frac{1}{n-1}\sum_{t\in V}\operatorname{dist}(s,t).$$

We build a graph where the nodes with biggest farness correspond to sets in $\mathcal{C}$, and the value of the farness does not depend on the corresponding set, if this set is not disjoint to any other set. If this latter condition is not satisfied, then the farness of the node is bigger. In particular, let us consider an instance $(X,\mathcal{C})$ of the k-TwoDisjointSets problem. To simplify the exposition, we need some simple assumptions on $(X,\mathcal{C})$.

- The empty set is not in $\mathcal{C}$: otherwise, we know that there are two disjoint sets.

- For each $x\in X$ there is a set $C\in\mathcal{C}$ containing x: otherwise, we can simply remove x from X.

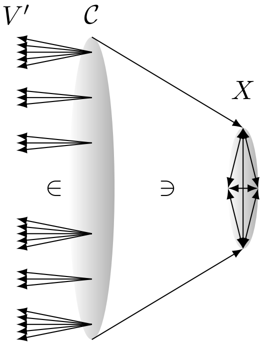


Figure 3.7. The reduction used to prove the hardness of the ClosenessCentralityMinimum problem.

- The cardinality of C is even for each $C \in \mathcal{C}$, and the cardinality of X is even: we can enforce this condition by considering $X = X_1 \sqcup X_2$ where X_1 and X_2 are two copies of X, and by redefining each set C as the union of the copy of C in X_1 and the copy of C in X_2.

Let us build a graph $G = (V, E)$ in the following way (see Figure 3.7):

- $V = X \sqcup \mathcal{C} \sqcup V'$, where V' is a set of cardinality $\sum_{C \in \mathcal{C}} v_C$, and the values of v_C will be chosen later;
- the nodes in X form a clique (that is, each pair of nodes in X is connected);
- for $x \in X$ and $C \in \mathcal{C}$, x is connected to $\mathcal{C}$ if and only if $x \in C$;
- each node $C \in \mathcal{C}$ is connected to v_C nodes in V', and different sets are connected to different nodes.

Clearly, in this graph, $m, n = \mathcal{O}(|X||\mathcal{C}|) = \mathcal{O}(|\mathcal{C}| \log(|\mathcal{C}|))$.

Claim 3.2. *The node with maximum farness is in V'.*

Proof of claim. For each node $C \in \mathcal{C}$, consider a node $v \in V'$ linked to C. Then, since the only neighbor of v is C,

$$\begin{aligned} f(v) &= \frac{1}{n-1} \sum_{w \in V} \operatorname{dist}(v, w) \\ &= \frac{1}{n-1} \sum_{w \neq v} (\operatorname{dist}(v, C) + \operatorname{dist}(C, w)) \\ &= \frac{1}{n-1} \left(n - 1 + \sum_{w \in V} \operatorname{dist}(C, w) - \operatorname{dist}(C, v) \right) \\ &= \frac{n-2}{n-1} + f(C). \end{aligned}$$

Hence, if $n > 2$, $f(v) > f(C)$.

Furthermore, for each node $x \in X$, let us consider a set C containing x, and a node $v \in V'$ linked to C. Clearly, for each node $w \neq v$, $\operatorname{dist}(v, w) = \operatorname{dist}(v, C) + \operatorname{dist}(C, w) = 1 + \operatorname{dist}(C, w) \leq \operatorname{dist}(x, w)$, and consequently $f(v) \geq f(x)$. Moreover, since there are nodes that are closer to x than v (for instance, all nodes in X), we conclude that $f(v) > f(x)$. □

At this point, let us compute the farness of nodes in $v \in V'$, that are linked to a set C. By the proof of the previous claim, it is enough to compute $f(C)$ for each set $C \in \mathcal{C}$. Let us compute the nodes at distance k from C.

- In X, there are $|C|$ nodes at distance 1 from C, and $|X| - |C|$ nodes at distance 2.
- In $\mathcal{C}$, there are $|R_C|$ nodes at distance 2 from $\mathcal{C}$, and $|\mathcal{C}| - |R_C|$ nodes at distance 3, where R_C is the subset of $\mathcal{C}$ made by all sets C' such that $C \neq C'$ and $C \cap C' \neq \emptyset$.
- In V', there are v_C nodes at distance 1 from C (the neighbors of C), $\sum_{C' \in R_C, C' \neq C} v_C$ nodes at distance 3, and $\sum_{C' \notin R_C} v_C$.

Hence, the farness of C satisfies

$$\begin{aligned}(n-1)f(C) &= |C| + 2(|X| - |C|) + 2R_C + 3(|\mathcal{C}| - |R_C|) + v_C + 3 \sum_{\substack{C' \in R_C \\ C' \neq C}} v_C + 4 \sum_{C' \notin R_C} v_C \\ &= -|C| + 2|X| + (|\mathcal{C}| - R_C) + 2|\mathcal{C}| - 2v_C + 3 \sum_{C' \in \mathcal{C}} v_{C'} + \sum_{C' \in \mathcal{C} - R_C} v_{C'} \\ &= \left(2|X| + 2|\mathcal{C}| + 3 \sum_{C' \in \mathcal{C}} v_{C'}\right) + \left((|\mathcal{C}| - R_C) + \sum_{C' \in \mathcal{C} - R_C} v_{C'}\right) - \big(2v_C + |C|\big).\end{aligned}$$

This expression is divided into three terms: the first term is a constant, for each set $C \in \mathcal{C}$, the second expression is 0 if $R_C = |\mathcal{C}|$, otherwise it is positive. We want to choose the v_Cs so that the third expression does not depend on C as well: for example, since we assumed that $|X|$ and $|C|$ are even, we can choose

$$v_C = \frac{|X| - |C|}{2}.$$

This way, for each $v \in V'$,

$$f(v) \;=\; \frac{1}{n-1}\left(\left(n - 2 + 3|X| + 2|\mathcal{C}| + 3 \sum_{C' \in \mathcal{C}} v_{C'}\right) + \left((|\mathcal{C}| - R_C) + \sum_{C' \in \mathcal{C} - R_C} v_{C'}\right)\right),$$

where C is the set connected to v. Hence, if we can compute the node v with maximum farness in time $\mathcal{O}(n^{2-\varepsilon})$, then we can check if

$$f(v) > \frac{1}{n-1}\left(n - 2 + 3|X| + 2|\mathcal{C}| + 3 \sum_{C' \in \mathcal{C}} v_{C'}\right).$$

If this is the case, then $R_C \neq \mathcal{C}$, and there are two disjoint sets; otherwise, for each C, $R_C = \mathcal{C}$, and there are not two disjoint sets. This means that we can solve the TwoDisjointSets problem in subquadratic time, against the Orthogonal Vector conjecture. □

Remark 3.17. The SubsetGraph problem is not solvable in subquadratic time unless the Orthogonal Vector conjecture is false.

Proof. A variation of the Orthogonal Vector conjecture says that it is hard to decide if a family of sets $\mathcal{C}$ is a Sperner family, that is, there are not two sets $C, C' \in \mathcal{C}$ such that $C \subseteq C'$. This means that it is hard to decide if the subset graph is empty, and consequently it is hard to compute the subset graph. □

Theorem 3.18. *The problem* Bipartite3DominatingSet *and the problem* BipartiteSubset2DominatingSet *are not solvable in subquadratic time unless the Orthogonal Vector conjecture is false.*

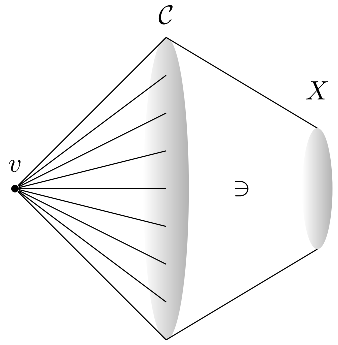


Figure 3.8. The reduction used to prove the hardness of the Bipartite3DominatingSet and the BipartiteSubset2DominatingSet problems.

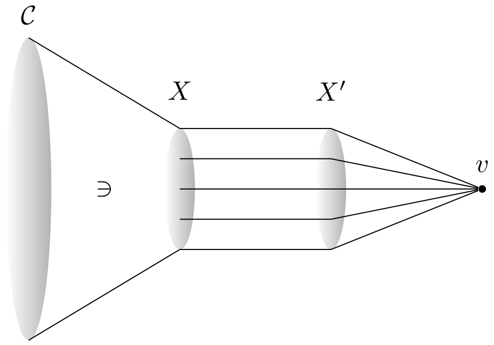


Figure 3.9. The reduction used to prove the hardness of the DominatedNode problems.

Proof. Let $(X, \mathcal{C})$ be an instance of k-TwoCovering, and let us construct a bipartite graph $G = (X \sqcup \mathcal{C}, E)$, where the bipartition is $\{X, \mathcal{C}\}$ (Figure 3.8). Then, each 2-covering of X with sets in $\mathcal{C}$ corresponds to a pair of nodes of $\mathcal{C}$ that covers X.

This way, we have proved that TwoCovering $\leq_{ql}$ BipartiteSubset2DominatingSet, by choosing X as the subset to cover.

In order to prove that TwoCovering $\leq_{ql}$ Bipartite3DominatingSet, we add another node v connected to any $C \in \mathcal{C}$: clearly, if there is a 2-covering of X, there is a dominating triple in G, which is the 2-covering and v.

Viceversa, if there is a dominating triple, one of the nodes of the triple must be in $\mathcal{C}$ because the graph is bipartite: hence, the other two nodes form a dominating pair (if only one of these nodes is in $\mathcal{C}$, the corresponding set is the whole X, and there is clearly a dominating pair). □

Corollary 3.19. *The problem* DominatingSet3 *is not solvable in subquadratic time unless the Orthogonal Vector conjecture is false.*

Remark 3.20. The problem DominatedNode is not solvable in subquadratic time unless the Orthogonal Vector conjecture is false.

Proof. We start from the k-SpernerFamily problem, and we start from the classic construction of a bipartite graph in which $V = \mathcal{C} \sqcup X$ and edges are created according to the $\in$-relation. Clearly, a node $C \in \mathcal{C}$ dominates another node $C' \in \mathcal{C}$ if and only if $C \supseteq C'$. However, we need to avoid that two nodes in X dominate each other: to this purpose, if we find two nodes $x, y \in X$ such that x dominates y, it is enough to add another copy X' of X and connect each node $x \in X$ to the same node $x \in X'$: this way, nodes in X cannot dominate each other, and nodes in X' cannot dominate each other. Finally, to avoid that a node in $\mathcal{C}$ dominates a node in X', we add a further node v which is connected to all the nodes in X' (Figure 3.9).

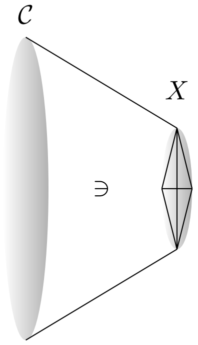


Figure 3.10. The reduction used to prove the hardness of the DiameterSplitGraph2Or3 problem.

Assuming that all sets have at least 2 nodes, there is no domination in the graph constructed. □

Theorem 3.21 ([138])**.** *The* DiameterSplitGraph2Or3 *problem is not solvable in subquadratic time unless the Orthogonal Vector conjecture is false.*

Proof. Given an input $(X, \mathcal{C})$ of k-TwoDisjointSets, we construct a split graph $G = (X \sqcup \mathcal{C}, E)$, where each pair in X is connected, and for each set $C \in \mathcal{C}$ we add an edge from C to its element, so that the number of edges is $\mathcal{O}(|X||\mathcal{C}|)$ (see Figure 3.10).

Since each node is at distance 1 from X, the diameter is 2 or 3: it is 3 if and only if there exist two different nodes $C, C' \in \mathcal{C}$ with no common neighbor, if and only if C and C' are disjoint. □

Corollary 3.22. *The* Diameter *problem is not solvable in subquadratic time unless the Orthogonal Vector conjecture is false.*

Theorem 3.23. *The three problems* Hyperbolicity*,* HyperWith1FixedNode*,* HyperWith2FixedNodes *are not solvable in subquadratic time unless the Orthogonal Vector conjecture is false.*

In order to prove this theorem, we need to anticipate two lemmas, that we will prove in Chapter 7.

Lemma 3.24 (see Lemma 7.1)**.** *For any quadruple (s, t, u, v) of nodes, if the maximum pairwise sum is $\operatorname{dist}(s, t) + \operatorname{dist}(u, v)$, $\delta(s, t, u, v) \leq \frac{1}{2} \min(\operatorname{dist}(s, t), \operatorname{dist}(u, v))$.*

Lemma 3.25 (see Lemma 7.6)**.** *For each quadruple of nodes (s, t, u, v),*

$$\delta(s, t, u, v) \leq \min_{w, w' \in \{s,t,u,v\}, w \neq w'} \operatorname{dist}(w, w').$$

Proof of Theorem 3.23. The construction used in the proof of the hardness of all hyperbolicity problems is the same. Given an input graph $G = (V, E)$ for the Diameter problem, the corresponding graph for the hyperbolicity problem is $H = (V', E')$, defined as follows (Figure 3.11). The set V' is $\{s\} \cup V_s \cup \tilde{V} \cup \{u\} \cup V_t \cup \{t\}$, where V_s, $\tilde{V}$ and V_t are disjoint copies of V, s, t, and u are new nodes. The set E' is defined as follows:

- s is connected to every node in V_s, t is connected to every node in V_t, and u is connected to every node in V_s and to every node in V_t;
- corresponding nodes in V_s and $\tilde{V}$ and corresponding nodes in $\tilde{V}$ and V_t are connected;
- if (v, w) is an edge of G, then the copies of v and w in $\tilde{V}$ are linked.

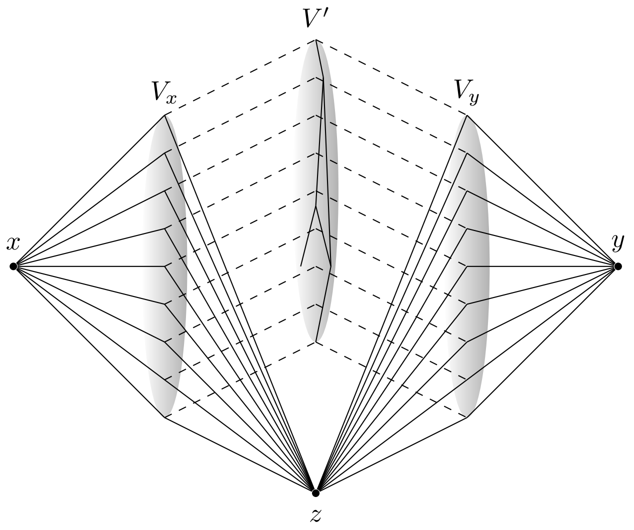


Figure 3.11. The reduction used to prove the hardness of the Hyperbolicity problems. Some edges are dashed to improve readability.

By this construction, if s, t are the aforementioned nodes and v, w is a pair of nodes in $\tilde{V}$, then $2\delta(s,t,v,w) = \mathrm{dist}(s,t)+\mathrm{dist}(v,w)-\max(\mathrm{dist}(x,v)+\mathrm{dist}(t,w), \mathrm{dist}(s,w)+\mathrm{dist}(t,v)) = 4+\mathrm{dist}(v,w)-4 = \mathrm{dist}(v,w)$, and $\mathrm{dist}(x,y)+\mathrm{dist}(v,w)$ is the biggest sum. Furthermore, by construction, the shortest paths between two nodes in $\tilde{V}$ remain in $\tilde{V}$, or they have length at least 4. This means that $\max_{v,w\in\tilde{V}} \delta(s,t,v,w) = \max_{v,w\in\tilde{V}} \frac{\mathrm{dist}(s,t)}{2}$ is at most 1 if and only if the maximum distance between two nodes in $\tilde{V}$ is at most 2, if and only if the diameter of G is at most 2. In order to conclude our reduction, we prove that other quadruples in H have smaller hyperbolicity, so that the hyperbolicity of H is exactly half the diameter of G.

Indeed, let us consider a quadruple v, w, v', w' having hyperbolicity bigger than 1, and let us first prove that s and t belong to this 4-tuple. This happens because, by Lemma 3.25, the distance between any pair of these nodes is at least 2, and consequently $2\delta(v,w,v',w') \leq S_1 - 4$. If we want the hyperbolicity to be bigger than 1, we need S_1 to be strictly bigger then 6, and a distance in S_1 must be at least 4. Since the only such distance in H is $\mathrm{dist}(s,t)$, s and t must be in the quadruple v, w, v', w', as we wanted to prove. We conclude the proof by showing that the hyperbolicity of a quadruple s, t, v, w with $v \notin \tilde{V}$ is at most 1. If $v = z$, this holds because $\mathrm{dist}(u,w) \leq 2$ for each w, and hence $\delta(s,t,u,w) \leq 1$ by Lemma 3.24. If $v \in V_x$ (resp. $v \in V_y$), then $\delta(s,t,v,w) \leq 1$ because $\mathrm{dist}(s,v) = 1$ (resp. $\mathrm{dist}(s,v) = 1$), and we may use Lemma 3.25.

As a consequence, we may conclude our reductions, because the hyperbolicity of H is bigger than 1 if and only if the diameter of G is bigger than 2. Since we know that s and t belong to the quadruple with maximum hyperbolicity, the hardness of all hyperbolicity problems follows from the hardness of the diameter computation. □

Remark 3.26. The problem AllEccentricities is not solvable in subquadratic time unless the Orthogonal Vector conjecture or the Hitting Set conjecture is false.

Proof. From all the eccentricities, one can find the minimum (radius) or the maximum (diameter) in time $\mathcal{O}(n)$. □

3.5 Bibliographic Notes

This chapter surveys several existing results in the field of polynomial-time reductions. The first reduction that appears in the literature link the complexity of the k-Sat* problem to the k-TwoDisjointSets problem, and it was proved in [168] (later, this reduction was restated by saying that SETH implies the Orthogonal Vector conjecture). The other existing reductions are taken from [171, 131, 138, 4, 2, 44, 1, 3], with some modifications in some constructions, used to simplify the proofs (due to the large number of reductions, we refer to the specific theorems for more information).

The equivalence of different formulations of the Orthogonal Vector conjecture and the Hitting Set conjecture is mainly folklore, but some of the versions were collected in [3].

The reductions dealing with minimum closeness centrality, betweenness centrality, and hyperbolicity are original, and they were published in [31], which also surveys many of the reductions available in this chapter. Finally, the hardness of the NumReachableNodes problem is original, and it was published in [28]; also the hardness of the LinIndexMaximum problem is original, and it was published in [20].

Chapter 4

Computing Diameter and Radius: the SumSweep Algorithm

Abstract

We study the computation of the diameter and the radius of graphs. In the directed, not strongly connected case, since the definition of radius is not well-established, we propose a new definition that generalizes all existing attempts.

Then, we present a new heuristic that lower bounds the diameter and upper bounds the radius of a graph, and a new algorithm that computes these quantities exactly (the source code is publicly available in the WebGraph library).

The new algorithms work on undirected and directed graphs, even if they are not necessarily (strongly) connected: as far as we know, this is the first nontrivial algorithm that is able to handle the directed, not strongly connected case.

In the particular case of undirected, or strongly connected graphs, we experimentally show that our algorithm outperforms all previous approaches.

Finally, as a case study, we apply the new algorithm to the computation of the diameter and the radius of the Wikipedia graph, and of several snapshots of the actor graph.

The diameter and the radius of a network are relevant measures (whose meaning depends on the semantics of the network itself), which have been almost always considered while analyzing real-world networks such as biological, collaboration, communication, road, social, and web networks (see, for example, [41]). Many algorithmic results have been presented in the last few decades concerning the computation of these quantities.

However, it is unlikely that we significantly improve the textbook approach, based on computing the distance between all pairs of nodes. Indeed, the complexity of computing radius and diameter is strictly linked to the complexity of many other hard problems: in the dense, weighted case, it is possible to prove that computing the diameter is at least as hard as computing betweenness or reach centrality, and computing the radius is even harder, since it is equivalent to computing the All Pairs Shortest Paths [1]. In both cases, there is no subcubic algorithm, where an algorithm is said to be subcubic if its running time is $\mathcal{O}(n^{3-\varepsilon})$ for some $\varepsilon > 0$ (clearly, there is a cubic algorithm based on computing the All Pairs Shortest Paths). Also in the sparse (unweighted) case, there are several results that link the complexity of computing the diameter and the radius to important conjectures, as we saw in Chapter 3. Overall, we can say that, very likely, the textbook algorithms cannot be significantly outperformed.

Unfortunately, such algorithms are not feasible whenever we deal with real-world networks, that are usually sparse, and that contain millions, and even billions of nodes and edges. In order to overcome this issue, three different approaches have been proposed in the literature:

approximation algorithms [47, 138, 3], restrictions to specific classes of graphs [60, 18], and heuristic approaches. Since our work is inscribed in the latter line of research, let us discuss it in more details.

The first attempts used the sampling algorithm [118, 108], which performs some BFSs from random nodes, and returns the maximum and minimum eccentricities found (we recall that, for a node s, the eccentricity $\mathrm{ecc}(s)$ is defined as $\max_{t \in V} \mathrm{dist}(s,t)$). In order to improve this approach, researchers tried to use better choice strategies for the nodes from which the BFSs have to be performed. For example, the so-called 2-Sweep heuristic picks one of the farthest nodes t from a random node s and returns the eccentricity of t [114], while the 4-Sweep picks the node in the middle of the longest path computed by a 2-Sweep execution and performs another 2-Sweep from that node [61]. Both methods work quite well, and they provide tight bounds very often. Furthermore, in the case of special classes of graphs, they can even be (almost) exact: for example, the 2-Sweep method gives the exact value of the diameter for trees [88], yields an approximation with additive error 1 for chordal graphs and interval graphs, and within 2 for AT-free graphs and hole-free graphs (see [60] for the proof and for the definitions of these graph classes). Adaptations of these methods to directed graphs have been proposed in [45, 63], and, even in this case, these techniques are very efficient and provide very good bounds on real-world networks.

However, in general, these heuristics cannot guarantee the correctness of the results obtained. For this reason, a major further step in the diameter computation was the design of bound-refinement algorithms. These methods apply a heuristic and try to validate the result found or improve it until they successfully validate it. Even if in the worst-case their time complexity is $\mathcal{O}(mn)$, they turn out to be almost linear in practice. The main algorithms developed until now are the BoundingDiameters algorithm [155] and the iFub algorithm [62, 61]. While the first works only on undirected graphs, the second is also able to deal with the directed strongly connected case (the adaptation is called diFub [63]). For the radius computation, the current best algorithm for undirected graphs is a modification of the BoundingDiameters algorithm [155] while for directed graphs it is possible to use the method in [116]. However, all these bound-refinement algorithms cannot deal with directed graphs that are not strongly connected.

4.1 Our Contribution

In this chapter, we propose the first bound-refinement algorithm that is able to find diameter and radius in directed, not necessarily strongly connected graphs (in the case of the radius, since there is no well-established definition, we propose a definition that generalizes all existing definitions). We also show that, when restricted to undirected or strongly connected graphs, our algorithm outperforms previous counterparts. The new algorithms are publicly available in the WebGraph library [23].

As a case study, we use our new algorithm in order to analyze the actors collaboration network and the Wikipedia citation network.

4.2 Notations and Preliminary Definitions

We have already defined the eccentricity of nodes, the diameter, and the radius of a graph in Chapter 2. However, in this section, we discuss a bit more thoroughly these concepts, and we motivate our choices when dealing with definitions that are not completely established in the literature.

In particular, the forward and backward eccentricities of a node s are usually defined as:

$$\mathrm{ecc}_1^F(s) := \max_{t \in V} \mathrm{dist}(s,t),$$
$$\mathrm{ecc}_1^B(t) := \max_{s \in V} \mathrm{dist}(s,t),$$

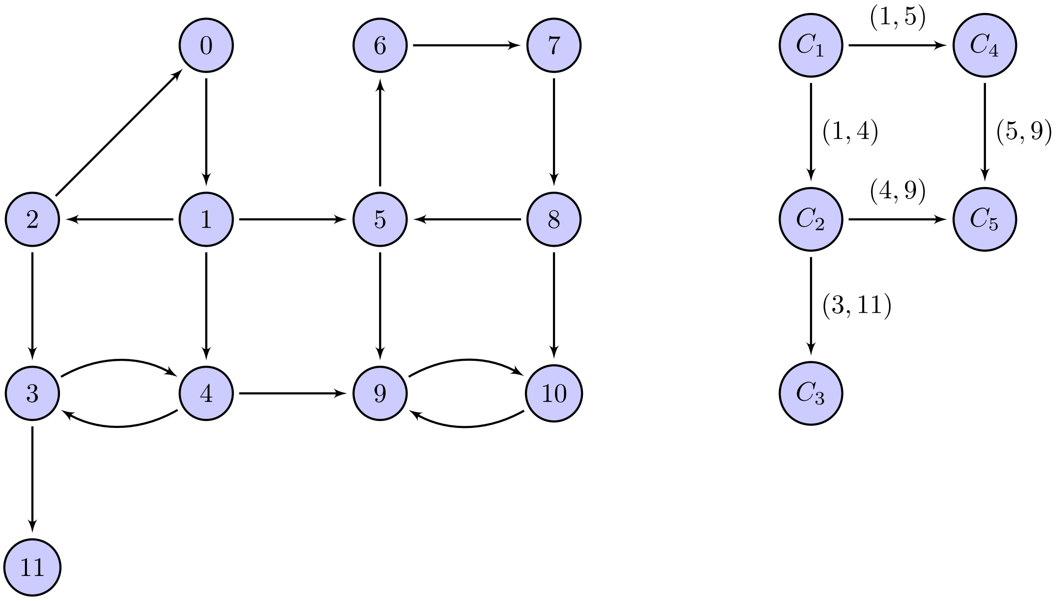


Figure 4.1. A weakly connected graph and the corresponding strong component graph. Each edge $(C_i, C_j) \in \mathcal{E}$ is labeled with one of the edges from the component C_i to the component C_j.

where $\operatorname{dist}(s,t)$ can be $+\infty$ if t is not reachable from s. However, since most of the real-world graphs are not (strongly) connected, these definitions would lead to infinite eccentricities in almost all cases. For this reason, we prefer to ignore infinite distances and we define the forward and backward eccentricities as:

$$\operatorname{ecc}^F(s) := \max_{t \in R^F(s)} \operatorname{dist}(s,t),$$
$$\operatorname{ecc}^B(t) := \max_{s \in R^B(t)} \operatorname{dist}(s,t),$$

where we recall that $R^F(s)$ is the set of nodes reachable from s, and $R^B(t)$ is the set of nodes that can reach t. The diameter is the maximum eccentricity of a node, that is, $D = \max_{s \in V} \operatorname{ecc}^F(s) = \max_{t \in V} \operatorname{ecc}^B(t)$: in other words, this is the length of "a longest shortest path" [89]. Note that in [14, 167] the diameter is defined through the original definition of eccentricity: however, this implies that the diameter of any disconnected graph is $+\infty$, leading to trivial behaviors.

The radius is usually defined as the minimum forward eccentricity of a node [14, 167]: in most real-world networks, using the old definition of eccentricity, we have that the radius is $+\infty$, while, by using the new definition of eccentricity, the radius is 0. Both these definitions are just related to reachability: in particular, the first is affected by nodes that cannot be reached, while the second is affected by the existence of nodes with out-degree 0. In order to be able to exclude such nodes, we consider a set V' of "meaningful" nodes, and define the radius as the minimum eccentricity of a node in V': if we choose V' to be the set of nodes s such that $\operatorname{ecc}^F(s) < +\infty$, or the set of all nodes, we simulate the two aforementioned definitions. In this work, if n_1 is the maximum size of a strongly connected component, we choose $V' = V_1' \cup V_2'$, where V_1' is the set of nodes in a component of size n_1, and V_2' is the set of nodes that are able to reach a node in V_1' (in almost all real-world networks there is only a component of size n_1, the so-called *giant component* [125]). For example, by referring to the graph in Figure 4.1, we have that $V' = \{0, 1, 2, 5, 6, 7, 8\}$. In any case, our algorithms work for any choice of V': in particular, they are also able to compute the radius according to the aforementioned definitions.

Algorithm 2: The SumSweepHeuristic.

```
Input: a graph $G = (V, E)$, a set $V' \subseteq V$, a node $s$, and an integer $k$
Output: an upper bound of the radius $R_U$ and a lower bound of the diameter $D_L$ of $G$
for $i \in V$ do $S^F(i) \leftarrow 0$; $S^B(i) \leftarrow 0$;
$F \leftarrow \{s\}$; $B \leftarrow \emptyset$;
FBFS($s$);
$D_L \leftarrow \mathrm{ecc}^F(s)$;
for $v \in V$ do
    if $\mathrm{dist}(s, v) < +\infty$ then $S^B(v) \leftarrow \mathrm{dist}(s, v)$;
end
for $i \in \{2, \dots, k-1\}$ do
    if $i$ is odd then
        $s \leftarrow \mathrm{argmax}_{v \in V-F} S^F(v)$;
        $F \leftarrow F \cup \{s\}$;
        FBFS($s$);
        $D_L \leftarrow \max(D_L, \mathrm{ecc}^F(s))$;
        for $v \in V$ do
            if $\mathrm{dist}(s, v) < +\infty$ then $S^B(v) \leftarrow S^B(v) + \mathrm{dist}(s, v)$;
        end
    else
        $s \leftarrow argmax_{v \in V-B} S^B(v)$;
        $B \leftarrow B \cup \{s\}$;
        BBFS($s$);
        $D_L \leftarrow \max(D_L, \mathrm{ecc}^B(s))$;
        for $v \in V$ do
            if $\mathrm{dist}(v, s) < +\infty$ then $S^F(v) \leftarrow S^F(v) + \mathrm{dist}(v, s)$;
        end
    end
end
$s \leftarrow \mathrm{argmin}_{v \in V'} S^F(v)$;
FBFS($s$);
$R_U \leftarrow \mathrm{ecc}^F(s)$;
return $D_L$ and $R_U$
```

4.3 The SumSweepHeuristic

The SumSweepHeuristic finds a lower bound on the eccentricity of all nodes, a lower bound D_L on the diameter, and an upper bound R_U on the radius. Intuitively, it is based on finding several "peripheral" nodes $t_1, \dots, t_k$, and estimating $\mathrm{ecc}(s) = \max_{i=1,\dots,k} \mathrm{dist}(s, t_i)$ for each node s. Then, the heuristic finds a very "central" node and sets R_U as the eccentricity of this node (the directed case is slightly more complicated, because we have to consider forward and backward eccentricities). The pseudo-code is shown in Algorithm 2.

Like other heuristics [45, 114, 63, 61], the SumSweepHeuristic is based on performing alternatively forward and backward BFSs from some nodes. By forward BFS or FBFS (resp., backward BFS or BBFS) we mean a visit in BFS order in which a directed edge (v, w) is traversed from v to w (resp., from w to v). The new feature with respect to the previous heuristics is the choice of the starting nodes of the BFSs, which is based on the two following quantities, that try to distinguish "central" and "peripheral" nodes:

$$S^F(v) := \sum_{s \in B \cap R^F(v)} \mathrm{dist}(v, s)$$

$$S^B(v) := \sum_{s \in F \cap R^B(v)} \mathrm{dist}(s, v)$$

where F (resp. B) is the set of starting nodes of the forward (resp. backward) BFSs already performed. These quantities resemble the farness $f(v) = \frac{1}{n-1} \sum_{w \in V} \mathrm{dist}(v, w)$ of the node v: this means that we are performing BFSs from nodes with high farness, and consequently low closeness centrality. In this case, since a high closeness centrality value means that a node is central, if $S^F(s)$ or $S^B(s)$ is big, it means that s is "peripheral" and hence a good candidate to have a big eccentricity.

It is worth observing that the starting node of the first BFS plays a slightly different role: it should not be a node with high forward eccentricity, but a node which helps us finding high-eccentricity nodes in the next steps. To do so, for instance, we suggest to choose the maximum out-degree node.

At the end of the procedure, we approximate the diameter with the maximum eccentricity found, and the radius with the eccentricity of the node $v \in V'$ minimizing $S^F(v)$, which should be rather central, according to the aforementioned intuition. Observe that if the graph is undirected, the forward and backward eccentricities coincide: this means that a single BFS is enough to perform a forward and backward step of the SumSweepHeuristic.

As an example, let us show the results of a SumSweepHeuristic on the graph in Figure 4.1 with $k = 4$, $V' = \{0, 1, 2, 5, 6, 7, 8\}$, and $s = 1$ (which is the maximum out-degree node). The first forward BFS visits the whole graph and finds a lower bound $D_L = 4$ and an upper bound $R_U = 4$. The second step performs a backward BFS from node 8 (the only node at distance 4 from 1) and sets $D_L = 6$, since the distance from 2 to 8 is 6. This BFS is followed by a forward BFS from node 2, which has eccentricity 6 and consequently does not improve the previous bounds. Finally, a BFS from 8 is performed: since 8 has forward eccentricity 3, R_U is set to 3. Then, SumSweepHeuristic returns $D_L = 6$ and $R_U = 3$ (note that these are the correct values of radius and diameter).

4.4 The ExactSumSweep Algorithm

In this section, we show how to compute the exact values of the diameter and the radius of a graph, by validating the bounds given by the SumSweepHeuristic. The general framework has been proposed in [154, 155] for undirected graphs: in this chapter, we adapt it to handle any directed graph. The general schema of our algorithm is shown in Algorithm 3.

Algorithm 3: The ExactSumSweep algorithm.

```
Input: a graph $G = (V, E)$, a set $V' \subseteq V$, a node $s$, and an integer $k$
Output: the radius $R$ and the diameter $D$ of $G$
for $i \in \{1, \ldots, |V|\}$ do $L^F(i) \leftarrow 0$; $L^B(i) \leftarrow 0$; $U^F(i) \leftarrow |V|$; $U^B(i) \leftarrow |V|$;
$D_L, R_U \leftarrow$ SumSweepHeuristic$(G, V', s, k)$;
while $(D_L < \max_{i \in V}\{U^F(i)\} \wedge D_L < \max_{i \in V}\{U^B(i)\}) \vee R_U > \min_{i \in V'}\{L^F(i)\}$ do
    Choose a technique from {StepForward, StepBackward,
     SingleCCUpperBound};
    Use it to update $D_L$, $R_U$, $L^F$, $L^B$, $U^F$, $U^B$;
end
return $D_L, R_U$;
```

After performing the SumSweepHeuristic, the algorithm tries to prove that the computed bounds are the exact values of the radius and of the diameter, or to improve them. This can be done by bounding the eccentricities of all the nodes in the graph. More specifically, for each node v, we store these values:

- $L^F(v)$ is a lower bound on the forward eccentricity of v;

- $U^F(v)$ is an upper bound on the forward eccentricity of v;
- $L^B(v)$ is a lower bound on the backward eccentricity of v;
- $U^B(v)$ is an upper bound on the backward eccentricity of v.

As soon as, for a node v, $L^F(v) = U^F(v)$ (resp. $L^B(v) = U^B(v)$), the forward (resp. backward) eccentricity of v is exactly computed: in this case, the algorithm might improve the values of D_L and R_U. The value of D is established as soon as $D_L \geq \max_{v \in V} U^F(v)$ or $D_L \geq \max_{v \in V} U^B(v)$, and the value of R is established as soon as $R_U \leq \max_{v \in V'} L^F(v)$. This is because if $D_L \geq \max_{v \in V} U^F(v)$ or $D_L \geq \max_{v \in V} U^B(v)$, then this lower bound cannot be improved anymore, since the forward eccentricity of each other node is smaller than D_L. Symmetrically, when $R_U \leq \min_{v \in V'} L^F(v)$, this upper bound cannot be improved anymore, and we can conclude that it is the actual value of the radius. Note that these inequalities are satisfied when all the eccentricities are known: since we ensure that at each iteration a forward or a backward eccentricity is exactly computed, we need at most $\mathcal{O}(n)$ iterations, so that the worst-case running time is $\mathcal{O}(mn)$ as in the naive algorithm.

The computation of new lower bounds is based on performing a BFS from a node s and bounding the eccentricity of a visited node v with $\operatorname{dist}(v, s)$ or $\operatorname{dist}(s, v)$ (the techniques are named StepForward and StepBackward, depending on the direction of the BFS). The upper bound techniques are a bit more complicated: they choose a pivot node for each strongly connected component, they bound the eccentricities of pivot nodes, and they propagate these bounds within each strongly connected component. The hardest part is bounding the eccentricities of pivot nodes: the simplest technique, AllCCUpperBound, uses a dynamic programming approach, based on the topological ordering of the SCCs. This technique is not used on its own, but it is a significant part of SingleCCUpperBound, a more sophisticated technique, which is more time consuming and provides better bounds. In particular, this technique performs a further forward and backward BFS from a given pivot q, allowing to improve the previous bounds when analyzing nodes reachable "by passing through q", while all other nodes are processed using the AllCCUpperBound technique. In the following three subsections, we describe each technique and we prove their correctness. In Sections 4.4.1 to 4.4.3, we define precisely these techniques, and in Section 4.4.4 we prove bounds on the running time of each technique, in Section 4.4.5 we show how these techniques apply to the special cases of strongly connected digraphs and of undirected graphs, and, finally, in Section 4.4.6 we discuss how to select the technique to use at each step.

4.4.1 StepForward and StepBackward

This technique performs a forward BFS from a "cleverly chosen" node s, setting $U^F(s) = L^F(s) = \operatorname{ecc}^F(s)$ and, for each visited node v, setting $L^B(v) = \max(L^B(v), \operatorname{dist}(s, v))$ (StepForward). A similar technique can be applied by performing a backward BFS (StepBackward). Note that, since the algorithm starts by running the SumSweepHeuristic, these bounds can also be computed during the BFSs performed by the heuristic. For example, after the aforementioned SumSweepHeuristic heuristic performed on the graph in Figure 4.1, the algorithm has already obtained the bounds in Table 4.1.

4.4.2 AllCCUpperBound

In order to compute upper bounds on the eccentricity of all nodes, we have to use more complicated techniques, based on the strong component graph (see Chapter 2). In particular, the AllCCUpperBound technique chooses a "pivot node" p_i for each SCC C_i of G and bounds the eccentricities of these nodes. Finally, it propagates these bounds by making use of the following inequalities, which are a simple consequence of the triangular inequality:

$$\operatorname{ecc}^F(v) \leq \operatorname{dist}(v, p_i) + \operatorname{ecc}^F(p_i) \tag{4.1}$$

Table 4.1. Bounds obtained after the initial SumSweepHeuristic, with $k = 4$. The sum of nodes whose eccentricity has already been computed exactly is set to -1 (in order to avoid these nodes to be chosen in subsequent BFSs).

Node	L^F	L^B	U^F	U^B	S^F	S^B
0	5	2	∞	∞	5	3
1	4	2	4	∞	-1	2
2	6	1	6	∞	-1	1
3	0	2	∞	∞	0	3
4	0	2	∞	∞	0	3
5	3	3	∞	∞	3	5
6	2	4	∞	∞	2	8
7	1	5	∞	∞	1	11
8	3	6	3	6	-1	-1
9	0	3	∞	∞	0	7
10	0	4	∞	∞	0	8
11	0	3	∞	∞	0	5

$$\mathrm{ecc}^B(v) \leq \mathrm{dist}(p_i, v) + \mathrm{ecc}^B(p_i), \tag{4.2}$$

where v belongs to the SCC C_i having pivot p_i. In order to apply these inequalities, we need to compute upper bounds on the forward and backward eccentricity of each pivot in the graph ($\mathrm{ecc}^F(p_i)$ and $\mathrm{ecc}^B(p_i)$): before explaining this in full detail, we show the main ideas of these bounds through an example, based on the graph in Figure 4.1.

Suppose we have chosen the pivots in Table 4.2 (actually this is the choice performed by the algorithm after the execution of the initial SumSweepHeuristic): we start to bound forward eccentricities in reverse topological order, that is, p_5, p_4, p_3, p_2, and p_1.

- Since no edge exits the SCC C_5, we set $U^F(p_5) = U^F(9) = \mathrm{ecc}^F_{scc}(9) = \mathrm{dist}(9, 10) = 1$, where $\mathrm{ecc}^F_{scc}(9)$ denotes the eccentricity of 9 if restricted to C_5.

- In order to bound the forward eccentricity of $p_4 = 5$, we observe that either the longest path stays in C_4, having length $\mathrm{ecc}^F_{scc}(5)$, or it passes through the SCC C_5, reachable in one step from C_4 through edge $(C_4, C_5) \in \mathcal{E}$, which corresponds (for example) to the edge $(5, 9) \in E$, according to Figure 4.1. We bound the eccentricity of 5 with the maximum between $\mathrm{ecc}^F_{scc}(5)$ and the length of a path from 5 to 9 passing through edge $(5, 9)$, plus $U^F(9)$ (this latter bound has already been computed thanks to the topological order). We obtain $U^F(5) = \max(\mathrm{ecc}^F_{scc}(5), \mathrm{dist}(5, 5) + 1 + \mathrm{dist}(9, 9) + U^F(9)) = \max(3, 2) = 3$.

- $U^F(11) = \mathrm{ecc}^F_{scc}(11) = 0$.

- There are two outgoing edges from C_2: $(C_2, C_3) \in \mathcal{E}$, which corresponds (for example) to $(3, 11) \in E$, and $(C_2, C_5) \in \mathcal{E}$, which corresponds (for example) to $(4, 9) \in E$. We bound $U^F(3)$ by considering the maximum among these possibilities: $U^F(3) = \max(\mathrm{ecc}^F_{scc}(3), \mathrm{dist}(3, 3)+1+\mathrm{dist}(11, 11)+\mathrm{ecc}^F(11), \mathrm{dist}(3, 4)+1+\mathrm{dist}(9, 9)+\mathrm{ecc}^F(9)) = \max(1, 1, 3) = 3$.

- Finally, $U^F(0) = \max(\mathrm{ecc}^F_{scc}(0), \mathrm{dist}(0, 1) + 1 + \mathrm{dist}(4, 3) + U^F(3), \mathrm{dist}(0, 1) + 1 + \mathrm{dist}(5, 5) + U^F(5)) = \max(2, 6, 5) = 6$.

The backward eccentricities are bounded similarly, considering SCCs in topological order. The forward and backward bounds computed in this way are summarized in Table 4.2.

Finally, we extend these bounds using inequalities (4.1) and (4.2): for instance, $U^F(1) = \mathrm{dist}(1, 0) + U^F(0) = 2 + 6 = 8$.

After showing the main ideas through this example, we may now formalize this intuition.

Table 4.2. The bounds computed on the pivots of the different SCCs.

Pivot	SCC	U^F	U^B
$p_1 = 0$	C_1	6	2
$p_2 = 3$	C_2	3	5
$p_3 = 11$	C_3	0	6
$p_4 = 5$	C_4	3	4
$p_5 = 9$	C_5	1	7

Lemma 4.1. *Given a SCC C_i with corresponding pivot p_i, for any j such that $(C_i, C_j) \in \mathcal{E}$ and for each edge $e_{ij} = (v_{ij}, w_{ij})$ from C_i to C_j, the following formulas hold:*

$$\mathrm{ecc}^F(p_i) \leq \max(\mathrm{ecc}^F_{scc}(p_i), \max_{(C_i,C_j)\in\mathcal{E}}(\mathrm{dist}(p_i, v_{ij}) + 1 + \mathrm{dist}(w_{ij}, p_j) + \mathrm{ecc}^F(p_j))) \tag{4.3}$$

$$\mathrm{ecc}^B(p_i) \leq \max(\mathrm{ecc}^B_{scc}(p_i), \max_{(C_j,C_i)\in\mathcal{E}}(\mathrm{dist}(w_{ji}, p_i) + 1 + \mathrm{dist}(p_j, v_{ji}) + \mathrm{ecc}^B(p_j))) \tag{4.4}$$

Proof. We prove the first formula, since the second is symmetric. Let t be one of the farthest nodes from p_i, so that $\mathrm{ecc}^F(p_i) = \mathrm{dist}(p_i, t)$. If $t \in C_i$, $\mathrm{ecc}^F(p_i) = \mathrm{dist}(p_i, t) = \mathrm{ecc}^F_{scc}(p_i)$ and the first formula holds. Otherwise, the shortest path from p_i to t passes through a component C_j such that $(C_i, C_j) \in \mathcal{E}$. Then, $\mathrm{ecc}^F(p_i) = \mathrm{dist}(p_i, t) \leq \mathrm{dist}(p_i, v_{ij}) + 1 + \mathrm{dist}(w_{ij}, p_j) + \mathrm{dist}(p_j, t) \leq \mathrm{dist}(p_i, v_{ij}) + 1 + \mathrm{dist}(w_{ij}, p_j) + \mathrm{ecc}^F(p_j)$, and the first formula holds again. □

At this point, we may observe that the inequalities in Equations (4.3) and (4.4) can be "solved" recursively by analyzing strongly connected components with their corresponding pivot nodes in topological (resp. reverse topological) order, as we did in the previous example (the choice of the edges e_{ij} is performed at the beginning of the algorithm). The pseudo-code for the computation of the pivot upper bounds and for the update of the upper bounds for each node is shown in Algorithm 4.

Before running these procedures, we just need to perform a forward and a backward BFS starting from p_i and restricted to C_i, for each i. This way, we compute the following values:

1. $\mathrm{dist}(p_i, v), \mathrm{dist}(v, p_i)$ for each SCC C_i and each node $v \in C_i$;

2. $\mathrm{ecc}^F_{scc}(p_i), \mathrm{ecc}^B_{scc}(p_i)$ for each pivot node p_i.

4.4.3 SingleCCUpperBound

In order to further improve the upper bounds defined by Lemma 4.1, we introduce the SingleCCUpperBound technique. It requires a further forward and a further backward BFS from a pivot node q, that we call "main pivot": the technique works for any pivot, but a suitable choice provides better bounds. As in the previous section, we first provide an example, then we explain the technique and in Section 4.4.6 we provide more details about the choice of the main pivot.

Our example deals again with the graph in Figure 4.1, choosing as main pivot node $q = 5$. We perform a forward and backward BFS from node 5, computing exactly its forward and backward eccentricities, thus setting $U^F(5) = 3$, and $U^B(5) = 3$. We now analyze how to improve the previous forward bounds (the backward case is completely analogous). First of all, we use the previous technique to bound the forward eccentricities of all pivot nodes not visited in the backward BFS from 5, namely $U^F(3) = 5, U^F(9) = 1, U^F(11) = 0$. Then, we want to upper bound the eccentricity of 0, reached in the backward BFS from 5. We observe that $\mathrm{ecc}^F(0) = \mathrm{dist}(0, t)$, for some node t: if t is reachable from the main pivot 5, $\mathrm{dist}(0, t) \leq \mathrm{dist}(0, 5) + \mathrm{dist}(5, t) \leq \mathrm{dist}(0, 5) + \mathrm{ecc}^F(5)$. Otherwise, t is reachable from 0 by remaining in the graph G' obtained from G by removing all nodes in $R^F(5)$, that is, by removing all nodes visited in the forward BFS. We then set $U^F(0) = \max(\mathrm{dist}(0, 5) +$

Algorithm 4: Computing upper bounds for all nodes. Note that the graph $\mathcal{G}$ can be precomputed in linear time as well as a topological order at the beginning of the ExactSumSweep algorithm.

```
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the graph of the SCCs in $G$.
$U_P^F \leftarrow$ computePivotBoundsF ($\mathcal{G}$);
$U_P^B \leftarrow$ computePivotBoundsB ($\mathcal{G}$);
for $i = 0$ to $|\mathcal{V}|$ do
    for $v \in C_i$ do
        $U^F(v) \leftarrow \min(U^F(v), \mathrm{dist}(v, p_i) + U_P^F(p_i))$;
        $U^B(v) \leftarrow \min(U^B(v), \mathrm{dist}(p_i, v) + U_P^B(p_i))$;
    end
end
Procedure computePivotBoundsF($\mathcal{G}$)
    for $i = |\mathcal{V}|$ to 1 do
        $U_P^F(p_i) \leftarrow \min(U^F(p_i), \max(\mathrm{ecc}_{scc}^F(p_i), \max_{(C_i,C_j)\in\mathcal{E}}(\mathrm{dist}(p_i, v_{ij}) + 1 +$
          $\mathrm{dist}(w_{ij}, p_j) + U_P^F(p_j))))$;
    end
    return $U_P^F$;
Procedure computePivotBoundsB($\mathcal{G}$)
    for $j = 1$ to $|\mathcal{V}|$ do
        $U_P^B(p_j) \leftarrow \min(U^B(p_j), \max(\mathrm{ecc}_{scc}^B(p_j), \max_{(C_i,C_j)\in\mathcal{E}}(\mathrm{dist}(w_{ij}, p_j) + 1 +$
          $\mathrm{dist}(p_i, v_{ij}) + U_P^B(p_i))))$;
    end
    return $U_P^B$;
```

$U^F(5), U_{G'}^F(0))$, where $U_{G'}^F$ is the bound on the forward eccentricity of 0 obtained by running Procedure computePivotBoundsF($\mathcal{G}'$) in Algorithm 4, and $\mathcal{G}'$ is the strong component digraph of G' (note that $\mathcal{G}'$ can be computed without computing explicitly G'). We finally obtain $U^F(0) = \max(\mathrm{dist}(0,5) + U^F(5), U_{G'}^F(0)) = \max(5,3) = 5$. Forward and backward results are summarized in Table 4.3. Note that better forward bounds have been found for pivot 0, and better backward bounds have been found for pivot 9.

Table 4.3. The bounds computed on the pivots of the different SCCs by the SingleCCUpperBound technique.

Pivot	SCC	U^F	U^B
0	C_1	5	2
3	C_2	3	5
11	C_3	0	6
5	C_4	3	3
9	C_5	1	4

More formally, the SingleCCUpperBound technique is based on the following lemma.

Lemma 4.2. *Let p_i be a pivot, q be the main pivot, and suppose $p_i \in R^B(q)$. If G' is the subgraph induced on G by removing $R^F(q)$, the following inequality holds:*

$$\mathrm{ecc}^F(p_i) \leq \max(\mathrm{dist}(p_i, q) + \mathrm{ecc}^F(q), \mathrm{ecc}_{G'}^F(p_i))$$

Proof. Let t be the farthest node from p_i: if $t \in R^F(q)$, $\mathrm{ecc}^F(p_i) = \mathrm{dist}(p_i, t) \leq \mathrm{dist}(p_i, q) + \mathrm{dist}(q, t) \leq \mathrm{dist}(p_i, q) + \mathrm{ecc}^F(q)$. Otherwise, all paths from p_i to t are paths in G', so $\mathrm{ecc}^F(p_i) = \mathrm{dist}(p_i, t) = \mathrm{dist}_{G'}(p_i, t) \leq \mathrm{ecc}_{G'}^F(p_i)$, where by $\mathrm{dist}_{G'}$ we mean distances in the graph G'. In both cases, the lemma holds. □

Algorithm 5: Computing better upper bounds for some nodes.

```
Procedure computeImprovedPivotBoundsF($\mathcal{G}$)
    Let $q$ be the main pivot.
    Let $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ be the subgraph of $\mathcal{G}$ induced by nodes not reachable from the
      SCC of $q$.
    $U^F_{G'} \leftarrow$ computePivotBoundsF($\mathcal{G}'$);
    $U^F_P \leftarrow$ computePivotBoundsF($\mathcal{G}$);
    for $p_i \in R^B(q)$ do
        $U^F_P(p_i) \leftarrow \min(U^F(p_i), \max(\mathrm{dist}(p_i, q) + \mathrm{ecc}^F(q), U^F_{G'}(p_i)))$;
    end
    return $U^F_P$;
Procedure computeImprovedPivotBoundsB($\mathcal{G}$)
    Let $q$ be the main pivot.
    Let $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ be the subgraph of $\mathcal{G}$ induced by nodes not reachable from the
      SCC of $q$.
    $U^F_{G'} \leftarrow$ computePivotBoundsB($\mathcal{G}'$);
    $U^F_P \leftarrow$ computePivotBoundsB($\mathcal{G}$);
    for $p_i \in R^F(q)$ do
        $U^B_P(p_i) \leftarrow \min(U^B(p_i), \max(\mathrm{dist}(p_i, q) + \mathrm{ecc}^B(q), U^B_{G'}(p_i)))$;
    end
    return $U^B_P$;
```

The pseudo-code for the computation of the improved forward bounds is provided by Algorithm 5: it is enough to replace functions `computePivotBoundsF` and `computePivotBoundsB` in Algorithm 4 with the improved versions, named `computeImprovedPivotBoundsF` and `computeImprovedPivotBoundsB`. Note that, in order to compute forward and backward bounds, we need to perform a forward and a backward BFS from the main pivot, and a forward and backward BFS inside each SCC not containing the main pivot (for the SCC of the main pivot, the results of the first two BFSs can be used).

4.4.4 Running Time Analysis

In all previous papers dealing with bound-refinement methods [62, 63, 61, 154, 155, 116], the efficiency of an algorithm was defined in terms of the total number of BFSs needed before the values of D and R are found. However, if the graph is not strongly connected, the time needed to perform a BFS highly depends on the starting node: for instance, a forward BFS from a node with outdegree 0 takes very little time. As a consequence, we consider as baseline the time needed to perform a BFS of the corresponding undirected graph, that is, the graph obtained by converting all directed edges into undirected edges. Since we deal with weakly connected graphs, this latter graph is connected, and the running time of a BFS on this graph does not depend on the starting node. The cost of the StepForward and StepBackward techniques is at most the cost of a BFS, because these techniques visit at most the same number of edges as a BFS of the corresponding undirected graph, and the operations performed when an edge is visited are the same. We then consider the cost of these operations as 1 BFS. For the SingleCCUpperBound technique, we need to perform the following operations:

1. a forward and a backward BFS from q and a BFS inside each SCC not containing q;

2. Algorithm 5, which does the following operations:

    (a) compute $\mathcal{G}'$;

Algorithm 6: Computing upper bounds for all nodes.

```
for $v \in V$ do
    $U^F(v) \leftarrow \min(U^F(v), \mathrm{dist}(v,p) + U^F(p))$;
    $U^B(v) \leftarrow \min(U^B(v), \mathrm{dist}(p,v) + U^B(p))$;
end
```

(b) perform twice `computePivotBoundsF` and `computePivotBoundsB` in Algorithm 4;

(c) improve the bounds for each pivot node.

3. update all bounds using the inequalities (4.1), and (4.2), as in Algorithm 4.

It is clear that steps 2(a), 2(b) and 2(c) can be performed in time $\mathcal{O}(|\mathcal{G}|)$, and that step 3 can be performed in time $\mathcal{O}(|V|)$. Step 1 performs, for each visited edge, the same operations as a standard BFS, and each edge is visited at most three times in the whole operation. Then, we consider the cost of running the SingleCCUpperBound technique as the cost of three BFSs, and we ignore operations that are performed in $\mathcal{O}(|V| + |\mathcal{G}|)$ time, similarly to what has been done in previous papers for operations needing $\mathcal{O}(|V|)$ time [62, 63, 61, 116, 155, 154]. This choice is justifiable also because, on average, in the dataset of our experiments, $|V| \approx 0.20 \cdot |E|$ and $|\mathcal{G}| \approx 0.17 \cdot |E|$.

4.4.5 Particular Cases

The Strongly Connected Case. In the strongly connected case, the aforementioned upper bound techniques AllCCUpperBound and SingleCCUpperBound collapse to a simpler technique, that performs a forward and backward BFS from the unique pivot node p, and bounds the eccentricity of any other node with $\mathrm{ecc}^F(v) \leq \mathrm{dist}(v,p) + \mathrm{ecc}^F(p)$ and $\mathrm{ecc}^B(v) \leq \mathrm{dist}(p,v) + \mathrm{ecc}^B(p)$. This technique costs two BFSs, differently from the SingleCCUpperBound technique that costs three BFSs. More specifically, Algorithm 4 becomes Algorithm 6.

The Undirected Case. In the undirected case, since we can deal with each connected component separately, again the two techniques are the same, and the cost reduces to one BFS, since the forward and backward BFSs coincide. Furthermore, we might improve the previous bounds by analyzing separately the first branch of the BFS tree and the other branches, as shown in the following lemma.

Lemma 4.3. *Suppose we have performed a BFS from p, and we have obtained a tree T; let p' be the first node in T having more than one child. Let Φ be the set of nodes on the (only) path from p to p', let Ψ be the set of nodes in the subtree of T rooted at the first child of p', and let h be the maximum distance from p' to a node outside Ψ. Then, for each $v \in V$, $\varepsilon(v) \leq U_p(v)$, where*

$$U_p(v) := \begin{cases} \max(\mathrm{dist}(p,v), \mathrm{ecc}(p) - \mathrm{dist}(p,v)) & v \in \Phi \\ \max(\mathrm{dist}(p',v) + \mathrm{ecc}(p') - 2, \mathrm{dist}(p',v) + h) & v \in \Psi \\ \mathrm{dist}(p',v) + \mathrm{ecc}(p') & \textit{otherwise} \end{cases}$$

Proof. If $v \in \Phi$ or $v \notin \Phi \cup \Psi$, the conclusion follows easily by the triangle inequality. If $v \in \Psi$, let t be the farthest node from v: if $t \notin \Psi$, then $\mathrm{dist}(t,v) \leq \mathrm{dist}(t,p') + \mathrm{dist}(p',v) \leq h + \mathrm{dist}(p',v)$. If $t \in \Psi$ and r is the root of the subtree of T consisting of nodes in Ψ, $\mathrm{dist}(v,t) \leq \mathrm{dist}(v,r) + \mathrm{dist}(r,t) = \mathrm{dist}(v,p') + \mathrm{dist}(p',t) - 2 \leq \mathrm{dist}(v,p') + \varepsilon(p') - 2$. □

In order to apply this lemma, after performing a BFS of an undirected graph, instead of bounding $\mathrm{ecc}(v)$ with $\mathrm{dist}(v,p) + \mathrm{ecc}(p)$, we may bound $\mathrm{ecc}(v)$ with $U_p(v)$ (which is smaller or

Algorithm 7: Computing upper bounds for all nodes.

```
for $v \in V$ do
    $U(v) \leftarrow U_p(v)$;
end
```

equal than $\mathrm{dist}(v, p) + \mathrm{ecc}(p)$). This bound can be used everytime a BFS is performed, that is, not only during the SingleCCUpperBound technique, but also during a StepForward or a StepBackward. The pseudo-code of this procedure is provided in Algorithm 7, which replaces Algorithm 4.

4.4.6 Choosing the Technique to Use

In the previous subsections, several bound techniques have been defined. Here, we explain how to put them together to be effective: in all previous papers, different techniques were alternated according to a fixed schema [154, 155]. In this paper, we provide a different approach: we run at each step a heuristic, that proposes a technique. The choice performed by this heuristic is based on the following definition, that quantifies how "close" we are to the solution.

Definition 4.4. Let V_U be the smallest set among $\{v \in V : U^F(v) > D_L\}$, and $\{v \in V : U^B(v) > D_L\}$, and let V_L be $\{v \in V' : L^F(v) < R_U\}$. A node v is *open* if $v \in V_U \cup V_L$.

At any point of the algorithm, the values of D_L and R_U can only be improved by the eccentricities of open nodes. For this reason, we consider $N = |V_U| + |V_L|$ as a measure of the distance from the final solution. Thanks to this definition, we may now state how we are going to use the bounding techniques. In particular, this can be done as follows (see Section 4.3, for the definition of $S^F(v)$ and $S^B(v)$).

- StepForward from a node maximizing U^F (maximizing S^F is used to break ties);
- StepBackward from a node maximizing U^B (maximizing S^B is used to break ties);
- StepForward from a node in V' minimizing L^F (minimizing S^F is used to break ties);
- StepBackward from a node maximizing S^B (maximizing U^B is used to break ties);
- SingleCCUpperBound: the pivot of a SCC is chosen by minimizing $L^F(v) + L^B(v)$ among all nodes whose exact eccentricity has not been determined, yet; the main pivot is chosen as the pivot of the component C having more open nodes.

The last problem to address is which of these techniques should be chosen at any step, filling the gap in Algorithm 3.

Definition 4.5. The utility U of a step is the difference between the value of N before the step and after the step.

In order to speed up the computation, we want to perform steps with high utility. For this reason, for each technique, we keep an "expected utility" value U_E and at each step we choose the technique with the biggest expected utility. After a technique is applied, U_E is updated as follows: the expected utility of the technique used is set to U, while the expected utility of all other techniques is increased by $\frac{2}{i}$, where i is the number of BFSs already performed. This is done in order to alternate different techniques at the beginning, and to focus on a single technique when the best one becomes clear (even if, in the long run, every technique is applied).

4.5 Experimental Results

This section experimentally shows the effectiveness of the aforementioned techniques, by testing them on several real-world networks taken from the well-known datasets SNAP [108] and KONECT [103], which cover a large set of network types. In Section 4.5.1, we show the effectiveness of the SumSweepHeuristic compared to other similar heuristics. In Section 4.5.2, we compare the ExactSumSweep algorithm with existing algorithms in the particular cases of undirected and directed strongly connected graphs. Then, we apply our algorithm to compute for the first time the diameter and radius of directed, not strongly connected graphs: even in this case, our algorithm has very good performances, even better than the strongly connected case (however, no comparison is possible, since this is the first such algorithm). More detailed results are available in Section 4.6.

4.5.1 Lower Bounding the Diameter

In this section, we compare the SumSweepHeuristic with the current most effective heuristics to compute lower bounds on the diameter [114, 63, 61], whose effectiveness has already been shown in [114, 63, 61]. In the case of undirected graphs, we have compared the following heuristics:

k-SumSweepHeuristic: performs a SumSweepHeuristic from a random node, stopping after k BFSs;

4-Sampling: returns the maximum eccentricity among four random-chosen nodes;

4-Sweep: the technique explained in [61];

2×2-Sweep performs twice a 2-Sweep [114] starting from two random nodes.

In the case of directed graphs, we have compared the following heuristics:

k-SumSweepHeuristic: performs a SumSweepHeuristic from a random node, stopping after k BFSs;

4-Sampling: returns the maximum eccentricity among four random-chosen nodes;

2-dSweep: the technique explained in [63].

All the competitors above perform four BFSs, so they should be compared to the 4-SumSweepHeuristic in order to obtain a fair comparison. We have run each heuristic ten times[1] for each graph, and we have considered the mean ratio r between the value returned and the diameter, among all ten experiments. In Tables 4.4 and 4.5 we have reported the average r among all the graphs in the dataset, with the corresponding standard error. In 4.6.2, the average r for each graph is provided.

We observe that the lower bounds provided by the 4-Sweep and 2-dSweep heuristics are usually tight, drastically outperforming the simple approach based on sampling, namely 4-Sampling. Tables 4.4 and 4.5 shows that the SumSweepHeuristic approaches are even improving these lower bounds. Moreover, it allows us to further improve the bounds found by performing some more BFSs. This is very useful on directed graphs, while on undirected graphs the bounds in the 4-SumSweepHeuristic are so good that they offer very little room for improvement.

[1] Very low variance has been observed: even increasing the number of experiments does not change the results.

Table 4.4. The average ratio r between the lower bound on the diameter returned by each heuristic and the diameter, in undirected graphs.

Method	r	Std Error
4-SumSweepHeuristic	99.9830 %	0.0315 %
3-SumSweepHeuristic	99.9671 %	0.0582 %
4-Sweep	99.9353 %	0.1194 %
2×2-Sweep	99.9295 %	0.1095 %
4-Sampling	76.9842 %	5.2841 %

Table 4.5. The average ratio r between the lower bound on the diameter returned by each heuristic and the diameter, in directed graphs.

Method	r	Std rror
8-SumSweepHeuristic	97.2835 %	4.9030 %
7-SumSweepHeuristic	97.2733 %	4.9010 %
6-SumSweepHeuristic	97.2733 %	4.9010 %
5-SumSweepHeuristic	96.8308 %	5.4324 %
4-SumSweepHeuristic	96.6091 %	5.6564 %
3-SumSweepHeuristic	95.5399 %	6.1901 %
2-dSweep	94.7607 %	6.5877 %
4RandomSamples	61.2688 %	15.1797 %

4.5.2 Computing the Radius and the Diameter

The following set of experiments aims to show that the ExactSumSweep algorithm improves the time bounds, the robustness, and the generality of all the existing methods, since they are outperformed for both radius and diameter computation, both in the directed and in the undirected case. Note that in the case of directed weakly connected graphs, there are no competitors (except the textbook algorithm), since ExactSumSweep is the first algorithm able to deal with this case. Since any (weakly) connected component can be analyzed separately, we have restricted our attention to the biggest one, which usually contains most of the nodes in the graph (see Tables 4.7 and 4.8).

Undirected Graphs. In the undirected case, we compared our method with the state of the art: the iFub algorithm for the diameter and the BoundingDiameters algorithm both for the radius and for the diameter.

Indeed, this latter algorithm, used in [154] just to compute the diameter, can be easily adjusted to also compute the radius [155], using the same node selection strategy and updating rules for the eccentricity bounds. In particular, it bounds the eccentricity of nodes similarly to our method, by using the fact that, after a visit from a node s is performed, $\mathrm{dist}(s,t) \leq \mathrm{ecc}(t) \leq \mathrm{dist}(s,t) + \mathrm{ecc}(s)$. It does not perform the initial SumSweepHeuristic and simply alternates between nodes v with the largest eccentricity upper bound and the smallest eccentricity lower bound.

For the diameter computation, we compared ExactSumSweep not only with BoundingDiameters, but also with two variations of iFub: iFubHd, starting from the node of highest degree, and iFub4S, starting by performing a 4-Sweep and choosing the central node of the second iteration (see [61] for more details).

The results of the comparison are summarized in Table 4.6: for each method and for each graph in our dataset, we have computed the *performance ratio*, that is the percentage of the number of visits performed by the algorithm with respect to the number of nodes of the network (i.e., the number of visits in the worstcase). In Table 4.6, we report the average of these values on the whole dataset, together with the corresponding standard error (more detailed results are available in Section 4.6).

Table 4.6. The average performance ratio p, i.e., percentage of the number of BFSs used by the different methods, with respect to the number of nodes (number of visits using the textbook algorithm).

Undirected Connected Graphs

Method	p	Std Error
ExactSumSweep	0.0572 %	0.0567 %
BoundingDiameters	0.2097 %	0.2509 %
iFub4S	1.6937 %	2.5603 %
iFubHd	3.2550 %	5.4185 %

(a) Diameter

Method	p	Std Error
ExactSumSweep	0.0279 %	0.0276 %
BoundingDiameters	0.0427 %	0.0486 %

(b) Radius

Directed Strongly Connected Graphs

Method	p	Std Error
ExactSumSweep	0.1990 %	0.2245 %
diFub2In	0.8429 %	1.1937 %
diFub2Out	0.8458 %	1.2026 %
diFubHdOut	1.7454 %	2.6369 %
diFubHdIn	2.3249 %	3.3871 %

(c) Diameter

Method	p	Std Error
ExactSumSweep	0.1935 %	0.2203 %
HR	6.0136 %	8.0915 %

(d) Radius

Directed Weakly Connected Graphs

Method	p	Std Error
ExactSumSweep	0.1220 %	0.0969 %

(e) Diameter

Method	p	Std Error
ExactSumSweep	0.1367 %	0.1045 %

(f) Radius

In the diameter computation, the improvement is shown in Table 4.6 (a). The new method is not only better than the previous ones on average, but it is even much more robust: the computation of the diameter for ExactSumSweep always ends in less than 180 BFSs, while the old methods need up to 1200 BFSs, as shown by Table 4.11.

In the radius computation, the ExactSumSweep method is slightly more effective than the BoundingDiameters algorithm on average, as shown in Table 4.6 (b). Again, we outline that the new method is much more robust: in our dataset, it never needs more than 18 BFSs, while the BoundingDiameters algorithm needs at most 156 BFSs. Moreover, in all the graphs in which the BoundingDiameters algorithm beats the ExactSumSweep algorithm, this happens always because of just one BFS. On the converse, when the ExactSumSweep algorithm beats the BoundingDiameters algorithm, the difference between the number of visits required can be much higher than 1: see Table 4.11 for the detailed results.

Directed Strongly Connected Graphs. For the computation of the diameter of directed, strongly connected graphs, the best previous algorithms are four variations of the diFub method [63]:

diFubHdIn: starts from the node with highest in-degree;
diFubHdOut: starts from the node with highest out-degree;
diFub2In: starts from the central node of a 2-Sweep performed from the node with highest in-degree;
diFub2Out: starts from the central node of a 2-Sweep performed from the node with highest out-degree.

The results of the comparison with the new algorithm are shown in Table 4.6 (c).

For the radius computation, the only efficient known method is explained in [116], which we refer to as HR. Basically, it works as follows: given the farthest pair of nodes s and t found by the directed version of 2-Sweep, order the nodes v according to $g(v) = \max\{\mathrm{dist}(v,s), \mathrm{dist}(v,t)\}$; scan the eccentricities of the nodes in this order and stop when the next node w has a value of $g(w)$ which is greater than the minimum eccentricity found. The results are shown in Table 4.6 (d).

In the diameter computation, the best previous method is diFub2In: the new ExactSumSweep method performs more than 4 times better. We note again the robustness: the maximum number of BFSs is 59, against the maximum number for diFub2Out which is 482 (note that the maximum for the best competitor of ExactSumSweep, diFub2In, is 510).

In the radius computation, the ExactSumSweep algorithm performs about 31 times better than the old method. We also remark that the robustness of ExactSumSweep applies also to the directed case: at most 36 BFSs are needed to find the radius of any graph of our dataset.

The Directed General Case. Finally, we have analyzed the performances of ExactSumSweep in computing diameter and radius of directed, not strongly connected graphs. Due to the lack of similar algorithms, it is possible to compare the new method only with the textbook one, namely, performing a BFS from each node. The performances of the new method are on average about 1000 times better than the textbook algorithm, allowing us to compute the diameter and radius of several real-world networks for the first time. Moreover, it is worth observing that in the case of weakly connected directed graphs, ExactSumSweep seems to perform even better with respect to strongly connected graphs, if the performance ratio is considered. However, this is mainly not due to a smaller number of BFSs needed, but to the increased number of nodes, that damage the performance in the worst-case (the biggest SCC, on average, has about half the nodes of the whole graph).

Overall, we conclude that the new method is more general: it is the only method which is able to deal with both directed strongly connected and undirected graphs, both for the radius and the diameter computation, with very small variations. Moreover, it is the only existing algorithm able to deal with weakly connected directed graphs, apart from the textbook one. Despite its generality, it is also faster than every existing algorithm, both on average running time and on robustness.

4.6 Detailed Experimental Results

4.6.1 Dataset

The following tables provide the graphs included in our dataset, showing for each network the number of nodes (n) and the number of edges (m). Moreover for each undirected network, we report the number of nodes (n_{cc}), the number of edges (m_{cc}), the diameter (D), and the radius (R) of the biggest connected component. Furthermore, for each directed network, we report the number of nodes (n_{wcc}), the number of edges (m_{wcc}), the diameter (D), and the radius (R) of the largest weakly connected component together with the number of nodes (n_{scc}), the number of edges (m_{scc}), the diameter (D_{scc}), and the radius (R_{scc}) of the biggest strongly connected component.

Table 4.7. Undirected graphs: n and m indicate respectively the number of nodes and the number of edges, D and R indicate diameter and radius, and the subscript cc refers to the largest connected component.

Network	n	m	n_{cc}	m_{cc}	D	R
as20000102	6474	12572	6474	12572	5	9
CA-AstroPh	18772	198050	17903	196972	8	14
CA-CondMat	23133	93439	21363	91286	8	15
ca-GrQc	5241	14484	4158	13422	9	17
ca-HepPh	12006	118489	11204	117619	7	13
ca-HepTh	9875	25973	8638	24806	10	18
com-amazon.all.cmty	134386	99433	7011	8955	20	39
com-amazon.ungraph	334863	925872	334863	925872	24	47
com-dblp.ungraph	317080	1049866	317080	1049866	12	23
com-lj.all.cmty	477998	530872	303526	427701	16	32
com-youtube.ungraph	1134890	2987624	1134890	2987624	12	24
email-Enron	36692	183831	33696	180811	7	13
facebook_combined	4039	88234	4039	88234	4	8
flickrEdges	105938	2316948	105722	2316668	5	9
gowalla_edges	196591	950327	196591	950327	8	16
loc-brightkite_edges	58228	214078	56739	212945	9	18
oregon1_010519	11051	22724	11051	22724	6	11
oregon1_010526	11174	23409	11174	23409	5	10
oregon2_010519	11375	32287	11375	32287	5	9
oregon2_010526	11461	32730	11461	32730	5	9
orkut-links	3072441	117185083	3072441	117185083	5	10
p2pGnutella09	8114	26013	8104	26008	6	10
roadNet-CA	1965206	2766607	1957027	2760388	494	865
roadNet-PA	1088092	1541898	1087562	1541514	402	794
roadNet-TX	1379917	1921660	1351137	1879201	540	1064
soc-pokec-relationships	1632803	44603928	1632803	44603928	14	7
youtube-u-growth	3223585	9375374	3216075	9369874	16	31

Table 4.8. Directed graphs: n and m indicate the number of nodes and the number of edges, respectively, D and R indicate diameter and radius, and subscripts wcc and scc refer respectively to the biggest weakly connected component and the biggest strongly connected component.

Network	n	m	n_{wcc}	m_{wcc}	n_{scc}	m_{scc}	D	R	D_{scc}	R_{scc}
amazon0302	262111	1234877	262111	1234877	241761	1131217	88	50	88	48
amazon0312	400727	3200440	400727	3200440	380167	3069889	53	28	52	26
amazon0505	410236	3356824	410236	3356824	390304	3255816	55	27	55	27
amazon0601	403394	3387388	403364	3387224	395234	3301092	54	29	52	25
as-caida2007	26475	106762	26475	106762	26475	106762	17	9	17	9
ca-AstroPh	18771	396100	17903	393944	17903	393944	14	8	14	8
ca-CondMat	23133	186878	21363	182572	21363	182572	15	8	15	8
ca-GrQc	5241	28968	4158	26844	4158	26844	17	9	17	9
ca-HepPh	12006	236978	11204	235238	11204	235238	13	7	13	7
ca-HepTh	9875	51946	8638	49612	8638	49612	18	10	18	10
cit-HepPh	34546	421534	34401	421441	12711	139965	49	12	49	15
cit-HepTh	27769	352768	27400	352504	7464	116252	37	12	35	13
cit-Patents	3774768	16518947	3764117	16511740	1	0	24	0	0	0
email-EuAll	265009	418956	224832	394400	34203	151132	11	5	10	5
flickr-growth	2302925	33140017	2173370	32948343	1605184	30338513	27	13	27	12
p2pGnutella04	10876	39994	10876	39994	4317	18742	26	15	25	15
p2pGnutella05	8846	31839	8842	31837	3234	13453	22	14	22	14
p2pGnutella06	8717	31525	8717	31525	3226	13589	21	13	19	12
p2pGnutella08	6301	20777	6299	20776	2068	9313	20	13	19	12
p2pGnutella09	8114	26013	8104	26008	2624	10776	20	14	19	13
p2pGnutella24	26518	65369	26498	65359	6352	22928	29	16	28	15
p2pGnutella25	22687	54705	22663	54693	5153	17695	22	14	21	13
p2pGnutella30	36682	88328	36646	88303	8490	31706	24	16	23	15
p2pGnutella31	62586	147892	62561	147878	14149	50916	31	20	30	19
soc-Epinions1	75879	508837	75877	508836	32223	443506	16	8	16	8
soc-sign-ep	131828	840799	119130	833390	41441	693507	16	8	16	7
soc-sign081106	77350	516575	77350	516575	26996	337351	15	7	15	7
soc-sign090216	81867	545671	81867	545671	27222	342747	15	7	15	7
soc-sign090221	82140	549202	82140	549202	27382	346652	15	7	15	7
soc-Slash0811	77360	828161	77360	828161	70355	818310	12	7	12	7
soc-Slash0902	82168	870161	82168	870161	71307	841201	13	7	13	7
trec-wt10g	1601787	8063026	1458316	7487449	470441	3012375	351	79	130	37
web-BerkStan	685230	7600595	654782	7499425	334857	4523232	694	283	679	249
web-Google	875713	5105039	855802	5066842	434818	3419124	51	27	51	24
web-NDame	325729	1469679	325729	1469679	53968	296228	93	44	93	44
web-Stanford	281903	2312497	255265	2234572	150532	1576314	580	134	210	97
wiki-Talk	2394385	5021410	2388953	5018445	111881	1477893	11	5	10	5
wiki-Vote	7115	103689	7066	103663	1300	39456	10	4	9	3
youtube-links	1138494	4942297	1134885	4938950	509245	4269142	23	13	20	10
zhishi-baidu	2141300	17632190	2107689	17607140	609905	8300678	39	17	39	17
zhishi-hudong	1984484	14682258	1962418	14672183	365558	4689296	31	19	31	19

4.6.2 The SumSweep Heuristic

Table 4.9. Efficiency of different lower bound techniques on undirected graphs.

Network	3-SS	4-SS	2×2-Sweep	4-Sweep	4-Sampling
as20000102	100.00%	100.00%	100.00%	100.00%	80.00%
CA-AstroPh	100.00%	100.00%	100.00%	100.00%	76.43%
CA-CondMat	100.00%	100.00%	100.00%	100.00%	74.00%
ca-GrQc	100.00%	100.00%	100.00%	100.00%	78.24%
ca-HepPh	100.00%	100.00%	100.00%	100.00%	79.23%
ca-HepTh	100.00%	100.00%	100.00%	100.00%	76.11%
com-amazon.all.cmty	100.00%	100.00%	100.00%	100.00%	77.18%
com-amazon.ungraph	100.00%	100.00%	100.00%	100.00%	75.74%
com-dblp.ungraph	100.00%	100.00%	100.00%	100.00%	71.74%
com-lj.all.cmty	100.00%	100.00%	100.00%	99.69%	70.31%
com-youtube.ungraph	100.00%	100.00%	100.00%	100.00%	65.42%
email-Enron	100.00%	100.00%	100.00%	100.00%	70.77%
facebook_combined	100.00%	100.00%	100.00%	100.00%	87.50%
flickrEdges	100.00%	100.00%	100.00%	100.00%	86.67%
gowalla_edges	100.00%	100.00%	100.00%	100.00%	71.88%
loc-brightkite_edges	100.00%	100.00%	100.00%	100.00%	67.78%
oregon1_010519	100.00%	100.00%	100.00%	100.00%	80.00%
oregon1_010526	100.00%	100.00%	100.00%	100.00%	76.00%
oregon2_010519	100.00%	100.00%	100.00%	100.00%	77.78%
oregon2_010526	100.00%	100.00%	100.00%	100.00%	76.67%
p2p-Gnutella09	100.00%	100.00%	100.00%	100.00%	85.00%
roadNet-CA	99.64%	99.88%	99.28%	99.76%	85.36%
roadNet-PA	99.67%	99.67%	99.67%	99.60%	90.43%
roadNet-TX	99.83%	100.00%	99.92%	99.28%	85.12%
soc-pokec-relationships	100.00%	100.00%	99.29%	100.00%	67.86%
youtube-u-growth	100.00%	100.00%	100.00%	100.00%	68.39%

Table 4.10. Efficiency of different lower bound techniques on directed graphs.

Network	3-SS	4-SS	5-SS	6-SS	7-SS	8-SS	2-dSweep	4-Sampling
amazon0302	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	79.43%
amazon0312	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	72.26%
amazon0505	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	64.36%
amazon0601	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	68.70%
as-caida20071105	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	88.82%
ca-AstroPh	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	75.00%
ca-CondMat	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	75.33%
ca-GrQc	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	76.47%
ca-HepPh	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	80.77%
ca-HepTh	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	77.22%
cit-HepPh	99.80%	100.00%	100.00%	100.00%	100.00%	100.00%	98.16%	71.02%
cit-HepTh	85.14%	86.76%	86.76%	86.76%	86.76%	86.76%	82.43%	72.97%
cit-Patents	67.08%	72.92%	72.92%	84.58%	84.58%	84.58%	59.58%	26.67%
email-EuAll	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	70.91%
flickr-growth	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	60.37%
p2pGnutella04	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	97.69%	73.08%
p2pGnutella05	94.55%	97.27%	97.27%	100.00%	100.00%	100.00%	97.73%	73.64%
p2pGnutella06	95.71%	100.00%	100.00%	100.00%	100.00%	100.00%	99.52%	50.95%
p2pGnutella08	88.00%	100.00%	100.00%	100.00%	100.00%	100.00%	95.50%	74.00%
p2pGnutella09	99.50%	100.00%	100.00%	100.00%	100.00%	100.00%	95.50%	86.50%
p2pGnutella24	94.48%	100.00%	100.00%	100.00%	100.00%	100.00%	93.79%	66.55%
p2pGnutella25	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	97.27%	32.27%
p2pGnutella30	95.83%	95.83%	95.83%	99.58%	99.58%	100.00%	95.42%	31.67%
p2pGnutella31	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	98.06%	48.06%
soc-Epinions1	98.13%	100.00%	100.00%	100.00%	100.00%	100.00%	92.50%	65.63%
soc-sign-epinions	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	90.63%	51.25%
Soc-sign081106	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	57.33%
Soc-sign090216	99.33%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	59.33%
Soc-sign090221	97.33%	100.00%	100.00%	100.00%	100.00%	100.00%	96.67%	56.00%
soc-Slashdot0811	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	75.83%
soc-Slashdot0902	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	74.62%
trec-wt10g	75.21%	75.21%	75.21%	75.21%	75.21%	75.21%	75.21%	28.35%
web-BerkStan	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	83.37%
web-Google	96.08%	100.00%	100.00%	100.00%	100.00%	100.00%	89.80%	60.39%
web-NotreDame	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	46.13%
web-Stanford	42.05%	42.07%	42.07%	42.07%	42.07%	42.07%	42.05%	24.00%
wiki-Talk	90.91%	90.91%	100.00%	100.00%	100.00%	100.00%	99.09%	0.00%
wiki-Vote	98.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	60.00%
youtube-links	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	49.13%
Zhishi-baidu	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	91.79%	49.74%
Zhishi-hudong	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	96.77%	73.87%

4.6.3 Computing Radius and Diameter

The following tables provide the number of iterations needed to compute diameter and radius, using various algorithms.

Undirected Graphs

Table 4.11. Number of iterations needed by various algorithms on undirected graphs.

Network	Diameter				Radius	
	SS	BD	iFub4S	iFubHd	SS	BD
as20000102	10	14	34	29	3	2
CA-AstroPh	15	18	12	12	8	9
CA-CondMat	11	13	14	6	4	3
ca-GrQc	4	24	6	11	4	10
ca-HepPh	11	20	39	10	6	9
ca-HepTh	10	14	12	9	6	6
com-amazon.all.cmty	4	16	16	10	4	7
com-amazon.ungraph	5	7	7	52	4	3
com-dblp.ungraph	9	8	34	9	4	3
com-lj.all.cmty	4	3	31	9	4	3
com-youtube.ungraph	3	2	6	2	3	2
email-Enron	6	10	14	19	4	3
facebook_combined	4	9	64	143	4	9
flickrEdges	36	1255	32178	11	3	156
gowalla_edges	6	5	6	2	6	5
loc-brightkite_edges	3	2	8	2	3	2
oregon1_010519	4	25	6	3	3	3
oregon1_010526	4	3	14	2	4	3
oregon2_010519	8	14	13	16	3	2
oregon2_010526	8	10	29	12	3	2
orkut-links	23	144	18722	103	8	13
p2pGnutella09	35	178	219	15	7	7
roadNet-CA	180	181	122817	354382	18	29
roadNet-PA	55	60	497	263606	10	11
roadNet-TX	68	84	25996	544280	6	9
soc-pokec-rel	6	3	74	2	6	3
youtube-u-growth	4	5	6	5	4	5

Directed Strongly Connected Graphs

Table 4.12. Number of iterations needed by various algorithms on directed strongly connected graphs.

Network	Diameter					Radius	
	SS	diFub2In	diFub2Out	diFubHdOut	diFubHdIn	SS	HR
amazon0302	17	482	482	264	146	20	284
amazon0312	36	68	68	2553	3198	29	242
amazon0505	17	41	24	173	194	22	2820
amazon0601	26	104	104	644	78	36	1021
as-caida20071105	15	8	8	10	10	15	2
ca-AstroPh	32	24	24	24	24	26	467
ca-CondMat	19	24	24	12	12	9	21
ca-GrQc	17	8	8	22	22	17	53
ca-HepPh	33	32	32	20	20	17	135
ca-HepTh	26	16	16	18	18	17	107
cit-HepPh	9	9	9	1719	5403	10	70
cit-HepTh	9	7	7	13	361	14	96
email-EuAll	18	10	10	11	11	19	4539
flickr-growth	15	18	18	17	17	12	4488
p2pGnutella04	15	38	38	44	117	18	38
p2pGnutella05	16	39	39	36	50	18	40
p2pGnutella06	21	172	172	113	117	17	167
p2pGnutella08	12	70	64	499	51	16	237
p2pGnutella09	33	307	307	137	187	32	28
p2pGnutella24	9	22	22	13	31	16	20
p2pGnutella25	59	152	152	88	1003	18	155
p2pGnutella30	38	246	288	1281	306	25	940
p2pGnutella31	26	255	255	132	212	22	306
soc-Epinions1	15	6	6	3	5	16	168
soc-pokec-rel	9	14	14	4	4	9	163
soc-sign-epinions	15	6	6	10	3	21	23
soc-sign081106	11	22	22	9	11	9	232
soc-sign090216	11	21	21	9	11	9	98
soc-sign090221	11	22	22	9	11	9	98
soc-Slashdot0811	48	28	28	10	10	13	17326
soc-Slashdot0902	9	21	21	10	10	11	12727
trec-wt10g	9	21	21	34	39	14	165
web-BerkStan	15	7	7	244	235	21	250
web-Google	15	8	8	51	12	17	243
web-NotreDame	9	7	7	6	5	11	45
web-Stanford	15	6	6	39	4	17	13941
wiki-Talk	20	13	13	45	6	18	61379
wiki-Vote	9	17	17	9	9	20	878
youtube-links	9	13	13	4	4	7	19247
zhishi-baidu	9	7	7	6	5	7	4990
zhishi-hudong	15	510	201	29	35	15	169

Directed Weakly Connected Graphs

Table 4.13. Number of iterations needed by the new algorithm on directed graphs.

Network	Diameter	Radius
amazon0302	21	18
amazon0312	21	30
amazon0505	10	329
amazon0601	53	37
as-caida20071105	15	15
ca-AstroPh	33	26
ca-CondMat	19	9
ca-GrQc	17	17
ca-HepPh	34	17
ca-HepTh	26	17
cit-HepPh	10	128
cit-HepTh	327	72
cit-Patents	1510	7
email-EuAll	33	36
flickr-growth	16	7
p2pGnutella04	17	19
p2pGnutella05	27	26
p2pGnutella06	20	27
p2pGnutella08	10	21
p2pGnutella09	43	34
p2pGnutella24	10	17
p2pGnutella25	78	24
p2pGnutella30	48	28
p2pGnutella31	37	39
soc-Epinions1	13	12
soc-sign-epinions	13	10
Soc-sign-Slash081106	20	15
Soc-sign-Slash090216	16	19
Soc-sign-Slash090221	13	17
soc-Slashdot0811	52	21
soc-Slashdot0902	20	16
trec-wt10g	559	7
web-BerkStan	17	19
web-Google	24	22
web-NotreDame	10	12
web-Stanford	69	7
wiki-Talk	10	10
wiki-Vote	10	16
youtube-links	21	13
Zhishi-baidu	10	7
Zhishi-hudong	17	17

4.7 Internet Movies Database Case Study

This section applies the ExactSumSweep algorithm to the Internet Movies Database (IMDB), in particular to the so-called actor graph, in which two actors are linked if they played together in a movie (we ignore TV-series in this work). All data have been taken from the website `http://www.imdb.com`. In line with `https://oracleofbacon.org/`, we decided to exclude some genres from our database: awards-shows, documentaries, game-shows, news, realities and talk-shows. We analyzed snapshots of the actor graph, taken every 5 years from 1940 to 2010, and 2014.

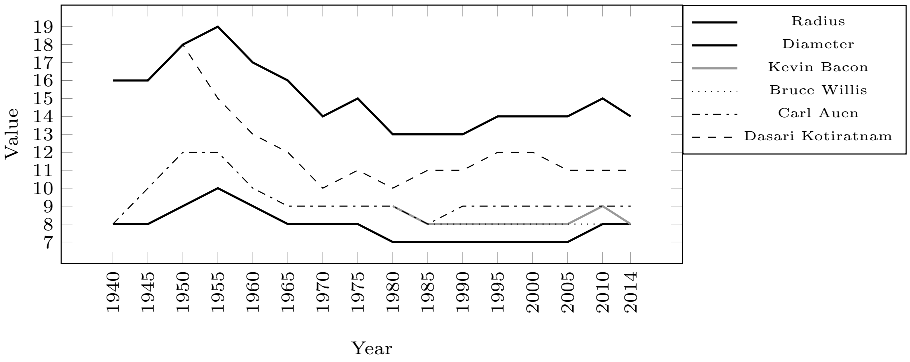


Figure 4.2. Actor graph evolution in terms of radius, diameter, and actor eccentricity.

4.7.1 Analysis of the Graph Stretch

Figure 4.2 shows the evolution of the diameter, the radius and the eccentricity of some actors. It shows that the stretch of the graph (in terms of radius and diameter) increased between 1940 and 1955, then it started decreasing, confirming the behavior of the diameter of real-world graphs analyzed in [107]. The first increase might be explained by the fact that the years between the Forties and the Sixties are known as the golden age for Asian cinema, especially Indian and Japanese.[2] Examples of popular movies of that period include *Tokyo Story* and *Godzilla*. This trend is also confirmed by the names of the central actors during that period. In 1940, they are all Western, usually German, like Carl Auen. The only non-western central actor is the Birmanian Abraham Sofaer, but he worked in England and America. On the other hand, in 1955, we find both Western actors like James Bell and Eastern actors like Sōjin Kamiyama.

Later, in the Sixties, the increase in independent producers and growth of production companies led to an increase of power of individual actors. This can explain the decreasing size of the graph during those years: the number of contacts between actors from different countries increased. An example of this trend is the first James Bond movie, *Dr. No*, starring Sean Connery (from Scotland), Ursula Andress (who is Swiss-American), Joseph Wiseman (from Canada), etc.

The decrease of the graph size halted in the Eighties, and there were little changes until the present. Now it seems that the size is slightly increasing again, but the number of central actors is increasing as well. It would be interesting to see if this trend will continue or there will soon be an actor who obtains again an eccentricity of 7.

4.7.2 Analysis of the Eccentricity of Actors

All actors seem to decrease their eccentricity as time passes. Even Dasari Kotiratnam, an Indian actress active between 1935 and 1939, is a diametral node when she is active, then she becomes more and more central. Also Carl Auen, who started from the center in 1940, remains quite central, having an eccentricity of 9 in the present.

Instead, the periphery is usually composed by recent actors: for example, in the last graph, the actress Neeta Dhungana is a diametral node who played only in the Nepalese movie *Masaan*. Another example is Steveanna Roose, an American actress who played only

[2] All other historical data in this section is taken from [156].

in *Lost in the Woods*: in 2010 she is diametral, having eccentricity 15, while in 2014 she obtained an eccentricity of 12 (without playing any movie in that period).

4.7.3 The Six Degrees of Separation Game

These results can be directly applied to analyze the solvability of the six degrees of separation game, which is a trivia game inspired by the well-known social experiment of Stanley Milgram [117], which was in turn a continuation of the empirical study of the structure of social networks by Michael Gurevich [86]. The game refers to a network, such as the (movie) actors collaboration network or the Wikipedia citation network, and can be played according to two main different variants. In the first variant, given two nodes s, i.e. the source, and t, i.e. the target, the player is asked to find a path of length at most six between s and t. In the second variant of the game, the node s is fixed and only the target node t is chosen during the game: for instance, in the case of the actor collaboration network, one very popular instance of this variant is the so-called "Six Degrees of Kevin Bacon" game, where the node s is the actor Kevin Bacon, who is considered one of the centers of the Hollywood universe (`https://oracleofbacon.org/`).

From our results, the Six Degrees of Separation game is not always solvable, and it has never been solvable, even if we fix the source. We also remark that Kevin Bacon has not minimum eccentricity until the present, and he never gets eccentricity 6, as suggested by the "Six Degrees of Kevin Bacon" game.

4.8 Wikipedia Case Study

The Wikipedia graph consists of all pagelinks between (English) Wikipedia articles and can be downloaded at DBpedia (`http://wiki.dbpedia.org/`). In the "Six Degrees of Wikipedia" game one is asked to connect two given Wikipedia pages, i.e., a source and a target page, by using at most six links. In this section, we analyze the Wikipedia directed graph, trying to understand whether the "Six Degrees of Wikipedia" game is always solvable whenever a path from the source to the target exists. We compute for the first time the radius and diameter of the whole Wikipedia graph, we further analyze the biggest SCC, and finally we try to avoid the problems generated by "long paths" inside the graph.

First of all, let us compute the radius and the diameter of the whole Wikipedia graph (which is composed by 4,229,722 nodes and 102,165,856 edges, with 452,488 strongly connected components). The diameter is 377 (254 iterations needed by using ExactSumSweep algorithm) and the radius is 42 (203 iterations). Note that these values are extremely high if compared with diameter and radius of real-world networks: in order to explain this phenomenon, it is worth analyzing the paths involved. In particular, the diameter starts from page `List of minor planets/108101-108200` and remains inside this list until page `List of minor planets/145701-145800`, in order to reach the last page `(145795) 1998 RA16` (which is a minor planet). Clearly the diameter only depends on this list and does not give any information about the connectivity of the remaining part of the graph.

A similar phenomenon holds for the radius: a radial node is the page `1954 in Ireland`, and a longest path from this node reaches in 6 steps the page `Papyrus Oxyrhynchus 116`, where a long path starts until the page `Papyrus Oxyrhynchus 158`, adding 36 steps.[3]

Moreover, the first step after the page `1954 in Ireland` reaches the page `England`, but this latter node has bigger eccentricity because of the existence of a very long path of pages with title `All-Ireland Minor Hurling Championship` the reason why the page `1954 in Ireland` is central is that it stays "between" the two very long lists `All-Ireland Minor Hurling Championship` and `Papyrus Oxyrhynchus`, not meaning that it is well connected to the rest of the graph.

[3] In the path of the pages `Papyrus Oxyrhynchus`, a jump is done from page `Papyrus Oxyrhynchus 145` to page `Papyrus Oxyrhynchus 152`

Similar results are found if we restrict ourselves to the biggest SCC of the graph, composed by 3,763,632 nodes. We obtain that the diameter is 49 (9 iterations) and the radius is 42 (14 iterations). Again, the longest path contains many pages in the `All-Ireland Minor Hurling Championship` path, starting from `Kickxellales` (an order of fungus) and ending into `All-Ireland Minor Hurling Championship 1928`. A radial node is `Play it Again Des`, and again the longest path from this node reaches `All-Ireland Minor Hurling Championship 1928`.

The last experiment tries to "eliminate" these long paths from the graph: to do so, we have modified all page titles by leaving only letters (case-sensitive), and we have collapsed all the pages having the same title. In this way, all numbers are deleted and the previous long paths of pages collapse to a single node: for instance, all pages like `All-Ireland Minor Hurling Championship YYYY` for years `YYYY` collapse to the single page `AllIrelandMinorHurlingChampionship`. After this procedure, the graph has 3,939,060 nodes, the diameter becomes 23 (14 iterations) and the radius becomes 17 (16 iterations). However, this is still not sufficient to analyze the structure of the graph, since many important different pages collapse together: for instance the collapsed page `s` (a radial node) corresponds to all pages like `1960s`, `1970s`, `1980s`, etc. Moreover, the problem of long lists is not completely solved yet, because the diameter starts from page `Advanced Diabetes Management Certification` and ends in page `Osieki Lęborskie railway station` (a Polish railway station): 15 steps of this path pass from a Polish railway station to another (the list starts from `Lębork railway station`). The same issues hold if only the biggest SCC is considered, since all considered paths are contained in the biggest SCC (the iterations become 15 for the diameter and 13 for the radius by using ExactSumSweep Algorithm). We argue that dealing with these long paths in this graph is a difficult task that require more sophisticated techniques exploiting the content of the pages.

More interesting results can be obtained by reversing all the edges: this way, a radial node is an "easily reachable node" and not a node that reaches easily all the others. In this case, the radius is very small, because "popular nodes" are cited by many other pages. Indeed, although the diameter remains obviously 377, the radius becomes 6 (7 iterations), and a radial node is the page `United States`. Intuitively, this happens because it is much easier "to reach very popular nodes" than to reach unpopular nodes by starting from very popular ones. This means that if we fix the target of the "Six Degrees of Wikipedia" game to be the `United States` the game becomes always solvable!

4.9 Bibliographic Notes

The definition and the implementation of previous algorithms was taken from [60, 114, 154, 63, 61, 155] (for more information, we refer to the introduction, at the beginning of the chapter, or to Section 4.5). The rest of this chapter is original, and it was published in [33, 32]: the paper [33] deals only with the strongly connected case, while the paper [32] explains the algorithm in full generality. In particular, Sections 4.2 to 4.8 are taken from [32]. Finally, another algorithm for the diameter of directed, not strongly connected graphs was proposed in [7]: in this chapter, we have not considered this algorithm because the techniques used seem to be a subset of our techniques, and the experimental results seem to be orders of magnitude worse than ours (for instance, in the graph `web-Google`, we need 24 BFSs, while the algorithm in [7] needs 12 074 BFSs).

Chapter 5

Computing Closeness Centrality: the BCM Algorithm

Abstract

The closeness centrality is a traditional measure of the centrality of a node in a complex network. In this chapter, we address the problem of computing the k most central nodes, for some small k: we propose a new algorithm, and we experimentally show that it outperforms all previous approaches.

For example, we are able to compute the top-10 or top-100 nodes in few dozens of seconds in real-world networks with millions of nodes and edges, such as the actors collaboration network and the Wikipedia citation network.

The problem of identifying the most central nodes is a fundamental question in network analysis, with applications in a plethora of research areas, such as biology, computer science, sociology, and psychology. Because of the importance of this question, dozens of centrality measures have been introduced in the literature (for a recent survey, see [25]). Among these measures, closeness centrality is certainly one of the oldest and of the most widely used [17]: it is studied in almost all books dealing with network analysis (for example, [126]), and almost all existing graph libraries implement algorithms to compute it.

Also for this measure, the textbook approach is based on computing the distance between all pairs of nodes: from these distances, one can easily compute the closeness centrality of every node, and output the k largest values found. As in the case of the diameter and the radius, the main bottleneck of this approach is solving the All Pairs Shortest Paths problem, which requires time $\mathcal{O}(mn)$ on sparse graphs. Again, our goal is to improve this approach.

Knowing that it is probably unlikely to achieve such result in the worst-case (see Chapter 3), we develop a new algorithm that works well in practice, even if the time complexity is still $\mathcal{O}(mn)$. This algorithm can also be easily adapted to the computation of similar measures, such as the harmonic centrality and the exponential centrality.

The new approach combines the BFS-based algorithm with a pruning technique: during the algorithm, we compute and update upper bounds on the closeness of all the nodes, and we exclude a node s from the computation as soon as its upper bound is "small enough", that is, we are sure that s does not belong to the top-k nodes. We propose two different strategies to set the initial bounds, and two different strategies to update the bounds during the computation: this means that our algorithm comes in four different variations. The experimental results show that different variations perform well on different kinds of networks, and the best variation of our algorithm drastically outperforms both a probabilistic approach [128], and the best exact algorithm available until now [129]. We have computed for the first time the 10 most central nodes in networks with millions of nodes and hundreds of millions of edges, in very little time. A significant example is the `wiki-Talk` network, which was also used in [142], where the authors propose an algorithm to update closeness centralities after edge additions or deletions. Our performance is about 30 000 times better than the performance of the textbook algorithm: if only the most central node is needed, we can recompute it from scratch more than 150 times faster than the geometric average update time in [142]. Finally, our approach is not only very efficient, but it is also very easy to code, making it a very good candidate to be implemented in existing graph libraries. Indeed, it is already implemented in NetworKit [151], and one of its variations is implemented in Sagemath [64] and in WebGraph [23]. We sketch the main ideas of the algorithm in Section 5.2, we provide all details in Sections 5.3 to 5.6. We experimentally evaluate the efficiency of the new algorithm in Section 5.7. In the last part of this chapter (Sections 5.8 and 5.9), we consider two case studies: the actor collaboration network (1 797 446 nodes, 72 880 156 edges) and the Wikipedia citation network (4 229 697 nodes, 102 165 832 edges). In the actor collaboration network, we analyze the evolution of the 10 most central nodes, considering snapshots taken every 5 years between 1940 and 2014. The computation was performed in little more than 45 minutes. In the Wikipedia case study, we consider both the standard citation network, that contains a directed edge (v, w) if the page v contains a link to w, and the reversed network, that contains a directed edge (v, w) if w contains a link to v. In a few minutes, we are able to compute the 10 most central pages of most of these graphs, making them available for future analyses.

5.1 Related Work

Closeness is a "traditional" definition of centrality, and consequently it was not "designed with scalability in mind", as stated in [96]. Also in [48], it is said that closeness centrality can "identify influential nodes", but it is "incapable to be applied in large-scale networks due to the computational complexity". The simplest solution considered was to define different measures, that might be related to closeness centrality [96].

A different line of research has tried to develop more efficient algorithms, or lower bounds for the complexity of this problem, as we saw in Chapter 3. In particular, we have proved that an algorithm that finds the least central node, or the most central node in directed, disconnected graphs, falsifies the Orthogonal Vector conjecture, and consequently the Hitting Set conjecture, and the Strong Exponential Time Hypothesis. Furthermore, it is also impossible to find the most central node, unless the Hitting Set conjecture is false [3]. Similar results hold in the dense weighted context [1]: the complexity of computing the most central node is the same as the complexity of computing the All Pairs Shortest Paths, a task that cannot be solved in time $\mathcal{O}(n^{3-\varepsilon})$ by any existing algorithm.

Knowing these hardness results, researchers tried to find ways to overcome them. A first possibility is to design approximation algorithms: the simplest approach samples the distance between a node s and l other nodes t, and returns the average of all values $\mathrm{dist}(s, t)$ found [72]. The time complexity is $\mathcal{O}(lm)$, to obtain an approximation $\tilde{c}(s)$ of the centrality of each node s such that

$$\mathbb{P}\left(\left|\frac{1}{\tilde{c}(s)} - \frac{1}{c(s)}\right| \geq \varepsilon D\right) \leq 2e^{-\Omega\left(l\varepsilon^2\right)}$$

where D is the diameter of the graph. A more refined approximation algorithm is provided in [55], which combines the sampling approach with a 3-approximation algorithm: this algorithm has running time $\mathcal{O}(lm)$, and it provides an estimate $\tilde{c}(v)$ of the centrality of each node v such that

$$\mathbb{P}\left(\left|\frac{1}{\tilde{c}(s)} - \frac{1}{c(s)}\right| \geq \frac{\varepsilon}{c(s)}\right) \leq 2e^{-\Omega\left(l\varepsilon^3\right)}$$

(note that, differently from the previous algorithm, this algorithm provides a guarantee on the relative error). However, even if these approximation algorithms work quite well, they are not suited to the ranking of nodes: indeed, we work with so-called *small world* networks, having a low diameter, and an even smaller average distance. Indeed, in a typical graph, the average distance between two nodes s and t is between 1 and 10, meaning that most of the n centrality values lie in this range. In order to obtain a ranking, we need the error to be close to $\frac{10}{n}$, which might be very small. Nevertheless, an approximation algorithm was proposed in [128], where the sampling technique developed in [72] was used to actually compute the top-k nodes: the result is not exact, but it is exact with high probability. The authors proved that the time complexity of their algorithm is $\mathcal{O}(mn^{\frac{2}{3}} \log n)$, under the rather strong assumption that closeness centralities are uniformly distributed between 0 and D (in the worst-case, the time complexity of this algorithm is $\mathcal{O}(mn)$).

Other approaches have tried to develop incremental algorithms that might be more suited to real-world networks. For instance, in [109], the authors develop heuristics to determine the k most central nodes in a varying environment. Furthermore, in [142], the authors consider the problem of updating the closeness centrality of all nodes after edge insertions or deletions: in some cases, the time needed for the update could be orders of magnitude smaller than the time needed to recompute all centralities from scratch.

Finally, some works have tried to exploit properties of real-world networks in order to find more efficient algorithms. In [106], the authors develop a heuristic to compute the k most central nodes according to different measures. The basic idea is to identify central nodes according to an "easy" centrality measure (for instance, degree of nodes), and then to inspect a small set of central nodes according to this measure, hoping it contains the top-k nodes according to the "hard" measure. The last approach [129], proposed by Olsen et al., tries to exploit the properties of real-world networks in order to develop exact algorithms with worst-case complexity $\mathcal{O}(mn)$, but performing much better in practice. As far as we know, this is the only exact algorithm that is able to efficiently compute the k most central nodes in networks with up to 1 million nodes.

However, despite this large amount of research, the major graph libraries still use the textbook algorithm: among them, Boost Graph Library [87], Sagemath [64], igraph [152], NetworkX [147], and NetworKit [151]. This is due to the fact that efficient available exact algorithms for top-k closeness centrality, like [129], are relatively recent and make use of several other non-trivial routines. Conversely, our algorithm is very simple, and it is already implemented in some graph libraries, such as NetworKit [151], WebGraph [23], and Sagemath [64].

5.2 Overview of the Algorithm

In this section, we describe our new approach for computing the k nodes with largest closeness (equivalently, the k nodes with smallest farness). If we have more than one node with the same score, we output all nodes having a centrality bigger than or equal to the centrality of the k-th node.

The basic idea is to keep track of a lower bound on the farness of each node, and to skip the analysis of a node s if this lower bound implies that s is not in the top-k. More formally, let us assume that we know the farness of some nodes $s_1, \dots, s_\ell$, and a lower bound $L(v)$ on the farness of any other node v. Furthermore, assume that there are k nodes among $s_1, \dots, s_l$ satisfying $f(s_i) < L(v) \; \forall v \in V - \{s_1, \dots, s_l\}$, and hence $f(v) \geq L(v) \geq \max_{w \in V - \{s_1, \dots, s_l\}} L(w) > f(s_i)$. Then, we can safely skip the exact computation of $f(v)$ for all remaining nodes v, because the k nodes with smallest farness are among $s_1, \dots, s_l$.

This idea is implemented in Algorithm 8: we use a list `Top` containing all "analyzed" nodes $s_1, \dots, s_l$ in increasing order of farness, and a priority queue `Q` containing all nodes "not analyzed, yet", in increasing order of lower bound L (this way, the head of `Q` always has the smallest value of L among all nodes in `Q`). At the beginning, using the function `computeBounds`(), we compute a first bound $L(v)$ for each node v, and we fill the queue `Q`

Algorithm 8: Pseudocode of our algorithm for top-k closeness centralities.

```
Input  : A graph $G = (V, E)$
Output: top-$k$ nodes with highest closeness and their closeness values $c(v)$
global $L, Q \leftarrow$ computeBounds($G$);
global Top $\leftarrow$ [ ];
global Farn;
for $v \in V$ do Farn[$v$] = $+\infty$;
while $Q$ is not empty do
    $s \leftarrow Q$.extractMin();
    if |Top| $\geq k$ and $L[s] >$ Top[$k$] then return Top;
    Farn[$s$] $\leftarrow$ updateBounds($s$); // This function might also modify $L$
    add $s$ to Top, and sort Top according to Farn;
    update $Q$ according to the new bounds;
end
```

according to this bound. Then, at each step, we extract the first element s of `Q`: if $L(s)$ is smaller than the k-th smallest farness computed until now (that is, the farness of the k-th node in variable `Top`), we can safely stop, because for each $v \in$ `Q`, $f(v) \leq L(v) \leq L(s) < f($`Top[`$k$`]`$)$, and v is not in the top-k. Otherwise, we run the function `updateBounds`(s), which performs a BFS from s, returns the farness of s, and improves the bounds L of all the other nodes. Finally, we insert s into `Top` in the right position, and we update `Q` if the lower bounds have changed.

The crucial point of the algorithm is the definition of the lower bounds, that is, the definition of the functions `computeBounds` and `updateBounds`. We propose two alternative strategies for each of these two functions: in both cases, one strategy is conservative, that is, it tries to perform as few operations as possible, while the other strategy is aggressive, that is, it needs many operations, but at the same time it improves many lower bounds.

Let us analyze the possible choices of the function `computeBounds`. The conservative strategy `computeBoundsDeg` needs time $\mathcal{O}(n)$: it simply sets $L(v) = 0$ for each v, and it fills `Q` by inserting nodes in decreasing order of degree (the idea is that nodes with high degree have small farness, and they should be analyzed as early as possible, so that the values in `Top` are correct as soon as possible). Note that the nodes can be sorted in time $\mathcal{O}(n)$ using counting sort.

The aggressive strategy `computeBoundsNB` needs time $\mathcal{O}(mD)$, where D is the diameter of the graph: it computes the neighborhood-based lower bound $L^{\mathrm{NB}}(s)$ for each node s (we explain shortly afterwards how it works), it sets $L(s) = L^{\mathrm{NB}}(s)$, and it fills Q by adding nodes in decreasing order of L. The idea behind the neighborhood-based lower bound is to count the number of paths of length ℓ starting from a given node s, which is also an upper bound U_ℓ on the number of nodes at distance ℓ from s. From U_ℓ, it is possible to define a lower bound on $\sum_{v \in V} \mathrm{dist}(s, v)$ by "summing U_ℓ times the distance ℓ", until we have summed n distances: this bound yields the desired lower bound on the farness of s. The detailed explanation of this function is provided in Section 5.3.

For the function `updateBounds`(s), the conservative strategy `updateBoundsBFSCut`(s) does not improve L, and it cuts the BFS as soon as it is sure that the farness of s is smaller than the k-th biggest farness found until now, that is, `Farn`[`Top`[k]]. If the BFS is cut, the function returns $+\infty$, otherwise, at the end of the BFS we have computed the farness of s, and we can return it. The running time of this procedure is $\mathcal{O}(m)$ in the worst-case, but it can be smaller in practice. It remains to define how the procedure can be sure that the farness of s is at least `Farn`[`Top`[k]]: to this purpose, during the BFS, we update a lower bound on the farness of s. The idea behind this bound is that, if we have already visited all nodes up to distance ℓ, we can upper bound the closeness centrality of s by setting distance $\ell + 1$ to a number of nodes equal to the number of edges "leaving" level ℓ, and distance $\ell + 2$ to all the remaining nodes. The details of this procedure are provided in Section 5.4.

The aggressive strategy `updateBoundsLB`(s) performs a complete BFS from s, and it bounds the farness of each node v using the level-based lower bound. The running time is $\mathcal{O}(m)$ for the BFS, and $\mathcal{O}(n)$ to compute the bounds. The idea behind the level-based lower bound is that $\operatorname{dist}(v,w) \geq |\operatorname{dist}(s,w) - \operatorname{dist}(s,v)|$, and consequently $\sum_{w \in V} \operatorname{dist}(v,w) \geq \sum_{w \in V} |\operatorname{dist}(s,w) - \operatorname{dist}(s,v)|$. The latter sum can be computed in time $\mathcal{O}(n)$ for each v, because it depends only on the level ℓ of v in the BFS tree, and because it is possible to compute in $\mathcal{O}(1)$ the sum for a node at level $\ell+1$, if we know the sum for a node at level ℓ. The details are provided in Section 5.5.

Finally, in order to transform these lower bounds on $\sum_{w \in V} \operatorname{dist}(v,w)$ into bounds on $f(v)$, we need to know the number of nodes reachable from a given node v. In Sections 5.3 to 5.5, we assume that these values are known: this assumption is true in undirected graphs, where we can compute the number of reachable nodes in linear time at the beginning of the algorithm, and in strongly connected directed graphs, where the number of reachable nodes is n. The only remaining case is when the graph is directed and not strongly connected: in this case, we need some additional machinery, which is presented in Section 5.6.

5.3 The `computeBoundsNB` Function

In this section, we define a lower bound $S^{\mathrm{NB}}(s, r(s))$ on the total distance sum $S(s) = \sum_{v \in R(s)} \operatorname{dist}(s,v)$ of a node v in an undirected or strongly-connected graph. If we know the number $r(s)$ of nodes reachable from s, this bound translates into a lower bound on the farness of s, simply multiplying by $(n-1)/(r(s)-1)^2$. The basic idea is to find an upper bound $\tilde{\gamma}^\ell(s)$ on the number of nodes $\boldsymbol{\gamma}^\ell(s)$ at distance exactly ℓ from s. Then, intuitively, if we assume that the number of nodes at distance ℓ is greater than its actual value and "stop counting" when we have $r(v)$ nodes, we get something that is smaller than the actual total distance. This is because we are assuming that the distances of some nodes are smaller than their actual values. This argument is formalized in Proposition 5.1.

Proposition 5.1. *If $\tilde{\gamma}^\ell(s)$ is an upper bound on $\boldsymbol{\gamma}^\ell(s)$, for $\ell = 0, ..., D$, where D is the diameter of the graph, then*

$$S^{NB}(s, r(s)) := \sum_{\ell=1}^{D} \ell \cdot \min\left\{\tilde{\gamma}^\ell(s),\ \max\left\{r(s) - \sum_{i=0}^{\ell-1} \tilde{\gamma}^i(s),\ 0\right\}\right\}$$

is a lower bound on $S(s) := \sum_{v \in R(v)} \operatorname{dist}(s,v)$.

Proof. Let us sort the nodes in increasing order of distance from s, obtaining $v_1, \dots, v_{r(v)}$, and let us define $\ell_i = 0$ for $i \leq \tilde{\gamma}^0(s) = 1$, $\ell_i = 1$ for $\tilde{\gamma}^0(s) < i \leq \tilde{\gamma}^0(s) + \tilde{\gamma}^1(s)$, and in general $\ell_i = \ell$ for each i such that $\sum_{l=0}^{\ell-1} \tilde{\gamma}^l(s) < i \leq \sum_{l=0}^{\ell} \tilde{\gamma}^l(s)$. We want to prove the following:

$$S^{\mathrm{NB}}(s, r(s)) = \sum_{i=0}^{r(s)} \ell_i \leq \sum_{i=0}^{r(s)} \operatorname{dist}(s, v_i) = S(s).$$

The first equality follows by definition of ℓ_i and $S^{\mathrm{NB}}((,s))$, while the last equality follows by definition of $S(s)$. For the second inequality, let us prove that $\ell_i \leq \operatorname{dist}(s, v_i)$ for each i: by definition of the ℓ_is, $i \leq \tilde{\gamma}^0(s) + \cdots + \tilde{\gamma}^{\ell_i}(s) \leq \boldsymbol{\gamma}^0(s) + \cdots + \boldsymbol{\gamma}^{\ell_i}(s)$. Consequently, there are at least i nodes at distance at most ℓ_i from s, and since the v_is are sorted, $\operatorname{dist}(s, v_i) \leq \ell_i$. □

In the following paragraphs, we propose upper bounds $\tilde{\gamma}^i(s)$ for trees, undirected graphs and directed strongly-connected graphs. In case of trees, the bound $\tilde{\gamma}^i(s)$ is actually equal to $\boldsymbol{\gamma}^i(v)$, which means that the algorithm can be used to compute closeness of all nodes in a tree exactly.

Computing closeness on trees. Let us consider a node s for which we want to compute the total distance $S(s)$. The number of nodes at distance 1 in the BFS tree from s is clearly the degree of s. What about distance 2? If $\mathbf{\Gamma}^\ell(s)$ is the set of nodes at distance ℓ from s, since there are no cycles, all the neighbors of the nodes in $\mathbf{\Gamma}^1(s)$ are nodes at distance 2 from s, with the only exception of s itself. Therefore, $\boldsymbol{\gamma}^2(s) = \sum_{v \in \mathbf{\Gamma}^1(s)} \boldsymbol{\gamma}^1(v) - \deg(s)$. In general, we can always relate the number of nodes at each distance ℓ from s to the number of nodes at distance $\ell - 1$ in the BFS trees of the neighbors of s. Let us now consider $\boldsymbol{\gamma}^\ell(s)$, for $\ell > 2$. Figure 5.1 shows an example where s has three neighbors w_1, w_2 and w_3. Suppose we want to compute $\mathbf{\Gamma}^4(s)$ using information from w_1, w_2 and w_3. Clearly, $\mathbf{\Gamma}^4(s) \subset \mathbf{\Gamma}^3(w_1) \cup \mathbf{\Gamma}^3(w_2) \cup \mathbf{\Gamma}^3(w_3)$; however, there are also other nodes in the union that are not in $\mathbf{\Gamma}^4(s)$. Furthermore, the nodes in $\mathbf{\Gamma}^3(w_1)$ (red nodes in the leftmost tree) are of two types: nodes in $\mathbf{\Gamma}^4(s)$ (the ones in the subtree of w_1) and nodes in $\mathbf{\Gamma}^2(s)$ (the ones in the subtrees of w_2 and w_3). An analogous behavior can be observed for w_2 and w_3 (central and rightmost trees). If we simply sum all the nodes in $\boldsymbol{\gamma}^3(w_1)$, $\boldsymbol{\gamma}^3(w_2)$ and $\boldsymbol{\gamma}^3(w_3)$, we would be counting each node at level 2 twice, i.e., once for each node in $\mathbf{\Gamma}^1(s)$ minus one. Hence, for each $\ell > 2$, we can write

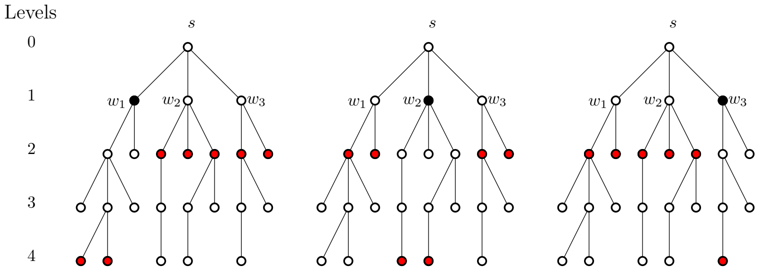


Figure 5.1. Relation between nodes at distance 4 for s and the neighbors of s. The red nodes represent the nodes at distance 3 for w_1 (left), for w_2 (center) and for w_3 (right).

$$\boldsymbol{\gamma}^\ell(s) = \sum_{w \in \mathbf{\Gamma}^1(s)} \boldsymbol{\gamma}^{\ell-1}(w) - \boldsymbol{\gamma}^{\ell-2}(s) \cdot (\deg(s) - 1). \tag{5.1}$$

From this observation, we define a new method to compute the total distance of all nodes, described in Algorithm 9. Instead of computing the BFS tree of each node one by one, at each step we compute the number $\boldsymbol{\gamma}^\ell(s)$ of nodes at level ℓ for *all* nodes s. First (Lines 1 - 3), we compute $\boldsymbol{\gamma}^1(s)$ for each node (and add that to $S(s)$). Then (Lines 7 - 26), we consider all the other levels ℓ one by one. For each ℓ, we use $\boldsymbol{\gamma}^{\ell-1}(w)$ of the neighbors w of s and $\boldsymbol{\gamma}^{\ell-2}(s)$ to compute $\boldsymbol{\gamma}^\ell(s)$ (Line 10 and 13). If, for some ℓ, $\boldsymbol{\gamma}^\ell(s) = 0$, all the nodes have been added to $S(s)$. Therefore, we can stop the algorithm when $\boldsymbol{\gamma}^\ell(s) = 0 \ \ \forall s \in V$.

Proposition 5.2. *Algorithm 9 requires $\mathcal{O}(D \cdot m)$ operations to compute the closeness centrality of all nodes in a tree T.*

Proof. The for loop in Lines 1 - 3 of Algorithm 9 clearly takes $\mathcal{O}(n)$ time. For each level of the while loop of Lines 7 - 26, each node scans its neighbors in Line 10 or Line 13. In total, this leads to $\mathcal{O}(m)$ operations per level. Since the maximum number of levels that a node can have is equal to the diameter of the tree, the algorithm requires $\mathcal{O}(D \cdot m)$ operations.

□

Note that closeness centrality on trees could even be computed in time $\mathcal{O}(n)$ in a different manner [42]. We choose to include Algorithm 9 here nonetheless since it paves the way for an algorithm computing a lower bound in general undirected graphs, described next.

Algorithm 9: Closeness centrality in trees.

```
Input  : a tree $T = (V, E)$
Output: the closeness centralities $c(s)$ of each node $s \in V$
foreach $s \in V$ do
    $\boldsymbol{\gamma}^{\ell-1}(s) \leftarrow \deg(s)$;
    $S(s) \leftarrow \deg(s)$;
end
$\ell \leftarrow 2$;
nFinished $\leftarrow 0$;
while nFinished $< n$ do
    foreach $s \in V$ do
        if $\ell = 2$ then
            $\boldsymbol{\gamma}^{\ell}(s) \leftarrow \sum_{w \in N(s)} \boldsymbol{\gamma}^{\ell-1}(w) - \deg(s)$;
        end
        else
            $\boldsymbol{\gamma}^{\ell}(s) \leftarrow \sum_{w \in N(s)} \boldsymbol{\gamma}^{\ell-1}(w) - \boldsymbol{\gamma}^{\ell-2}(s)(\deg(s) - 1)$;
        end
    end
    foreach $s \in V$ do
        $\boldsymbol{\gamma}^{\ell-2}(s) \leftarrow \boldsymbol{\gamma}^{\ell-1}(s)$;
        $\boldsymbol{\gamma}^{\ell-1}(s) \leftarrow \boldsymbol{\gamma}^{\ell}(s)$;
        if $\boldsymbol{\gamma}^{\ell-1}(s) > 0$ then
            $S(s) \leftarrow S(s) + \ell \cdot \boldsymbol{\gamma}^{\ell-1}(s)$;
        end
        else
            nFinished $\leftarrow$ nFinished $+ 1$;
        end
    end
    $\ell \leftarrow \ell + 1$;
end
foreach $s \in V$ do
    $c(v) \leftarrow (n - 1)/S(v)$;
end
return $c$
```

Lower bound for undirected graphs. For general undirected graphs, Eq. (5.1) is not true anymore – but a related upper bound $\tilde{\gamma}^{\ell}(\cdot)$ on $\boldsymbol{\gamma}^{\ell}(\cdot)$ is still useful. Let $\tilde{\gamma}^{\ell}(s)$ be defined recursively as in Equation (5.1): in a tree, $\tilde{\gamma}^{\ell}(s) = \boldsymbol{\gamma}^{\ell}(s)$, while in this case we prove that $\tilde{\gamma}^{\ell}(s)$ is an upper bound on $\boldsymbol{\gamma}^{\ell}(s)$. Indeed, there could be nodes v for which there are multiple paths between s and v and that are therefore contained in the subtrees of more than one neighbor of s. This means that we would count v multiple times when considering $\tilde{\gamma}^{\ell}(s)$, overestimating the number of nodes at distance ℓ. However, we know for sure that at level ℓ there cannot be *more nodes* than in Equation (5.1). If, for each node s, we assume that the number $\tilde{\gamma}^{\ell}(s)$ of nodes at distance ℓ is that of Equation (5.1), we can apply Proposition 5.1 and get a lower bound $S^{\mathrm{NB}}(s, r(s))$ on the total sum for undirected graphs. The procedure is described in Algorithm 10. The computation of $S^{\mathrm{NB}}(S, r(v))$ works basically like Algorithm 9, with the difference that here we keep track of the number of the nodes found in all the levels up to ℓ (nVisited) and stop the computation when nVisited becomes equal to $r(s)$ (if it becomes larger, in the last level we consider only $r(s) -$ nVisited nodes, as in Proposition 5.1 (Lines 26 - 29).

Proposition 5.3. *For an undirected graph G, computing the lower bound $S^{NB}(v, r(v))$ described in Algorithm 10 takes $\mathcal{O}(D \cdot m)$ time.*

Proof. Like in Algorithm 9, the number of operations performed by Algorithm 10 at each level of the while loop is $\mathcal{O}(m)$. At each level i, all the nodes at distance i are accounted for (possibly multiple times) in Lines 12 and 15. Therefore, at each level, the variable nVisited is always greater than or equal to the the number of nodes v at distance $\mathrm{dist}(v) \leq i$. Since $\mathrm{dist}(v) \leq D$ for all nodes v, the maximum number of levels scanned in the while loop cannot be larger than D, therefore the total complexity is $\mathcal{O}(D \cdot m)$.

Algorithm 10: Neighborhood-based lower bound for undirected graphs.

```
   Input  : A graph $G = (V, E)$
   Output: Lower bounds $L^{\mathrm{NB}}(v, r(v))$ of each node $v \in V$
 1 foreach $s \in V$ do
 2     $\boldsymbol{\gamma}^{k-1}(s) \leftarrow \deg(s)$;
 3     $\tilde{S}^{(\mathrm{un})}(s) \leftarrow \deg(s)$;
 4     nVisited[$s$] $\leftarrow \deg(s) + 1$;
 5     finished[$s$] $\leftarrow false$;
 6 end
 7 $k \leftarrow 2$;
 8 nFinished $\leftarrow 0$;
 9 while nFinished $< n$ do
10     foreach $s \in V$ do
11         if $k = 2$ then
12             $\boldsymbol{\gamma}^{k}(s) \leftarrow \sum_{w \in N(s)} \boldsymbol{\gamma}^{k-1}(w) - \deg(s)$;
13         end
14         else
15             $\boldsymbol{\gamma}^{k}(s) \leftarrow \sum_{w \in N(s)} \boldsymbol{\gamma}^{k-1}(w) - \boldsymbol{\gamma}^{k-2}(s)(\deg(s) - 1)$;
16         end
17     end
18     foreach $s \in V$ do
19         if finished[$v$] then continue;;
20         $\boldsymbol{\gamma}^{k-2}(s) \leftarrow \boldsymbol{\gamma}^{k-1}(s)$;
21         $\boldsymbol{\gamma}^{k-1}(s) \leftarrow \boldsymbol{\gamma}^{k}(s)$;
22         nVisited[$s$] $\leftarrow$ nVisited[$s$] $+ \boldsymbol{\gamma}^{k-1}(s)$;
23         if nVisited[$s$] $< r(v)$ then
24             $\tilde{S}^{(\mathrm{un})}(s) \leftarrow \tilde{S}^{(\mathrm{un})}(s) + k \cdot \boldsymbol{\gamma}^{k-1}(s)$;
25         end
26         else
27             $\tilde{S}^{(\mathrm{un})}(s) \leftarrow \tilde{S}^{(\mathrm{un})}(s) + k(r(v) - ($nVisited[$s$]$ - \boldsymbol{\gamma}^{k-1}(s)))$;
28             nFinished $\leftarrow$ nFinished $+ 1$;
29             finished[$s$] $\leftarrow true$;
30         end
31     end
32     $k \leftarrow k + 1$;
33 end
34 foreach $v \in v$ do
35     $L^{\mathrm{NB}}(v, r(v)) \leftarrow \frac{(n-1)\tilde{S}^{(\mathrm{un})}}{(r(v)-1)^2}$;
36 end
37 return $L^{NB}(\cdot, r(\cdot))$
```

□

Lower bound on directed graphs. In directed graphs, we can simply consider the out-neighbors, without subtracting the number of nodes discovered in the subtrees of the other neighbors in Equation (5.1). The lower bound (which we still refer to as $S^{\mathrm{NB}}(v, r(v))$) is obtained by replacing Equation (5.1) with the following in Lines 12 and 15 of Algorithm 10:

$$\tilde{\gamma}^{\ell}(s) = \sum_{w \in \mathbf{\Gamma}^1(s)} \tilde{\gamma}^{\ell-1}(w) \tag{5.2}$$

5.4 The `updateBoundsBFSCut` Function

The `updateBoundsBFSCut` function is based on a simple idea: if the k-th biggest farness found until now is x, and if we are performing a BFS from node s to compute its farness $f(s)$, we can stop as soon as we can guarantee that $f(s) \geq x$.

Informally, assume that we have already visited all nodes up to distance ℓ: we can lower bound $S(s) = \sum_{v \in V} \mathrm{dist}(s, v)$ by setting distance $\ell + 1$ to a number of nodes equal to the number of edges "leaving" level ℓ, and distance $\ell + 2$ to all the remaining reachable nodes. Then, this bound yields a lower bound on the farness of s. As soon as this lower bound is bigger than x, the `updateBoundsBFSCut` function may stop; if this condition never occurs, at the end of the BFS we have exactly computed the farness of s.

More formally, the following lemma defines a lower bound $S_\ell^{\mathrm{CUT}}(s, r(s))$ on $S(s)$, which is computable after we have performed a BFS from s up to level ℓ, assuming we know the number $r(s)$ of nodes reachable from s (this assumption is lifted in Section 5.6).

Lemma 5.4. *Given a graph $G = (V, E)$, a node $s \in V$, and an integer $\ell \geq 0$, let $\boldsymbol{N}^\ell(s)$ be the set of nodes at distance at most ℓ from s, $\boldsymbol{n}^\ell(s) = |\boldsymbol{N}^\ell(s)|$, and let $\tilde{\gamma}^{\ell+1}(v)$ be an upper bound on the number of nodes at distance $\ell + 1$ from s. Then,*

$$S(s) \geq S_\ell^{CUT}(s, r(s)) := \sum_{v \in \boldsymbol{N}^\ell(s)} \mathrm{dist}(s, v) - \tilde{\gamma}^{\ell+1}(s) + (\ell + 2)(r(s) - \boldsymbol{n}^\ell(s)).$$

Proof. The sum of all the distances from s is lower bounded by setting the correct distance to all nodes at distance at most ℓ from s, by setting distance $\ell + 1$ to all nodes at distance $\ell + 1$ (there are $\boldsymbol{\gamma}^{\ell+1}(s)$ such nodes), and by setting distance $\ell + 2$ to all other nodes (there are $r(s) - \boldsymbol{n}^{\ell+1}(s)$ such nodes). In formula,

$$S(s) \geq \sum_{v \in \boldsymbol{N}^\ell(s)} \mathrm{dist}(s, v) + (\ell + 1)\boldsymbol{\gamma}^{\ell+1}(s) + (\ell + 2)(r(s) - \boldsymbol{n}^{\ell+1}(s)).$$

Since $\boldsymbol{n}^{\ell+1}(s) = \boldsymbol{\gamma}^{\ell+1}(s) + \boldsymbol{n}^\ell(s)$, we obtain that $S(s) \geq \sum_{v \in \boldsymbol{N}^\ell(s)} \mathrm{dist}(s, v) - \boldsymbol{\gamma}^{\ell+1}(s) + (\ell + 2)(r(s) - \boldsymbol{n}^\ell(s))$. We conclude because, by assumption, $\tilde{\gamma}^{\ell+1}(s)$ is an upper bound on $\boldsymbol{\gamma}^{\ell+1}(s)$.

□

Corollary 5.5. *For each node s and for each $\ell \geq 0$,*

$$f(s) \geq L_\ell^{CUT}(s, r(s)) := \frac{(n-1) S_\ell^{CUT}(s, r(s))}{(r(s) - 1)^2}.$$

It remains to define the upper bound $\tilde{\gamma}^{\ell+1}(s)$: in the directed case, this bound is simply the sum of the out-degrees of nodes at distance ℓ from s. In the undirected case, since at least an edge from each node $v \in \boldsymbol{\Gamma}^\ell(s)$ is directed towards $\boldsymbol{\Gamma}^{\ell-1}(s)$, we may define $\tilde{\gamma}^{\ell+1}(s) = \sum_{v \in \boldsymbol{\Gamma}^\ell(s)} \deg(v) - 1$ (the only exception is $\ell = 0$: in this case, $\tilde{\gamma}^1(s) = \boldsymbol{\gamma}^1(s) = \deg(s)$).

Remark 5.6. When we are processing nodes at level ℓ, if we process an edge (v, w) where w is already in the BFS tree, we can decrease $\tilde{\gamma}^{\ell+1}(s)$ by one, obtaining a better bound.

Assuming we know $r(s)$, all quantities necessary to compute $L_\ell^{\mathrm{CUT}}(s, r(s))$ are available as soon as all nodes in $\boldsymbol{N}^\ell(s)$ are visited by a BFS. Hence, the `updateBoundsBFSCut` function performs a BFS starting from s, continuously updating the upper bound $L_\ell^{\mathrm{CUT}}(s, r(s)) \leq f(s)$ (the update is done whenever all nodes in $\boldsymbol{\Gamma}^\ell(s)$ have been reached, or Remark 5.6 can be used). As soon as $L_\ell^{\mathrm{CUT}}(s, r(s)) \geq x$, we know that $f(s) \geq L_\ell^{\mathrm{CUT}}(s, r(s)) \geq x$, and we return $+\infty$.

Algorithm 11 is the pseudo-code of the function `updateBoundsBFSCut` when implemented for directed graphs, assuming we know the number $r(v)$ of nodes reachable from each v (for example, if the graph is strongly connected). This code can be easily adapted to all the other cases.

Algorithm 11: The `updateBoundsBFSCut`(s, x) function in the case of directed graphs, if $r(s)$ is known for each s.

```
$x \leftarrow \mathsf{Farn}(\mathsf{Top}[k])$; // Farn and Top are global variables, as in Algorithm 8.
Create queue $Q$;
$Q$.enqueue($s$);
Mark $s$ as visited;
$\ell \leftarrow 0$; $S \leftarrow 0$; $\tilde{\gamma} \leftarrow \text{outdeg}(s)$; $n\ell \leftarrow 1$;
while $Q$ is not empty do
    $v \leftarrow Q$.dequeue();
    if $\text{dist}(s, v) > \ell$ then
        $\ell \leftarrow \ell + 1$;
        $L_\ell^{\text{CUT}}(s, r(s)) \leftarrow \frac{(n-1)\left(S-\tilde{\gamma}+(\ell+2)(r(s)-n\ell)\right)}{(r(s)-1)^2}$;
        if $L_\ell^{CUT}(s, r(s)) > x$ then return $+\infty$;
        $\tilde{\gamma} \leftarrow 0$
    end
    for $w$ in adjacency list of $v$ do
        if $w$ is not visited then
            $S \leftarrow S + \text{dist}(s, w)$;
            $\tilde{\gamma} \leftarrow \tilde{\gamma} + \text{outdeg}(w)$;
            $n\ell \leftarrow n\ell + 1$;
            $Q$.enqueue($w$);
            Mark $w$ as visited
        end
        else
            // we use Remark 5.6
            $L_\ell^{\text{CUT}}(s, r(s)) \leftarrow L_\ell^{\text{CUT}}(s, r(s)) + \frac{(n-1)}{(r(s)-1)^2}$;
            if $L_\ell^{CUT}(s, r(s)) > x$ then return $+\infty$;
        end
    end
end
return $\frac{S(n-1)}{(r(s)-1)^2}$;
```

5.5 The `updateBoundsLB` Function

Differently from `updateBoundsBFSCut` function, `updateBoundsLB` computes a complete BFS traversal, but uses information acquired during the traversal to update the lower bounds. Let us first consider an undirected graph G and let s be the source node from which we are computing the BFS. We can see the distances $\text{dist}(s, v)$ between s and all the nodes v reachable from s as *levels*: node v is at level ℓ if and only if the distance between s and v is ℓ, and we write $v \in \mathbf{\Gamma}^\ell(s)$. Let i and j be two levels, $i \leq j$. Then, the distance between any two nodes v at level i and w at level j must be at least $j - i$. Indeed, if $\text{dist}(v, w)$ was smaller than $j - i$, w would be at level $i + \text{dist}(v, w) < j$, which contradicts our assumption. It follows directly that $\sum_{w \in V} |\text{dist}(s, w) - \text{dist}(s, v)|$ is a lower bound on $S(v)$, for all $v \in R(s)$:

Lemma 5.7. $\sum_{w \in R(s)} |\text{dist}(s, w) - \text{dist}(s, v)| \leq S(v) \quad \forall v \in R(s)$.

To improve the approximation, we notice that the number of nodes at distance 1 from v is exactly the degree of v. Therefore, all the other nodes w such that $|\text{dist}(s, v) - \text{dist}(s, w)| \leq 1$ must be at least at distance 2 (with the only exception of v itself, whose distance is of course 0). This way we can define the following lower bound on $S(v)$:

Algorithm 12: The `updateBoundsLB` function for undirected graphs.

```
Input  : A graph $G = (V, E)$, a source node $s$
Output: Lower bounds $L_s^{\mathrm{LB}}(v, r(v))$ of each node $v \in R(s)$
$d \leftarrow$ BFSfrom($s$);
$\mathsf{D} \leftarrow \max_{v \in V} \mathrm{dist}(s, v)$;
$\mathsf{sum}\Gamma_{\leq 0} \leftarrow 0$; $\mathsf{sum}\Gamma_{\leq -1} \leftarrow 0$; $\mathsf{sum}\Gamma_{>\mathsf{D}+1} \leftarrow 0$;
for $i = 1, 2, ..., \mathsf{D}$ do
    $\boldsymbol{\Gamma}^i(s) \leftarrow \{w \in V : \mathrm{dist}(s, w) = i\}$;
    $\boldsymbol{\gamma}^i(s) \leftarrow \boldsymbol{\gamma}^i(s)$;
    $\mathsf{sum}\Gamma_{\leq i} \leftarrow \mathsf{sum}\Gamma_{\leq i-1} + \boldsymbol{\gamma}^i(s)$;
    $\mathsf{sum}\Gamma_{>i} \leftarrow |V| - \mathsf{sum}\Gamma_{\leq i}$;
end
$L(1) \leftarrow \boldsymbol{\gamma}^1(s) + \boldsymbol{\gamma}^2(s) + \mathsf{sum}\Gamma_{>2} - 2$;
for $i = 2, ..., \mathsf{D}$ do
    $L(i) \leftarrow L(i-1) + \mathsf{sum}\Gamma_{\leq i-3} - \mathsf{sum}\Gamma_{>i+1}$;
end
for $i = 1, ..., \mathsf{D}$ do
    foreach $v \in \boldsymbol{\Gamma}^i(s)$ do
        $L_s^{\mathrm{LB}}(v, r(v)) \leftarrow (L(i) - \deg(v)) \cdot \frac{(n-1)}{(r(v)-1)^2}$;
    end
end
return $L_s^{LB}(v, r(v)) \quad \forall v \in V$
```

$$2(|\{w \in R(s) : |\operatorname{dist}(s,w) - \operatorname{dist}(s,v)| \leq 1\}| - \deg(v) - 1) \\ + \deg(v) + \sum_{\substack{w \in R(s) \\ |\operatorname{dist}(s,w) - \operatorname{dist}(s,v)| > 1}} |\operatorname{dist}(s,w) - \operatorname{dist}(s,v)|,$$

that is:

$$2 \cdot \sum_{|j - \operatorname{dist}(s,v)| \leq 1} \boldsymbol{\gamma}^j(s) + \sum_{|j - \operatorname{dist}(s,v)| > 1} \boldsymbol{\gamma}^j(s) \cdot |j - \operatorname{dist}(s,v)| - \deg(v) - 2, \tag{5.3}$$

Multiplying the bound of Equation (5.3) by $\frac{(n-1)}{(r(v)-1)^2}$, we obtain a lower bound on the farness $f(v)$ of node v, named $L_s^{\mathrm{LB}}(v, r(v))$. A straightforward way to compute $L_s^{\mathrm{LB}}(v, r(v))$ would be to first run the BFS from s and then, for each node v, to consider the level difference between v and all the other nodes. This would require $\mathcal{O}(n^2)$ operations, which is clearly too expensive. However, we can notice two things: First, the bounds of two nodes at the same level differ only by their degree. Therefore, for each level i, we can compute $2 \cdot \sum_{|j-i| \leq 1} \boldsymbol{\gamma}^j(v) + \sum_{|j-i|>1} \boldsymbol{\gamma}^j(v) \cdot |j - i| - 2$ only once and then subtract $\deg(v)$ for each node at level i. We call the quantity $2 \cdot \sum_{|j-i| \leq 1} \boldsymbol{\gamma}^j(s) + \sum_{|j-i|>1} \boldsymbol{\gamma}^j(s) \cdot |j - i| - 2$ the level-bound $L(i)$ of level i. Second, we can prove that $L(i)$ can actually be written as a function of $L(i-1)$.

Lemma 5.8. *Let $L(i) := 2 \cdot \sum_{|j-i| \leq 1} \boldsymbol{\gamma}^j(s) + \sum_{|j-i|>1} \boldsymbol{\gamma}^j(s) \cdot |j - i| - 2$. Also, let $\boldsymbol{\gamma}^j(s) = 0$ for $j \leq 0$ and $j > D$, where D is the diameter of the graph. Then $L(i) - L(i-1) = \sum_{j<i-2} \boldsymbol{\gamma}^j(s) - \sum_{j>i+1} \boldsymbol{\gamma}^j(s)$, $\forall i \in \{1, ..., D\}$.*

Proof. Since $\boldsymbol{\gamma}^j(s) = 0$ for $j \leq 0$ and $j > D$, we can write $L(i)$ as $2 \cdot (\boldsymbol{\gamma}^{i-1}(s) + \boldsymbol{\gamma}^i(s) + \boldsymbol{\gamma}^{i+1}(s)) + \sum_{|j-i|>1} \boldsymbol{\gamma}^j(s) \cdot |j - i| - 2$, $\forall i \in \{1, ..., D\}$. The difference between $L(i)$ and $L(i-1)$ is: $2 \cdot (\boldsymbol{\gamma}^{i-1}(s) + \boldsymbol{\gamma}^i(s) + \boldsymbol{\gamma}^{i+1}(s)) + \sum_{|j-i|>1} |j - i| \cdot \boldsymbol{\gamma}^j(s) - 2 \cdot (\boldsymbol{\gamma}^{i-2}(s) + \boldsymbol{\gamma}^{i-1}(s) + \boldsymbol{\gamma}^i(s)) + \sum_{|j-i+1|>1} |j - i + 1| \cdot \boldsymbol{\gamma}^j(s) = 2 \cdot (\boldsymbol{\gamma}^{i+1}(s) - \boldsymbol{\gamma}^{i-2}(s)) + 2 \cdot \boldsymbol{\gamma}^{i-2}(s) - 2 \cdot \boldsymbol{\gamma}^{i+1}(s) + \sum_{j<i-2 \cup j>i+1} (|j - i| - |j - i + 1|) \cdot \boldsymbol{\gamma}^j(s) = \sum_{j<i-2} \boldsymbol{\gamma}^j(s) - \sum_{j>i+1} \boldsymbol{\gamma}^j(s)$. □

Algorithm 12 describes the computation of $L_s^{\mathrm{LB}}(v, r(v))$. First, we compute all the distances between s and the nodes in $R(s)$ with a BFS, storing the number of nodes in each level and the number of nodes in levels $j \leq i$ and $j > i$ respectively (Lines 1 - 9). Then we compute the level bound $L(1)$ of level 1 according to its definition (Line 10) and those of the other level according to Lemma 5.8 (Line 12). The lower bound $L_s^{\mathrm{LB}}(v, r(v))$ is then computed for each node v by subtracting its degree to $L(\mathrm{dist}(s, v))$ and normalizing (Line 16). The complexity of Lines 1 - 9 is that of running a BFS, i.e., $\mathcal{O}(n + m)$. Line 12 is repeated once for each level (which cannot be more than n) and Line 16 is repeated once for each node in $R(s)$. Therefore, the following proposition holds.

Proposition 5.9. *Computing the lower bound $L_s^{LB}(v, r(v))$ takes $\mathcal{O}(n + m)$ time.*

For directed strongly-connected graphs, the result does not hold for nodes w whose level is smaller than the level of v, since there might be a directed edge or a shortcut from v to w. Yet, for nodes w such that $\mathrm{dist}(s, w) > \mathrm{dist}(s, v)$, it is still true that $\mathrm{dist}(v, w) \geq \mathrm{dist}(s, w) - \mathrm{dist}(s, v)$. For the remaining nodes (apart from the outgoing neighbors of v), we can only say that the distance must be at least 2. The upper bound $L_s^{\mathrm{LB}}(v, r(v))$ for directed graphs can therefore be defined as:

$$2 \cdot |\{w \in R(s) : \mathrm{dist}(s, w) - \mathrm{dist}(s, v) \leq 1\}| \\ + \sum_{\substack{w \in R(s) \\ \mathrm{dist}(s,w) - \mathrm{dist}(s,v) > 1}} (\mathrm{dist}(s, w) - \mathrm{dist}(s, v)) - \deg(v) - 2. \quad (5.4)$$

The computation of $L_s^{\mathrm{LB}}(v, r(v))$ for directed strongly-connected graphs is analogous to the one described in Algorithm 12.

5.6 The Directed Disconnected Case

In the directed disconnected case, even if the time complexity of computing strongly connected components is linear in the input size, the time complexity of computing the number of reachable nodes is much larger (assuming the Strong Exponential Time Hypothesis, or the Orthogonal Vector conjecture, we cannot solve this problem in $\mathcal{O}(n^{2-\varepsilon})$ on sparse graphs, as we have shown in Chapter 3 and in [31]). For this reason, when computing our upper bounds, we cannot rely on the exact value of $r(s)$: for now, let us assume that we know a lower bound $\alpha(s) \leq r(s)$ and an upper bound $\omega(s) \geq r(s)$. The definition of these bounds is postponed to Section 5.6.4.

Furthermore, let us assume that we have a lower bound $L(s, r(s))$ on the farness of s, depending on the number $r(s)$ of nodes reachable from s: in order to obtain a bound not depending on $r(s)$, the simplest approach is $f(s) \geq L(s, r(s)) \geq \min_{\alpha(s) \leq r \leq \omega(s)} L(s, r)$. However, during the algorithm, computing the minimum among all these values might be quite expensive, if $\omega(s) - \alpha(s)$ is large. In order to solve this issue, we find a small set $R \subseteq [\alpha(s), \omega(s)]$ such that $\min_{\alpha(s) \leq r \leq \omega(s)} L(s, r) = \min_{r \in R} L(s, r)$.

More specifically, we find a condition that is satisfied by "many" values of r, and that implies $L(s, r) \geq \min\big(L(s, r - 1), L(s, r + 1)\big)$: this way, we may define R as the set of values of r that either do not satisfy this condition, or that are extremal points of the interval $[\alpha(v), \omega(v)]$ (indeed, all other values cannot be minima of $L(s, r)$). Since all our bounds are of the form $L(s, r) = \frac{(n-1)S(s,r)}{(r-1)^2}$, where $S(s, r)$ is a lower bound on $\sum_{v \in R(s)} \mathrm{dist}(s, v)$, we state our condition in terms of the function $S(s, r)$. For instance, in the case of the `updateBoundsBFSCut` function, $S_\ell^{\mathrm{CUT}}(s, r) = \sum_{v \in \boldsymbol{N}^d(s)} \mathrm{dist}(s, v) - \tilde{\gamma}^{\ell+1}(s) + (\ell + 2)(r - \boldsymbol{n}^\ell(s))$, as in Lemma 5.4.

Lemma 5.10. *Let s be a node, and let $S(s, r)$ be a positive function such that $S(s, r(s))) \leq \sum_{v \in R(s)} \mathrm{dist}(s, v)$ (where $r(s)$ is the number of nodes reachable from s). Assume that $S(s, r + 1) - S(s, r) \leq S(s, r) - S(s, r - 1)$. Then, if $L(s, r) := \frac{(n-1)S(s,r)}{(r-1)^2}$ is the corresponding bound on the farness of s, $\min\big(L(s, r + 1), L(s, r - 1)\big) \leq L(s, r)$.*

Proof. Let us define $d = S(s,r+1) - S(s,r)$. Then, $L(s,r+1) \leq L(s,r)$ if and only if $\frac{(n-1)S(s,r+1)}{r^2} \leq \frac{(n-1)S(s,r)}{(r-1)^2}$ if and only if $\frac{S(s,r)+d}{r^2} \leq \frac{S(s,r)}{(r-1)^2}$ if and only if $(r-1)^2(S(s,r)+d) \leq r^2 S(s,r)$ if and only if $S(s,r)(r^2-(r-1)^2) \geq (r-1)^2 d$ if and only if $S(s,r)(2r-1) \geq (r-1)^2 d$.

Similarly, if $d' = S(s,r) - S(s,r-1)$, $L(s,r-1) \leq L(s,r)$ if and only if $\frac{(n-1)S(s,r-1)}{(r-2)^2} \leq \frac{(n-1)S(s,r)}{(r-1)^2}$ if and only if $\frac{S(s,r)-d'}{(r-2)^2} \leq \frac{S(s,r)}{(r-1)^2}$ if and only if $(r-1)^2(S(s,r)-d') \leq (r-2)^2 S(s,r)$ if and only if $S(s,r)((r-1)^2-(r-2)^2) \leq (r-1)^2 d'$ if and only if $S(s,r)(2r-3) \leq (r-1)^2 d'$ if and only if $S(s,r)(2r-1) \leq (r-1)^2 d' + 2S(s,r)$.

We conclude that, assuming $d \leq d'$, $(r-1)^2 d \leq (r-1)^2 d' \leq (r-1)^2 d + 2S(s,r)$, and one of the two previous conditions is always satisfied. □

5.6.1 The `computeBoundsNB` Function

In the neighborhood-based lower bound, we computed upper bounds $\tilde{\gamma}^{\ell}(s)$ on $\mathbf{\Gamma}^{\ell}(s)$, and we defined the lower bound $S^{\mathrm{NB}}(s,r(s)) \leq \sum_{v \in R(s)} \mathrm{dist}(s,v)$, by

$$S^{\mathrm{NB}}(s,r(s)) := \sum_{\ell=1}^{D} \ell \cdot \min \left\{ \tilde{\gamma}^{\ell}(s),\ r(s) - \sum_{i=0}^{\ell-1} \tilde{\gamma}^{i}(s),\ 0 \right\}.$$

The corresponding bound on $f(v)$ is $L^{\mathrm{NB}}(s,r(s)) := \frac{(n-1)S^{\mathrm{NB}}(s,r(s))}{(r(s)-1)^2}$: let us apply Lemma 5.10 with $S(s,r) = S^{\mathrm{NB}}(s,r)$ and $L(s,r) = L^{\mathrm{NB}}(s,r)$. We obtain that the local minima of $L^{\mathrm{NB}}(s,r(s))$ are obtained on values r such that $S^{\mathrm{NB}}(s,r+1) - S^{\mathrm{NB}}(s,r) > S^{\mathrm{NB}}(s,r) - S^{\mathrm{NB}}(s,r-1)$, that is, when $r = \sum_{i=0}^{\ell} \tilde{\gamma}^{i}(s)$ for some ℓ. Hence, our final bound $L^{\mathrm{NB}}(s)$ becomes:

$$\min \left(L^{\mathrm{NB}}(s,\alpha(s)), L^{\mathrm{NB}}(s,\omega(s)), \min \left\{ L^{\mathrm{NB}}(s,r) : \alpha(s) < r < \omega(s), r = \sum_{i=0}^{\ell} \tilde{\gamma}^{i}(s) \right\} \right). \tag{5.5}$$

This bound can be computed with no overhead, by modifying Lines 23 - 29 in Algorithm 10. Indeed, when $r(s)$ is known, we have two cases: either `nVisited[s]` $< r(s)$, and we continue, or `nVisited[s]` $\geq r(s)$, and $S^{\mathrm{NB}}(v,r(s))$ is computed. In the disconnected case, we need to distinguish three cases:

- if `nVisited[s]` $< \alpha(s)$, we simply continue the computation;
- if $\alpha(s) \leq$ `nVisited[s]` $< \omega(s)$, we compute $L^{\mathrm{NB}}(s,$`nVisited[s]`$)$, and we update the minimum in Equation (5.5) (if this is the first occurrence of this situation, we also have to compute $L^{\mathrm{NB}}(s,\alpha(s))$);
- if `nVisited[s]` $\geq \omega(s)$, we compute $L^{\mathrm{NB}}(s,\omega(s))$, and we update the minimum in Equation (5.5).

Since this procedure needs time $\mathcal{O}(1)$, it has no impact on the running time of the computation of the neighborhood-based lower bound.

5.6.2 The `updateBoundsBFSCut` Function

Let us apply Lemma 5.10 to the bound used in the `updateBoundsBFSCut` function. In this case, by Lemma 5.4, $S_{\ell}^{\mathrm{CUT}}(s,r) = \sum_{v \in \boldsymbol{N}^{\ell}(s)} \mathrm{dist}(s,v) - \tilde{\gamma}^{\ell+1}(v) + (\ell+2)(r - \boldsymbol{n}^{\ell}(s))$, and

$S_\ell^{\mathrm{CUT}}(s,r+1) - S_\ell^{\mathrm{CUT}}(s,r) = d+2$, which does not depend on r. Hence, the condition in Lemma 5.10 is always satisfied, and the only values we have to analyze are $\alpha(s)$ and $\omega(s)$. Then, the lower bound becomes $f(s) \geq L_\ell^{\mathrm{CUT}}(s,r(s)) \geq \min_{\alpha(s)\leq r\leq\omega(s)} L_\ell^{\mathrm{CUT}}(s,r) = \min(L_\ell^{\mathrm{CUT}}(s,\alpha(s)), L_\ell^{\mathrm{CUT}}(s,\omega(s)))$ (which does not depend on $r(s)$).

This means that, in order to adapt the `updateBoundsBFSCut` function (Algorithm 11), it is enough to replace Lines 10, 24 in order to compute both $L_\ell^{\mathrm{CUT}}(s,\alpha(s))$ and $L_\ell^{\mathrm{CUT}}(s,\omega(s)))$, and to replace Lines 11, 25 in order to stop if $\min(L_\ell^{\mathrm{CUT}}(s,\alpha(s)), L_\ell^{\mathrm{CUT}}(s,\omega(s))) \geq x$.

5.6.3 The `updateBoundsLB` Function

In this case, we do not apply Lemma 5.10 to obtain simpler bounds. Indeed, the `updateBoundsLB` function improves the bounds of nodes that are quite close to the source of the BFS, and hence are likely to be in the same component as this node. Consequently, if we perform a BFS from a node s, we can simply compute $L_s^{\mathrm{LB}}(v,r(v))$ for all nodes in the same strongly connected component as s, and for these nodes we know the value $r(v) = r(s)$. The computation of better bounds for other nodes is left as an open problem.

5.6.4 Computing $\alpha(s)$ and $\omega(s)$

The computation of $\alpha(s)$ and $\omega(s)$ can be done during the preprocessing phase of our algorithm, in linear time. To this purpose, let us consider the strong component graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$ of G (see Chapter 2). Furthermore, let us define the weight $w(C)$ of a SCC C as the number of nodes in C. Note that the weighted graph $\mathcal{G}$ is computable in linear time.

For each node $v \in C$, $r(v) = \sum_{D\in R(C)} w(D)$, where $R(C)$ denotes the set of SCCs that are reachable from C in $\mathcal{G}$. This means that we simply need to compute a lower (respectively, upper) bound $\alpha_{SCC}(C)$ (respectively, $\omega_{SCC}(C)$) on $\sum_{D\in\mathcal{R}(C)} w(D)$, for every SCC C. To this aim, we first compute a topological sort $\{C_1,\dots,C_l\}$ of $\mathcal{V}$ (that is, if $(C_i,C_j)\in\mathcal{E}$, then $i<j$). Successively, we use a dynamic programming approach, and, by starting from C_l, we process the SCCs in reverse topological order, and we set:

$$\alpha_{SCC}(C) = w(C) + \max_{(C,D)\in\mathcal{E}} \alpha_{SCC}(D)$$
$$\omega_{SCC}(C) = w(C) + \sum_{(C,D)\in\mathcal{E}} \omega_{SCC}(D).$$

Note that processing the SCCs in reverse topological ordering ensures that the values $\alpha(D)$ and $\omega(D)$ on the right hand side of these equalities are available when we process the SCC C. Clearly, the complexity of computing $\alpha(C)$ and $\omega(C)$, for each SCC C, is linear in the size of $\mathcal{G}$, which in turn is smaller than G.

Observe that the bounds obtained through this simple approach can be improved by using some "tricks". First of all, when the biggest SCC $\tilde{C}$ is processed, we do not use the dynamic programming approach and we exactly compute $\sum_{D\in\mathcal{R}(\tilde{C})} w(D)$ by performing a BFS starting from any node in $\tilde{C}$. This way, not only $\alpha(\tilde{C})$ and $\omega(\tilde{C})$ are exact, but also $\alpha_{SCC}(C)$ and $\omega_{SCC}(C)$ are improved for each SCC C from which it is possible to reach $\tilde{C}$. Finally, in order to compute the upper bounds for the SCCs that are able to reach $\tilde{C}$, we can run the dynamic programming algorithm on the graph obtained from $\mathcal{G}$ by removing all components reachable from $\tilde{C}$, and we can then add $\sum_{D\in\mathcal{R}(\tilde{C})} w(D)$.

The pseudo-code is available in Algorithm 13.

5.7 Experimental Results

In this section, we test the four variations of our algorithm on several real-world networks, in order to evaluate their performances. All the networks used in our experiments come from the datasets SNAP (`snap.stanford.edu/`), NEXUS (`nexus.igraph.org`),

Algorithm 13: Estimating the number of reachable nodes in directed, disconnected graphs.

```
Input  : A graph $G = (V, E)$
Output: Lower and upper bounds $\alpha(v), \omega(v)$ on the number of nodes reachable from $v$
$(\mathcal{V}, \mathcal{E}, w) \leftarrow$ computeSCCGraph($G$);
$\tilde{C} \leftarrow$ the biggest SCC;
$\alpha_{SCC}(\tilde{C}), \omega_{SCC}(\tilde{C}) \leftarrow$ the number of nodes reachable from $\tilde{C}$;
for $X \in \mathcal{V}$ in reverse topological order do
    if $X == \tilde{C}$ then continue;
    $\alpha_{SCC}(X), \omega_{SCC}(X), \omega'_{SCC}(X) \leftarrow 0$ for $Y$ neighbor of $X$ in $\mathcal{G}$ do
        $\alpha_{SCC}(X) \leftarrow \max(\alpha_{SCC}(X), \alpha_{SCC}(Y))$;
        $\omega_{SCC}(X) \leftarrow \omega_{SCC}(X) + \omega_{SCC}(Y)$;
        if $W$ not reachable from $\tilde{C}$ then $\omega'_{SCC}(X) \leftarrow \omega'_{SCC}(X) + \omega_{SCC}(Y)$;
    end
    if $X$ reaches $\tilde{C}$ then $\omega_{SCC}(X) \leftarrow \omega'_{SCC}(X) + \omega_{SCC}(\tilde{C})$;
    $\alpha_{SCC}(X) \leftarrow \alpha_{SCC}(X) + w(X)$;
    $\omega_{SCC}(X) \leftarrow \omega_{SCC}(X) + w(X)$;
end
for $v \in V$ do
    $\alpha(v) = \alpha_{SCC}$(the component of $v$);
    $\omega(v) = \omega_{SCC}$(the component of $v$);
end
return $\alpha, \omega$
```

LASAGNE (`piluc.dsi.unifi.it/lasagne`), LAW (`law.di.unimi.it`), KONECT (`http://konect.uni-koblenz.de/networks/`, and IMDB (`www.imdb.com`). The platform for our tests is a shared-memory server with 256 GB RAM and 2x8 Intel(R) Xeon(R) E5-2680 cores (32 threads due to hyperthreading) at 2.7 GHz. The algorithms are implemented in C++, and they are available in the open-source *NetworKit* framework [151] (other implementations are available in WebGraph [23] and Sagemath [152]).

5.7.1 Comparison with the State of the Art

In order to compare the performance of our algorithm with state of the art approaches, we select 19 directed complex networks, 17 undirected complex networks, 6 directed road networks, and 6 undirected road networks (the undirected versions of the previous ones). The number of nodes of most of these networks ranges between 5 000 and 100 000. We test four different variations of our algorithm, that provide different implementations of the functions `computeBounds` and `updateBounds` (for more information, we refer to Section 5.2):

DegCut uses the conservative strategies `computeBoundsDeg` and `updateBoundsBFSCut`;

DegBound uses the conservative strategy `computeBoundsDeg` and the aggressive strategy `updateBoundsLB`;

NBCut uses the aggressive strategy `computeBoundsNB` and the conservative strategy `updateBoundsBFSCut`;

NBBound uses the aggressive strategies `computeBoundsNB` and `updateBoundsLB`.

We compare these algorithms with our implementations of the best existing algorithms for top-k closeness centrality.[1] The first one [129] is based on a pruning technique and on Δ-BFS, a method to reuse information collected during a BFS from a node to speed up a BFS from one of its in-neighbors; we denote this algorithm as Olh. The second one, Ocl, provides top-k closeness centralities with high probability [128]. It performs some BFSes from

[1] The source code of our competitors is not available.

Table 5.1. Complex networks: geometric mean and standard deviation of the improvement factors of the algorithm in [129] (Olh), the algorithm in [128] (Ocl), and the four variations of the new algorithm (DegCut, DegBound, NBCut, NBBound).

		Directed		Undirected		Both	
k	Algorithm	GMean	GStdDev	GMean	GStdDev	GMean	GStdDev
1	Olh	21.24	5.68	11.11	2.91	15.64	4.46
	Ocl	1.71	1.54	2.71	1.50	2.12	1.61
	DegCut	104.20	6.36	171.77	6.17	131.94	6.38
	DegBound	3.61	3.50	5.83	8.09	4.53	5.57
	NBCut	**123.46**	7.94	**257.81**	8.54	**174.79**	8.49
	NBBound	17.95	10.73	56.16	9.39	30.76	10.81
10	Olh	21.06	5.65	11.11	2.90	15.57	4.44
	Ocl	1.31	1.31	1.47	1.11	1.38	1.24
	DegCut	56.47	5.10	60.25	4.88	58.22	5.00
	DegBound	2.87	3.45	2.04	1.45	2.44	2.59
	NBCut	**58.81**	5.65	**62.93**	5.01	**60.72**	5.34
	NBBound	9.28	6.29	10.95	3.76	10.03	5.05
100	Olh	20.94	5.63	11.11	2.90	15.52	4.43
	Ocl	1.30	1.31	1.46	1.11	1.37	1.24
	DegCut	22.88	4.70	15.13	3.74	18.82	4.30
	DegBound	2.56	3.44	1.67	1.36	2.09	2.57
	NBCut	**23.93**	4.83	**15.98**	3.89	**19.78**	4.44
	NBBound	4.87	4.01	4.18	2.46	4.53	3.28

a random sample of nodes to estimate the closeness centrality of all the other nodes, then it computes the exact centrality of all the nodes whose estimate is big enough. Note that this algorithm requires the input graph to be (strongly) connected: for this reason, differently from the other algorithms, we have run this algorithm on the largest (strongly) connected component of the input graph. Furthermore, this algorithm offers different tradeoffs between the time needed by the sampling phase and the second phase: in our tests, we try all possible tradeoffs, and we choose the best alternative in each input graph (hence, our results are upper bounds on the real performance of the Ocl algorithm).

In order to perform a fair comparison, we consider the *improvement factor*, which is defined as $\frac{mn}{m_{\text{vis}}}$ in directed graphs, $\frac{2mn}{m_{\text{vis}}}$ in undirected graphs, where m_{vis} is the number of arcs visited during the algorithm, and mn (resp., $2mn$) is an estimate of the number of arcs visited by the *textbook* algorithm in directed (resp., undirected) graphs (this estimate is correct whenever the graph is connected). Note that the improvement factor does not depend on the implementation, nor on the machine used for the algorithm, and it does not consider parts of the code that need subquadratic time in the worst-case. These parts are negligible in our algorithm, because their worst-case running time is $\mathcal{O}(n \log n)$ or $\mathcal{O}(mD)$ where D is the diameter of the graph, but they can be significant when considering the competitors. For instance, in the particular case of Olh, we have just counted the arcs visited in BFS and Δ-BFS, ignoring all the operations done in the pruning phases (see [129]).

We consider the geometric mean of the improvement factors over all graphs in the dataset. In our opinion, this quantity is more informative than the arithmetic mean, which is highly influenced by the maximum value: for instance, in a dataset of 20 networks, if all improvement factors are 1 apart from one, which is 10 000, the arithmetic mean is more than 500, which makes little sense, while the geometric mean is about 1.58. Our choice is further confirmed by the geometric standard deviation, which is always quite small.

The results are summarized in Table 5.1 for complex networks and Table 5.2 for street networks. For the improvement factors of all graphs, we refer to Section 5.7.3.

On complex networks, the best algorithm is NBCut: when $k = 1$, the improvement factors are always bigger than 100, up to 258, when $k = 10$ they are close to 60, and when $k = 100$ they are close to 20. Another good option is DegCut, which achieves results similar to NBCut, but it has almost no overhead at the beginning (while NBCut needs a preprocessing phase with cost $\mathcal{O}(mD)$). Furthermore, DegCut is very easy to implement, becoming a very good candidate for state-of-the-art graph libraries. The improvement factors of the competitors are smaller: Olh has improvement factors between 10 and 20, and Ocl provides almost no improvement with respect to the *textbook* algorithm.

Table 5.2. Street networks: geometric mean and standard deviation of the improvement factors of the algorithm in [129] (Olh), the algorithm in [128] (Ocl), and the four variations of the new algorithm (DegCut, DegBound, NBCut, NBBound).

		Directed		Undirected		Both	
k	Algorithm	GMean	GStdDev	GMean	GStdDev	GMean	GStdDev
1	Olh	4.11	1.83	4.36	2.18	4.23	2.01
	Ocl	3.39	1.28	3.23	1.28	3.31	1.28
	DegCut	4.14	2.07	4.06	2.06	4.10	2.07
	DegBound	187.10	1.65	272.22	1.67	225.69	1.72
	NBCut	4.12	2.07	4.00	2.07	4.06	2.07
	NBBound	**250.66**	1.71	**382.47**	1.63	**309.63**	1.74
10	Olh	4.04	1.83	4.28	2.18	4.16	2.01
	Ocl	2.93	1.24	2.81	1.24	2.87	1.24
	DegCut	4.09	2.07	4.01	2.06	4.05	2.07
	DegBound	172.06	1.65	245.96	1.68	205.72	1.72
	NBCut	4.08	2.07	3.96	2.07	4.02	2.07
	NBBound	**225.26**	1.71	**336.47**	1.68	**275.31**	1.76
100	Olh	4.03	1.82	4.27	2.18	4.15	2.01
	Ocl	2.90	1.24	2.79	1.24	2.85	1.24
	DegCut	3.91	2.07	3.84	2.07	3.87	2.07
	DegBound	123.91	1.56	164.65	1.67	142.84	1.65
	NBCut	3.92	2.08	3.80	2.09	3.86	2.08
	NBBound	**149.02**	1.59	**201.42**	1.69	**173.25**	1.67

We also test our algorithm on the three complex unweighted networks analyzed in [129], respectively called `web-Google` (`Web` in [129]), `wiki-Talk` (`Wiki` in [129]), and `com-dblp` (`DBLP` in [129]). In the `com-dblp` graph (resp. `web-Google`), our algorithm NBCut computed the top-10 nodes in about 17 seconds (resp., less than 2 minutes) on the whole graph, having 1 305 444 nodes (resp., 875 713), while Olh needed about 25 minutes (resp. 4 hours) on a subgraph of 400 000 nodes. In the graph `wiki-Talk`, NBCut needed 8 seconds for the whole graph having 2 394 385 nodes, instead of about 15 minutes on a subgraph with 1 million nodes. These results are available in Table 5.7 in Section 5.7.3.

On street networks, the best option is NBBound: for $k = 1$, the average improvement is about 250 in the directed case and about 382 in the undirected case, and it always remains bigger than 150, even for $k = 100$. It is worth noting that also the performance of DegBound are quite good, being at least 70% of NBBound. Even in this case, the DegBound algorithm offers some advantages: it is very easy to be implemented, and there is no overhead in the first part of the computation. All the competitors perform relatively poorly on street networks, since their improvement is always smaller than 5.

Overall, we conclude that the preprocessing function `computeBoundsNB` always leads to better results (in terms of visited edges) than `computeBoundsDeg`, but the difference is quite small: hence, in some cases, `computeBoundsDeg` could be even preferred, because of its simplicity. Conversely, the performance of `updateBoundsBFSCut` is very different from the performance of `updateBoundsLB`: the former works much better on complex networks, while the latter works much better on street networks. Currently, these two approaches exclude each other: an open problem left by this work is the design of a “combination” of the two, that works both in complex networks and in street networks. Finally, the experiments show that the best variation of our algorithm outperforms all competitors in all frameworks considered: both in complex and in street networks, both in directed and undirected graphs.

5.7.2 Real-World Large Networks

In this section, we run our algorithm on bigger inputs, by considering a dataset containing 23 directed networks, 15 undirected networks, and 5 road networks, with up to 3 774 768 nodes and 117 185 083 edges. On this dataset, we run the fastest variant of our algorithm (DegBound in complex networks, NBBound in street networks), using 64 threads (however, the server used only runs 16 threads, or 32 with hyperthreading).

Once again, we consider the *improvement factor*, which is defined as $\frac{mn}{m_{\text{vis}}}$ in directed graphs, $\frac{2mn}{m_{\text{vis}}}$ in undirected graphs. It is worth observing that we are able to compute for the first time the k most central nodes of networks with millions of nodes and hundreds of

Table 5.3. Large networks: geometric mean and standard deviation of the improvement factors of the best variation of the new algorithm (DegBound in complex networks, NBBound in street networks).

		Directed		Undirected		Both	
Input	k	GMean	GStdDev	GMean	GStdDev	GMean	GStdDev
Street	1	742.42	2.60	1681.93	2.88	1117.46	2.97
	10	724.72	2.67	1673.41	2.92	1101.25	3.03
	100	686.32	2.76	1566.72	3.04	1036.95	3.13
Complex	1	247.65	11.92	551.51	10.68	339.70	11.78
	10	117.45	9.72	115.30	4.87	116.59	7.62
	100	59.96	8.13	49.01	2.93	55.37	5.86

millions of arcs, with $k = 1$, $k = 10$, and $k = 100$. The detailed results are shown in Table 5.7 in Section 5.7.3, where for each network we report the running time and the improvement factor. A summary of these results is available in Table 5.3, which contains the geometric means of the improvement factors, with the corresponding standard deviations.

For $k = 1$, the geometric mean of the improvement factors is always above 200 in complex networks, and above 700 in street networks. In undirected graphs, the improvement factors are even bigger: close to 500 in complex networks and close to 1 600 in street networks. For bigger values of k, the performance does not decrease significantly: on complex networks, the improvement factors are bigger than or very close to 50, even for $k = 100$. In street networks, the performance loss is even smaller, always below 10% for $k = 100$.

Regarding the robustness, we outline that the algorithm always achieves performance improvements bigger than $\sqrt{n}$ in street networks, and that in complex networks, with $k = 1$, 64% of the networks have improvement factors above 100, and 33% of the networks above 1 000. In some cases, the improvement factor is even bigger: in the `com-Orkut` network, our algorithm for $k = 1$ is almost 35 000 times faster than the *textbook* algorithm.

In our experiments, we also report the running time of our algorithm. Even for $k = 100$, a few minutes are sufficient to conclude the computation on most networks, and, in all but two cases, the total time is smaller than 3 hours. For $k = 1$, the computation always terminates in at most 1 hour and a half, apart from two street networks where it needs less than 2 hours and a half. Overall, the total time needed to compute the most central node in all the networks is smaller than 1 day. This is quite impressive if we consider that many input graphs have millions of nodes, and tens of millions of edges.

5.7.3 Detailed Experimental Results

In this section, we report the detailed results of our algorithm on each graph in our dataset, together with the results of the competitors.

Table 5.4. Detailed comparison of the improvement factors, with $k = 1$.

Directed Street

Network	Olh	Ocl	DegCut	DegBound	NBCut	NBBound
faroe-islands	4.080	3.742	4.125	338.011	4.086	437.986
liechtenstein	2.318	2.075	2.114	130.575	2.115	137.087
isle-of-man	2.623	3.740	2.781	224.566	2.769	314.856
malta	5.332	4.351	4.147	73.836	4.141	110.665
belize	2.691	3.969	2.606	253.866	2.595	444.849
azores	13.559	3.038	19.183	230.939	19.164	266.488

Undirected Street

Network	Olh	Ocl	DegCut	DegBound	NBCut	NBBound
faroe-islands	4.126	3.276	4.118	361.593	3.918	444.243
liechtenstein	2.318	2.027	2.107	171.252	2.122	183.240
isle-of-man	2.613	3.661	2.767	266.734	2.676	370.194
malta	4.770	4.164	3.977	122.729	3.958	232.622
belize	2.565	3.945	2.510	340.270	2.481	613.778
azores	22.406	2.824	18.654	589.985	18.810	727.528

Directed Complex

Network	Olh	Ocl	DegCut	DegBound	NBCut	NBBound
polblogs	3.201	1.131	31.776	1.852	31.974	5.165
out.opsahl-openflights	13.739	1.431	73.190	2.660	73.888	18.255
ca-GrQc	9.863	1.792	36.673	3.630	38.544	6.307
out.subelj_jung-j_jung-j	125.219	1.203	79.559	1.024	79.882	1.897
p2p-Gnutella08	5.696	1.121	66.011	4.583	81.731	6.849
out.subelj_jdk_jdk	116.601	1.167	74.300	1.023	74.527	1.740
wiki-Vote	9.817	2.760	261.242	1.479	749.428	395.278
p2p-Gnutella09	5.534	1.135	41.214	4.650	43.236	6.101
ca-HepTh	7.772	2.121	40.068	3.349	42.988	5.217
freeassoc	33.616	1.099	12.638	2.237	12.700	2.199
ca-HepPh	7.682	2.836	10.497	3.331	10.516	4.387
out.lasagne-spanishbook	13.065	2.553	1871.296	7.598	6786.506	3160.750
out.cfinder-google	16.725	1.782	38.321	2.665	25.856	3.020
ca-CondMat	7.382	3.526	409.772	5.448	517.836	29.282
out.subelj_cora_cora	14.118	1.700	14.098	1.345	14.226	2.299
out.ego-twitter	2824.713	1.000	1870.601	28.995	3269.183	278.214
out.ego-gplus	722.024	1.020	3481.943	236.280	3381.029	875.111
as-caida20071105	20.974	3.211	2615.115	1.737	2837.853	802.273
cit-HepTh	4.294	3.045	16.259	1.514	16.398	3.290

Undirected Complex

Network	Olh	Ocl	DegCut	DegBound	NBCut	NBBound
HC-BIOGRID	5.528	1.581	15.954	3.821	14.908	3.925
facebook_combined	10.456	3.726	56.284	18.786	56.517	98.512
Mus_musculus	18.246	1.743	70.301	3.253	104.008	7.935
Caenorhabditis_elegans	11.446	2.258	86.577	2.140	110.677	9.171
ca-GrQc	6.567	1.904	38.279	3.551	41.046	6.824
as20000102	19.185	2.402	1550.351	3.213	1925.916	498.000
advogato	8.520	2.018	315.024	18.181	323.163	142.654
p2p-Gnutella09	3.744	2.336	90.252	1.708	100.427	13.846
hprd_pp	6.543	2.397	392.853	2.091	407.261	63.953
ca-HepTh	7.655	2.075	42.267	3.308	46.326	5.593
Drosophila_melanogaster	5.573	2.346	69.457	1.822	75.456	6.904
oregon1_010526	20.474	3.723	1603.739	2.703	1798.822	399.071
oregon2_010526	17.330	4.748	1138.475	2.646	1227.105	520.955
Homo_sapiens	6.689	2.700	1475.113	1.898	1696.909	130.381
GoogleNw	15.591	8.389	107.902	15763.000	15763.000	15763.000
dip20090126_MAX	2.883	3.826	5.833	6.590	5.708	7.392
com-amazon.all.cmty	415.286	2.499	5471.982	3.297	8224.693	373.294

Table 5.5. Detailed comparison of the improvement factors, with $k = 10$.

Directed Street

Network	Olh	Ocl	DegCut	DegBound	NBCut	NBBound
faroe-islands	3.713	2.884	4.037	290.626	4.025	361.593
liechtenstein	2.318	2.002	2.104	111.959	2.106	116.713
isle-of-man	2.623	2.933	2.711	209.904	2.720	288.123
malta	5.325	3.861	4.094	70.037	4.086	101.546
belize	2.690	3.638	2.592	244.275	2.580	416.210
azores	13.436	2.644	19.043	222.073	19.045	254.206

Undirected Street

Network	Olh	Ocl	DegCut	DegBound	NBCut	NBBound
faroe-islands	3.702	2.594	4.046	320.588	3.848	388.713
liechtenstein	2.316	1.965	2.097	142.047	2.114	150.608
isle-of-man	2.612	2.889	2.695	241.431	2.636	323.185
malta	4.768	3.615	3.920	115.574	3.910	208.192
belize	2.564	3.634	2.496	323.257	2.469	563.820
azores	22.392	2.559	18.541	539.032	18.712	653.372

Directed Complex

Network	Olh	Ocl	DegCut	DegBound	NBCut	NBBound
polblogs	3.199	1.039	13.518	1.496	13.544	2.928
out.opsahl-openflights	13.739	1.130	32.297	1.984	32.405	6.867
ca-GrQc	9.863	1.356	25.238	3.096	25.786	4.565
out.subelj_jung-j_jung-j	124.575	1.000	79.284	1.024	79.657	1.884
p2p-Gnutella08	5.684	1.064	12.670	3.241	12.763	3.599
out.subelj_jdk_jdk	116.228	1.000	74.106	1.023	74.363	1.730
wiki-Vote	9.812	1.205	166.941	1.453	174.775	25.411
p2p-Gnutella09	5.532	1.084	16.293	3.624	16.265	4.213
ca-HepTh	7.772	1.586	31.314	3.013	32.604	4.356
freeassoc	33.414	1.034	10.612	2.210	10.704	2.178
ca-HepPh	7.682	2.077	10.322	3.042	10.340	4.010
out.lasagne-spanishbook	13.063	1.483	303.044	1.067	351.262	94.351
out.cfinder-google	16.725	1.413	36.364	2.665	24.765	3.017
ca-CondMat	7.382	2.318	91.209	3.507	93.548	7.027
out.subelj_cora_cora	13.699	1.287	12.763	1.334	12.909	2.072
out.ego-twitter	2689.884	1.000	1817.032	28.157	2872.213	218.411
out.ego-gplus	722.024	1.000	951.983	201.949	1085.361	482.204
as-caida20071105	20.974	1.615	997.996	1.371	1266.443	448.729
cit-HepTh	4.030	2.179	11.361	1.486	11.423	2.832

Undirected Complex

Network	Olh	Ocl	DegCut	DegBound	NBCut	NBBound
HC-BIOGRID	5.528	1.240	10.714	3.102	10.036	3.058
facebook_combined	10.456	1.292	9.103	2.236	9.371	2.694
Mus_musculus	18.246	1.316	18.630	2.279	20.720	3.288
Caenorhabditis_elegans	11.445	1.405	58.729	1.904	68.905	7.605
ca-GrQc	6.567	1.340	26.050	3.052	26.769	5.011
as20000102	19.185	1.529	196.538	1.314	209.674	52.210
advogato	8.520	1.405	131.173	2.043	132.207	11.155
p2p-Gnutella09	3.744	1.632	79.093	1.623	87.357	12.941
hprd_pp	6.543	1.436	47.945	1.837	47.866	8.620
ca-HepTh	7.655	1.546	32.612	2.961	34.407	4.677
Drosophila_melanogaster	5.573	1.672	50.840	1.646	54.637	5.743
oregon1_010526	20.474	1.451	418.099	1.282	429.161	109.549
oregon2_010526	17.330	1.560	364.277	1.302	371.929	71.186
Homo_sapiens	6.689	1.599	81.496	1.620	82.250	15.228
GoogleNw	15.591	1.320	23.486	1.252	23.053	2.420
dip20090126_MAX	2.881	1.836	4.055	4.556	4.065	4.498
com-amazon.all.cmty	414.765	1.618	3407.016	3.279	3952.370	199.386

Table 5.6. Detailed comparison of the improvement factors, with $k = 100$.

Directed Street

Network	Olh	Ocl	DegCut	DegBound	NBCut	NBBound
faroe-islands	3.713	2.823	3.694	150.956	3.691	168.092
liechtenstein	2.318	1.998	2.078	84.184	2.086	86.028
isle-of-man	2.620	2.902	2.551	139.139	2.567	167.808
malta	5.282	3.850	3.933	56.921	3.942	76.372
belize	2.688	3.617	2.526	184.718	2.516	268.634
azores	13.334	2.628	18.380	194.724	18.605	220.013

Undirected Street

Network	Olh	Ocl	DegCut	DegBound	NBCut	NBBound
faroe-islands	3.702	2.548	3.693	159.472	3.523	171.807
liechtenstein	2.311	1.959	2.072	96.782	2.095	99.768
isle-of-man	2.607	2.847	2.533	153.859	2.468	183.982
malta	4.758	3.605	3.745	89.929	3.730	137.538
belize	2.562	3.629	2.428	226.582	2.406	323.257
azores	22.345	2.548	18.092	411.760	18.384	476.253

Directed Complex

Network	Olh	Ocl	DegCut	DegBound	NBCut	NBBound
polblogs	3.198	1.037	3.951	1.245	3.961	1.731
out.opsahl-openflights	13.739	1.124	5.524	1.456	5.553	1.740
ca-GrQc	9.863	1.339	11.147	2.353	10.407	2.926
out.subelj_jung-j_jung-j	123.393	1.000	78.473	1.021	78.798	1.787
p2p-Gnutella08	5.684	1.063	6.611	2.935	7.750	3.278
out.subelj_jdk_jdk	114.210	1.000	73.522	1.020	73.755	1.669
wiki-Vote	9.812	1.186	61.375	1.236	60.475	9.436
p2p-Gnutella09	5.531	1.083	6.370	3.109	7.650	3.508
ca-HepTh	7.772	1.570	16.135	2.477	16.747	3.135
freeassoc	33.266	1.032	6.314	2.154	6.428	2.138
ca-HepPh	7.682	2.032	9.605	2.549	9.619	3.340
out.lasagne-spanishbook	13.063	1.467	56.689	1.043	80.069	33.271
out.cfinder-google	16.725	1.392	13.521	2.655	12.298	2.722
ca-CondMat	7.382	2.288	16.884	2.602	16.950	2.824
out.subelj_cora_cora	13.231	1.280	11.171	1.315	11.350	1.870
out.ego-twitter	2621.659	1.000	1574.836	26.893	1908.731	110.236
out.ego-gplus	722.024	1.000	522.333	181.754	522.576	236.280
as-caida20071105	20.974	1.606	17.971	1.216	18.694	5.479
cit-HepTh	3.969	2.143	8.867	1.466	9.068	2.662

Undirected Complex

Network	Olh	Ocl	DegCut	DegBound	NBCut	NBBound
HC-BIOGRID	5.528	1.236	4.452	2.154	4.345	1.999
facebook_combined	10.456	1.292	3.083	1.470	3.074	1.472
Mus_musculus	18.245	1.305	7.940	1.944	9.518	2.631
Caenorhabditis_elegans	11.445	1.391	11.643	1.463	12.296	3.766
ca-GrQc	6.567	1.331	11.311	2.346	10.389	3.105
as20000102	19.185	1.512	7.318	1.174	7.956	3.593
advogato	8.520	1.398	32.629	1.706	33.166	7.784
p2p-Gnutella09	3.744	1.625	11.378	1.374	11.867	3.695
hprd_pp	6.543	1.422	21.053	1.547	22.191	3.468
ca-HepTh	7.655	1.539	16.406	2.454	17.030	3.301
Drosophila_melanogaster	5.573	1.655	29.115	1.487	30.979	4.614
oregon1_010526	20.474	1.443	13.300	1.163	14.611	6.569
oregon2_010526	17.330	1.530	18.203	1.173	21.758	7.258
Homo_sapiens	6.689	1.577	19.350	1.445	20.182	3.080
GoogleNw	15.591	1.320	16.224	1.172	16.506	2.010
dip20090126_MAX	2.880	1.815	2.789	2.602	2.784	2.546
com-amazon.all.cmty	414.765	1.605	1368.675	3.236	1654.150	97.735

Table 5.7. Detailed comparison of the improvement factors of the new algorithm, in a dataset made of big networks.

Directed Street

			$k = 1$		$k = 10$		$k = 100$	
Input	Nodes	Edges	Impr.	Time	Impr.	Time	Impr.	Time
egypt	1054242	2123036	144.91	0:03:55	132.86	0:04:25	116.74	0:04:48
new_zealand	2759124	5562944	447.55	0:02:34	443.95	0:02:35	427.31	0:02:38
india	16230072	33355834	1370.32	0:43:42	1369.05	0:44:17	1326.31	0:45:05
california	16905319	34303746	1273.66	0:54:56	1258.12	0:56:00	1225.73	0:56:02
north_am	35236615	70979433	1992.68	2:25:58	1967.87	2:29:25	1877.78	2:37:14

Undirected Street

			$k = 1$		$k = 10$		$k = 100$	
Input	Nodes	Edges	Impr.	Time	Impr.	Time	Impr.	Time
egypt	1054242	1159808	344.86	0:01:54	340.30	0:01:54	291.71	0:02:11
new_zealand	2759124	2822257	811.75	0:02:47	786.52	0:03:02	734.20	0:03:02
india	16230072	17004400	2455.38	0:44:21	2484.70	0:44:38	2422.40	0:44:21
california	16905319	17600566	2648.08	0:39:15	2620.17	0:42:04	2504.86	0:44:19
north_am	35236615	36611653	7394.88	1:13:37	7530.80	1:15:01	7263.78	1:10:28

Directed Complex

			$k = 1$		$k = 10$		$k = 100$	
Input	Nodes	Edges	Impr.	Time	Impr.	Time	Impr.	Time
cit-HepTh	27769	352768	16.34	0:00:01	11.41	0:00:01	9.06	0:00:02
cit-HepPh	34546	421534	23.68	0:00:01	19.88	0:00:01	14.41	0:00:02
p2p-Gnut31	62586	147892	194.19	0:00:01	44.24	0:00:01	19.34	0:00:04
soc-Eps1	75879	508837	243.14	0:00:01	43.75	0:00:01	33.60	0:00:05
soc-Slash0811	77360	828161	1007.70	0:00:00	187.46	0:00:00	21.09	0:00:18
twitter_comb	81306	2684592	1024.32	0:00:01	692.96	0:00:01	145.68	0:00:05
Slash090221	82140	549202	177.82	0:00:02	162.30	0:00:02	108.53	0:00:03
gplus_comb	107614	24476570	1500.35	0:00:04	235.17	0:00:04	62.54	0:02:19
soc-sign-eps	131828	840799	225.91	0:00:03	161.58	0:00:03	39.26	0:00:16
email-EuAll	265009	418956	4724.80	0:00:00	3699.48	0:00:00	1320.22	0:00:01
web-Stanford	281903	2312497	13.59	0:04:00	8.70	0:04:00	7.47	0:07:15
web-NotreD	325729	1469679	1690.08	0:00:02	132.83	0:00:02	66.88	0:00:49
amazon0601	403394	3387388	10.81	0:14:54	8.87	0:14:54	6.84	0:22:04
web-BerkStan	685230	7600595	3.95	1:36:21	3.67	1:36:21	3.47	1:49:12
web-Google	875713	5105039	228.61	0:01:51	96.63	0:01:51	38.69	0:10:29
youtube-links	1138494	4942297	662.78	0:01:33	200.68	0:01:33	125.72	0:07:02
in-2004	1382870	16539643	43.68	0:41:45	29.89	0:41:45	16.68	1:48:42
trec-wt10g	1601787	8063026	33.86	0:36:01	20.39	0:36:01	16.73	1:10:54
soc-pokec	1632803	22301964	21956.64	0:00:17	2580.43	0:06:14	1106.90	0:12:35
zhishi-hudong	1984484	14682258	30.37	1:25:38	27.71	1:25:38	24.95	1:53:27
zhishi-baidu	2141300	17632190	44.05	1:17:52	38.61	1:17:52	23.17	3:08:05
wiki-Talk	2394385	5021410	34863.42	0:00:08	28905.76	0:00:08	9887.18	0:00:18
cit-Patents	3774768	16518947	9454.04	0:02:07	8756.77	0:02:07	8340.18	0:02:13

Undirected Complex

			$k = 1$		$k = 10$		$k = 100$	
Input	Nodes	Edges	Impr.	Time	Impr.	Time	Impr.	Time
ca-HepPh	12008	118489	10.37	0:00:00	10.20	0:00:00	9.57	0:00:01
CA-AstroPh	18772	198050	62.47	0:00:00	28.87	0:00:01	14.54	0:00:01
CA-CondMat	23133	93439	247.35	0:00:00	84.48	0:00:00	17.06	0:00:01
email-Enron	36692	183831	365.92	0:00:00	269.80	0:00:00	41.95	0:00:01
loc-brightkite	58228	214078	308.03	0:00:00	93.85	0:00:01	53.49	0:00:02
flickrEdges	105938	2316948	39.61	0:00:23	17.89	0:00:55	15.39	0:01:16
gowalla	196591	950327	2412.26	0:00:01	33.40	0:01:18	28.13	0:01:33
com-dblp	317080	1049866	500.83	0:00:10	300.61	0:00:17	99.64	0:00:52
com-amazon	334863	925872	37.76	0:02:21	31.33	0:02:43	18.68	0:04:34
com-lj.all	477998	530872	849.57	0:00:07	430.72	0:00:13	135.14	0:00:45
com-youtube	1134890	2987624	2025.32	0:00:32	167.45	0:06:44	110.39	0:09:16
soc-pokec	1632803	30622564	46725.71	0:00:18	8664.33	0:02:16	581.52	0:18:12
as-skitter	1696415	11095298	185.91	0:19:06	164.24	0:21:53	132.38	0:27:06
com-orkut	3072441	117185083	23736.30	0:02:32	255.17	2:54:58	69.23	15:02:06
youtube-u-g	3223585	9375374	11473.14	0:01:07	91.17	2:07:23	66.23	2:54:12

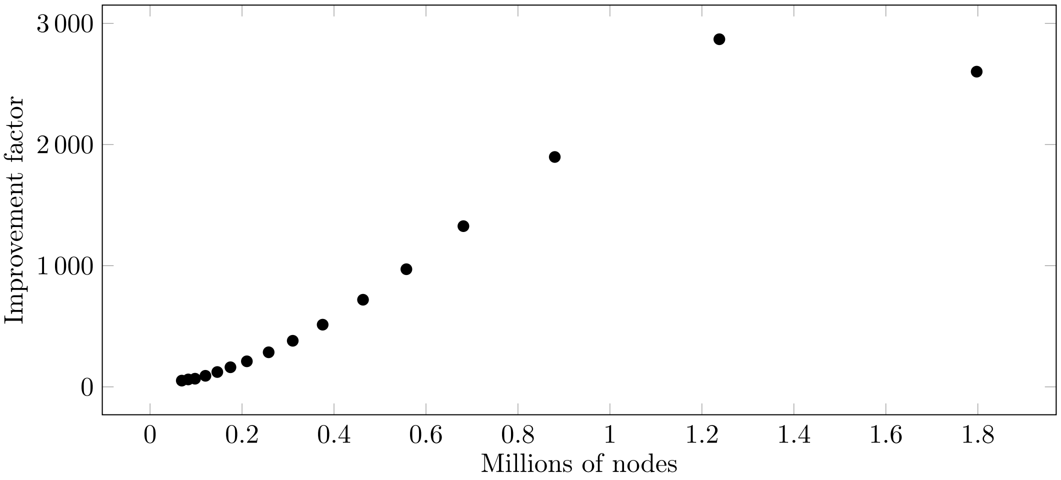


Figure 5.2. Improvement factor of the new closeness centrality algorithm, in increasing snapshots of the actor graph ($k = 1$).

5.8 Internet Movies Database Case Study

In this section, we apply the new algorithm NBBound to analyze the actors collaboration network, where nodes are actors, and two actors are linked if they played together in a movie (see Section 4.7). The performances of the algorithm are reported in Table 5.8, and the results obtained are reported in Table 5.9.

The Algorithm. Thanks to this experiment, we can evaluate the performance of our algorithm on increasing snapshots of the same graph. This way, we can have an informal idea on the asymptotic behavior of its complexity. In Figure 5.2, we have plotted the improvement factor with respect to the number of nodes: if the improvement factor is I, the running time is $\mathcal{O}(\frac{mn}{I})$. Hence, assuming that $I = cn$ for some constant c (which is approximately verified in the actor graph, as shown by Figure 5.2), the running time is linear in the input size. Indeed, the total time needed to perform the computation on all snapshots is little more than 30 minutes for $k = 1$, and little more than 45 minutes for $k = 10$.

The Results. In 2014, the most central actor is Michael Madsen, whose career spans 25 years and more than 170 films. Among his most famous appearances, he played as *Jimmy Lennox* in *Thelma & Louise* (Ridley Scott, 1991), as *Glen Greenwood* in *Free Willy* (Simon Wincer, 1993), as *Bob* in *Sin City* (Frank Miller, Robert Rodriguez, Quentin Tarantino), and as *Deadly Viper Budd* in *Kill Bill* (Quentin Tarantino, 2003-2004). The second is Danny Trejo, whose most famous movies are *Heat* (Michael Mann, 1995), where he played as *Trejo*, *Machete* (Ethan Maniquis, Robert Rodriguez, 2010) and *Machete Kills* (Robert Rodriguez, 2013), where he played as *Machete*. The third “actor” is not really an actor: he is the German dictator Adolf Hitler: he was also the most central actor in 2005 and 2010, and he was in the top-10 since 1990. This a consequence of his appearances in several archive footages, that were re-used in several movies (he counts 775 credits, even if most of them are in documentaries or TV-shows, that were eliminated). Among the movies where Adolf Hitler is credited, we find *Zelig* (Woody Allen, 1983), and *The Imitation Game* (Morten Tyldum, 2014). Among the other most central actors, we find many people who played a lot of movies, and most of them are quite important actors. However, this ranking does not discriminate between important roles and marginal roles: for instance, the actress Bess Flowers is not widely known, because she rarely played significant roles, but she appeared in over 700 movies in her 41 years career, and for this reason she was the most central for 30 years, between 1950 and 1980. Finally, it is worth noting that we never find Kevin Bacon in the top-10, even if he became famous for the “Six Degrees of Kevin Bacon” game (`http://oracleofbacon.org`), where the player

Table 5.8. Detailed results of the new improvement factors on the actor graph.

Year	**1940**	**1945**	**1950**	**1955**
Nodes	69 011	83 068	97 824	120 430
Edges	3 417 144	5 160 584	6 793 184	8 674 159
Impr ($k = 1$)	51.74	61.46	67.50	91.46
Impr ($k = 10$)	32.95	40.73	44.72	61.52
Year	**1960**	**1965**	**1970**	**1975**
Nodes	146 253	174 826	210 527	257 896
Edges	11 197 509	12 649 114	14 209 908	16 080 065
Impr ($k = 1$)	122.63	162.06	211.05	285.57
Impr ($k = 10$)	80.50	111.51	159.32	221.07
Year	**1980**	**1985**	**1990**	**1995**
Nodes	310 278	375 322	463 078	557 373
Edges	18 252 462	20 970 510	24 573 288	28 542 684
Impr ($k = 1$)	380.52	513.40	719.21	971.11
Impr ($k = 10$)	296.24	416.27	546.77	694.72
Year	**2000**	**2005**	**2010**	**2014**
Nodes	681 358	880 032	1 237 879	1 797 446
Edges	33 564 142	41 079 259	53 625 608	72 880 156
Impr ($k = 1$)	1326.53	1897.31	2869.14	2601.52
Impr ($k = 10$)	838.53	991.89	976.63	1390.32

receives an actor s, and he has to find a path of length at most 6 from s to Kevin Bacon in the actor graph (see Section 4.7.3 for more information). Kevin Bacon was chosen as the goal because he played in several movies, and he was thought to be one of the most central actors: this work shows that, actually, he is quite far from the top. Indeed, his closeness centrality is 0.336, while the most central actor has centrality 0.354, the 10th actor has centrality 0.350, and the 100th actor (Rip Torn) has centrality 0.341.

5.9 Wikipedia Case Study

In this section, we apply the new algorithm NBBound to analyze the Wikipedia citation network (for more information, see Section 4.8). We analyze both the standard graph and the reversed graph (where all edges are inverted). The 10 most central pages are available in Table 5.10.

The Algorithm. In the standard graph, the improvement factor is 1 784 for $k = 1$, 1 509 for $k = 10$, and 870 for $k = 100$. The total running time is about 39 minutes for $k = 1$, 45 minutes for $k = 10$, and less than 1 hour and 20 minutes for $k = 100$. In the reversed graph, the algorithm performs even better: the improvement factor is 87 918 for $k = 1$, 71 923 for $k = 10$, and 21 989 for $k = 100$. The total running times are less than 3 minutes for both $k = 1$ and $k = 10$, and less than 10 minutes for $k = 100$.

The Results. If we consider the standard graph, the results are quite unexpected: indeed, all the most central pages are years (the first is 1989). However, this is less surprising if we consider that these pages contain a lot of links to events that happened in that year: for instance, the out-degree of 1989 is 1 560, and the links contain pages from very different topics: historical events, like the fall of Berlin wall, days of the year, different countries where particular events happened, and so on. A similar argument also works for other years: indeed, the second page is 1967 (with out-degree 1 438), and the third is 1979 (with out-degree

Table 5.9. Detailed ranking of the IMDB actor graph.

	1940	**1945**	**1950**	**1955**
1	Semels, Harry (I)	Corrado, Gino	Flowers, Bess	Flowers, Bess
2	Corrado, Gino	Steers, Larry	Steers, Larry	Harris, Sam (II)
3	Steers, Larry	Flowers, Bess	Corrado, Gino	Steers, Larry
4	Bracey, Sidney	Semels, Harry (I)	Harris, Sam (II)	Corrado, Gino
5	Lucas, Wilfred	White, Leo (I)	Semels, Harry (I)	Miller, Harold (I)
6	White, Leo (I)	Mortimer, Edmund	Davis, George (I)	Farnum, Franklyn
7	Martell, Alphonse	Boteler, Wade	Magrill, George	Magrill, George
8	Conti, Albert (I)	Phelps, Lee (I)	Phelps, Lee (I)	Conaty, James
9	Flowers, Bess	Ring, Cyril	Ring, Cyril	Davis, George (I)
10	Sedan, Rolfe	Bracey, Sidney	Moorhouse, Bert	Cording, Harry

	1960	**1965**	**1970**	**1975**
1	Flowers, Bess	Flowers, Bess	Flowers, Bess	Flowers, Bess
2	Harris, Sam (II)	Harris, Sam (II)	Harris, Sam (II)	Harris, Sam (II)
3	Farnum, Franklyn	Farnum, Franklyn	Tamiroff, Akim	Tamiroff, Akim
4	Miller, Harold (I)	Miller, Harold (I)	Farnum, Franklyn	Welles, Orson
5	Chefe, Jack	Holmes, Stuart	Miller, Harold (I)	Sayre, Jeffrey
6	Holmes, Stuart	Sayre, Jeffrey	Sayre, Jeffrey	Miller, Harold (I)
7	Steers, Larry	Chefe, Jack	Quinn, Anthony (I)	Farnum, Franklyn
8	Parìs, Manuel	Parìs, Manuel	O'Brien, William H.	Kemp, Kenner G.
9	O'Brien, William H.	O'Brien, William H.	Holmes, Stuart	Quinn, Anthony (I)
10	Sayre, Jeffrey	Stevens, Bert (I)	Stevens, Bert (I)	O'Brien, William H.

	1980	**1985**	**1990**	**1995**
1	Flowers, Bess	Welles, Orson	Welles, Orson	Lee, Christopher (I)
2	Harris, Sam (II)	Flowers, Bess	Carradine, John	Welles, Orson
3	Welles, Orson	Harris, Sam (II)	Flowers, Bess	Quinn, Anthony (I)
4	Sayre, Jeffrey	Quinn, Anthony (I)	Lee, Christopher (I)	Pleasence, Donald
5	Quinn, Anthony (I)	Sayre, Jeffrey	Harris, Sam (II)	Hitler, Adolf
6	Tamiroff, Akim	Carradine, John	Quinn, Anthony (I)	Carradine, John
7	Miller, Harold (I)	Kemp, Kenner G.	Pleasence, Donald	Flowers, Bess
8	Kemp, Kenner G.	Miller, Harold (I)	Sayre, Jeffrey	Mitchum, Robert
9	Farnum, Franklyn	Niven, David (I)	Tovey, Arthur	Harris, Sam (II)
10	Niven, David (I)	Tamiroff, Akim	Hitler, Adolf	Sayre, Jeffrey

	2000	**2005**	**2010**	**2014**
1	Lee, Christopher (I)	Hitler, Adolf	Hitler, Adolf	Madsen, Michael (I)
2	Hitler, Adolf	Lee, Christopher (I)	Lee, Christopher (I)	Trejo, Danny
3	Pleasence, Donald	Steiger, Rod	Hopper, Dennis	Hitler, Adolf
4	Welles, Orson	Sutherland, Donald (I)	Keitel, Harvey (I)	Roberts, Eric (I)
5	Quinn, Anthony (I)	Pleasence, Donald	Carradine, David	De Niro, Robert
6	Steiger, Rod	Hopper, Dennis	Sutherland, Donald (I)	Dafoe, Willem
7	Carradine, John	Keitel, Harvey (I)	Dafoe, Willem	Jackson, Samuel L.
8	Sutherland, Donald (I)	von Sydow, Max (I)	Caine, Michael (I)	Keitel, Harvey (I)
9	Mitchum, Robert	Caine, Michael (I)	Sheen, Martin	Carradine, David
10	Connery, Sean	Sheen, Martin	Kier, Udo	Lee, Christopher (I)

1 452). Furthermore, all the 10 most central pages have out-degree at least 1 269. Overall, we conclude that the central page in the Wikipedia standard graph are not the "intuitively important" pages, but they are the pages that have a biggest number of links to pages with different topics, and this maximum is achieved by pages related to years.

Conversely, if we consider the reversed graph, the most central page is `United States`, confirming a common conjecture. Indeed, in `http://wikirank.di.unimi.it/`, it is shown that the United States are the center according to harmonic centrality, and many other

Table 5.10. Top-10 pages in Wikipedia directed graph, both in the standard graph and in the reversed graph.

Position	Standard Graph	Reversed Graph
1st	1989	United States
2nd	1967	World War II
3rd	1979	United Kingdom
4th	1990	France
5th	1970	Germany
6th	1991	English language
7th	1971	Association football
8th	1976	China
9th	1945	World War I
10th	1965	Latin

measures (however, in that work, the ranking is only approximated). A further evidence for this conjecture comes from the Six Degree of Wikipedia game (`http://thewikigame.com/6-degrees-of-wikipedia`), where a player is asked to go from one page to the other following the smallest possible number of link: a hard variant of this game forces the player not to pass from the `United States` page, which is considered to be central. In this work, we show that this conjecture is true. The second page is `World War II`, and the third is `United Kingdom`, in line with the results obtained by other centrality measures (see `http://wikirank.di.unimi.it/`), especially for the first two pages.

Overall, we conclude that most of the central pages in the reversed graph are nations, and that the results capture our intuitive notion of "important" pages in Wikipedia. Thanks to this new algorithm, we can compute these pages in a bit more than 1 hour for the original graph, and less than 10 minutes for the reversed one.

5.10 Bibliographic Notes

In [34], P. Crescenzi, A. Marino, we have developed the functions `computeBoundsDeg`, `updateBoundsBFSCut`, and we have defined the first and simplest variation of the BCM algorithm. Independently, E. Bergamini and H. Meyerhenke developed the `computeBoundsNB` and the `updateBoundsLB` functions, only in the (strongly) connected case, defining the other three variations of the BCM algorithm (see [20]). The two approaches were combined in [20], in the case of (strongly) connected graphs. The generalization to disconnected graphs is a collaboration between all the aforementioned authors, but it is not published, yet.

Chapter 6

Computing Betweenness Centrality: the KADABRA Algorithm

Abstract

We present KADABRA, a new algorithm to approximate betweenness centrality in directed and undirected graphs, which significantly outperforms all previous approaches on real-world complex networks.

The new algorithm has two significant improvements with respect to previous counterparts: first, we adopt for the first time the balanced bidirectional BFS, in order sample shortest paths faster. Furthermore, we provide a new rigorous application of the adaptive sampling technique. This approach decreases the total number of shortest paths that need to be sampled to compute all betweenness centralities with a given absolute error, and it also handles more general problems, such as computing the k most central nodes.

In this chapter, we focus on estimating the *betweenness centrality*, which is one of the most famous measures of *centrality* for nodes and edges of real-world complex networks [70, 126]. The rigorous definition of betweenness centrality has its roots in sociology, dating back to the Seventies [78], when Freeman formalized the informal concept discussed in the previous decades in different scientific communities [16, 146, 145, 37], although the definition already appeared in [11]. Since then, this notion has been very successful in network science [166, 122, 82, 126].

A probabilistic way to define the betweenness centrality[1] $\mathrm{bc}(v)$ of a node v in a graph $G = (V, E)$ is the following. We choose two nodes s and t, and we go from s to t through a shortest path $\boldsymbol{\pi}$; if the choices of s, t and $\boldsymbol{\pi}$ are made uniformly at random, the betweenness centrality of v is the probability that we pass through v.

In a seminal paper [39], Brandes showed that it is possible to exactly compute the betweenness centrality of all the nodes in a graph in time $\mathcal{O}(mn)$, where n is the number of nodes and m is the number of edges. We have already proved a corresponding lower bound in Chapter 3 (see also [30]): if we are able to compute the betweenness centrality of a single node in time $\mathcal{O}(mn^{1-\varepsilon})$ for some $\varepsilon > 0$, then the Strong Exponential Time Hypothesis [92], the Orthogonal Vector conjecture [3], and the Hitting Set conjecture [3], are false. A similar result is available in the context of dense weighted graphs [1]: computing the betweenness centrality exactly is equivalent to computing the All Pairs Shortest Paths, and providing a relative approximation is equivalent to computing the diameter.

[1] As explained in Section 6.3, to simplify notation we consider the *normalized* betweenness centrality.

This result further motivates the rich line of research on computing absolute approximations of betweenness centrality, with the goal of trading precision with efficiency. The main idea is to define a probability distribution over the set of all paths, by choosing two uniformly random nodes s, t, and then a uniformly distributed st-path $\boldsymbol{\pi}$, so that $\mathbb{P}(v \in \boldsymbol{\pi}) = \mathrm{bc}(v)$. As a consequence, we can approximate $\mathrm{bc}(v)$ by sampling paths $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\tau$ according to this distribution, and estimating $\tilde{\boldsymbol{b}}(v) := \frac{1}{\tau} \sum_{i=1}^{\tau} \boldsymbol{X}_i(v)$, where $\boldsymbol{X}_i(v) = 1$ if $v \in \boldsymbol{\pi}_i$ (and $v \neq s, t$), 0 otherwise.

The tricky part of this approach is to provide probabilistic guarantees on the quality of this approximation: the goal is to obtain a $1 - \eta$ confidence interval $\boldsymbol{I}(v) = [\tilde{\boldsymbol{b}}(v) - \lambda_L, \tilde{\boldsymbol{b}}(v) + \lambda_U]$ for $\mathrm{bc}(v)$, which means that $\mathbb{P}(\forall v \in V, \mathrm{bc}(v) \in \boldsymbol{I}(v)) \geq 1 - \eta$. Thus, the research for approximating betweenness centrality has been focusing on obtaining, as fast as possible, the smallest possible $\boldsymbol{I}$.

6.1 Our Contribution

In this chapter, we propose a new and faster algorithm to perform this task, named KADABRA. In the standard task of approximating betweenness centralities with absolute error at most λ, we show that, on average, the new algorithm is more than 100 times faster than the previous ones, on graphs with approximately 10 000 nodes. Moreover, differently from previous approaches, our algorithm can perform more general tasks, since it does not need all confidence intervals to be equal. As an example, we consider the computation of the k most central nodes: all previous approaches compute all centralities with an error λ, and use this approximation to obtain the ranking. Conversely, our approach allows us to use small confidence interval only when they are needed, and allows bigger confidence intervals for nodes whose centrality values are "well separated". This way, we can compute for the first time an approximation of the k most central nodes in networks with millions of nodes and hundreds of millions of edges, such as the IMDB actors collaboration network and the Wikipedia citation network.

Our results rely on two main theoretical contributions, which are interesting in their own right, since their generality naturally extends to other applications.

Balanced bidirectional breadth-first search. Differently from previous approaches, in order to sample a uniformly random shortest path, we use a balanced bidirectional BFS. We show that this technique heavily improves with respect to the standard BFS performed by existing approaches, yielding a factor-50 improvement on graphs with tens of thousands of nodes. Furthermore, in Chapter 8, we show that this technique has running time $\mathcal{O}(n^{\frac{1}{2}+\varepsilon})$ on realistic random graph models.

Adaptive sampling made rigorous. To speed up the estimation of the betweenness centrality, previous work make use of the technique of adaptive sampling, which consists in testing during the execution of the algorithm whether some condition on the sample obtained so far has been met, and terminating the execution of the algorithm as soon as this happens. However, this technique introduces a subtle stochastic dependence between the time in which the algorithm terminates and the correctness of the given output, which previous papers claiming a formal analysis of the technique did not realize (see Section 6.4 for details). With an argument based on martingale theory, we provide a general analysis of such useful technique. Through this result, we do not only improve previous estimators, but we also make it possible to define more general stopping conditions, that can be decided "on the fly": this way, with little modifications, we can adapt our algorithm to perform more general tasks than previous ones.

To better illustrate the power of our techniques, we focus on the unweighted, static graphs, and to the centrality of nodes. However, our algorithm can be easily adapted to compute the centrality of edges, to handle weighted graphs and, since its core part consists merely in sampling paths, we conjecture that it may be coupled with the existing techniques in [21] to handle dynamic graphs.

6.2 Related Work

6.2.1 Computing Betweenness Centrality

With the recent event of big data, the major shortcoming of betweenness centrality has been the lack of efficient methods to compute it [39]. In the worst-case, the best exact algorithm to compute the centrality of all the nodes is due to Brandes [39], and its time complexity is $\mathcal{O}(mn)$: the basic idea of the algorithm is to define the dependency $\delta_s(v) = \sum_{t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$, where $\sigma_{st}(v)$ is the number of st-shortest paths passing through v, and σ_{st} is the number of st-shortest paths. The idea of the algorithm is to prove that the dependencies $\delta_s(v)$ can be computed in time $\mathcal{O}(m)$ for each $v \in V$. In Chapter 3 (see also [30]), we also show that Brandes algorithm is almost optimal, assuming the Orthogonal Vector conjecture [3].

The latter result further motivates the already rich line of research on approaches that overcome this barrier. A first possibility is to use heuristics, that do not provide analytical guarantees on their performance [143, 73, 163]. Another line of research has defined variants of betweenness centrality, that might be easier to compute [40, 132, 68]. Finally, a third line of research has investigated approximation algorithms, which trade accuracy for speed [93, 43, 82, 109, 136, 137]. Our work follows the latter approach.

The first approximation algorithm proposed in the literature [93] adapts Eppstein and Wang's approach for computing closeness centrality [72], using Hoeffding inequality and the union bound technique. This way, it is possible to obtain an estimate of the betweenness centrality of every node that is correct up to an additive error λ with probability $1 - \eta$, by sampling $\mathcal{O}(\frac{D^2}{\lambda^2} \log \frac{n}{\eta})$ nodes, where D is the diameter of the graph. In [82], it is shown that this can lead to an overestimation. Riondato and Kornaropoulos improve this sampling-based approach by sampling single shortest paths instead of the whole dependency of a node [136], introducing the use of the VC-dimension. As a result, the number of samples is decreased to $\frac{c}{\lambda^2}(\lfloor \log_2(\mathrm{VD} - 2) \rfloor + 1 + \log(\frac{1}{\eta}))$, where VD is the vertex diameter, that is, the minimum number of nodes in a shortest path in G (it can be different from $D+1$ if the graph is weighted). This use of the VC-dimension is further developed and generalized in [137]. Finally, many of these results were adapted to handle dynamic networks [21, 137].

6.2.2 Approximating the Top-k Betweenness Centrality Set

Let us order the nodes $v_1, ..., v_n$ such that $\mathrm{bc}(v_1) \geq ... \geq \mathrm{bc}(v_n)$ and define $TOP(k) = \{(v_i, \mathrm{bc}(v_i)) : i \leq k\}$. In [136] and [137], the authors provide an algorithm that, for any given η, ε, with probability $1 - \eta$ outputs a set $\widetilde{TOP}(k) = \{(v_i, \tilde{\boldsymbol{b}}(v_i))\}$ such that: i) If $v \in TOP(k)$ then $v \in \widetilde{TOP}(k)$ and $|\,\mathrm{bc}(v) - \tilde{\boldsymbol{b}}(v)| \leq \varepsilon\,\mathrm{bc}(v)$; ii) If $v \in \widetilde{TOP}(k)$ but $v \notin TOP(k)$ then $\tilde{\boldsymbol{b}}(v) \leq (\mathbf{b}_k - \varepsilon)(1 + \varepsilon)$ where $\mathbf{b}_k$ is the k-th largest betweenness given by a preliminary phase of the algorithm.

6.2.3 Adaptive Sampling

In [13, 137], the number of samples required is substantially reduced using the adaptive sampling technique introduced by Lipton and Naughton in [111, 112]. Let us clarify that, by adaptive sampling, we mean that the termination of the sampling process depends on the sample observed so far (in other cases, the same expression refers to the fact that the distribution of the new samples is a function of the previous ones [6], while the sample size is fixed in advance). Except for [133], previous approaches tacitly assume that there is little dependency between the stopping time and the correctness of the output: indeed, they prove that, for each *fixed* τ, the probability that the estimate is wrong at time τ is below η. However, the stopping time $\boldsymbol{\tau}$ is a random variable, and in principle there might be dependency between the event $\boldsymbol{\tau} = \tau$ and the event that the estimate is correct at time τ. As for [133], they consider a specific stopping condition and their proof technique does not seem to extend to other settings. For a more thorough discussion of this issue, we defer the reader to Section 6.4.

6.2.4 Balanced Bidirectional Breadth-First Search

The possibility of speeding up a breadth-first search for the shortest-path problem by performing, at the same time, a BFS from the final end-point, has been considered since the Seventies [134]. Unfortunately, because of the lack of theoretical results dealing with its efficiency, the bidirectional BFS has apparently not been considered a fundamental heuristic improvement [95], at least in the context of complex networks (there are some applications of bidirectional searches in route planning). However, in [136] (and in some public talks by M. Riondato), the bidirectional BFS was proposed as a possible way to improve the performance of betweenness centrality approximation algorithms.

6.3 Algorithm Overview

To simplify notation, we always consider the *normalized* betweenness centrality of a node v, which is defined by:

$$\mathrm{bc}(v) = \frac{1}{n(n-1)} \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where σ_{st} is the number of shortest paths between s and t, and $\sigma_{st}(v)$ is the number of shortest paths between s and t that pass through v. Furthermore, to simplify the exposition, we use bold symbols to denote random variables, and light symbols to denote deterministic quantities.

On the same line of previous work, our algorithm samples random paths $\boldsymbol{\pi}_1, \dots, \boldsymbol{\pi}_\tau$, where $\boldsymbol{\pi}_i$ is chosen by selecting uniformly at random two nodes s, t, and then selecting uniformly at random one of the shortest paths from s to t. Then, it estimates $\mathrm{bc}(v)$ with

$$\tilde{\boldsymbol{b}}(v) := \frac{1}{\tau} \sum_{i=1}^{\tau} \boldsymbol{X}_i(v),$$

where $\boldsymbol{X}_i(v) = 1$ if $v \in \boldsymbol{\pi}_i$, 0 otherwise. By definition of $\boldsymbol{\pi}_i$, $\mathbb{E}[\tilde{\boldsymbol{b}}(v)] = \mathrm{bc}(v)$.

The tricky part is to bound the distance between $\tilde{\boldsymbol{b}}(v)$ and its expected value. The simplest technique uses Hoeffding inequality.

Lemma 6.1 (Hoeffding inequality, for a proof see [54])**.** *Let $\boldsymbol{X}_1, \dots, \boldsymbol{X}_k$ be independent random variables such that $a_i < \boldsymbol{X}_i < b_i$ almost surely, and let $\boldsymbol{S} = \sum_{i=1}^{k} \boldsymbol{X}_i$. Then,*

$$\mathbb{P}\left(|\boldsymbol{S} - \mathbb{E}[\boldsymbol{S}]| > \lambda\right) \leq 2e^{-\frac{2\lambda^2}{\sum_{i=1}^{k} |b_i - a_i|^2}}$$

Using this inequality, it is possible to prove that

$$\mathbb{P}\left(\left|\tilde{\boldsymbol{b}}(v) - \mathrm{bc}(v)\right| \geq \lambda\right) \leq 2e^{-2\tau\lambda^2}.$$

Consequently, if we consider a union bound on all possible nodes v, we obtain $\mathbb{P}(\forall v \in V, |\tilde{\boldsymbol{b}}(v) - \mathrm{bc}(v)| \geq \lambda) \leq 2ne^{-2\tau\lambda^2}$. This means that the algorithm can safely stop as soon as $2ne^{-2\tau\lambda^2} \leq \eta$, that is, after $\tau = \frac{2}{\lambda^2}\log(\frac{2n}{\eta})$ steps.

In order to improve this idea, we can start from the Chernoff bound, instead of Hoeffding's inequality.

Lemma 6.2 (Chernoff bound, for a proof see [54])**.** *Let $\boldsymbol{X}_1, \dots, \boldsymbol{X}_k$ be independent random variables such that $\boldsymbol{X}_i \leq M$ for each $1 \leq i \leq n$, and let $\boldsymbol{X} = \frac{1}{k}\sum_{i=1}^{k} \boldsymbol{X}_i$. Then,*

$$\Pr\left(\boldsymbol{X} \geq \mathbb{E}[\boldsymbol{X}] + \lambda\right) \leq \exp\left\{-\frac{\lambda^2}{2(\sum_{i=1}^{n} \mathbb{E}[\boldsymbol{X}_i^2] + M\lambda/3)}\right\}.$$

We obtain that

$$\mathbb{P}\left(\left|\tilde{\boldsymbol{b}}(v) - \mathrm{bc}(v)\right| \geq \lambda\right) \leq 2e^{-\frac{\tau\lambda^2}{2(\mathrm{bc}(v)+\lambda/3)}}.$$

If we assume the error λ to be small, this inequality is stronger than the previous one for all values of $\mathrm{bc}(v) < \frac{1}{4}$ (a condition which holds for almost all nodes, in almost all graphs considered). However, in order to apply this inequality, we have to deal with the fact that we do not know $\mathrm{bc}(v)$ in advance, and hence we do not know when to stop. Intuitively, to solve this problem, we make a “change of variable”, and we rewrite the previous inequality as

$$\mathbb{P}\left(\mathrm{bc}(v) \leq \tilde{\boldsymbol{b}}(v) - f\right) \leq \eta_L(v) \quad \text{and} \quad \mathbb{P}\left(\mathrm{bc}(v) \geq \tilde{\boldsymbol{b}}(v) + g\right) \leq \eta_U(v), \tag{6.1}$$

for some functions $f = f(\tilde{\boldsymbol{b}}(v), \eta_L(v), \tau), g = g(\tilde{\boldsymbol{b}}(v), \eta_U(v), \tau)$. Our algorithm fixes at the beginning the values $\eta_L(v), \eta_U(v)$ for each node v, and, at each step, it tests if $f(\tilde{\boldsymbol{b}}(v), \eta_L(v), \tau)$ and $g(\tilde{\boldsymbol{b}}(v), \eta_U(v), \tau)$ are small enough. If this condition is satisfied, the algorithm stops. Note that this approach lets us define very general stopping conditions, that might depend on the centralities computed until now, on the single nodes, and so on.

Remark 6.3. Instead of fixing the values $\eta_L(v), \eta_U(v)$ at the beginning, one might want to decide them during the algorithm, depending on the outcome. However, this is not formally correct, because of dependency issues (for example, Equation (6.1) does not even make sense, if $\eta_L(v), \eta_U(v)$ are random). Finding a way to overcome this issue is left as a challenging open problem (more details are provided in Section 6.4).

In order to implement this idea, we still need to solve an issue: Equation (6.1) holds for each *fixed* time τ, but the stopping time of our algorithm is a random variable $\boldsymbol{\tau}$, and there might be dependency between the value of $\boldsymbol{\tau}$ and the probability in Equation (6.1). To this purpose, we use (a strenghtened version of) Azuma's inequality, which is a stronger version of Chernoff bound, that holds even if $\boldsymbol{\tau}$ is a random variable.

Lemma 6.4 (strengthened version of Azuma inequality; for the definition of supermartingale and for a proof see [54])**.** *Let $\boldsymbol{X}_k$ be a supermartingale associated with a filter $\mathcal{F}$, and assume that* $\mathrm{Var}(\boldsymbol{X}_k|\mathcal{F}_{k-1}) \leq \sigma^2$*, and* $\boldsymbol{X}_k - \mathbb{E}(\boldsymbol{X}_k|\mathcal{F}_{k-1}) \leq M$*. Then,*

$$\mathbb{P}\left(\boldsymbol{X}_k \geq \boldsymbol{X}_0 + \lambda\right) \leq e^{\frac{-\lambda^2}{2k\sigma^2+M\lambda/3}}.$$

However, to use this inequality, we need to assume that $\boldsymbol{\tau} < \omega$ for some deterministic ω: in our algorithm, we choose

$$\omega = \frac{0.5}{\lambda^2}\left(\lfloor \log_2(\mathrm{VD} - 2)\rfloor + 1 + \log\left(\frac{2}{\eta}\right)\right),$$

because, by the results in [136], after ω samples, the maximum error is at most λ, with probability $1 - \frac{\eta}{2}$. Furthermore, also f and g should be modified, since they now depend on the value of ω. The pseudocode of the algorithm obtained is available in Algorithm 14 (as was done previous approaches, we can easily parallelize the while loop in Line 5).

The correctness of the algorithm follows from the following theorem, which is the base of our adaptive sampling, and which we prove in Section 6.4 (where we also define the functions f and g).

Algorithm 14: Our algorithm for approximating betweenness centrality.

```
Input  : a graph $G = (V, E)$
Output: for each $v \in V$, an approximation $\tilde{\boldsymbol{b}}(v)$ of $\mathrm{bc}(v)$ such that $\mathbb{P}\left(\forall v, |\tilde{\boldsymbol{b}}(v) - \mathrm{bc}(v)| \leq \lambda\right) \geq 1 - \eta$
$\omega \leftarrow \frac{c}{\lambda^2}\left(\lfloor \log_2(\mathrm{VD}-2) \rfloor + 1 + \log\left(\frac{2}{\eta}\right)\right)$;
$(\eta_L(v), \eta_U(v)) \leftarrow$ computeEta();
$\tau \leftarrow 0$;
foreach $v \in V$ do $\tilde{\boldsymbol{b}}(v) \leftarrow 0$;
while $\tau < \omega$ and not haveToStop $(\tilde{\boldsymbol{b}}, \eta_L, \eta_U, \omega, \tau)$ do
    $\boldsymbol{\pi}$ = samplePath();
    foreach $v \in \boldsymbol{\pi}$ do $\tilde{\boldsymbol{b}}(v) \leftarrow \tilde{\boldsymbol{b}}(v) + 1$;
    $\tau \leftarrow \tau + 1$;
end
foreach $v \in V$ do $\tilde{\boldsymbol{b}}(v) \leftarrow \tilde{\boldsymbol{b}}(v)/\tau$;
return $\tilde{\boldsymbol{b}}$
```

Theorem 6.5. *Let $\tilde{\boldsymbol{b}}(v)$ be the output of Algorithm 14, let $\boldsymbol{\tau}$ be the number of samples at the end of the algorithm, and let*

$$f\left(\tilde{\boldsymbol{b}}(v), \eta_L, \omega, \boldsymbol{\tau}\right) = \frac{1}{\boldsymbol{\tau}} \log \frac{1}{\eta_L} \left(\frac{1}{3} - \frac{\omega}{\boldsymbol{\tau}} + \sqrt{\left(\frac{1}{3} - \frac{\omega}{\boldsymbol{\tau}}\right)^2 + \frac{2\tilde{\boldsymbol{b}}(v)\omega}{\log \frac{1}{\eta_L}}} \right), \tag{6.2}$$

$$g\left(\tilde{\boldsymbol{b}}(v), \eta_U, \omega, \boldsymbol{\tau}\right) = \frac{1}{\boldsymbol{\tau}} \log \frac{1}{\eta_U} \left(\frac{1}{3} + \frac{\omega}{\boldsymbol{\tau}} + \sqrt{\left(\frac{1}{3} + \frac{\omega}{\boldsymbol{\tau}}\right)^2 + \frac{2\tilde{\boldsymbol{b}}(v)\omega}{\log \frac{1}{\eta_U}}} \right). \tag{6.3}$$

Then, with probability $1 - \eta$, the following conditions hold:

- *if $\boldsymbol{\tau} = \omega$, $|\tilde{\boldsymbol{b}}(v) - \mathrm{bc}(v)| < \lambda$ for all v;*
- *if $\boldsymbol{\tau} < \omega$, $-f(\boldsymbol{\tau}, \tilde{\boldsymbol{b}}(v), \eta_L(v), \omega) \leq \mathrm{bc}(v) - \tilde{\boldsymbol{b}}(v) \leq g(\boldsymbol{\tau}, \tilde{\boldsymbol{b}}(v), \eta_U(v), \omega)$ for all v.*

Remark 6.6. This theorem says that, at the beginning of the algorithm, we know that, with probability $1-\eta$, one of the two conditions will hold when the algorithm stops, independently on the final value of $\boldsymbol{\tau}$. This is essential to avoid the stochastic dependence that we discuss in Section 6.4.

In order to apply this theorem, we choose λ such that our goal is reached if all centralities are known with error at most λ. Then, we choose the function `haveToStop` in a way that our goal is reached if the stopping condition is satisfied. This way, our algorithm is correct, both if $\boldsymbol{\tau} = \omega$ and if $\boldsymbol{\tau} < \omega$. For example, if we want to compute all centralities with bounded absolute error, we simply choose λ as the bound we want to achieve, and we plug the stopping condition $f, g \leq \lambda$ in the function `haveToStop`. Instead, if we want to compute an approximation of the k most central nodes, we need a different definition of f and g, which is provided in Section 6.7.

To complete the description of this algorithm, we need to specify the following functions.

`samplePath` In order to sample a path between two random nodes s and t, we use a balanced bidirectional BFS, which is defined in Section 6.5.

`computeEta` The algorithm works for any choice of the $\eta_L(v), \eta_U(v)$s, but a good choice yields better running times. We propose a heuristic way to choose them in Section 6.6.

6.4 Correctness of the Algorithm

The algorithm explained in the previous section relies on the notion of adaptive sampling, that is, the stopping time depends on the results obtained. The goal of this section is to prove the correctness of the algorithm: first, we highlight the main technical difficulty in the formalization of adaptive sampling, which previous works claiming analogous results did not address. Then, we prove that our algorithm is correct, that is, we prove Theorem 6.5. Our argument is quite general, and it could be easily adapted to formalize claims by previous papers.

As already said, the problem is the stochastic dependence between the time $\boldsymbol{\tau}$ in which the algorithm terminates and the event $\boldsymbol{E}_\tau$ = "at time τ, the estimate is within the required distance from the true value", since both $\boldsymbol{\tau}$ and $\boldsymbol{E}_\tau$ are functions of the same random sample. Since it is typically possible to prove that $\mathbb{P}(\neg\boldsymbol{E}_\tau) \leq \eta$ for every fixed τ, one may be tempted to argue that also $\mathbb{P}(\neg\boldsymbol{E}_{\boldsymbol{\tau}}) \leq \eta$, by applying these inequalities at time $\boldsymbol{\tau}$. However, this is not correct: indeed, if we have no assumptions on $\boldsymbol{\tau}$, $\boldsymbol{\tau}$ could even be defined as the smallest τ such that $\boldsymbol{E}_\tau$ does not hold!

More formally, if we want to link $\mathbb{P}(\neg\boldsymbol{E}_{\boldsymbol{\tau}})$ to $\mathbb{P}(\neg\boldsymbol{E}_\tau)$, we have to use the law of total probability, that says that:

$$\mathbb{P}(\neg\boldsymbol{E}_{\boldsymbol{\tau}}) = \sum_{\tau=1}^{\infty} \mathbb{P}(\neg\boldsymbol{E}_{\boldsymbol{\tau}} \mid \boldsymbol{\tau} = \tau)\mathbb{P}(\boldsymbol{\tau} = \tau) \tag{6.4}$$

$$= \mathbb{P}(\neg\boldsymbol{E}_{\boldsymbol{\tau}} \mid \boldsymbol{\tau} \leq \tau)\mathbb{P}(\boldsymbol{\tau} \leq \tau) + \mathbb{P}(\neg\boldsymbol{E}_{\boldsymbol{\tau}} \mid \boldsymbol{\tau} \geq \tau)\mathbb{P}(\boldsymbol{\tau} \geq \tau). \tag{6.5}$$

Then, if we want to bound $\mathbb{P}(\neg\boldsymbol{E}_{\boldsymbol{\tau}})$, we need to assume that

$$\mathbb{P}(\neg A_{\boldsymbol{\tau}} \mid \boldsymbol{\tau} = \tau) \leq \mathbb{P}(\neg A_\tau) \quad \text{or that} \quad \mathbb{P}(\neg A_\tau \mid \boldsymbol{\tau} \geq \tau) \leq \mathbb{P}(\neg A_\tau), \tag{6.6}$$

which would allow to bound Equation (6.4) or Equation (6.5) from above. Equation (6.6) is implicitly assumed to be true in previous works adopting adaptive sampling techniques. Unfortunately, because of the stochastic dependence, it is quite difficult to prove such inequalities, even if some approaches managed to overcome these difficulties [133].

In our case, we use a different technique, based on martingale theory. More specifically, we assume that the stopping time $\boldsymbol{\tau}$ is smaller than a given (fixed) integer ω, and we apply our inequalities at time ω. Then, we consider two possible cases.

- If the algorithm stops at time ω, our algorithm yields the correct bounds with probability $1 - \frac{\eta}{2}$, by Equation (3) in [136].
- Otherwise, we can use inequalities that hold at time ω to bound probabilities that hold at time $\boldsymbol{\tau}$.

These items can be formalized in the following.

Definition 6.7. Given a graph $G = (V, E)$, the vertex diameter VD of G is the maximum number of nodes in a shortest path in G.

Remark 6.8. If the graph G is unweighted, the vertex diameter is always equal to $D + 1$, where D is the diameter of the graph.

Lemma 6.9 ([136], Equation (3)). *For some universal positive constant $c \approx 0.5$ (see [113]), for each $\eta, \lambda > 0$, and for*

$$\omega = \frac{c}{\lambda^2}\left(\lfloor \log_2(\mathrm{VD} - 2) \rfloor + 1 + \log\left(\frac{2}{\eta}\right)\right),$$

it holds

$$\mathbb{P}(\exists v \in V, |\tilde{\boldsymbol{b}}_\omega(v) - \mathrm{bc}(v)| \geq \lambda) \leq \frac{\eta}{2}.$$

Lemma 6.10. *For each node v, for every fixed real numbers η_L, η_U, and for every stopping time $\boldsymbol{\tau} < \omega$,*

$$\mathbb{P}\left(\mathrm{bc}(v) \leq \tilde{\boldsymbol{b}}(v) - f\left(\tilde{\boldsymbol{b}}(v), \eta_L, \omega, \boldsymbol{\tau}\right)\right) \leq \eta_L$$

$$\mathbb{P}\left(\mathrm{bc}(v) \geq \tilde{\boldsymbol{b}}(v) + g\left(\tilde{\boldsymbol{b}}(v), \eta_U, \omega, \boldsymbol{\tau}\right)\right) \leq \eta_U.$$

It remains to prove Lemma 6.10 and to show that these lemmas imply Theorem 6.5. Let us start with the latter implication: to simplify notation, we often omit the arguments of the function f and g.

Proof of Theorem 6.5 assuming Lemmas 6.9 and 6.10. Let $\boldsymbol{E}_1$ be the event $(\boldsymbol{\tau} = \omega \wedge \exists v \in V, |\tilde{\boldsymbol{b}}(v) - \mathrm{bc}(v)| > \lambda)$, and let $\boldsymbol{E}_2$ be the event $(\boldsymbol{\tau} < \omega \wedge (\exists v \in V, -f \geq \mathrm{bc}(v) - \tilde{\boldsymbol{b}}(v) \vee \mathrm{bc}(v) - \tilde{\boldsymbol{b}}(v) \geq g))$. Let us also denote $\tilde{\boldsymbol{b}}_\tau(v) = \frac{1}{\tau}\sum_{i=1}^{\tau} \boldsymbol{X}_i(v)$ (note that $\tilde{\boldsymbol{b}}_{\boldsymbol{\tau}}(v) = \tilde{\boldsymbol{b}}(v)$).

By our choice of ω and by Equation (3) in [136],

$$\mathbb{P}(\boldsymbol{E}_1) \leq \mathbb{P}(\exists v \in V, |\tilde{\boldsymbol{b}}_\omega(v) - \mathrm{bc}(v)| > \lambda) \leq \frac{\eta}{2}$$

where $\tilde{\boldsymbol{b}}_\omega(v)$ is the approximate betweenness of v after ω samples. Furthermore, by Lemma 6.10,

$$\begin{aligned}\mathbb{P}(\boldsymbol{E}_2) &\leq \sum_{v \in V} \mathbb{P}(\boldsymbol{\tau} < \omega \wedge -f \geq \mathrm{bc}(v) - \tilde{\boldsymbol{b}}(v)) + \mathbb{P}(\boldsymbol{\tau} < \omega \wedge \mathrm{bc}(v) - \tilde{\boldsymbol{b}}(v) \leq g) \\ &\leq \sum_{v \in V} \eta_L(v) + \eta_U(v) \leq \frac{\eta}{2}.\end{aligned}$$

By a union bound, $\mathbb{P}(\boldsymbol{E}_1 \vee \boldsymbol{E}_2) \leq \mathbb{P}(\boldsymbol{E}_1) + \mathbb{P}(\boldsymbol{E}_1) \leq \eta$, concluding the proof of Theorem 6.5. □

It only remains to prove Lemma 6.10

Proof of Lemma 6.10. Since Lemma 6.10 deals with a single node v, let us simply write $\mathrm{bc} = \mathrm{bc}(v), \tilde{\boldsymbol{b}} = \tilde{\boldsymbol{b}}(v), \boldsymbol{X}_i = \boldsymbol{X}_i(v)$. Let us consider $\boldsymbol{Y}^\tau = \sum_{i=1}^{\tau}(\boldsymbol{X}_i - \mathrm{bc})$ (we recall that $\boldsymbol{X}_i = 1$ if v is in the i-th path sampled, $\boldsymbol{X}_i = 0$ otherwise). Clearly, $\boldsymbol{Y}^\tau$ is a martingale, and $\boldsymbol{\tau}$ is a stopping time for $\boldsymbol{Y}^\tau$: this means that also $\boldsymbol{Z}^\tau = \boldsymbol{Y}^{\min(\boldsymbol{\tau}, \tau)}$ is a martingale.

Let us apply Lemma 6.4 to the martingales $\boldsymbol{Z}$ and $-\boldsymbol{Z}$: for each fixed $\lambda_L, \lambda_U > 0$ we have

$$\mathbb{P}\left(\boldsymbol{Z}^\omega \geq \lambda_L\right) = \mathbb{P}\left(\boldsymbol{\tau}\tilde{\boldsymbol{b}} - \boldsymbol{\tau}\,\mathrm{bc} \geq \lambda_L\right) \leq \exp\left(-\frac{\lambda_L^2}{2\left(\omega\,\mathrm{bc} + \lambda_L/3\right)}\right) = \eta_L, \tag{6.7}$$

$$\mathbb{P}\left(-\boldsymbol{Z}^\omega \geq \lambda_U\right) = \mathbb{P}\left(\boldsymbol{\tau}\tilde{\boldsymbol{b}} - \boldsymbol{\tau}\,\mathrm{bc} \leq -\lambda_U\right) \leq \exp\left(-\frac{\lambda_U^2}{2\left(\omega\,\mathrm{bc} + \lambda_U/3\right)}\right) = \eta_U. \tag{6.8}$$

We now show how to prove Equation (6.2) from Equation (6.7). The way to derive Equation (6.3) from Equation (6.8) is analogous.

If we express λ_L as a function of η_L we get

$$\lambda_L^2 = 2\log\frac{1}{\eta_L}\left(\omega\,\mathrm{bc} + \frac{\lambda_L}{3}\right) \iff \lambda_L^2 - \frac{2}{3}\lambda_L\log\frac{1}{\eta_L} - 2\omega\,\mathrm{bc}\log\frac{1}{\eta_L} = 0,$$

which implies that

$$\lambda_L = \frac{1}{3}\log\frac{1}{\eta_L} \pm \sqrt{\frac{1}{9}\left(\log\frac{1}{\eta_L}\right)^2 + 2\omega\,\mathrm{bc}\log\frac{1}{\eta_L}}.$$

Since Equation (6.7) holds for any positive value λ_L, it also holds for the value corresponding to the positive solution of this equation, that is,

$$\lambda_L = \frac{1}{3}\log\frac{1}{\eta_L} + \sqrt{\frac{1}{9}\left(\log\frac{1}{\eta_L}\right)^2 + 2\omega\,\mathrm{bc}\log\frac{1}{\eta_L}}.$$

Plugging this value into Equation (6.7), we obtain

$$\mathbb{P}\left(\boldsymbol{\tau}\tilde{\boldsymbol{b}} - \boldsymbol{\tau}\,\mathrm{bc} \geq \frac{1}{3}\log\frac{1}{\eta_L} + \sqrt{\frac{1}{9}\left(\log\frac{1}{\eta_L}\right)^2 + 2\omega\,\mathrm{bc}\log\frac{1}{\eta_L}}\right) \leq \eta_L. \tag{6.9}$$

By assuming $\tilde{\boldsymbol{b}} - \mathrm{bc} \geq \frac{1}{3\boldsymbol{\tau}}\log(\frac{1}{\eta_L})$, the event in Equation (6.9) can be rewritten as

$$(\boldsymbol{\tau}\,\mathrm{bc})^2 - 2\,\mathrm{bc}\left(\boldsymbol{\tau}^2\tilde{\boldsymbol{b}} + \omega\log\frac{1}{\eta_L} - \frac{1}{3}\boldsymbol{\tau}\log\frac{1}{\eta_L}\right) - \frac{2}{3}\log\frac{1}{\eta_L}\boldsymbol{\tau}\tilde{\boldsymbol{b}} + \left(\boldsymbol{\tau}\tilde{\boldsymbol{b}}\right)^2 \geq 0.$$

By solving the previous quadratic equation w.r.t. bc we get

$$\mathrm{bc} \;\leq\; \tilde{\boldsymbol{b}} + \log\frac{1}{\eta_L}\left(\frac{\omega}{\boldsymbol{\tau}^2} - \frac{1}{3\boldsymbol{\tau}} - \sqrt{\left(\frac{\tilde{\boldsymbol{b}}}{\log\frac{1}{\eta_L}} + \frac{\omega}{\boldsymbol{\tau}^2} - \frac{1}{3\boldsymbol{\tau}}\right)^2 - \left(\frac{\tilde{\boldsymbol{b}}}{\log\frac{1}{\eta_L}}\right)^2 + \frac{2}{3\boldsymbol{\tau}}\frac{\tilde{\boldsymbol{b}}}{\log\frac{1}{\eta_L}}}\right),$$

where we only considered the solution which upper bounds bc, since we assumed $\tilde{\boldsymbol{b}} - \mathrm{bc} \geq \frac{1}{3\boldsymbol{\tau}}\log(\frac{1}{\eta_L})$. After simplifying the terms under the square root in the previous expression, we get

$$\mathrm{bc} \leq \tilde{\boldsymbol{b}} + \log\frac{1}{\eta_L}\left(\frac{\omega}{\boldsymbol{\tau}^2} - \frac{1}{3\boldsymbol{\tau}} - \sqrt{\left(\frac{\omega}{\boldsymbol{\tau}^2} - \frac{1}{3\boldsymbol{\tau}}\right)^2 + \frac{2\tilde{\boldsymbol{b}}\omega}{\boldsymbol{\tau}^2\log\frac{1}{\eta_L}}}\right),$$

which means that

$$\mathbb{P}\left(\mathrm{bc} \leq \tilde{\boldsymbol{b}} - f\left(\tilde{\boldsymbol{b}}, \eta_L, \omega, \boldsymbol{\tau}\right)\right) \leq \eta_L,$$

concluding the proof.

□

6.5 Balanced Bidirectional BFS

The main routine of our algorithm, like in many previous algorithms, is to sample a random path between two nodes s and t. The *textbook* algorithm performs a BFS from s, until it hits t, needing time $\mathcal{O}(m)$. In the literature, researchers proposed various techniques to improve this bound [148, 65, 95, 83]: one of several possibilities is to use the bidirectional approach [148, 65, 95], which performs at the same time a BFS from s and a BFS from t. Although bidirectional heuristic searches are common in route planning, this approach is not established in the context of complex networks, probably because it is not clear if and why it outperforms the standard BFS approach. As far as we know, bidirectional heuristic searches are only implemented in NetworkX [87] and Sagemath [152], and there are very few references in the literature. Here, we define a bidirectional BFS where we "grow the smallest ball", and we show that this heuristic can yield very good improvements in practice: indeed, in many cases, it achieves a speedup close to $\sqrt{m}$.

Our procedure starts by setting $\ell_s = 0$ and $\ell_t = 0$, where ℓ_s is the current level in the BFS from s and ℓ_t is the current level in the BFS from t. Assume that we have visited some levels from s and t, and we choose to proceed in the visit from s (we define below how this choice is made). If $\boldsymbol{\Gamma}^{\ell}(s)$ is the set of nodes at distance ℓ from s, for each node $v \in \boldsymbol{\Gamma}^{\ell_s}(s)$, and for each (out-)neighbor w of v we do the following:

- if w was never visited, we add w to $\mathbf{\Gamma}^{\ell_s+1}(s)$;
- if w was already visited in the BFS from s, we do not do anything;
- if w was visited in the BFS from t, we know that the distance between s and t is $\mathrm{dist}(s,v)+1+\mathrm{dist}(w,t)$, and consequently we can conclude the visit as soon as the current level is finished.

If we choose to proceed in the visit from t, we follow the same procedure (apart from the fact that, in the directed case, w is an in-neighbor of v instead of an out-neighbor).

The algorithm stops as soon as $\mathbf{\Gamma}^{\ell_s}(s)$ or $\mathbf{\Gamma}^{\ell_t}(t)$ is empty (in this case, s and t are not connected), or as soon as we have visited a node v from both s and t, and we have concluded the visit of the level of v (since we have to sample a random path, we need to compute all the possible nodes v that are in the middle of a shortest path from s to t).

In order to extract the path, we compute all edges (v,w) such that v has been visited from s, and w has been visited from t, and we choose at random one of these edges, where the probability of choosing (v,w) is proportional to the number of σ_{sv} of shortest paths from s to v, times the number σ_{wt} of shortest paths from w to t (note that σ_{sv}, σ_{wt} can be computed during the BFS as in [43]).

Then, the path is selected by considering the concatenation of a random path from s to v, the edge (v,w), and a random path from w to t (these random paths can be easily chosen by backtracking, as shown in [136]).

Remark 6.11. Note that an aspect often neglected in previous work when it comes to computing shortest paths is the fact that the number of shortest paths between a pair of nodes may be exponential, thus requiring to work with a linear number of bits. While real-world complex networks are typically sparse with logarithmic diameter, in order to avoid such issue it is sufficient to assume that addition and comparison require constant time.

It remains to define which level should be visited at each step. To this purpose, since the time needed to process all nodes at distance ℓ_s from s is proportional to $\sum_{v\in\mathbf{\Gamma}^{\ell_s}(s)}\deg(v)$, we decide to proceed in the visit from s if $\sum_{v\in\mathbf{\Gamma}^{\ell_s}(s)}\deg(v)\leq\sum_{w\in\mathbf{\Gamma}^{\ell_t}(t)}\deg(w)$, from t otherwise.

6.6 How to Choose $\eta_L(v), \eta_U(v)$

In Section 6.4, we proved that our algorithm works for any choice of the values $\eta_L(v), \eta_U(v)$. In this section, we show how we can heuristically compute such values, in order to obtain the best performances.

For each node v, let $\lambda_L(v), \lambda_U(v)$ be the lower and the upper maximum error that we want to obtain on the betweenness of v: if we simply want all errors to be smaller than λ, we choose $\lambda_L(v), \lambda_U(v)=\lambda$, but for other purposes different values might be needed. We want to minimize the time τ such that the approximation of the betweenness at time τ is in the confidence interval required. In formula, we want to minimize

$$\min\left\{\tau\in\mathbb{N}:\forall v\in V,\left(f\left(\tilde{\boldsymbol{b}}_\tau(v),\eta_L(v),\omega,\tau\right)\leq\lambda_L(v)\wedge\right.\right.$$
$$\left.\left.g\left(\tilde{\boldsymbol{b}}_\tau(v),\eta_U(v),\omega,\tau\right)\leq\lambda_U(v)\right)\right\} \quad (6.10)$$

where $\tilde{\boldsymbol{b}}_\tau(v)$ is the approximation of $\mathrm{bc}(v)$ obtained at time τ, and

$$f\left(\tau,\tilde{\boldsymbol{b}}_\tau,\eta_L,\omega\right)=\frac{1}{\tau}\log\frac{1}{\eta_L}\left(\frac{1}{3}-\frac{\omega}{\tau}+\sqrt{\left(\frac{1}{3}-\frac{\omega}{\tau}\right)^2+\frac{2\tilde{\boldsymbol{b}}_\tau\omega}{\log\frac{1}{\eta_L}}}\right) \text{ and}$$

$$g\left(\tau, \tilde{\boldsymbol{b}}_\tau, \eta_U, \omega\right) = \frac{1}{\tau}\log\frac{1}{\eta_U}\left(\frac{1}{3} + \frac{\omega}{\tau} + \sqrt{\left(\frac{1}{3}+\frac{\omega}{\tau}\right)^2 + \frac{2\tilde{\boldsymbol{b}}_\tau\omega}{\log\frac{1}{\eta_U}}}\right).$$

The goal of this section is to provide deterministic values of $\eta_L(v), \eta_U(v)$ that minimize the value in Equation (6.10), and such that $\sum_{v\in V}\eta_L(v)+\eta_U(v) < \frac{\eta}{2}$. To obtain our estimate, we replace $\tilde{\boldsymbol{b}}_\tau(v)$ with an approximation $\tilde{b}(v)$, that we compute by sampling α paths, before starting the algorithm (in our code, $\alpha = \frac{\omega}{100}$). Furthermore, we consider a simplified version of Equation (6.10): in most cases, λ_L is much smaller than all other quantities in play, and since ω is proportional to $\frac{1}{\lambda_L^2}$, we can safely assume

$$f(\tau, \tilde{b}(v), \eta_L(v), \omega) \approx \sqrt{\frac{2\tilde{b}(v)\omega}{\tau^2}\log\frac{1}{\eta_L}},$$
$$g(\tau, \tilde{b}(v), \eta_U(v), \omega) \approx \sqrt{\frac{2\tilde{b}(v)\omega}{\tau^2}\log\frac{1}{\eta_U}}.$$

Hence, in place of the value in Equation (6.10), our heuristic tries to minimize

$$\min\left\{\tau\in\mathbb{N} : \forall v \in V, \sqrt{\frac{2\tilde{b}(v)\omega}{\tau^2}\log\frac{1}{\eta_L(v)}} \leq \lambda_L(v) \wedge \sqrt{\frac{2\tilde{b}(v)\omega}{\tau^2}\log\frac{1}{\eta_U(v)}} \leq \lambda_U(v)\right\}.$$

Solving with respect to τ, we are trying to minimize

$$\max_{v\in V}\left(\max\left(\sqrt{\frac{2\tilde{b}(v)\omega}{\left(\lambda_L(v)\right)^2}\log\frac{1}{\eta_L(v)}}, \sqrt{\frac{2\tilde{b}(v)\omega}{\left(\lambda_U(v)\right)^2}\log\frac{1}{\eta_U(v)}}\right)\right).$$

which is the same as minimizing

$$\max_{v\in V}\max\left(c_L(v)\log\frac{1}{\eta_L(v)}, c_U(v)\log\frac{1}{\eta_U(v)}\right)$$

for some constants $c_L(v), c_U(v)$, conditioned on $\sum_{v\in V}\eta_L(v)+\eta_U(v) < \frac{\eta}{2}$. We claim that, among the possible choices of $\eta_L(v), \eta_U(v)$, the best choice makes all the terms in the maximum equal: otherwise, if two terms were different, we would be able to slightly increase and decrease the corresponding values, in order to decrease the maximum. This means that, for some constant C, for each v, $c_L(v)\log\frac{1}{\eta_L(v)} = c_U(v)\log\frac{1}{\eta_L(v)} = C$, that is, $\eta_L(v) = \exp(-\frac{C}{c_L(v)})$, $\eta_U(v) = \exp(-\frac{C}{c_U(v)})$. In order to find the largest constant C such that $\sum_{v\in V}\eta_L(v)+\eta_U(v) \leq \frac{\eta}{2}$, we use a binary search procedure on all possible constants C.

Finally, if $c_L(v) = 0$ or $c_U(v) = 0$, this procedure chooses $\eta_L(v) = 0$: to avoid this problem, we impose $\sum_{v\in V}\eta_L(v)+\eta_U(v) \leq \frac{\eta}{2} - \varepsilon\eta$, and we add $\frac{\varepsilon\eta}{2n}$ to all the $\eta_L(v)$s and all the $\eta_U(v)$s (in our code, we choose $\varepsilon = 0.001$). The pseudocode of the algorithm is available in Algorithm 15.

6.7 Computing the k Most Central Nodes

Differently from previous approaches, our algorithm is more flexible, making it possible to compute the betweenness centrality of different nodes with different precision. This feature can be exploited if we only want to rank the nodes: for instance, if v is much more central than all the other nodes, we do not need a very precise estimation on the centrality of v to say that it is the top node. Following this idea, in this section we adapt our approach to

Algorithm 15: The function `computeEta`.

```
Input  : a graph $G = (V, E)$, and two values $\lambda_L(v), \lambda_U(v)$ for each $v \in V$
Output: for each $v \in V$, two values $\eta_L(v), \eta_U(v)$
$\alpha \leftarrow \frac{\omega}{100}$;
$\varepsilon \leftarrow 0.0001$;
foreach $i \in [1, \alpha]$ do
    $\boldsymbol{\pi}$ = samplePath();
    foreach $v \in \boldsymbol{\pi}$ do $\tilde{\boldsymbol{b}}(v) \leftarrow \tilde{\boldsymbol{b}}(v) + 1$;
end
foreach $v \in V$ do
    $\tilde{\boldsymbol{b}}(v) \leftarrow \tilde{\boldsymbol{b}}(v)/\alpha$;
    $c_L(v) \leftarrow \frac{2\tilde{b}(v)\omega}{(\lambda_L(v))^2}$;
    $c_U(v) \leftarrow \frac{2\tilde{b}(v)\omega}{(\lambda_U(v))^2}$;
end
Binary search to find $C$ such that $\sum_{v \in V} \exp\left(-\frac{C}{c_L(v)}\right) + \exp\left(-\frac{C}{c_U(v)}\right) = \frac{\eta}{2} - \varepsilon\eta$;
foreach $v \in V$ do
    $\eta_L(v) \leftarrow \exp\left(-\frac{C}{c_L(v)}\right) + \frac{\varepsilon\eta}{2n}$;
    $\eta_U(v) \leftarrow \exp\left(-\frac{C}{c_U(v)}\right) + \frac{\varepsilon\eta}{2n}$;
end
return $\boldsymbol{b}$;
```

the approximation of the ranking of the k most central nodes: as far as we know, this is the first approach which computes the ranking without computing a λ-approximation of all betweenness centralities, allowing significant speedups. Clearly, we cannot expect our ranking to be always correct, otherwise the algorithm does not terminate if two of the k most central nodes have the same centrality. For this reason, the user fixes a parameter λ, and, for each node v, the algorithm does one of the following:

- it provides the exact position of v in the ranking;
- it guarantees that v is not in the top-k;
- it provides a value $\tilde{\boldsymbol{b}}(v)$ such that $|\operatorname{bc}(v) - \tilde{\boldsymbol{b}}(v)| \leq \lambda$.

In other words, similarly to what is done in [136], the algorithm provides a set of $k' \geq k$ nodes containing the top-k nodes, and for each pair of nodes v, w in this subset, either we can rank correctly v and w, or v and w are almost even, that is, $|\operatorname{bc}(v) - \operatorname{bc}(w)| \leq 2\lambda$. In order to obtain this result, we plug into Algorithm 14 the aforementioned conditions in the function `haveToStop` (see Algorithm 16).

Then, we have to adapt the function `computeEta` to optimize the $\eta_L(v)$s and the $\eta_U(v)$s to the new stopping condition: in other words, we have to choose the values of $\lambda_L(v)$ and $\lambda_U(v)$ that should be plugged into the function `computeEta` (we recall that the heuristic `computeEta` chooses the $\eta_L(v)$s so that we can guarantee as fast as possible that $\tilde{\boldsymbol{b}}(v) - \lambda_L(v) \leq \operatorname{bc}(v) \leq \tilde{\boldsymbol{b}}(v) + \lambda_U(v)$). To this purpose, we estimate the betweenness of all nodes with few samples and we sort all nodes according to these approximate values $\tilde{b}(v)$, obtaining $v_1, \ldots, v_n$. The basic idea is that, for the first k nodes, we set $\lambda_U(v_i) = \frac{\tilde{b}(v_{i-1}) - \tilde{b}(v_i)}{2}$, and $\lambda_L(v_i) = \frac{\tilde{b}(v_i) - \tilde{b}(v_{i+1})}{2}$ (the goal is to find confidence intervals that separate the betweenness of v_i from the betweenness of v_{i+1} and v_{i-1}). For nodes that are not in the top-k, we choose $\lambda_L(v) = 1$ and $\lambda_U(v) = \tilde{b}(v_k) - \lambda_L(v_k) - \tilde{b}(v_i)$ (the goal is to prove that v_i is not in the top-k). Finally, if $\tilde{b}(v_i) - \tilde{b}(v_{i+1})$ is small, we simply set $\lambda_L(v_i) = \lambda_U(v_i) = \lambda_L(v_{i+1}) = \lambda_U(v_{i+1}) = \lambda$, because we do not know if $\operatorname{bc}(v_{i+1}) > \operatorname{bc}(v_i)$, or viceversa.

Algorithm 16: The function `haveToStop` to compute the top-k nodes.

```
Input  : for each node $v$, the values of $\tilde{\boldsymbol{b}}(v), \eta_L(v), \eta_U(v)$, and the values of $\omega$ and $\tau$
Output: True if the algorithm should stop, False otherwise
Sort nodes in decreasing order of $\tilde{\boldsymbol{b}}(v)$, obtaining $v_1, \dots, v_n$;
for $i \in [1, \dots, k]$ do
    if $f(\tilde{\boldsymbol{b}}(v_i), \eta_L(v_i), \omega, \tau) > \lambda$ or $g(\tilde{\boldsymbol{b}}(v_i), \eta_U(v_i), \omega, \tau) > \lambda$ then
        if $\tilde{\boldsymbol{b}}(v_{i-1}) - f(\tilde{\boldsymbol{b}}(v_{i-1}), \eta_L(v_{i-1}), \omega, \tau) < \tilde{\boldsymbol{b}}(v_i) + g(\tilde{\boldsymbol{b}}(v_i), \eta_U(v_i), \omega, \tau)$ or
          $\tilde{\boldsymbol{b}}(v_i) - f(\tilde{\boldsymbol{b}}(v_i), \eta_L(v_i), \omega, \tau) < \tilde{\boldsymbol{b}}(v_{i+1}) + g(\tilde{\boldsymbol{b}}(v_{i+1}), \eta_U(v_{i+1}), \omega, \tau)$ then
            return False;
        end
    end
end
for $i \in [k+1, \dots, n]$ do
    if $f(\tilde{\boldsymbol{b}}(v_i), \eta_L(v_i), \omega, \tau) > \lambda$ or $g(\tilde{\boldsymbol{b}}(v_i), \eta_U(v_i), \omega, \tau) > \lambda$ then
        if $\tilde{\boldsymbol{b}}(v_k) - f(\tilde{\boldsymbol{b}}(v_k), \eta_L(v_k), \omega, \tau) < \tilde{\boldsymbol{b}}(v_i) + g(\tilde{\boldsymbol{b}}(v_i), \eta_U(v_i), \omega, \tau)$ then
            return False;
        end
    end
end
return True;
```

6.8 Experimental Results

In this section, we test our algorithm on several real-world networks, in order to evaluate its performances. The platform for our tests is a server with 1515 GB RAM and 48 Intel(R) Xeon(R) CPU E7-8857 v2 cores at 3.00GHz, running Debian GNU Linux 8. The algorithms are implemented in C++, and they are compiled using gcc 5.3.1. The source code of our algorithm is available at `https://sites.google.com/a/imtlucca.it/borassi/publications`.

6.8.1 Comparison with the State of the Art

The first experiment compares the performances of our algorithm KADABRA with the state of the art. First, we compare our algorithm with the RK algorithm [136], available in the open-source *NetworKit* framework [151]. This algorithm uses the same estimator as our algorithm, but the stopping condition is different: it simply stops after sampling

$$k = \frac{c}{\varepsilon^2}\left(\left\lfloor \log_2(\mathrm{VD}-2)\right\rfloor + 1 + \log\left(\frac{1}{\eta}\right)\right),$$

and it uses a heuristic to upper bound the vertex diameter. Following suggestions by the author of the *NetworKit* implementation, we set to 20 the number of samples used in the latter heuristic [19].

The other algorithm we compare with is the ABRA algorithm [137], available at `http://matteo.rionda.to/software/ABRA-radebetw.tbz2`. This algorithm samples pairs of nodes (s, t), and it adds the fraction of st-paths passing from v to the approximation of the betweenness of v, for each node v. The stopping condition is based on a key result in statistical learning theory, and there is a scheduler that decides when it should be tested. Following the suggestions by the authors, we use both the automatic scheduler ABRA-Aut, which uses a heuristic approach to decide when the stopping condition should be tested, and the geometric scheduler ABRA-1.2, which tests the stopping condition after $(1.2)^i k$ iterations, for each integer i.

The test is performed on a dataset made by 15 undirected and 15 directed real-world networks, taken from the datasets SNAP (`snap.stanford.edu/`), LASAGNE (`piluc.dsi.unifi.it/lasagne`), and KONECT (`http://konect.uni-koblenz.de/networks/`). As in [137], we have considered all values of $\lambda \in \{0.03, 0.025, 0.02, 0.015, 0.01, 0.005\}$, and $\eta = 0.1$.

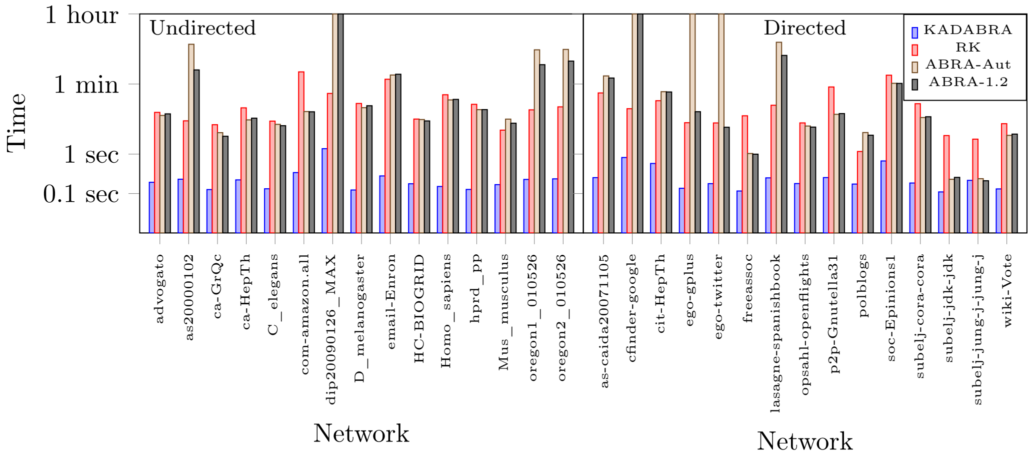


Figure 6.1. The time needed by the different algorithms, on all the graphs of our dataset.

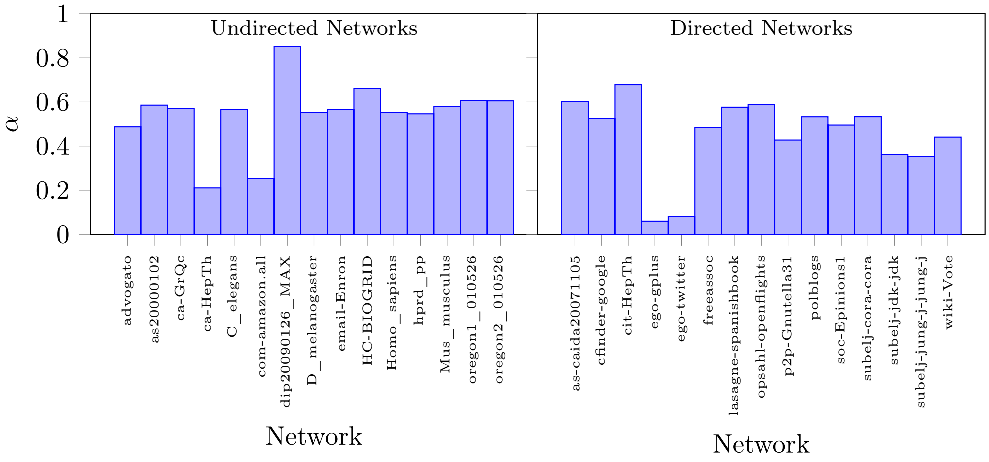


Figure 6.2. The exponent α such that the average number of edges visited during a bidirectional BFS is n^{α}.

All the algorithms have to provide an approximation $\tilde{\boldsymbol{b}}(v)$ of $\mathrm{bc}(v)$ for each v such that

$$\mathbb{P}\left(\forall v, \left|\tilde{\boldsymbol{b}}(v) - \mathrm{bc}(v)\right| \leq \lambda\right) \geq 1 - \eta.$$

In Figure 6.1, we report the time needed by the different algorithms on every graph for $\lambda = 0.005$ (the behavior with different values of λ is very similar). More detailed results are reported in Section 6.8.2.

From the figure, we see that KADABRA is much faster than all the other algorithms, on all graphs: on average, our algorithm is about 100 times faster than RK in undirected graphs, and about 70 times faster in directed graphs; it is also more than 1 000 times faster than ABRA. The latter value is due to the fact that the ABRA algorithm has large running times on few networks: in some cases, it did not even conclude its computation within one hour. The authors confirmed that this behavior might be due to some bug in the code, which seems to affect it only on specific graphs: indeed, in many networks, the performances of ABRA are better than those of the RK algorithm (but, still, not better than KADABRA).

In order to explain these data, we take a closer look at the improvements obtained through the bidirectional BFS, by considering the average number of edges m_{avg} that the algorithm visits in order to sample a shortest path (for all existing approaches, $m_{\mathrm{avg}} = m$, since they

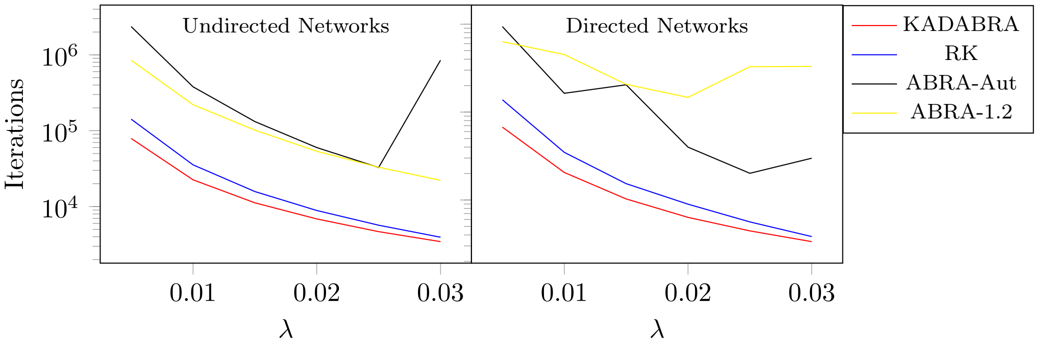


Figure 6.3. The average number of samples needed by the different algorithms.

perform a full BFS). In Figure 6.2, for each graph in our dataset, we plot $\alpha = \frac{\log(m_{\text{avg}})}{\log(m)}$ (intuitively, this means that the average number of edges visited is m^α).

The figure shows that, apart from few cases, the number of edges visited is close to $n^{\frac{1}{2}}$, confirming the results in Section 6.5. This means that, since many of our networks have approximately 10 000 edges, the bidirectional BFS is about 100 times faster than the standard BFS. Finally, for each value of λ, we report in Figure 6.3 the number of samples needed by all the algorithms, averaged over all the graphs in the dataset.

From the figure, KADABRA needs to sample the smallest amount of shortest paths, and the average improvement over RK grows when λ tends to 0, from a factor 1.14 (resp., 1.14) if $\lambda = 0.03$, to a factor 1.79 (resp., 2.05) if $\lambda = 0.005$ in the case of undirected (resp., directed) networks. Again, the behavior of ABRA is highly influenced by the behavior on few networks, and as a consequence the average number of samples is higher. In any case, also in the graphs where ABRA has good performances, KADABRA still needs a smaller number of samples.

6.8.2 Detailed Experimental Results

In this section, we provide detailed results of the running time and the number of iteration performed by the different algorithms, and the average number of edges visited by the KADABRA algorithm (for all the other algorithm, this number is m, because they perform a whole BFS to compute a shortest path).

Table 6.1. Detailed experimental results (undirected graphs). Empty values correspond to graphs on which the algorithm needed more than 1 hour.

	Number of iterations				Time (s)				Edges
Graph	KADABRA	RK	ABRA-Aut	ABRA-1.2	KADABRA	RK	ABRA-Aut	ABRA-1.2	KADABRA
$\lambda = 0.005$									
advogato	64427	126052	174728	185998	0.193	11.450	9.557	10.498	261.2
as20000102	115797	126052	18329844	4126626	0.231	6.990	611.584	136.764	377.6
ca-GrQc	61611	146052	142982	129165	0.126	5.574	3.500	2.839	353.4
ca-HepTh	31735	146052	121587	129165	0.222	14.921	7.389	8.168	9.9
C_elegans	69729	146052	204634	185998	0.132	6.876	5.693	5.261	270.7
com-amazon.all	40711	166052	69708	74747	0.340	122.020	12.011	11.849	21.9
dip20090126_MAX	156552	166052			1.374	34.595			15354.9
D_melanogaster	51227	126052	144680	154998	0.123	19.253	15.061	16.882	520.8
email-Enron	74745	146052	257989	267838	0.280	79.296	101.529	106.278	1408.0
HC-BIOGRID	78804	146052	245780	223198	0.177	7.751	7.534	6.951	713.2
Homo_sapiens	60060	146052	156973	154998	0.151	32.078	23.716	24.449	643.8
hprd_pp	59125	146052	151499	154998	0.127	18.323	13.425	13.458	456.4
Mus_musculus	92081	146052	504669	385688	0.168	4.058	7.723	6.083	226.6
oregon1_010526	114829	126052	6798931	2865712	0.228	13.281	442.370	185.711	681.6
oregon2_010526	115764	126052	5714183	2865712	0.236	15.823	452.554	229.234	822.2
$\lambda = 0.010$									
advogato	19811	31513	47076	48243	0.081	2.804	2.576	2.788	258.2
as20000102	29062	31513	2688614	1070372	0.071	1.777	88.886	35.049	377.3
ca-GrQc	18535	36513	37529	33501	0.049	1.417	0.987	0.753	350.6
ca-HepTh	13761	36513	31721	33501	0.188	3.771	2.078	2.275	10.0
C_elegans	19888	36513	54327	48243	0.048	1.803	1.586	1.483	269.4
com-amazon.all	14641	41513	18007	19386	0.312	31.004	5.196	7.623	21.5
dip20090126_MAX	39314	41513			0.395	8.578			15301.7
D_melanogaster	15136	31513	37219	40202	0.063	4.983	3.891	4.715	519.9
email-Enron	21637	36513	65392	69471	0.198	19.877	24.997	27.296	1387.2
HC-BIOGRID	22924	36513	62413	57892	0.052	1.979	1.989	1.906	712.5
Homo_sapiens	20273	36513	41006	40202	0.085	7.876	6.442	6.636	642.7
hprd_pp	18403	36513	39994	40202	0.074	4.348	4.097	3.714	456.4
Mus_musculus	25146	36513	130384	100040	0.061	1.055	1.965	1.718	223.9
oregon1_010526	30514	31513	1104167	743313	0.087	3.254	70.383	47.740	683.3
oregon2_010526	29117	31513	954515	743313	0.088	3.983	73.942	59.103	822.1
$\lambda = 0.015$									
advogato	9570	14006	21027	22204	0.050	1.428	1.227	1.299	261.0
as20000102	13035	14006	705483	492651	0.047	0.776	22.939	16.136	377.6
ca-GrQc	8668	16228	17419	15419	0.031	0.637	0.493	0.361	345.8
ca-HepTh	7524	16228	15002	15419	0.167	1.641	0.939	1.050	11.5
C_elegans	10956	16228	25233	22204	0.034	0.782	0.740	0.732	267.6
com-amazon.all	8228	18451		15419	0.301	13.814		7.785	21.9
dip20090126_MAX	17578	18451			0.203	3.851			15197.2
D_melanogaster	9350	14006	17229	18503	0.053	2.216	1.904	2.182	519.3
email-Enron	11209	16228	29134	31974	0.170	8.845	10.510	12.423	1367.4
HC-BIOGRID	12694	16228	28805	26645	0.043	0.858	0.946	0.947	708.6
Homo_sapiens	10142	16228	18491	18503	0.072	3.717	3.076	3.061	640.4
hprd_pp	10659	16228	17969	18503	0.056	1.919	1.719	1.752	451.5
Mus_musculus	11825	16228	59756	46043	0.033	0.458	0.906	0.812	222.8
oregon1_010526	13662	14006	426845	342118	0.056	1.522	26.420	21.871	681.4
oregon2_010526	13024	14006	333638	342118	0.060	1.773	26.070	27.298	833.6

Table 6.2. Detailed experimental results (undirected graphs). Empty values correspond to graphs on which the algorithm needed more than 1 hour.

	Number of iterations				Time (s)				Edges
Graph	KADABRA	RK	ABRA-Aut	ABRA-1.2	KADABRA	RK	ABRA-Aut	ABRA-1.2	KADABRA
$\lambda = 0.020$									
advogato	5874	7879	11993	12915	0.054	0.710	0.665	0.765	260.3
as20000102	7436	7879	312581	238814	0.037	0.441	10.066	7.819	376.2
ca-GrQc	5313	9129	9939	10762	0.032	0.356	0.293	0.268	347.9
ca-HepTh	5115	9129	8708	8968	0.191	0.891	0.694	0.611	10.5
C_elegans	7172	9129	14871	12915	0.030	0.439	0.436	0.439	263.5
com-amazon.all	5467	10379	12232	10762	0.331	7.683	4.338	5.459	17.9
dip20090126_MAX	9966	10379			0.148	2.165			15188.3
D_melanogaster	5610	7879	10201	10762	0.056	1.236	1.265	1.306	520.9
email-Enron	7458	9129	16443	15498	0.174	4.916	6.102	6.034	1371.7
HC-BIOGRID	8459	9129	17406	15498	0.026	0.505	0.602	0.582	716.6
Homo_sapiens	6292	9129	10481	10762	0.064	1.944	1.672	1.814	644.8
hprd_pp	6611	9129	10501	10762	0.050	1.089	0.930	1.050	449.8
Mus_musculus	7227	9129	31634	26782	0.026	0.255	0.507	0.532	221.0
oregon1_010526	7733	7879	220948	199011	0.051	0.863	13.584	12.989	679.2
oregon2_010526	7381	7879	152242	165842	0.059	1.031	11.676	13.290	836.0
$\lambda = 0.025$									
advogato	3883	5043	7439	7110	0.052	0.450	0.421	0.468	263.4
as20000102	4829	5043	130506	157779	0.033	0.285	4.097	5.108	373.5
ca-GrQc	3982	5843	6427	5925	0.028	0.242	0.180	0.162	342.1
ca-HepTh	3773	5843	6016	5925	0.176	0.573	0.374	0.416	11.8
C_elegans	4477	5843	9557	8532	0.025	0.292	0.293	0.293	266.6
com-amazon.all	4059	6643	58995	14745	0.338	4.744	9.644	7.217	21.3
dip20090126_MAX	6457	6643			0.125	1.397			15193.8
D_melanogaster	3993	5043	6279	7110	0.056	0.793	0.827	0.870	522.6
email-Enron	4576	5843	11001	12287	0.574	3.289	3.888	4.705	1381.5
HC-BIOGRID	5940	5843	11109	10239	0.029	0.321	0.414	0.404	714.0
Homo_sapiens	4796	5843	7109	7110	0.077	1.245	1.154	1.215	647.2
hprd_pp	5071	5843	6772	7110	0.052	0.687	0.579	0.647	446.3
Mus_musculus	4477	5843	18626	17694	0.026	0.168	0.302	0.385	219.8
oregon1_010526	5027	5043	92520	109568	0.058	0.516	5.762	7.014	681.0
oregon2_010526	4763	5043	86287	91306	0.050	0.638	7.140	7.420	847.5
$\lambda = 0.030$									
advogato	3256	3502	5521	5090	0.048	0.361	0.335	0.322	260.6
as20000102	3388	3502	122988	94140	0.029	0.199	3.899	3.182	378.7
ca-GrQc	2981	4057	4686	4241	0.025	0.169	0.145	0.175	344.7
ca-HepTh	2992	4057	4022	4241	0.190	0.435	0.286	0.341	7.9
C_elegans	3707	4057	6905	6108	0.026	0.198	0.218	0.217	265.9
com-amazon.all	3157	4613	39917	12668	0.330	3.631	8.491	6.852	17.5
dip20090126_MAX	4499	4613	12373086		0.300	0.972	1958.083		15199.0
D_melanogaster	2893	3502	4883	5090	0.052	0.562	0.620	0.807	510.4
email-Enron	3619	4057	7321	7330	0.172	2.735	2.724	2.806	1399.7
HC-BIOGRID	3883	4057	7499	7330	0.024	0.367	0.316	0.307	720.8
Homo_sapiens	3322	4057	4982	5090	0.066	0.897	0.842	0.877	654.2
hprd_pp	3355	4057	5028	5090	0.048	0.478	0.458	0.503	448.8
Mus_musculus	3806	4057	14290	10556	0.033	0.127	0.237	0.233	221.4
oregon1_010526	3542	3502	85854	78450	0.052	0.366	5.402	5.039	675.7
oregon2_010526	3355	3502	61841	65375	0.048	0.509	4.972	5.302	822.8

Table 6.3. Detailed experimental results (directed graphs). Empty values correspond to graphs on which the algorithm needed more than 1 hour.

	Number of iterations				Time (s)				Edges
Graph	KADABRA	RK	ABRA-Aut	ABRA-1.2	KADABRA	RK	ABRA-Aut	ABRA-1.2	KADABRA
$\lambda = 0.005$									
as-caida20071105	103488	146052	546951	462826	0.253	35.652	96.312	85.201	1066.4
cfinder-google	137313	146052			0.820	14.190			554.4
cit-HepTh	98054	166052	481476	462826	0.579	22.651	38.339	37.720	5773.1
ego-gplus	37862	66052		2388093	0.136	6.266		11.912	1.9
ego-twitter	37125	66052		154998	0.178	6.181		4.804	2.3
freeassoc	41602	166052	89424	89697	0.116	9.384	1.036	0.997	223.5
lasagne-spanishbook	112266	146052	8918751	4126626	0.250	17.374	687.815	318.784	552.8
opsahl-openflights	73744	146052	200164	185998	0.179	6.191	5.165	4.849	431.1
p2p-Gnutella31	39193	166052	81335	89697	0.254	50.542	10.213	10.662	162.1
polblogs	71423	126052	387278	321406	0.174	1.165	3.522	3.017	190.3
soc-Epinions1	58223	146052	109607	107637	0.671	100.516	62.524	62.167	671.9
subelj-cora-cora	68112	186052	180740	185998	0.185	19.012	8.464	8.873	440.4
subelj-jdk-jdk	42361	146052	84549	89697	0.110	2.955	0.230	0.257	51.5
subelj-jung-j-jung-j	43637	126052	84225	89697	0.216	2.397	0.238	0.211	45.9
wiki-Vote	47003	126052	100153	107637	0.131	5.916	2.990	3.219	162.4
$\lambda = 0.010$									
as-caida20071105	30382	36513	132997	120048	0.135	8.902	22.251	20.315	1066.1
cfinder-google	34452	36513			0.156	3.664			553.2
cit-HepTh	27203	41513	117633	120048	0.255	5.654	8.803	9.677	5798.8
ego-gplus	13123	16513		4602412	0.085	1.584		22.510	2.3
ego-twitter	13310	16513		83366	0.086	1.518		3.500	2.2
freeassoc	13222	41513	23586	23264	0.080	2.335	0.238	0.227	220.7
lasagne-spanishbook	32527	36513	1366576	1070372	0.101	4.339	104.916	83.610	553.4
opsahl-openflights	22473	36513	52196	48243	0.059	1.475	1.348	1.339	432.0
p2p-Gnutella31	13101	41513	21567	23264	0.192	12.950	2.677	2.831	162.1
polblogs	22286	31513	101466	83366	0.046	0.298	1.078	0.834	190.6
soc-Epinions1	17061	36513	28493	27917	0.320	27.194	16.516	15.974	659.5
subelj-cora-cora	23078	46513	47936	48243	0.128	4.797	1.988	2.101	432.4
subelj-jdk-jdk	14047	36513	22038	23264	0.066	0.734	0.099	0.075	52.2
subelj-jung-j-jung-j	14894	36513	22266	23264	0.064	0.696	0.113	0.083	46.4
wiki-Vote	17380	31513	26352	27917	0.088	1.446	0.792	0.870	155.7
$\lambda = 0.015$									
as-caida20071105	14157	16228	55049	55252	0.477	3.963	8.518	8.914	1059.6
cfinder-google	15400	16228			0.123	1.666			558.1
cit-HepTh	13002	18451	47035	46043	0.232	2.529	3.807	3.766	5883.0
ego-gplus	7205	7340		2118317	0.080	0.710		12.808	2.2
ego-twitter	7403	7340	1958981	114573	0.082	0.704	14.021	5.304	2.3
freeassoc	7095	18451	10956	10707	0.297	1.072	0.115	0.110	222.0
lasagne-spanishbook	14542	16228	437041	410542	0.068	1.936	34.098	33.153	552.8
opsahl-openflights	11550	16228	24433	22204	0.034	0.649	0.643	0.648	433.9
p2p-Gnutella31	7227	18451	10002	10707	0.190	5.732	1.317	1.444	157.1
polblogs	10296	14006	46648	38369	0.029	0.136	0.516	0.435	189.5
soc-Epinions1	9273	16228	13571	12849	0.450	12.115	7.661	7.629	662.0
subelj-cora-cora	11297	20673	20940	22204	0.502	2.135	0.937	1.073	445.6
subelj-jdk-jdk	8360	14006	10045	10707	0.052	0.288	0.080	0.049	51.6
subelj-jung-j-jung-j	8712	16228	10319	10707	0.046	0.312	0.068	0.042	45.6
wiki-Vote	8668	14006	12406	12849	0.408	0.659	0.380	0.429	152.6

Table 6.4. Detailed experimental results (directed graphs). Empty values correspond to graphs on which the algorithm needed more than 1 hour.

	Number of iterations				Time (s)				Edges
Graph	KADABRA	RK	ABRA-Aut	ABRA-1.2	KADABRA	RK	ABRA-Aut	ABRA-1.2	KADABRA
$\lambda = 0.020$									
as-caida20071105	9086	9129	31242	32139	0.104	2.226	4.954	5.087	1064.2
cfinder-google	8745	9129			0.353	0.946			551.9
cit-HepTh	8679	10379	27755	32139	1.249	1.442	2.225	2.684	5758.0
ego-gplus	4785	4129		1478684	0.081	0.395		9.234	2.6
ego-twitter	4950	7879		138201	0.083	0.743		5.079	2.4
freeassoc	4268	10379	6509	6227	0.065	0.609	0.078	0.073	216.4
lasagne-spanishbook	8338	9129	294793	286577	0.058	1.074	22.405	22.468	555.0
opsahl-openflights	7392	9129	14202	12915	0.029	0.364	0.390	0.391	432.3
p2p-Gnutella31	4697	10379	5700	6227	0.190	3.162	0.695	0.816	156.7
polblogs	6325	7879	25593	22318	0.023	0.076	0.283	0.252	188.4
soc-Epinions1	5489	9129	7686	7473	0.457	6.738	4.506	4.335	651.8
subelj-cora-cora	6325	11629	12437	12915	0.500	1.203	0.571	0.520	450.8
subelj-jdk-jdk	5456	9129	6070	6227	0.191	0.192	0.062	0.044	52.3
subelj-jung-j-jung-j	5643	9129		6227	0.217	0.176		0.045	46.6
wiki-Vote	4939	7879	7125	7473	0.075	0.368	0.221	0.259	152.2
$\lambda = 0.025$									
as-caida20071105	5723	5843	21020	21233	0.022	1.465	3.129	3.340	1093.4
cfinder-google	6275	5843			0.019	0.648			758.0
cit-HepTh	5206	6643	15915	21233	0.034	0.940	1.351	1.891	6130.5
ego-gplus	2989	5043		4200646	0.013	0.485		20.309	2.6
ego-twitter	2958	2643		157779	0.012	0.248		6.291	2.4
freeassoc	2804	6643	4285	4114	0.009	0.399	0.061	0.058	261.5
lasagne-spanishbook	5409	5043	129999	131482	0.013	0.592	10.040	10.221	626.1
opsahl-openflights	4557	5843	10116	8532	0.009	0.236	0.290	0.267	561.3
p2p-Gnutella31	3069	6643	3931	4114	0.043	2.149	0.590	0.663	176.8
polblogs	3880	5043	15986	14745	0.007	0.049	0.185	0.176	241.9
soc-Epinions1	3689	5843	5060	4937	0.188	4.158	2.798	2.791	888.1
subelj-cora-cora	5264	7443	7699	8532	0.020	0.781	0.360	0.408	436.5
subelj-jdk-jdk	3201	5843	9428	4937	0.008	0.122	0.065	0.036	57.2
subelj-jung-j-jung-j	3168	5043	13471	5925	0.007	0.098	0.057	0.045	57.7
wiki-Vote	3265	5043	4566	4937	0.009	0.241	0.137	0.178	174.7
$\lambda = 0.030$									
as-caida20071105	3956	4057	12696	15202	0.017	1.029	1.973	2.434	1285.2
cfinder-google	4419	4057			0.013	0.412			770.5
cit-HepTh	4062	4613	13172	12668	0.033	0.672	1.195	1.059	6131.6
ego-gplus	2434	1835		4330990	0.009	0.188		21.395	3.1
ego-twitter	2270	1835	98839	135562	0.008	0.174	4.909	5.510	2.2
freeassoc	2105	4613	3008	3534	0.006	0.285	0.101	0.091	250.7
lasagne-spanishbook	3820	4057	158028	94140	0.010	0.487	12.564	7.855	656.8
opsahl-openflights	3450	4057	6556	6108	0.007	0.165	0.184	0.195	481.4
p2p-Gnutella31	2367	4613	2874	2945	0.036	1.412	0.422	0.445	166.5
polblogs	3567	3502	11357	8796	0.007	0.036	0.151	0.122	207.9
soc-Epinions1	2659	4057	3585	3534	0.312	3.211	2.186	2.046	918.3
subelj-cora-cora	3790	5169	5681	5090	0.016	0.564	0.272	0.265	422.6
subelj-jdk-jdk	2425	4057	25575	5090	0.006	0.097	0.100	0.064	57.4
subelj-jung-j-jung-j	2436	3502	43584	5090	0.006	0.079	0.140	0.059	57.0
wiki-Vote	2633	3502	3467	3534	0.006	0.188	0.148	0.149	188.2

6.9 Internet Movies Database Case Study

After showing that KADABRA outperforms all previous approaches, we use the new algorithm to compute the top-k betweenness centralities of large graphs, which were unfeasible to handle with the previous algorithms.

The first set of graph is a series of temporal snapshots of the IMDB actor collaboration network, in which two actors are connected if they played together in a movie (for more information on the graph, we refer to Section 4.7).

We have run our algorithm with $\lambda = 0.0002$ and $\eta = 0.1$: as discussed in Section 6.7, this means that either two nodes are ranked correctly, or their centrality is known with precision at most λ. As a consequence, if two nodes are not ranked correctly, the difference between their real betweenness is at most 2λ. In the latter case, in the results, we report a lower and an upper bound on the ranking of the node. The running time of the algorithm is plotted in Figure 6.4, while full results are available in Tables 6.5 to 6.20.[2]

[2] We remark that, as for the IMDB database, the top-k betweenness centralities of a single snapshot of a

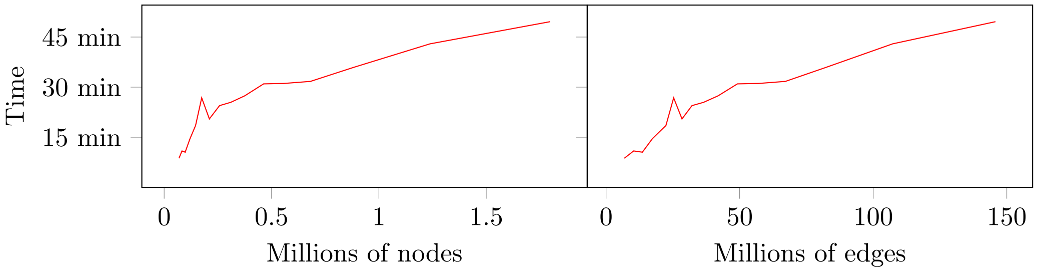


Figure 6.4. The total time of computation of KADABRA on increasing snapshots of the IMDB graph.

The Algorithm. From Figure 6.4, we see that all the graphs were processed in less than one hour, and it seems that the time needed by our algorithm scales slightly sublinearly with respect to the size of the graph. This result respects the analytical results in Section 8.13.2, because the degrees in the actor collaboration network are power law distributed with exponent $\beta \approx 2.13$ (`http://konect.uni-koblenz.de/networks/actor-collaboration`). Finally, we observe that the ranking is quite precise: indeed, most of the times, there are very few nodes in the top-5 with the same ranking, and the ranking rarely contains significantly more than 10 nodes.

The Results. In 2014, the most central actor is Ron Jeremy, who is listed in the Guinness Book of World Records for "Most Appearances in Adult Films", with more than 2000 appearances. Among his non-adult ones, we mention *The Godfather Part III*, *Ghostbusters*, *Crank: High Voltage* and *Family Guy*[3]. His topmost centrality in the actor collaboration network has been previously observed by similar experiments on betweenness centrality [164]. Indeed, around 3 actors out of 100 in the IMDB database played in adult movies, which explains why the high number of appearances of Ron Jeremy both in the adult and non-adult film industry rises his betweenness to the top.

The second most-central actor is Lloyd Kaufman, which is best known as a co-founder of *Troma Entertainment Film Studio* and as the director of many of their feature films, including the cult movie *The Toxic Avenger*. His high betweenness score is likely due to his central role in the low-budget independent film industry.

The third "actor" is the historical German dictator Adolf Hitler, since his appearances in several historical footages, that were re-used in several movies (e.g. in *The Imitation Game*), are credited by IMDB as cameo role. Indeed, he appears among the topmost actors since the 1984 snapshot, being the first one in the 1989 and 1994 ones, and during those years many movies about the World War II were produced. .

Observe that the betweenness centrality measure on our graph does not discriminate between important and marginal roles. For example, the actress Bess Flowers, who appears among the top actors in the snapshots from 1959 to 1979, rarely played major roles, but she appeared in over 700 movies in her 41 years career.

similar graph (`hollywood-2009` in `http://law.di.unimi.it/datasets.php`) have been previously computed exactly, with one week of computation on a 40-core machine [164]

[3] The latter is a TV-series, which are not taken into account in our data.

Table 6.5. The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1939 (69011 nodes), computed by KADABRA with $\eta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Est. betweenness	Upper bound
1)	Meyer, Torben	0.022331	0.022702	0.023049
2)	Roulien, Raul	0.021361	0.021703	0.022071
3)	Myzet, Rudolf	0.014229	0.014525	0.014747
4)	Sten, Anna	0.013245	0.013460	0.013723
5)	Negri, Pola	0.012509	0.012768	0.012943
6-7)	Jung, Shia	0.012250	0.012379	0.012509
6-7)	Ho, Tai-Hau	0.012195	0.012324	0.012454
8)	Goetzke, Bernhard	0.010721	0.010978	0.011201
9-10)	Yamamoto, Togo	0.010095	0.010224	0.010354
9-10)	Kamiyama, Sōjin	0.010087	0.010215	0.010344

Table 6.6. The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1944 (83068 nodes), computed by KADABRA with $\eta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Est. betweenness	Upper bound
1)	Meyer, Torben	0.018320	0.018724	0.019136
2)	Kamiyama, Sōjin	0.012629	0.012964	0.013308
3-4)	Jung, Shia	0.010751	0.010901	0.011053
3-4)	Ho, Tai-Hau	0.010704	0.010854	0.011005
5)	Myzet, Rudolf	0.010365	0.010514	0.010666
6-7)	Sten, Anna	0.009778	0.009928	0.010080
6-7)	Goetzke, Bernhard	0.009766	0.009915	0.010066
8)	Yamamoto, Togo	0.009108	0.009327	0.009539
9)	Parìs, Manuel	0.008649	0.008859	0.009108
10)	Hayakawa, Sessue	0.007916	0.008158	0.008369

Table 6.7. The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1949 (97824 nodes), computed by KADABRA with $\eta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Est. betweenness	Upper bound
1)	Meyer, Torben	0.016139	0.016679	0.017236
2)	Kamiyama, Sōjin	0.012351	0.012822	0.013312
3)	Parìs, Manuel	0.011104	0.011552	0.011861
4)	Yamamoto, Togo	0.010342	0.010639	0.011086
5-6)	Jung, Shia	0.008926	0.009120	0.009318
5-6)	Goetzke, Bernhard	0.008567	0.008762	0.008962
7-9)	Paananen, Tuulikki	0.008147	0.008341	0.008539
7-9)	Sten, Anna	0.007969	0.008164	0.008363
7-9)	Mayer, Ruby	0.007967	0.008162	0.008362
10-12)	Ho, Tai-Hau	0.007538	0.007732	0.007930
10-12)	Hayakawa, Sessue	0.007399	0.007593	0.007792
10-12)	Haas, Hugo (I)	0.007158	0.007352	0.007552

Table 6.8. The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1954 (120430 nodes), computed by KADABRA with $\eta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Est. betweenness	Upper bound
1)	Meyer, Torben	0.013418	0.013868	0.014334
2)	Kamiyama, Sōjin	0.010331	0.010726	0.011089
3-4)	Ertugrul, Muhsin	0.009956	0.010141	0.010331
3-4)	Jung, Shia	0.009643	0.009826	0.010013
5-6)	Singh, Ram (I)	0.008657	0.008841	0.009030
5-6)	Paananen, Tuulikki	0.008383	0.008567	0.008755
7-9)	Parìs, Manuel	0.007886	0.008070	0.008257
7-10)	Goetzke, Bernhard	0.007802	0.007987	0.008176
7-10)	Yamaguchi, Shirley	0.007531	0.007716	0.007905
8-10)	Hayakawa, Sessue	0.007473	0.007657	0.007845

Table 6.9. The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1959 (146253 nodes), computed by KADABRA with $\eta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Est. betweenness	Upper bound
1-2)	Singh, Ram (I)	0.010683	0.010877	0.011075
1-2)	Frees, Paul	0.010372	0.010566	0.010763
3)	Meyer, Torben	0.009478	0.009821	0.010235
4-5)	Jung, Shia	0.008623	0.008816	0.009013
4-5)	Ghosh, Sachin	0.008459	0.008651	0.008847
6-7)	Myzet, Rudolf	0.007085	0.007278	0.007476
6-7)	Yamaguchi, Shirley	0.006908	0.007101	0.007299
8)	de Còrdova, Arturo	0.006391	0.006582	0.006778
9-11)	Kamiyama, Sōjin	0.005861	0.006054	0.006254
9-12)	Paananen, Tuulikki	0.005810	0.006003	0.006202
9-12)	Flowers, Bess	0.005620	0.005813	0.006012
10-12)	Parìs, Manuel	0.005442	0.005635	0.005835

Table 6.10. The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1964 (174826 nodes), computed by KADABRA with $\eta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Est. betweenness	Upper bound
1)	Frees, Paul	0.013140	0.013596	0.014067
2)	Meyer, Torben	0.007279	0.007617	0.007856
3-4)	Harris, Sam (II)	0.006813	0.006967	0.007124
3-5)	Myzet, Rudolf	0.006696	0.006849	0.007005
4-5)	Flowers, Bess	0.006422	0.006572	0.006726
6)	Kong, King (I)	0.005909	0.006104	0.006422
7)	Yuen, Siu Tin	0.005114	0.005264	0.005420
8)	Miller, Marvin (I)	0.004708	0.004859	0.005015
9-12)	de Còrdova, Arturo	0.004147	0.004299	0.004457
9-18)	Haas, Hugo (I)	0.003888	0.004039	0.004197
9-18)	Singh, Ram (I)	0.003854	0.004004	0.004160
9-18)	Kamiyama, Sōjin	0.003848	0.003999	0.004155
10-18)	Sauli, Anneli	0.003827	0.003978	0.004135
10-18)	King, Walter Woolf	0.003774	0.003923	0.004078
10-18)	Vanel, Charles	0.003716	0.003867	0.004024
10-18)	Kowall, Mitchell	0.003684	0.003834	0.003990
10-18)	Holmes, Stuart	0.003603	0.003752	0.003907
10-18)	Sten, Anna	0.003582	0.003733	0.003890

Table 6.11. The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1969 (210527 nodes), computed by KADABRA with $\eta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Est. betweenness	Upper bound
1)	Frees, Paul	0.010913	0.011446	0.012005
2-3)	Yuen, Siu Tin	0.006157	0.006349	0.006547
2-3)	Tamiroff, Akim	0.006097	0.006291	0.006490
4-6)	Meyer, Torben	0.005675	0.005869	0.006069
4-7)	Harris, Sam (II)	0.005639	0.005830	0.006027
4-8)	Rubener, Sujata	0.005427	0.005618	0.005815
5-8)	Myzet, Rudolf	0.005253	0.005444	0.005641
6-8)	Flowers, Bess	0.005136	0.005328	0.005526
9-10)	Kong, King (I)	0.004354	0.004544	0.004741
9-10)	Sullivan, Elliott	0.004208	0.004398	0.004596

Table 6.12. The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1974 (257896 nodes), computed by KADABRA with $\eta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Est. betweenness	Upper bound
1)	Frees, Paul	0.008507	0.008958	0.009295
2)	Chen, Sing	0.007734	0.008056	0.008507
3)	Welles, Orson	0.006115	0.006497	0.006903
4-5)	Loren, Sophia	0.005056	0.005221	0.005392
4-7)	Rubener, Sujata	0.004767	0.004933	0.005106
5-8)	Harris, Sam (II)	0.004628	0.004795	0.004967
5-8)	Tamiroff, Akim	0.004625	0.004790	0.004962
6-10)	Meyer, Torben	0.004382	0.004548	0.004720
8-12)	Flowers, Bess	0.004259	0.004425	0.004598
8-12)	Yuen, Siu Tin	0.004229	0.004397	0.004571
9-12)	Carradine, John	0.004026	0.004192	0.004364
9-12)	Myzet, Rudolf	0.003984	0.004151	0.004325

Table 6.13. The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1979 (310278 nodes), computed by KADABRA with $\eta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Est. betweenness	Upper bound
1)	Chen, Sing	0.007737	0.008220	0.008647
2)	Frees, Paul	0.006852	0.007255	0.007737
3-5)	Welles, Orson	0.004894	0.005075	0.005263
3-6)	Carradine, John	0.004623	0.004803	0.004989
3-6)	Loren, Sophia	0.004614	0.004796	0.004985
4-6)	Rubener, Sujata	0.004284	0.004464	0.004651
7-17)	Tamiroff, Akim	0.003516	0.003696	0.003885
7-17)	Meyer, Torben	0.003479	0.003657	0.003844
7-17)	Quinn, Anthony (I)	0.003447	0.003626	0.003815
7-17)	Flowers, Bess	0.003446	0.003625	0.003815
7-17)	Mitchell, Gordon (I)	0.003417	0.003596	0.003785
7-17)	Sullivan, Elliott	0.003371	0.003551	0.003740
7-17)	Rietty, Robert	0.003368	0.003547	0.003735
7-17)	Tanba, Tetsurō	0.003360	0.003537	0.003724
7-17)	Harris, Sam (II)	0.003331	0.003510	0.003699
7-17)	Lewgoy, Josè	0.003223	0.003402	0.003590
7-17)	Dalio, Marcel	0.003185	0.003364	0.003553

Table 6.14. The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1984 (375322 nodes), computed by KADABRA with $\eta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Est. betweenness	Upper bound
1)	Chen, Sing	0.007245	0.007716	0.008218
2-4)	Welles, Orson	0.005202	0.005391	0.005587
2-4)	Frees, Paul	0.005174	0.005363	0.005559
2-5)	Hitler, Adolf	0.004906	0.005094	0.005290
4-6)	Carradine, John	0.004744	0.004932	0.005127
5-7)	Mitchell, Gordon (I)	0.004418	0.004606	0.004802
6-8)	Jürgens, Curd	0.004169	0.004356	0.004551
7-8)	Kinski, Klaus	0.003938	0.004123	0.004318
9-12)	Rubener, Sujata	0.003396	0.003585	0.003785
9-12)	Lee, Christopher (I)	0.003391	0.003576	0.003771
9-12)	Loren, Sophia	0.003357	0.003542	0.003738
9-12)	Harrison, Richard (II)	0.003230	0.003417	0.003614

Table 6.15. The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1989 (463078 nodes), computed by KADABRA with $\eta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Est. betweenness	Upper bound
1-2)	Hitler, Adolf	0.005282	0.005467	0.005658
1-3)	Chen, Sing	0.005008	0.005192	0.005382
2-4)	Carradine, John	0.004648	0.004834	0.005027
3-4)	Harrison, Richard (II)	0.004515	0.004697	0.004887
5-6)	Welles, Orson	0.004088	0.004271	0.004462
5-9)	Mitchell, Gordon (I)	0.003766	0.003948	0.004139
6-9)	Kinski, Klaus	0.003691	0.003874	0.004065
6-11)	Lee, Christopher (I)	0.003610	0.003793	0.003984
6-11)	Frees, Paul	0.003582	0.003766	0.003960
8-13)	Jürgens, Curd	0.003306	0.003486	0.003676
8-13)	Pleasence, Donald	0.003299	0.003479	0.003670
10-13)	Mitchell, Cameron (I)	0.003105	0.003285	0.003476
10-13)	von Sydow, Max (I)	0.002982	0.003161	0.003350

Table 6.16. The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1994 (557373 nodes), computed by KADABRA with $\eta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Est. betweenness	Upper bound
1)	Hitler, Adolf	0.005227	0.005676	0.006164
2-6)	Harrison, Richard (II)	0.003978	0.004165	0.004362
2-6)	von Sydow, Max (I)	0.003884	0.004069	0.004264
2-7)	Lee, Christopher (I)	0.003718	0.003907	0.004106
2-7)	Carradine, John	0.003696	0.003883	0.004079
2-7)	Chen, Sing	0.003683	0.003871	0.004068
4-10)	Jeremy, Ron	0.003336	0.003524	0.003722
7-11)	Pleasence, Donald	0.003253	0.003439	0.003637
7-11)	Rey, Fernando (I)	0.003234	0.003420	0.003617
7-15)	Smith, William (I)	0.003012	0.003199	0.003397
8-15)	Welles, Orson	0.002885	0.003072	0.003271
10-15)	Mitchell, Gordon (I)	0.002851	0.003036	0.003232
10-15)	Kinski, Klaus	0.002705	0.002890	0.003087
10-15)	Mitchell, Cameron (I)	0.002671	0.002858	0.003058
10-15)	Quinn, Anthony (I)	0.002640	0.002826	0.003026

Table 6.17. The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1999 (681358 nodes), computed by KADABRA with $\eta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Est. betweenness	Upper bound
1)	Jeremy, Ron	0.007380	0.007913	0.008484
2)	Hitler, Adolf	0.004601	0.005021	0.005480
3-4)	Lee, Christopher (I)	0.003679	0.003849	0.004028
3-4)	von Sydow, Max (I)	0.003604	0.003775	0.003953
5-6)	Harrison, Richard (II)	0.003041	0.003211	0.003390
5-7)	Carradine, John	0.002943	0.003114	0.003296
6-11)	Chen, Sing	0.002662	0.002834	0.003018
7-14)	Rey, Fernando (I)	0.002569	0.002740	0.002922
7-14)	Smith, William (I)	0.002559	0.002729	0.002910
7-14)	Pleasence, Donald	0.002556	0.002725	0.002906
7-14)	Sutherland, Donald (I)	0.002449	0.002617	0.002796
8-14)	Quinn, Anthony (I)	0.002307	0.002476	0.002658
8-14)	Mastroianni, Marcello	0.002271	0.002440	0.002621
8-14)	Saxon, John	0.002251	0.002420	0.002602

Table 6.18. The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 2004 (880032 nodes), computed by KADABRA with $\eta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Est. betweenness	Upper bound
1)	Jeremy, Ron	0.010653	0.011370	0.012136
2)	Hitler, Adolf	0.005333	0.005840	0.006396
3-4)	von Sydow, Max (I)	0.003424	0.003608	0.003802
3-4)	Lee, Christopher (I)	0.003403	0.003587	0.003781
5-6)	Kier, Udo	0.002898	0.003081	0.003275
5-8)	Keitel, Harvey (I)	0.002646	0.002828	0.003023
6-12)	Hopper, Dennis	0.002424	0.002607	0.002804
6-16)	Smith, William (I)	0.002322	0.002504	0.002700
7-17)	Sutherland, Donald (I)	0.002241	0.002422	0.002617
7-23)	Carradine, David	0.002149	0.002329	0.002526
7-23)	Carradine, John	0.002147	0.002328	0.002524
7-23)	Harrison, Richard (II)	0.002054	0.002234	0.002430
8-23)	Sharif, Omar	0.002043	0.002222	0.002418
8-23)	Steiger, Rod	0.001988	0.002165	0.002358
8-23)	Quinn, Anthony (I)	0.001974	0.002151	0.002344
8-23)	Depardieu, Gèrard	0.001966	0.002148	0.002346
9-23)	Sheen, Martin	0.001913	0.002093	0.002291
10-23)	Rey, Fernando (I)	0.001866	0.002044	0.002238
10-23)	Kane, Sharon	0.001857	0.002038	0.002237
10-23)	Pleasence, Donald	0.001859	0.002037	0.002232
10-23)	Skarsgård, Stellan	0.001848	0.002026	0.002221
10-23)	Mueller-Stahl, Armin	0.001789	0.001969	0.002166
10-23)	Hong, James (I)	0.001780	0.001957	0.002152

Table 6.19. The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 2009 (1237879 nodes), computed by KADABRA with $\eta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Est. betweenness	Upper bound
1)	Jeremy, Ron	0.010531	0.011237	0.011991
2)	Hitler, Adolf	0.005500	0.006011	0.006568
3-4)	Kaufman, Lloyd	0.003620	0.003804	0.003997
3-4)	Kier, Udo	0.003472	0.003654	0.003845
5-6)	Lee, Christopher (I)	0.003056	0.003240	0.003435
5-8)	Carradine, David	0.002866	0.003050	0.003245
6-8)	Keitel, Harvey (I)	0.002659	0.002840	0.003034
6-9)	von Sydow, Max (I)	0.002532	0.002713	0.002907
8-13)	Hopper, Dennis	0.002237	0.002419	0.002616
9-15)	Skarsgård, Stellan	0.002153	0.002333	0.002529
9-15)	Depardieu, Gèrard	0.002001	0.002181	0.002377
9-15)	Hauer, Rutger	0.001894	0.002074	0.002271
9-15)	Sutherland, Donald (I)	0.001875	0.002054	0.002250
10-15)	Smith, William (I)	0.001811	0.001990	0.002186
10-15)	Dafoe, Willem	0.001805	0.001986	0.002186

Table 6.20. The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken in 2014 (1797446 nodes), computed by KADABRA with $\eta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Est. betweenness	Upper bound
1)	Jeremy, Ron	0.009360	0.010058	0.010808
2)	Kaufman, Lloyd	0.005936	0.006492	0.007100
3)	Hitler, Adolf	0.004368	0.004844	0.005373
4-6)	Kier, Udo	0.003250	0.003435	0.003631
4-6)	Roberts, Eric (I)	0.003178	0.003362	0.003557
4-6)	Madsen, Michael (I)	0.003120	0.003305	0.003501
7-9)	Trejo, Danny	0.002652	0.002835	0.003030
7-9)	Lee, Christopher (I)	0.002551	0.002734	0.002931
7-12)	Estevez, Joe	0.002350	0.002534	0.002732
9-17)	Carradine, David	0.002116	0.002296	0.002492
9-17)	von Sydow, Max (I)	0.002023	0.002206	0.002405
9-17)	Keitel, Harvey (I)	0.001974	0.002154	0.002352
10-17)	Skarsgȧrd, Stellan	0.001945	0.002125	0.002323
10-17)	Dafoe, Willem	0.001899	0.002080	0.002279
10-17)	Hauer, Rutger	0.001891	0.002071	0.002269
10-17)	Depardieu, Gèrard	0.001763	0.001943	0.002142
10-17)	Rochon, Debbie	0.001745	0.001926	0.002126

6.10 Wikipedia Case Study

The other large graph considered is the Wikipedia citation network, whose nodes are Wikipedia pages, and which contains an edge from page p_1 to page p_2 if the text of page p_1 contains a link to page p_2 (for more information, see Section 4.8).

The time needed to compute an approximation of the 10 most central actors was approximately 1 hour and 38 minutes, and the results are available in Table 6.21.

All topmost pages in the betweenness centrality ranking, except for the `World War II`, are countries. This is not surprising if we consider that, for most topics (such as important people or events), the corresponding Wikipedia page refers to their geographical context (since it mentions the country of origin of the given person or where a given event took place). It is also worth noting the correlation between the high centrality of the `World War II` Wikipedia page and that of Adolf Hitler in the IMDB graph.

Interestingly, a similar ranking is obtained by considering the closeness centrality measure in the inverse graph, where a link from page p_1 to page p_2 exists if a link to page p_1 appears in page p_2 (see Section 5.9). However, in contrast with the results in Section 5.9 when edges are oriented in the usual way, the pages about specific years do not appear in the top ranking. We note that the betweenness centrality of a node in a directed graph does not change if the orientation of all edges is flipped.

Finally, the most important page is the `United States`, confirming a common conjecture. Indeed, in `http://wikirank.di.unimi.it/`, it is shown that the `United States` page is the center according to harmonic centrality, and many other measures. Further evidence for this conjecture comes from the Six Degree of Wikipedia game (`http://thewikigame.com/6-degrees-of-wikipedia`), where a player is asked to go from one page to the other following the smallest possible number of links: a hard variant of this game forces the player not to pass from the `United States` page, which is considered to be central. Our results thus confirm that the conjecture is indeed true for the betweenness centrality measure.

Table 6.21. The top-k betweenness centralities of the Wikipedia graph computed by KADABRA with $\eta = 0.1$ and $\lambda = 0.0002$.

Ranking	Wikipedia page	Lower bound	Estimated betweenness	Upper bound
1)	United States	0.046278	0.047173	0.048084
2)	France	0.019522	0.020103	0.020701
3)	United Kingdom	0.017983	0.018540	0.019115
4)	England	0.016348	0.016879	0.017428
5-6)	Poland	0.012092	0.012287	0.012486
5-6)	Germany	0.011930	0.012124	0.012321
7)	India	0.009683	0.010092	0.010518
8-12)	World War II	0.008870	0.009065	0.009265
8-12)	Russia	0.008660	0.008854	0.009053
8-12)	Italy	0.008650	0.008845	0.009045
8-12)	Canada	0.008624	0.008819	0.009018
8-12)	Australia	0.008620	0.008814	0.009013

6.11 Bibliographic Notes

This chapter builds on a large amount of research on approximation algorithms for betweenness centrality [93, 43, 82, 109, 136, 137]. The code used in the experimental results is taken from these papers. Furthermore, our new algorithm is a rigorous application of the technique of adaptive sampling: although this technique was widely used in the past [111, 112, 13, 133, 137], as far as we know, the only rigorous application is [133], because all the other works ignored the stochastic dependency discussed in Section 6.4. The definition of the new algorithm and the techniques used to formalize the adaptive sampling are original, and they are published in [36].

Chapter 7

Computing Hyperbolicity: the HYP Algorithm

Abstract

The (Gromov) hyperbolicity is a metric property of a graph, which has been recently applied in different contexts, such as the design of routing schemes, network security, computational biology, the analysis of graph algorithms, and the classification of complex networks.

Computing the hyperbolicity of a graph can be very time consuming: indeed, the best available algorithm has running time $\mathcal{O}(n^{3.69})$ in the worst-case, which is clearly prohibitive for big graphs. In this chapter, we provide a new and more efficient algorithm: although its worst-case complexity is $\mathcal{O}(n^4)$, in practice it is much faster, allowing, for the first time, the computation of the hyperbolicity of graphs with up to 200 000 nodes. We experimentally show that our new algorithm drastically outperforms the best previously available algorithms, by analyzing a big dataset of real-world networks.

Using the new algorithm, we experimentally show that real-world networks are usually not hyperbolic, at least using the Gromov definition, shading further light on a long-standing discussion. Furthermore, we provide a new definition of hyperbolicity, and we provide evidences that this definition is more suited to the analysis of real-world graphs. Finally, we apply the new algorithm to compute the hyperbolicity of random graphs generated with the Erdös-Renyi model, the Chung-Lu model, and the Configuration Model.

The analysis of complex networks has provided several significant results, with a huge amount of applications in sociology, biology, economics, statistical physics, electrical engineering, and so on. These results are based on the analysis of large real-world networks, now made available by improvements in computer technology and by the pervasive presence of the Internet. One of the major challenges in this field is to understand which properties distinguish these networks from other kinds of graphs, such as random graphs [125], and which properties distinguish networks of different kinds [90], in order to classify general and particular behavior.

In this context, a significant role is played by the hyperbolic structure underlying a complex network, that is usually not present in random graphs [121, 49]. For instance, if we draw points from a hyperbolic space and we connect nearby points, we obtain a graph that shares many properties with real-world networks [102]. Furthermore, the Internet graph can be embedded in the hyperbolic space, preserving some metric properties [101, 22].

From these evidences, one might be tempted to say that hyperbolicity is a characteristic of many real-world complex networks: for this reason, several works have tried to measure this phenomenon. One of the most successful attempts is based on Gromov's definitions of hyperbolicity [84], which works in any metric space, and does not rely on complicated struc-

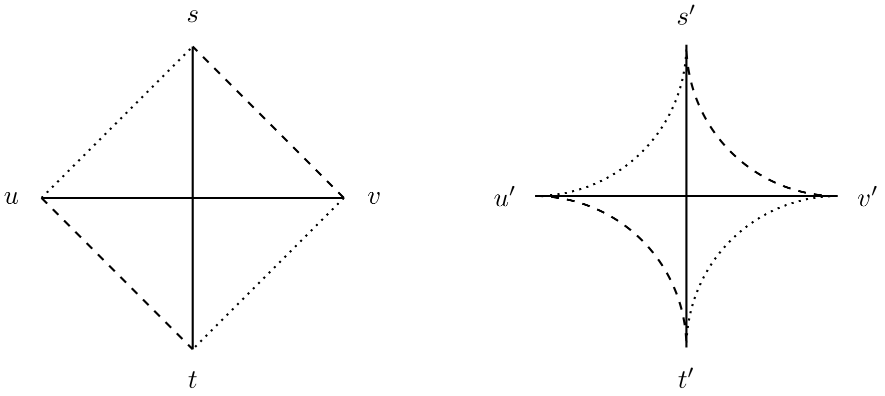


Figure 7.1. An intuition of the definition of δ. In both quadruples, assuming the lines are shortest paths, the biggest sum is $\mathrm{dist}(s,t)+\mathrm{dist}(u,v)$ (straight lines), and the other two sums are equal (dashed and dotted lines). However, $\delta(s,t,u,v) > \delta(s',t',u',v')$, because in the second case the two small sums are closer to the two big sums, since the underlying space is hyperbolic.

tures not available in graphs (geodesics, connections, and so on). Intuitively, this quantity reflects how the metric space of a graph is close to the metric space of a tree.

More formally, given an undirected graph $G = (V,E)$ (in this chapter, all graphs are undirected), the Gromov hyperbolicity of a quadruple of nodes $\delta(s,t,u,v)$ is defined as half the difference between the biggest two of the three sums

$$\mathrm{dist}(s,t)+\mathrm{dist}(u,v) \qquad \mathrm{dist}(s,u)+\mathrm{dist}(t,v) \qquad \mathrm{dist}(s,v)+\mathrm{dist}(t,u).$$

The hyperbolicity of G is $\delta(G) = \max_{s,t,u,v \in V} \delta(s,t,u,v)$. A small value of δ corresponds to an hyperbolic metric space, according to the intuition in Figure 7.1.

Several network properties are connected to the value of the hyperbolicity: here we just recall some of them. In [51], it is shown that a small hyperbolicity implies the existence of efficient distance and routing labeling schemes. In [120], the authors observe that a small hyperbolicity, that is, a negative curvature of an interconnection network, implies a faster congestion within the core of the network itself, and in [94] it is suggested that this property is significant in the context of network security and can, for example, mitigate the effect of distributed denial of service attacks. Instead, in [69], the hyperbolicity is used to implement a distance between trees, successively applied to the estimation of phylogenetic trees. From a more algorithmic point of view, it has been shown that several approximation algorithms for problems related to distances in graphs (such as diameter and radius computation [50], and minimum ball covering [52]) have an approximation ratio which depends on the hyperbolicity of the input graph. Moreover, some approximation algorithms with constant approximation factor rely on a data-structure whose size is proportional to the hyperbolicity of the input graph [100]. More in general, the hyperbolicity is connected to other important graph quantities, like treelength [50] and chordality [172]. In the field of the analysis of complex networks, the hyperbolicity and its connection with the size and the diameter of a network has been used in [10] in order to classify networks into three different classes, that is, strongly hyperbolic, hyperbolic, and non-hyperbolic, and to apply this classification to a small dataset of small biological networks. Finally, the hyperbolicity of random graphs has been analyzed in the case of several random graph models, such as the Erdös-Renyi model [121] and the Kleinberg model [49]. Moreover, in the latter paper, it is stated that the design of more efficient exact algorithms for the computation of the hyperbolicity would be of interest.

However, despite this large amount of work, there is no definitive proof that real-world graphs are indeed Gromov hyperbolic: some papers claim that they are [98, 10], but some other papers claim that they are not [9]. This is due to the fact that existing analyses are performed on very small graphs, because there is no algorithm that can handle large instances.

Indeed, even if the hyperbolicity computation problem is polynomial-time solvable by the trivial algorithm that computes $\delta(s,t,u,v)$ for each quadruple of nodes, the running time is $\mathcal{O}(n^4)$, which is prohibitive for most real-world networks. The best known algorithm uses fast max-min matrix multiplication algorithm to obtain a running time $\mathcal{O}(n^{3.69})$ [77], and it has been shown that hyperbolicity cannot be computed in $\mathcal{O}(n^{3.05})$ time, unless there exists a faster algorithm for max-min matrix multiplication than currently known. Such running times are prohibitive for analyzing large-scale graphs with more than 10 000 nodes.

Recently, new algorithms have been developed [56, 58]. Although these algorithms have worst-case running time $\mathcal{O}(n^4)$, they perform well in practice, making it possible to compute the hyperbolicity of graphs with up to 50 000 nodes.

Following this line of research, we propose a new algorithm to compute the hyperbolicity of a graph, taking some ideas from the algorithm in [58]. The new algorithm is available in the Sagemath graph library [64]. We heavily improve the performances of the algorithm in [58] through significant speed-ups in the most time consuming part: we obtain that the running time of the new algorithm is dominated by the preprocessing part, which needs time $\mathcal{O}(mn)$. This way, the $\mathcal{O}(n^4)$ bottleneck is almost removed, at least in practical instances. For this reason, we are able for the first time to compute the hyperbolicity of graphs with up to 200 000 nodes. We experimentally show these claims by analyzing a big dataset of real-world networks of different kinds.

As a case study, we apply the new algorithm to study the hyperbolicity of real-world graphs. First, our experiments show that real-world graphs are usually not hyperbolic in the Gromov sense, and we can quantify this phenomenon by showing that the hyperbolicity values are approximately normally distributed, between 0 and $\frac{D}{2}$, where D is the diameter of the graph (it is possible to analytically prove that $0 \leq \delta \leq \frac{D}{2}$ on all graphs). This result closes a long-standing discussion in real-world network analysis.

However, there are evidences that real-world graphs have an underlying hyperbolic structure, as shown in [101, 102, 22]. For this reason, we propose another approach to study the hyperbolicity of real-world graphs: considering the average of $\delta(s,t,u,v)$ instead of the maximum. We show that, in this sense, real-world networks are hyperbolic, and we provide an interpretation of this behavior in terms of "democracy", by proving results that link a low hyperbolicity to the existence of few nodes that "control" all shortest paths in the graph, and consequently to an aristocratic network. Conversely, if the graph is not hyperbolic, there is no "lobby" that controls all shortest paths, and the network is democratic. Then, we use the average hyperbolicity to define the "influence area" of a node in a complex network. This result lets us classify networks into two different categories: distributed and centralized networks. The former class contains all networks where every node tries to reach a specific goal, such as social networks and peer-to-peer networks, while the latter class contains networks that try to reach global goals, such as biological networks and power grid networks. We show that, in distributed networks, the influence area of a node is much smaller than in centralized networks, confirming again our interpretation of hyperbolicity, and our definition of influence area.

Finally, we apply our algorithm to the computation of the hyperbolicity of random graphs. For example, in the Chung-Lu model, we compute the hyperbolicity of graphs with up to 200 000 nodes, improving previous experiments that stop at 1 100 nodes [75].

7.1 The Currently Best Available Algorithm: ccl

In this section, we sketch the algorithm proposed in [58], whose main ideas and lemmas are the base of our new algorithm. The main idea is to improve the trivial algorithm by analyzing quadruples in a specific order, and by cutting the exploration of the quadruples as soon as some conditions are satisfied. We name this algorithm ccl, from the initials of the surnames of the authors. In particular, for each quadruple (s,t,u,v) of nodes, ccl computes $\tau(s,t;u,v)$

as defined below, instead of computing $\delta(s,t,u,v)$.

$$\tau(s,t;u,v) = \frac{1}{2}(\operatorname{dist}(s,t) + \operatorname{dist}(u,v) - \max\{\operatorname{dist}(s,u) + \operatorname{dist}(t,v), \operatorname{dist}(s,v) + \operatorname{dist}(t,u)\}).$$

Note that $\delta(G) = \max_{s,t,u,v \in V} \tau(s,t;u,v)$, because if $\operatorname{dist}(s,t) + \operatorname{dist}(u,v)$ is the maximum sum, then $\tau(s,t;u,v) = \delta(s,t,u,v)$, otherwise $\tau(s,t;u,v) \leq 0$.

Lemma 7.1 (Lemma 3.2 of [58])**.** *For any quadruple (s,t,u,v) of nodes, $\tau(s,t;u,v) \leq \frac{1}{2}\min(\operatorname{dist}(s,t), \operatorname{dist}(u,v))$.*

Proof. The following inequalities hold:

$$\begin{aligned}
2\tau(s,t;u,v) &= \operatorname{dist}(s,t) + \operatorname{dist}(u,v) - \max\{\operatorname{dist}(s,u) + \operatorname{dist}(t,v), \operatorname{dist}(s,v) + \operatorname{dist}(t,u)\} \\
&\leq \operatorname{dist}(s,t) + \operatorname{dist}(u,v) - \frac{1}{2}(\operatorname{dist}(s,u) + \operatorname{dist}(t,v) + \operatorname{dist}(s,v) + \operatorname{dist}(t,u)) \\
&\leq \operatorname{dist}(s,t) + \operatorname{dist}(u,v) - \frac{1}{2}\big(2\operatorname{dist}(u,v)\big) \\
&\leq \operatorname{dist}(s,t).
\end{aligned}$$

In a very similar way, one can prove that $2\tau(s,t;u,v) \leq \operatorname{dist}(u,v)$. □

In order to exploit this lemma, CCL stores all the $N = \frac{n(n-1)}{2}$ pairs of nodes inside a sequence $\mathtt{P} = (\{s_1,t_1\},\dots,\{s_N,t_N\})$, in decreasing order of distance (that is, if $\operatorname{dist}(s_i,t_i) > \operatorname{dist}(s_j,t_j)$, then $i < j$). For each i, CCL iterates over all pairs $\{s_j,t_j\}$ with $j < i$, and computes $\tau(s_i,t_i;s_j,t_j)$, storing the maximum value found in a variable δ_L (clearly, δ_L is a lower bound for $\delta(G)$). Even if iterating over the whole sequence P would lead us to the trivial algorithm, by applying Lemma 7.1 we may cut the exploration as soon as $\operatorname{dist}(s_i,t_i) \leq 2\delta_L$, because the τ value of all remaining quadruples is at most $\operatorname{dist}(s_i,t_i)$.

A further improvement is provided by the following lemma.

Lemma 7.2 ([150])**.** *Let s,t,u,v be four nodes, and let us assume that there exists an edge (s,s') such that $\operatorname{dist}(s',t) = \operatorname{dist}(s,t) + 1$. Then, $\tau(s,t;u,v) \leq \tau(s',t;u,v)$.*

Proof. The following inequalities hold:

$$\begin{aligned}
2\tau(s',t;u,v) = &\operatorname{dist}(s',t) + \operatorname{dist}(u,v) - \max\{ \\
&\quad \operatorname{dist}(s',u) + \operatorname{dist}(t,v), \operatorname{dist}(s',v) + \operatorname{dist}(t,u)\} \\
\geq &\ 1 + \operatorname{dist}(s,t) + \operatorname{dist}(u,v) - \max\{ \\
&\quad \operatorname{dist}(s,u) + \operatorname{dist}(t,v) + 1, \operatorname{dist}(s,v) + \operatorname{dist}(t,u) + 1\} \\
= &\ 2\tau(s,t;u,v).
\end{aligned}$$

□

Definition 7.3. A pair $\{s,t\}$ is far apart if there is no edge (s,s') such that $\operatorname{dist}(s',t) = \operatorname{dist}(s,t) + 1$ and no edge (t,t') such that $\operatorname{dist}(s,t') = \operatorname{dist}(s,t) + 1$.

By Lemma 7.2, CCL only needs to analyze far apart pairs, and, hence, in the following we denote by P the list of far apart pairs, and by N its cardinality. The pseudo-code of the algorithm CCL is provided in Algorithm 17.

Other improvements of this algorithm involve pre-processing the graph: first of all, we may analyze each biconnected component separately [58, Section 2], then, we may decompose the graph by modular decomposition, split decomposition [150], and clique decomposition [56].

7.2 The New Algorithm: HYP

In this section, we propose a new algorithm, named HYP, that improves upon CCL by further reducing the number of quadruples to consider.

Algorithm 17: Hyperbolicity algorithm proposed in [58], ccl.

```
Let P = $(\{s_1, t_1\}, \dots, \{s_N, t_N\})$ be the list of far apart pairs, in decreasing order of
  distance.
$\delta_L \leftarrow 0$;
for $i \in [1, N]$ do
    if $\mathrm{dist}(s_i, t_i) \leq 2\delta_L$ then
        return $\delta_L$;
    end
    for $j \in [1, i-1]$ do
        $\delta_L \leftarrow \max(\delta_L, \tau(s_i, t_i; s_j, t_j))$;
    end
end
return $\delta_L$;
```

7.2.1 Overview

The new algorithm hyp speeds-up the inner `for` loop in Algorithm 17, by decreasing the number of pairs to be analyzed. In particular, let us fix a pair (s_i, t_i) in the outer `for` loop and a lower bound δ_L: a node u is (i, δ_L)-*skippable* or simply *skippable* if, for any v, $\tau(s_i, t_i; u, v) \leq \delta_L$. It is clear that if a node u is skippable, the algorithm could skip the analysis of all quadruples containing s_i, t_i, and u. Even if it is not easy to compute the set of skippable nodes, we can define easy-to-verify conditions that imply that a node u is skippable (Section 7.2.2): a node not satisfying any of these conditions is named (i, δ_L)-*acceptable* or *acceptable*. Then, our algorithm discards all quadruples (s_i, t_i, u, v) where either u or v is not acceptable.

Furthermore, we define another condition such that if $\tau(s_i, t_i; u, v) > \delta_L$, then either u or v must satisfy this condition (an acceptable node also satisfying this condition is defined (i, δ_L)-*valuable* or *valuable*). Hence, our algorithm does not only discard all quadruples $(s_i, t_i, u.v)$ where either u or v is not acceptable, but also all quadruples where both u and v are not valuable.

In order to apply these conditions, when analyzing a pair (s_i, t_i), hyp computes the set of acceptable and valuable nodes in time $\mathcal{O}(n)$ (actually, several nodes are skipped, thanks to implementation tricks, so that the time might be even smaller). Then, for each valuable node u, it analyzes pairs (u, v) preceding (s_i, t_i) such that u is acceptable. For this latter loop, we record for each node u the list `mate`$[u]$ of previously seen pairs (u, v), and then test each time if v is acceptable. The pseudo-code for hyp is provided by Algorithm 18.

Lemma 7.4. *The algorithm is correct.*

Proof. First of all, $\delta_L \leq \delta(G)$ during the whole algorithm, so we only have to rule out the possibility that the output is strictly smaller than $\delta(G)$. Let s, t, u, v be a quadruple such that $\tau(s, t; u, v) = \delta(G)$. We may assume without loss of generality that $\{s, t\}$ and $\{u, v\}$ are far-apart (otherwise, we change the pairs using Lemma 7.2), and that $\{u, v\}$ is before $\{s, t\}$ in the ordering of pairs (otherwise, we swap the pairs). By Lemma 7.1, $\mathrm{dist}(s, t) \geq 2\delta(G) \geq 2\delta_L$ at any step of the algorithm: if $2\delta_L = \mathrm{dist}(s, t) \geq 2\delta(G)$ at some step, the algorithm is correct because δ_L never decreases. Otherwise, the pair $\{s, t\}$ is analyzed at some step i, u and v are (i, δ_L)-acceptable, and either u or v is be (i, δ_L)-valuable (by definition of acceptable and valuable). Hence, in the inner loop, $\tau(s, t; u, v)$ is computed, and afterwards $\delta_L = \tau(s, t; u, v) = \delta(G)$. □

It remains to define how we compute an approximation of the set of acceptable and valuable nodes.

Algorithm 18: The new algorithm, HYP.

```
Let P = $(\{s_1,t_1\},\dots,\{s_N,t_N\})$ be the ordered list of far apart pairs.
$\delta_L \leftarrow 0$;
mate[$u$]$\leftarrow \emptyset$ for each $u$;
for $i \in [1,N]$ do
    if $\text{dist}(s_i,t_i) \leq 2\delta_L$ then
        return $\delta_L$;
    end
    (acceptable, valuable) $\leftarrow$ computeAccVal ();
    for $u \in$ valuable do
        for $v \in$ mate[$v$] do
            if $v \in$ acceptable then
                $\delta_L \leftarrow \max(\delta_L, \tau(s_i,t_i;u,v))$;
            end
        end
    end
    add $t_i$ to mate[$s_i$];
    add $s_i$ to mate[$t_i$];
end
return $\delta_L$
```

7.2.2 Acceptable and Valuable Nodes

First of all, let us fix i and δ_L, since in this section they play the role of parameters. Moreover, for the sake of clarity, we denote s_i and t_i simply by s and t. The following lemmas provide conditions implying that u is skippable, that is, there is no pair $\{u,v\}$ appearing in P before $\{s,t\}$ such that $\tau(s,t;u,v) > \delta_L$. An acceptable node must not satisfy these conditions. The first lemma holds by definition of skippable.

Lemma 7.5. *If u does not belong to any far-apart pair $\{u,v\}$ before $\{s,t\}$ in P, then u is skippable.*

A second possibility to prove that a node is skippable is given by a simple corollary of the following lemma.

Lemma 7.6 ([58])**.** *For each quadruple of nodes (s,t,u,v),*

$$\tau(s,t;u,v) \leq \min_{w,w' \in \{s,t,u,v\}, w \neq w'} \text{dist}(w,w').$$

Proof. Let us assume that $\text{dist}(s,t)+\text{dist}(u,v)$ is the biggest sum (otherwise, $\tau(s,t;u,v) \leq 0$, and the result follows directly). We already know by Lemma 7.1 that $2\tau(s,t;u,v) \leq \text{dist}(s,t)$ and $2\tau(s,t;u,v) \leq \text{dist}(u,v)$. Let us prove that $\tau(s,t;u,v) \leq \text{dist}(s,u)$:

$$\begin{aligned} 2\tau(s,t;u,v) &= \text{dist}(s,t)+\text{dist}(u,v)-\max\{\text{dist}(s,u)+\text{dist}(t,v), \text{dist}(s,v)+\text{dist}(t,u)\} \\ &\leq \text{dist}(s,t)+\text{dist}(u,v)-\text{dist}(s,v)-\text{dist}(t,u) \\ &\leq \text{dist}(s,u)+\text{dist}(u,t)+\text{dist}(u,s)+\text{dist}(s,v)-\text{dist}(s,v)-\text{dist}(t,u) \\ &\leq 2\,\text{dist}(s,u). \end{aligned}$$

The proof of the remaining inequalities is very similar: it is enough to swap the roles of the nodes s,t,u,v. □

Corollary 7.7. *If $\text{dist}(s,u) \leq \delta_L$ or $\text{dist}(t,u) \leq \delta_L$, then u is skippable.*

Proof. If the assumptions are satisfied, for each v, either $\tau(s,t;u,v) \leq \text{dist}(s,u) \leq \delta_L$, or $\tau(s,t;u,v) \leq \text{dist}(t,u) \leq \delta_L$. □

The next lemmas make use of the notion of the eccentricity of a node u, defined as $\operatorname{ecc}(u) = \max_{v \in V} \operatorname{dist}(u, v)$.

Lemma 7.8. *If* $2\operatorname{ecc}(u) - \operatorname{dist}(s, u) - \operatorname{dist}(t, u) < 4\delta_L + 2 - \operatorname{dist}(s, t)$*, then* u *is skippable.*

Proof. By contradiction, let us suppose that there exists a node v such that $\delta_L < \tau(s, t; u, v)$. Then,

$$\begin{aligned}\delta_L + 1 &\leq 2\tau(s, t; u, v)\\ &= \operatorname{dist}(s, t) + \operatorname{dist}(u, v) - \max(\operatorname{dist}(s, u) + \operatorname{dist}(t, v), \operatorname{dist}(s, v) + \operatorname{dist}(t, u))\\ &\leq \operatorname{dist}(s, t) + \operatorname{dist}(u, v) - \frac{1}{2}(\operatorname{dist}(s, u) + \operatorname{dist}(t, v) + \operatorname{dist}(s, v) + \operatorname{dist}(t, u))\\ &\leq \operatorname{dist}(s, t) + \operatorname{ecc}(u) - \frac{1}{2}(\operatorname{dist}(s, u) + \operatorname{dist}(t, u)) - \frac{1}{2}\operatorname{dist}(s, t).\end{aligned}$$

By rearranging this inequality, we would contradict the hypothesis. □

Lemma 7.9. *If* $\operatorname{ecc}(u) + \operatorname{dist}(s, t) - 3\delta_L - \frac{3}{2} < \max\{\operatorname{dist}(s, u), \operatorname{dist}(t, u)\}$*, then* u *is skippable.*

Proof. By contradiction, let us suppose that there exists a node v such that $\delta_L < \tau(s, t; u, v)$. By Corollary 7.7, $\operatorname{dist}(t, v) > \delta_L$, that is, $\operatorname{dist}(t, v) \geq \delta_L + \frac{1}{2}$. Consequently,

$$\begin{aligned}2\delta_L + 1 &\leq 2\tau(s, t; u, v)\\ &= \operatorname{dist}(s, t) + \operatorname{dist}(u, v) - \max(\operatorname{dist}(s, u) + \operatorname{dist}(t, v), \operatorname{dist}(s, v) + \operatorname{dist}(t, u))\\ &\leq \operatorname{dist}(s, t) + \operatorname{dist}(u, v) - \operatorname{dist}(s, u) - \operatorname{dist}(t, v)\\ &\leq \operatorname{dist}(s, t) + \operatorname{ecc}(u) - \operatorname{dist}(s, u) - \delta_L - 1/2.\end{aligned}$$

By exchanging the roles of s and t, we obtain

$$2\delta_L + 1 \leq \operatorname{dist}(s, t) + \operatorname{ecc}(u) - \operatorname{dist}(t, u) - \delta_L - \frac{1}{2}.$$

These two inequalities contradict the hypothesis. □

Definition 7.10. A node is acceptable if it does not satisfy the assumptions of Lemmas 7.5, 7.8 and 7.9 and Corollary 7.7.

Remark 7.11. Lemma 7.5 can be verified “on the fly”, by keeping track of already-seen nodes. The other items are clearly verifiable in time $\mathcal{O}(1)$ for each node, and consequently the running time of this operation is $\mathcal{O}\left(|\{u \in V : \exists\{u, v\} < \{s, t\}\}|\right)$, which is less than or equal to $\mathcal{O}(n)$.

Remark 7.12. A variation of hyp verifies on the fly Lemma 7.9 and not Lemma 7.5. At the beginning of the algorithm, for each node s, we pre-compute a list of all nodes u in decreasing order of $\operatorname{ecc}(u) - \operatorname{dist}(s, u)$ (in time $\mathcal{O}(n^2 \log n)$). Then, when computing acceptable nodes, we scan the list corresponding to s, and we stop as soon as we find a node u such that $\operatorname{ecc}(u) + \operatorname{dist}(s, t) - 3\delta_L - \frac{3}{2} < \operatorname{dist}(s, u)$. In this case, the running time of this operation is $\mathcal{O}\left(|\{u \in V : \operatorname{ecc}(u) + \operatorname{dist}(s, t) - 3\delta_L - \frac{3}{2} \geq \operatorname{dist}(s, u)\}|\right)$. Since we may swap the roles of s and t, at each step, our algorithm chooses between s and t the less central node, according to *closeness centrality* measure [17].

The two remarks above correspond to two versions of our algorithm hyp, that we call hyp1 and hyp2, respectively. Now we need to define valuable nodes, using the following lemma, which involves a given node c (formally, we would need to write c-valuable instead of valuable). All choices of c are feasible, but if c is “central”, the running time improves. We decided to set c as the most central node according to *closeness centrality* measure [17].

Lemma 7.13. *Let* c *be any fixed node, and, for any node* w*, let* $f_c(w) := \frac{1}{2}(\operatorname{dist}(s, t) - \operatorname{dist}(s, c) - \operatorname{dist}(w, t)) + \operatorname{dist}(w, c)$*. Then, for any two nodes* u *and* v*, we have* $2\tau(s, t; u, v) \leq f_c(u) + f_c(v)$*.*

Proof. We have that

$$\begin{aligned}2\tau(s,t;u,v) &= \mathrm{dist}(s,t)+\mathrm{dist}(u,v)-\max(\\&\quad \mathrm{dist}(s,u)+\mathrm{dist}(t,v),\mathrm{dist}(s,v)+\mathrm{dist}(t,u))\\&\leq \mathrm{dist}(s,t)+\mathrm{dist}(u,c)+\mathrm{dist}(c,v)-\frac{1}{2}(\\&\quad \mathrm{dist}(s,u)+\mathrm{dist}(t,v)+\mathrm{dist}(s,v)+\mathrm{dist}(t,u))\\&= f_c(v)+f_c(w).\end{aligned}$$

The lemma is thus proved. □

As a consequence, if $2\tau(s,t;u,v) > 2\delta_L$, either $f_c(u) > \delta_L$ or $f_c(v) > \delta_L$. This justifies the following definition.

Definition 7.14. An acceptable node u is c-valuable or valuable if $f_c(u) > \delta_L$.

We conclude that, if $\tau(s,t;u,v) > \delta_L$, then at least one of u and v must be valuable.

Remark 7.15. It is possible to compute if an acceptable node is valuable in time $\mathcal{O}(1)$, so there is no time overhead for the computation of valuable nodes.

7.3 Experimental Results

In this section, we compare the best algorithm available until now [58] (CCL, whose pseudocode is Algorithm 17), with the two versions of our new algorithm, denoted as HYP1 and HYP2 (using Remark 7.11 and Remark 7.12, respectively). Other available algorithms are the trivial algorithm, which is significantly outperformed by CCL in [58], and the algorithm in [77]. The latter is not practical, because it is based on fast matrix multiplication: indeed, using $\mathcal{O}(n^3)$ matrix multiplication implementation, we get the same running time of the trivial algorithm. As far as we know, no other competitors are available.

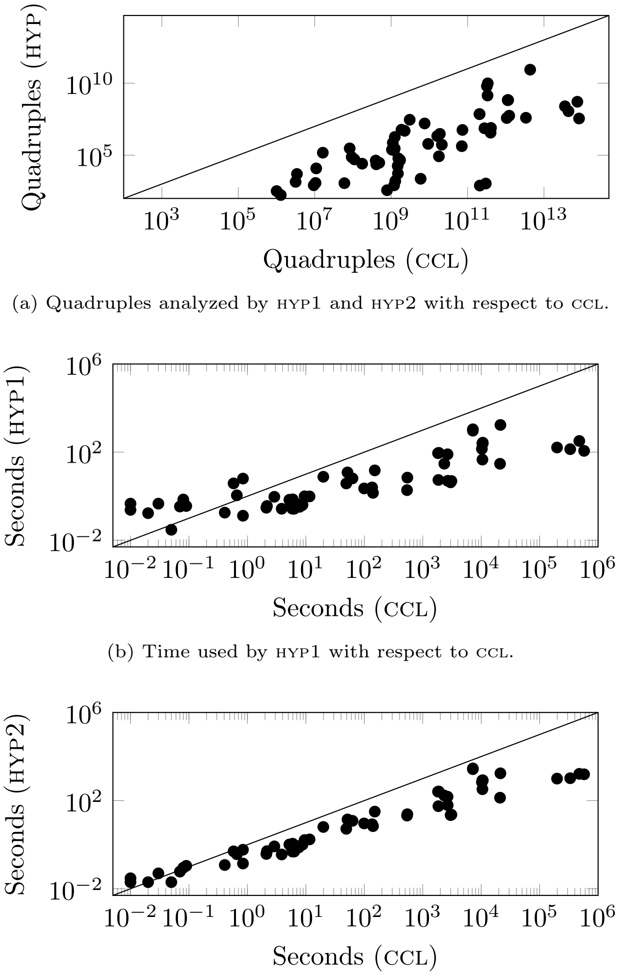
Both CCL and our algorithm, in both versions HYP1 and HYP2, share the following preprocessing (see [58]):

- compute biconnected components to treat them separately;
- computing the distances between all pairs of nodes;
- computing and sorting the list `P` of all far-apart pairs.

All the operations above need time $\mathcal{O}(mn)$ and they are ignored in the comparison since they are common to all three algorithms. Our tests were performed on an AMD Opteron(TM) Processor 6276, running Fedora release 21. Our source code has been written in C and compiled with gcc 4.9.2 with optimization level 3. The code is available at `piluc.dsi.unifi.it/lasagne`, and another implementation is available in the Sagemath library [64].

We have collected a dataset composed by 62 graphs (available with the code) of different kinds: social, peer-to-peer, autonomous systems, citation networks, and so on. The networks were selected from the well-known SNAP dataset (`http://snap.stanford.edu/`), and from CAIDA (`http://www.caida.org`). The number of nodes varies between 4 039 and 265 009 (1 396 and 50 219 after the preprocessing).

Number of quadruples. The first comparison analyzes how many quadruples are processed before the hyperbolicity is computed - note that HYP1 and HYP2 analyze the same number of quadruples, since the only difference between them is how acceptable and valuable nodes are computed. The results are summarized in Figure 7.2a, which plots the number of quadruples processed by the new algorithms with respect to CCL. More precisely, for each graph G, we draw a point in position (x, y) if CCL analyzed x quadruples and both HYP1 and HYP2



(a) Quadruples analyzed by hyp1 and hyp2 with respect to ccl.

(b) Time used by hyp1 with respect to ccl.

(c) Time used by hyp2 with respect to ccl.

Figure 7.2. Comparisons of quadruples analyzed and running time of hyp1, hyp2, and ccl. The line $y = x$ separates the region where ccl is better (above) from the region where hyp1 and hyp2 are better (below).

analyzed y quadruples to compute the hyperbolicity of G. More detailed results are available in Table 7.1. The results show that the new algorithm analyzes a much smaller number of quadruples, ranging from one hundred to few millions, drastically outperforming ccl, which often analyzes millions of millions of quadruples, and even billions of millions. Of course, the new algorithm is never outperformed, because the quadruples analyzed by hyp1 and hyp2 are a subset of the quadruples analyzed by ccl by definition.

Running time. Since the computation of acceptable and valuable nodes has a non-negligible impact on the total running time, for a more fair comparison, we have also considered the running time of the algorithms. In Figures 7.2b and 7.2c we report the time used by hyp1 and hyp2 with respect to the time used by ccl (again, more detailed results are available in Table 7.1). Also in this experiment, both hyp1 and hyp2 drastically outperform ccl: the running time is lower in most of the graphs, and the only cases where ccl is faster need a very small amount of time (a few seconds at most). On the other hand, the new algorithms are much faster when the total time is big: for instance, on input `as-20120601.caida`, ccl needs at least one week (this lower bound was computed from the actual hyperbolicity and all the distances between the nodes), while hyp1 is 367 times faster, needing less than half an hour, and hyp2 is more than 5 000 times faster, needing less than two minutes. Similar

Table 7.1. Summary of the results of our experiments. For the graphs marked with $*$, CCL did not finish after two hours. The number of quadruples reported is the lower bound $\frac{k(k-1)}{2}$, where k is the number of pairs (s,t) such that $\mathrm{dist}(s,t) > 2\delta$; the time reported is $\frac{T}{1\,000\,000\,000}\frac{k(k-1)}{2}$ where T is the time needed to process 1 000 000 000 quadruples (in two cases, we reported > 7200, because the lower bound was not tight, and the estimate was smaller than 2 hours). We also report our improvement with respect to CCL, that is, the ratio between the quadruples (resp. time) of CCL and the quadruples (resp. time) of our algorithms.

				Quadruples Analyzed		Quadruple	Time (seconds)			Time Improv.	
Graph	Nodes	Edges	δ	CCL	HYP1-HYP2	Improv.	CCL	HYP1	HYP2	HYP1	HYP2
ca-GrQc	1396	10382	3.5	10953540	12180	899.31	0.05	0.02	0.03	2.51	1.53
com-amazon.all.cmty	2617	7682	8	82272378	298472	275.65	0.41	0.12	0.18	3.38	2.33
as20000102	2680	14468	2.5	174555270	25868	6747.92	0.84	0.14	0.13	5.96	6.28
facebook-combined*	3421	169874	1.5	4349543592331	86614009902	50.22	21314.60	1733.66	1713.63	12.29	12.44
ca-HepTh	3702	28784	4	3428271	4990	687.03	0.02	0.02	0.17	0.80	0.11
oregon1-010331	4218	24972	2.5	1284966165	1617	794660.58	6.32	0.49	0.27	12.93	23.11
oregon1-010407	4279	25086	2.5	1195433856	810	1475844.27	5.89	0.49	0.27	12.00	21.62
oregon1-010414	4290	25712	2.5	1049667471	243033	4319.03	5.22	1.02	0.69	5.09	7.56
oregon1-010421	4336	26202	2.5	780855921	373	2093447.51	3.86	0.36	0.27	10.86	14.49
oregon1-010428	4392	25772	2.5	1192062378	220894	5396.54	5.96	1.11	0.72	5.35	8.31
oregon1-010512	4399	25756	2.5	1240194306	289910	4277.86	6.21	0.69	0.33	8.96	18.77
oregon1-010505	4435	25928	2.5	1504562940	5362	280597.34	7.54	0.75	0.35	10.04	21.70
oregon1-010519	4442	25886	2.5	1539653286	60315	25526.87	7.81	0.88	0.33	8.85	23.87
oregon1-010526	4487	26982	3	407982029	23417	17422.47	2.11	0.38	0.30	5.57	7.04
p2p-Gnutella08	4527	37992	3	1007490	324	3109.54	0.01	0.02	0.24	0.31	0.03
wiki-Vote	4583	196064	2	16293337	149456	109.02	0.09	0.11	0.36	0.81	0.25
oregon2-010331	4667	42910	2	400262386560	3663772	109248.72	2995.00	22.04	4.26	135.88	703.60
oregon2-010407	4729	42376	2	410741958903	7857070	52276.73	3092.18	23.02	4.90	134.30	630.55
oregon2-010414	4766	44138	2.5	1245329371	1845137	674.93	6.29	0.93	0.55	6.77	11.39
oregon2-010421	4811	43600	2.5	1105698825	733049	1508.36	5.61	0.78	0.46	7.15	12.24
oregon2-010428	4870	43452	2.5	1865841328	5878928	317.38	9.52	1.63	0.99	5.83	9.61
oregon2-010505	4895	42412	2.5	1687311186	42131	40049.16	8.64	1.00	0.39	8.63	22.05
ca-HepPh	4950	128620	3	394622371	42674	9247.37	2.16	0.51	0.36	4.26	5.93
oregon2-010512	4955	42980	2.5	1710978753	46489	36803.95	8.77	1.01	0.41	8.68	21.29
oregon2-010519	5010	44726	2.5	1482537378	18041	82176.01	7.53	0.86	0.40	8.72	18.66
oregon2-010526	5030	45338	2.5	2258222410	4934010	457.69	11.57	1.74	0.98	6.66	11.82
p2p-Gnutella09	5590	46956	3	9419970	821	11473.78	0.07	0.06	0.34	1.04	0.19
p2p-Gnutella06	6708	58994	3	1292028	172	7511.79	0.01	0.03	0.46	0.31	0.02
p2p-Gnutella05	6813	59578	3	3163870	1411	2242.29	0.03	0.05	0.46	0.57	0.06
as-20040105.caida	7332	41046	2.5	21234792321	540599	39280.12	140.82	6.95	1.42	20.25	99.21
as-20040607.caida*	7776	44582	2	3370014286246	39712688	84859.89	21005.11	135.43	29.59	155.09	709.97
p2p-Gnutella04	8362	74926	3	10513405	1190	8834.79	0.08	0.09	0.71	0.93	0.11
ca-CondMat	8773	100384	3.5	488609430	29851	16368.28	2.89	0.84	0.94	3.44	3.07
as-20050905.caida	8895	50020	3	7525743270	15991110	470.62	49.17	5.26	3.79	9.34	12.98
ca-AstroPh	9577	286950	3	109423150	51097	2141.48	0.66	0.37	1.10	1.79	0.60
ASEdges10-2011	9710	95858	2	18861383976	3042167	6199.98	136.15	8.47	2.51	16.07	54.28
ASEdges4-2012	9795	98222	2	70723160371	424079	166768.83	538.81	20.96	1.89	25.70	284.87
ASEdges12-2010	10708	128922	2	3060396657	28895395	105.91	19.91	6.37	7.62	3.12	2.61
email-Enron	10969	262726	2.5	16069135356	2148540	7479.10	99.27	9.23	2.25	10.76	44.08
as-caida20071105	11935	68132	2.5	73495671315	5671011	12959.89	545.03	24.16	6.98	22.55	78.08
p2p-Gnutella25	13301	90562	3	300322187578	1081	277818859.92	2672.52	62.40	5.00	42.83	534.51
p2p-Gnutella24	15455	108504	3	209384412126	781	268097838.83	1852.04	55.95	5.46	33.10	339.33
as-20100120.caida*	15549	101900	2	43782839513031	113921805	384323.61	333184.15	1038.52	136.42	320.83	2442.29
as-20110116.caida*	17030	152576	2	35387143423885	248996963	142118.78	198417.40	997.34	162.05	198.95	1224.44
as-20120101.caida*	18543	186746	2	74744549775451	513362824	145597.90	476614.17	1641.20	321.31	290.41	1483.33
as-20120601.caida*	19068	181882	2	82852372357930	35436735	2338036.29	576275.86	1567.35	114.38	367.68	5038.13
as-20130101.caida	20045	216338	2.5	333864201231	1399456972	238.57	2656.53	150.39	80.02	17.66	33.20
p2p-Gnutella30	20095	143304	3.5	59825391	1147	52158.14	0.58	0.50	3.82	1.16	0.15
as-20130601.caida*	20908	236708	2.5	320649529266	6249608655	51.31	> 7200.00	2678.17	940.84	0.77	2.20
as-20131101.caida*	21476	250960	2.5	338186916990	9772268809	34.61	> 7200.00	2889.48	1087.69	0.67	1.77
email-EuAll	22153	240734	3	9309369475	610516	15248.36	62.77	12.09	6.44	5.19	9.74
cit-HepTh	24982	696474	4	92854378	74784	1241.63	0.84	0.59	6.30	1.41	0.13
loc-brightkite-edges*	29136	358092	3	1232144653915	55546114	22182.37	10496.85	329.02	45.93	31.90	228.55
soc-Epinions1	30935	706214	2.5	276357092628	7425184	37218.89	2336.56	170.16	29.79	13.73	78.44
cit-HepPh	32233	834586	3.5	17804316753	83828	212391.05	151.08	31.75	14.86	4.76	10.17
p2p-Gnutella31	33645	237584	3.5	5916272253	2318	2552317.62	51.64	13.98	12.02	3.69	4.30
sign-Slashdot081106	45414	868420	2.5	205817260078	71550147	2876.55	1883.35	259.44	92.76	7.26	20.30
soc-Slashdot0811	45425	869602	2.5	205827525606	71589472	2875.11	1824.68	255.67	89.57	7.14	20.37
sign-Slashdot090216*	49148	924130	2.5	1127364491100	666496152	1691.48	10638.72	829.71	268.45	12.82	39.63
sign-Slashdot090221*	49313	929482	2.5	1126754933865	673410974	1673.21	10629.82	830.22	250.08	12.80	42.51
soc-Slashdot0902*	49384	937030	2.5	1156307585356	672901135	1718.39	10464.75	831.93	249.52	12.58	41.94
soc-sign-epinions*	50219	1246832	2.5	1060615290790	38110649	27829.89	10277.35	687.49	140.81	14.95	72.99

results hold in all graphs where the total running time is large. This does not only mean that we have improved upon CCL, but also that the improvement is concentrated on inputs where the running time is high. Furthermore, we observe that on all graphs the total running time of HYP2 is less than half an hour: this means that, even if the worst-case complexity of this algorithm is $\mathcal{O}(n^4)$, in practice, the time used by the second part is comparable to the preprocessing time, which is $\mathcal{O}(mn)$. Hence, from a practical point of view, since real-world graphs are usually sparse, the algorithm may be considered quadratic.

7.4 Hyperbolicity of Real-World Graphs

In this section, we apply the new algorithm to the analysis of the hyperbolicity of real-world graphs. First, we study the standard definition of hyperbolicity, that is, the maximum of $\delta(s,t,u,v)$ where s,t,u,v range over all nodes in the graph. Since this analysis deals with many more graphs than previous ones, our results are able to outline that real-world graphs are usually not hyperbolic, because $\delta \approx \frac{D}{4}$, and by analytical results we already know that $0 \leq \delta \leq \frac{D}{2}$. Furthermore, we show that the hyperbolicity does not capture specific properties of graphs of different kinds: indeed, the ratio $\frac{2\delta}{D}$ follows approximately a normal distribution, with average value $\frac{1}{2}$, even if we restrict our attention to specific classes of graphs. This suggests that the value of δ is highly influenced by "random events".

As an alternative, we propose to use the *average hyperbolicity*, that is simply defined as $\delta_{\text{avg}} = \text{avg}_{s,t,u,v \in V} \delta(s,t,u,v)$ (in other words, we have replaced the maximum with an average). We show that, using this definition, real-world graphs are usually hyperbolic, because the ratio $\frac{\delta_{\text{avg}}}{2\,\text{dist}_{\text{avg}}}$ is usually quite small, where dist_{avg} is the average distance between two nodes. Building on this definition, we consider an interpretation of hyperbolicity as "lack of democracy": if a graph is hyperbolic, it means that few nodes are on many shortest paths, and consequently the graph is aristocratic; conversely, if a graph is not hyperbolic, then shortest paths are "spread around the graph", and consequently the graph is democratic. As far as we know, this is the first measure of democracy in a complex network, apart from assortativity [123, 124]. In any case, our measure is quite different from the latter one, because it is based on shortest paths and not on neighbors: consequently, the new measure is global. Moreover, it is more robust: for instance, if we "break" all edges by "adding a node in the middle", the hyperbolicity of the graph does not change much, but the assortativity decreases drastically. Our experiments show that the value of δ_{avg} is much more robust than the standard hyperbolicity with respect to random events, allowing us to effectively distinguish networks of different kind. Our classification is different from the classification provided by assortativity [123, 124]: for instance, a network with few influential hubs not connected to each other is democratic if we consider assortativity, while it is aristocratic in our framework.

Finally, we introduce for the first time the average hyperbolicity of neighborhoods of a given node, which measures the "importance" of a node (the k-neighborhood of a node s is the subgraph induced by the k nodes closest to s). Applications include the classification of complex networks (hyperbolic networks have interesting properties, as outlined in the literature), the analysis of nodes in a given network, and possibly the detection of communities using hyperbolicity.

7.4.1 Using the Classical Definition

Our first experiment computes the ratio $\frac{2\delta}{D}$ in all the networks in our dataset (Figure 7.3).

These results show that the distribution of the ratio $\frac{2\delta}{D}$ is approximately Gaussian, both in the whole dataset and in each single kind of network. The average ratio is 0.521, and the standard deviation is 0.085. Moreover, a Chi-square goodness of fit test applied to the previous data does not reject the hypothesis that the distribution is Gaussian with mean 0.5 and variance 0.085, with a very high confidence level [139]. This result confirms that the value of δ in real-world networks is not much "smaller than expected". In other words, real-world networks are not hyperbolic, at least in the Gromov sense: this is the first main result of our analysis. However, we are able to perform a further step: the Gaussian probability distribution makes us think that δ is influenced by random events. Indeed it does not reflect particular characteristics of the network, since the same distribution arises from networks of different kinds.

Social networks show a slightly different behavior, since many of them have a larger value of $\frac{2\delta}{D}$, between 0.65 and 0.75. However, this is due to the presence of several financial networks (e-MID, a platform for interbank lending), where the ratio is often $\frac{2}{3}$ or $\frac{3}{4}$ since the diameter is 3 or 4.

Despite this particular case, we may conclude that the ratio $\frac{2\delta}{D}$ is not a characteristic of the network, but it mainly depends on "random events" that have a deep impact on this value. This conclusion is further confirmed by the particular case of the e-MID networks: this parameter changed from 0.750 in 2011 to 0.286 in 2012, only because a simple path of length

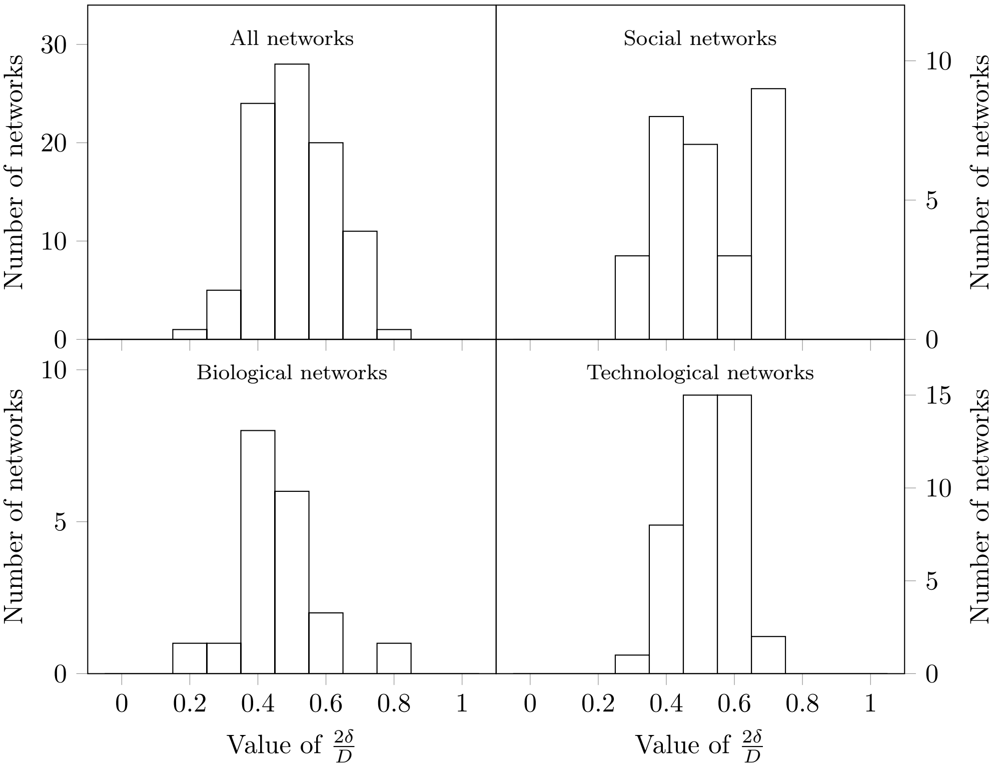


Figure 7.3. The distribution of $\frac{2\delta}{D}$ in the graphs in our dataset. The bar corresponding to the value p contains all networks where $p - 0.5 < \frac{2\delta}{D} \leq p + 0.5$.

3 increased the diameter from 4 to 7.

7.4.2 Average Hyperbolicity and Democracy

In the past, the average hyperbolicity δ_{avg} of a quadruple of nodes was rarely analyzed: the only known result is that it is usually significantly smaller than δ [9]. In order to fill this gap, we have computed the ratio $\frac{2\delta_{\text{avg}}}{\text{dist}_{\text{avg}}}$, where dist_{avg} denotes the average distance in the network (also this quantity lies in the interval $[0, 1]$ by Lemma 7.1).

Although the exact computation of δ_{avg} is quite hard, the value $\frac{2\delta_{\text{avg}}}{\text{dist}_{\text{avg}}}$ can be easily approximated through sampling, as shown by the following lemma.

Lemma 7.16. *Let G be a graph with hyperbolicity $\delta = 2.5$, and let us sample the hyperbolicity of $N = 10\,000\,000$ quadruples of nodes, obtaining $\delta_1, \ldots, \delta_N$. Let $h_i := \frac{2\delta_i}{\text{dist}_{\text{avg}}}$, and let $t = 0.01$ be the tolerance. Then,*

$$\mathbb{P}\left(\left|\frac{\sum_{i=1}^{N} h_i}{N} - \frac{2\delta_{avg}}{\text{dist}_{\text{avg}}}\right| \geq t\right) \leq 2e^{-\frac{Nt^2}{2\delta^2}} = 0.07\%.$$

Proof. By Hoeffding inequality (see Lemma 6.1 in Chapter 6) applied with $a_i = 0 \leq h_i \leq \frac{2\delta}{\text{dist}_{\text{avg}}} = b_i$, we obtain:

$$\mathbb{P}\left(\left|\frac{\sum_{i=1}^{N} h_i}{N} - \frac{2\delta_{\text{avg}}}{\text{dist}_{\text{avg}}}\right| \geq t\right) \leq 2e^{\frac{-2N^2t^2}{\sum_{i=1}^{N}(b_i - a_i)^2}} \leq$$

$$\leq 2e^{-\frac{Nt^2 \operatorname{dist}_{\text{avg}}^2}{2\delta^2}}$$
$$\leq 2e^{-\frac{Nt^2}{2\delta^2}}$$

because $\operatorname{dist}_{\text{avg}} \geq 1$. The previous inequality applied with $N = 10\,000\,000$, $t = 0.01$, and $\delta = 2.5$ yields:

$$\mathbb{P}\left(\left|\frac{\sum_{i=1}^{10\,000\,000} \delta_i}{10\,000\,000} - \delta_{\text{avg}}\right| \geq 0.01\right) \leq 0.07\%.$$

□

Detailed results containing the average hyperbolicity of each single network are plotted in Figure 7.4.

From the figure, we outline that the average hyperbolicity is usually an order of magnitude smaller than the average distance: in this sense, real-world networks are hyperbolic. This result suggests that this quantity can be meaningful in the analysis and classification of complex networks. Following this line, we propose an interpretation of the average hyperbolicity in terms of democracy, through the following two lemmas. The first one shows that, if for some nodes s, t, $\max_{u,v} \delta(s, t, u, v)$ is not high, then there is a set of small diameter that "controls" all approximately shortest paths from u to v. Consequently, a hyperbolic network is not democratic, because δ is small, and shortest paths are controlled by small sets.

Lemma 7.17 ([56], Lemma 2)**.** *Let s, t be two nodes in a network $G = (V, E)$, let $\boldsymbol{N}^{\ell}(s)$ be the ℓ-neighborhood of s (that is, the set $\{v \in V : \operatorname{dist}(s, v) \leq \ell\}$), and let $\boldsymbol{N}^{\ell'}(t)$ be the ℓ'-neighborhood of t. Then, the diameter of the set $X = \boldsymbol{N}^{\ell}(s) \cap \boldsymbol{N}^{\ell'}(t)$ is at most $2\max_{u,v} \delta(s, t, u, v) + \ell + \ell' - \operatorname{dist}(s, t)$.*

Proof. Let u, v be two nodes in $\boldsymbol{N}^{\ell}(s) \cap \boldsymbol{N}^{\ell'}(t)$: $2\delta(s, t, u, v) \geq \operatorname{dist}(s, t) + \operatorname{dist}(u, v) - \max(\operatorname{dist}(s, u) + \operatorname{dist}(t, v), \operatorname{dist}(s, v) + \operatorname{dist}(t, u)) \geq \operatorname{dist}(s, t) + \operatorname{dist}(u, v) - \ell - \ell'$, and consequently $\operatorname{dist}(u, v) \leq 2\delta(s, t, u, v) + \ell + \ell' - \operatorname{dist}(s, t) \leq 2\max_{u,v \in V} \delta(s, t, u, v) + \ell + \ell' - \operatorname{dist}(s, t)$. □

The second lemma is a sort of converse: if there is a set of nodes controlling the shortest paths of a given quadruple (s, t, u, v), $\delta(s, t, u, v)$ is low. Consequently, if δ is high, then there is not a small set of nodes controlling many shortest paths, and the network is democratic.

Lemma 7.18 ([55], Lemma 2)**.** *Let s, t, u, v be a quadruple of nodes, and let us assume that there exists a set $V' \subseteq V$ of diameter Δ such that all shortest paths between s, t, u, v pass through V'. Then, $\delta(s, t, u, v) \leq \Delta$.*

Proof. We can assume without loss of generality that

$$2\delta(s, t, u, v) = \operatorname{dist}(s, t) + \operatorname{dist}(u, v) - \operatorname{dist}(s, u) - \operatorname{dist}(t, v). \tag{7.1}$$

Since all the shortest paths pass through V', we can say that $\operatorname{dist}(s, V') + \operatorname{dist}(V', t) \leq \operatorname{dist}(s, t) = \operatorname{dist}(s, V') + \Delta + \operatorname{dist}(V', t)$, where $\operatorname{dist}(s, V') = \min_{w \in V'} \operatorname{dist}(s, w)$. Plugging this inequality into Equation (7.1) (and all similar inequalities), we obtain that

$$\begin{aligned} 2\delta(s, t, u, v) &= \operatorname{dist}(s, t) + \operatorname{dist}(u, v) - \operatorname{dist}(s, u) - \operatorname{dist}(t, v) \\ &\leq \operatorname{dist}(s, V') + \Delta + \operatorname{dist}(V', t) + \operatorname{dist}(u, V') + \Delta + \operatorname{dist}(V', v) \\ &\quad - \operatorname{dist}(s, V') - \operatorname{dist}(V', u) - \operatorname{dist}(t, V') - \operatorname{dist}(V', v) \\ &\leq 2\Delta. \end{aligned}$$

□

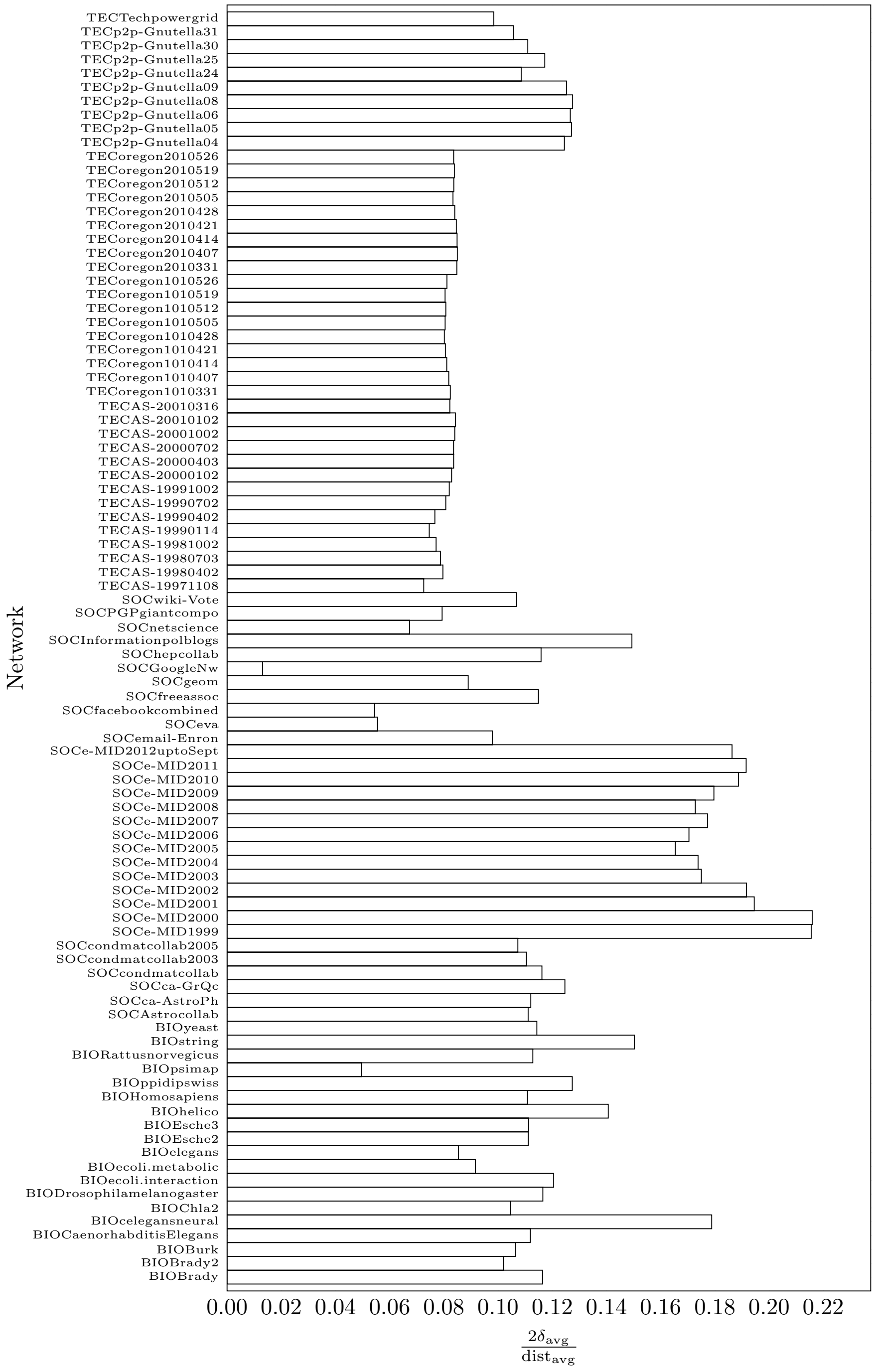


Figure 7.4. The value $\frac{\delta_{\text{avg}}}{\text{dist}_{\text{avg}}}$ in all the networks in our dataset.

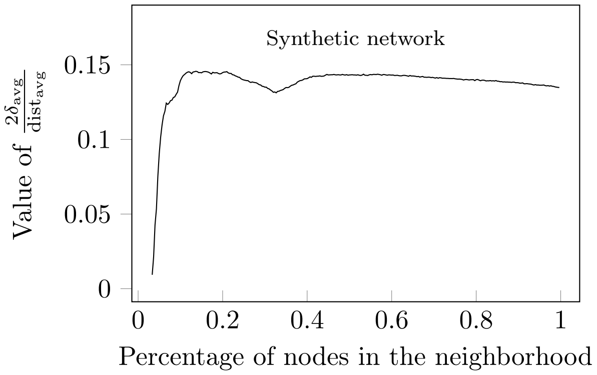


Figure 7.5. The value of $\frac{2\delta_{\text{avg}}}{\text{dist}_{\text{avg}}}$ for neighborhoods of the maximum degree node in a synthetic network generated with the techniques in [104].

7.4.3 Hyperbolicity of Neighborhoods

As an application of the results in the previous subsection, we consider the average hyperbolicity of k-neighborhoods of a given node s, where the k-neighborhood of a node s is the subgraph induced by the k nodes closest to s (in case of tie, we use a random tie-break). The intuitive idea is that, since the hyperbolicity of a graph is closely related to the existence of a small part of the graph controlling most shortest paths, neighborhoods of a given node s should be "less democratic" than the whole graph, in the sense that they are contained in the "influence area" of s, and most shortest paths should pass through s.

To highlight this quantity, we let k range from the degree of s to the number n of nodes in the graph, with steps of 10 nodes, and we measure $\frac{2\delta_{\text{avg}}}{\text{dist}_{\text{avg}}}$ in all these k-neighborhoods of s. In order to prove the effectiveness of this approach, we first test a synthetic power-law graph [104] made by three communities of 1 000 nodes each (see Figure 7.5). We have computed the hyperbolicity of neighborhoods of the node s with highest degree: we can see a local minimum close to the size of a community. In our opinion, this minimum appears because the neighborhood is "dominated by the community", and consequently by the center s of the community. This result confirms the link between the value of $\frac{2\delta_{\text{avg}}}{\text{dist}_{\text{avg}}}$ and the influence area of a node.

Then, we analyzed neighborhoods in real-world networks. The upper plots in Figure 7.6 show the same results for one network of each kind:

- a social network, the General Relativity and Quantum Cosmology collaboration network;
- a biological network, the yeast metabolic network;
- a technological network, the peer-to-peer Gnutella network in 2004.

As a benchmark of comparison, we have also considered the hyperbolicity of neighborhoods of a random node.

The plots show that the value of $\frac{\delta_{\text{avg}}}{\text{dist}_{\text{avg}}}$ in increasing-size neighborhoods of the maximum degree node grows almost linearly with the neighborhood size, until it converges to the value of $\frac{\delta_{\text{avg}}}{\text{dist}_{\text{avg}}}$ in the whole graph. Convergence time differs from graph to graph. In biological networks, convergence was reached at size close to $\frac{n}{2}$, while in the social and in the technological networks convergence is reached before. For neighborhoods of a random node, we outline a different behavior: at the beginning, the growth is not monotone, like in the previous case, and it is much more irregular. In our opinion, these results are due to the fact that, when the neighborhood grows, it reaches more and more "influential" nodes, and the first neighborhood

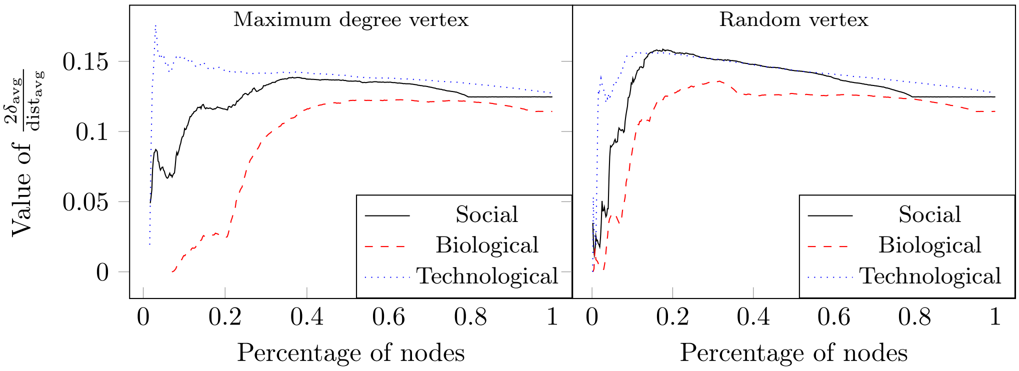


Figure 7.6. The value of $\frac{2\delta_{\text{avg}}}{\text{dist}_{\text{avg}}}$ for neighbours of a randomly chosen node (up left), or the maximum degree node (up right). Results are shown for a social network, a biological network, and a technological network.

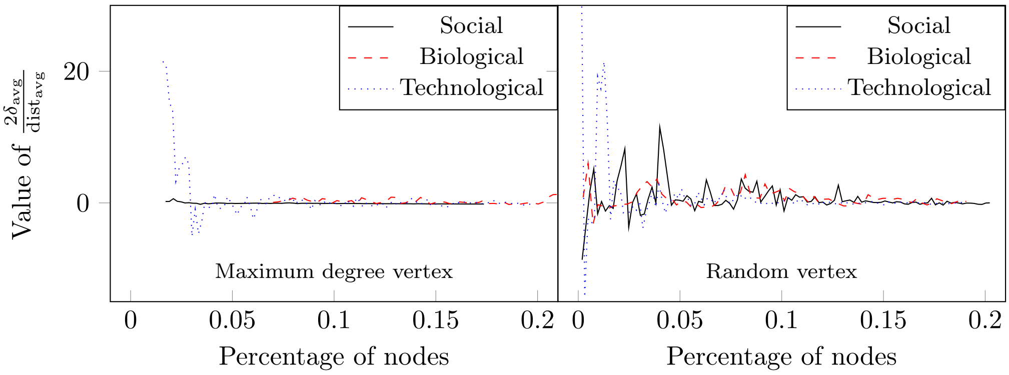


Figure 7.7. The derivative with respect to the neighborhoods size of the value of $\frac{2\delta_{\text{avg}}}{\text{dist}_{\text{avg}}}$, in neighborhoods of the maximum degree node and of random nodes.

that touches such a node corresponds to a local maximum in the plot. This issue is further confirmed by Figure 7.7, where the discrete derivative of $\frac{2\delta_{\text{avg}}}{\text{dist}_{\text{avg}}}$ is shown.

For this reason, we have focused on the maximum degree node, and, in order to have more general results, we have analyzed all graphs in the dataset. Figure 7.8 shows the size of the maximum neighborhood having ratio $\frac{2\delta_{\text{avg}}}{\text{dist}_{\text{avg}}}$ at least half than the same ratio in the whole graph. Actually, we have plotted the fourth neighborhood where this condition is satisfied, in order to avoid random deviations.

We outline that the influence area of an individual is small in social and peer-to-peer networks, compared to biological or autonomous system network. This standard behavior has few exceptions: first of all, protein-protein interaction networks (`string`, `ecoli.interaction`) are different from other biological networks, and the influence area is smaller. Furthermore, the social network `GoogleNW` contains a node with an enormous influence area: this network is the set of Google pages, and the central node s considered is the page `www.google.com`, which clearly dominates all the others. Another particular case is the social network `facebook_combined`: this network is a collection of ego-networks from Facebook, and links are made if common interests are retrieved. We think that this network is different from the others because it is a small subgraph of a bigger graph (where all Facebook users are considered), and the choice of the subgraph has a strong impact on the topology of the network, which does not reflect the standard behavior.

In our opinion, this is due to the fact that the networks with small influence areas are "distributed", in the sense that each node has a goal (downloading in peer-to-peer networks, and creating relationships in social networks), and edges are created locally by nodes that

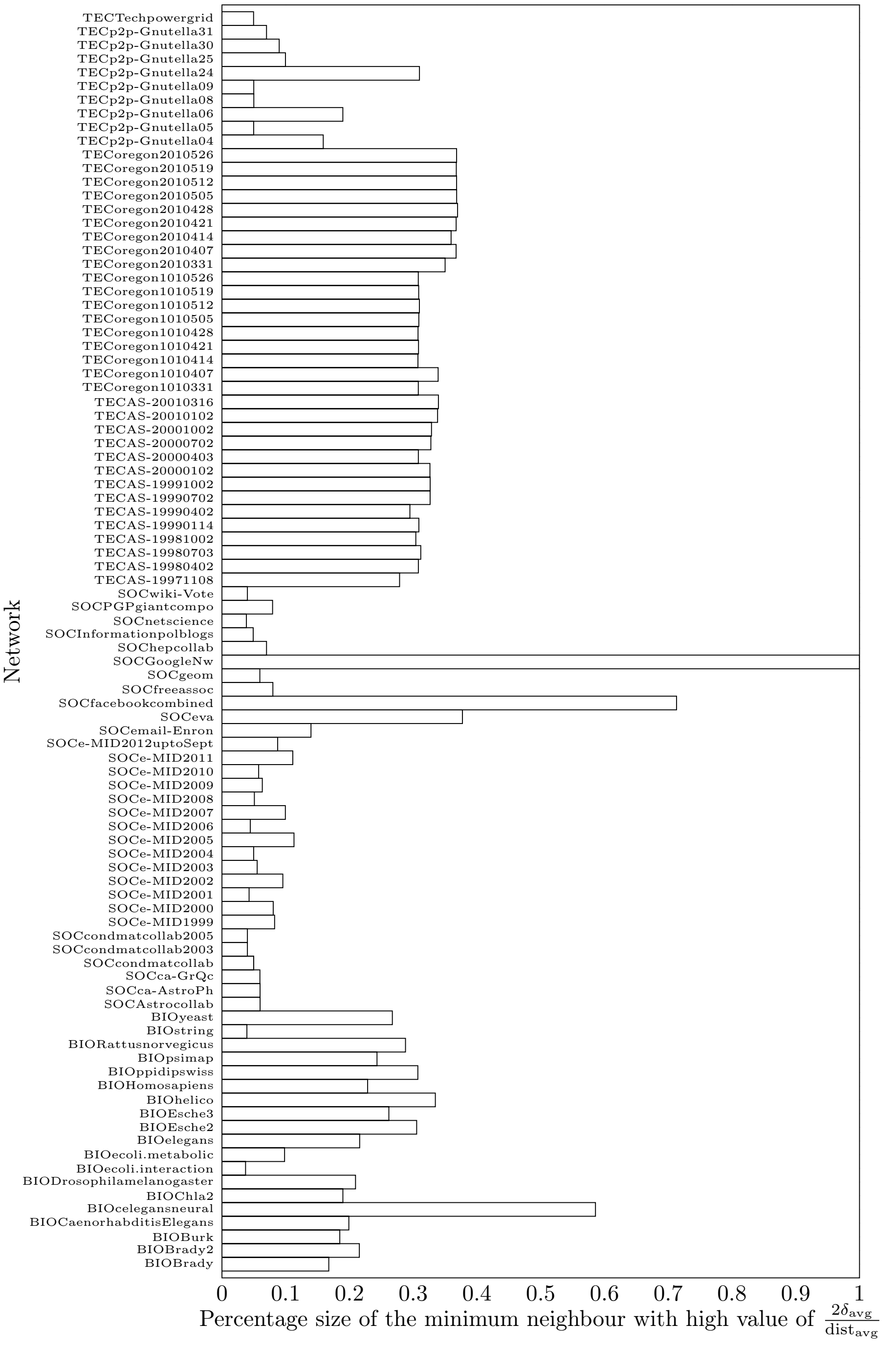


Figure 7.8. The fourth neighborhood of the maximum degree node whose ratio $\frac{2\delta_{\text{avg}}}{\text{dist}_{\text{avg}}}$ is half this value in the whole graph.

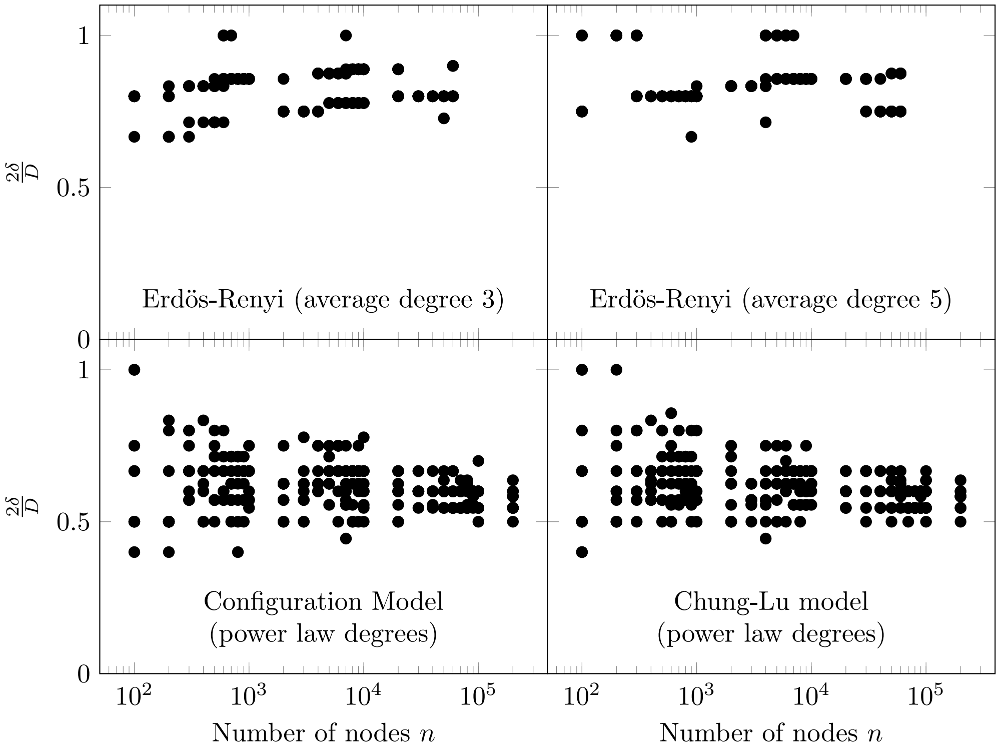


Figure 7.9. For each of the models considered, the ratio $\frac{2\delta}{D}$ in 10 random graphs of each size.

try to reach the goal. On the other hand, the latter networks have global goals (connecting everyone in the network, or making a cell live), and the creation of edges is "centralized".

As far as we know, this is the first work that provides this interpretation of the average hyperbolicity. Possible applications include not only the classification of networks according to this parameter, but also the classifications of nodes in a network, or the classification of different communities. These communities might be democratic, if everyone has "the same role" and δ_{avg} is high, or not democratic, if there is a group of few nodes that keeps the community together, making δ_{avg} small.

7.5 Synthetic Graphs

Recently, some works have tried to compute asymptotic values for the hyperbolicity of random graphs, when the number of nodes n tends to infinity. The simplest model considered is the Erdös-Renyi random graph $G_{n,m}$, that is, we choose a graph with n nodes and m edges uniformly at random. In this model, it has been proved that the hyperbolicity tends to infinity [121], and, if $m > n \log^5 n$, exact asymptotics for δ have been computed [119]. Instead, the hyperbolicity of sparse Erdös-Renyi graphs is not known, and it is mentioned as an open problem in [119]. Among the other possible models, the Chung-Lu model and the Configuration Model stand out for their simplicity: basically, they generalize Erdös-Renyi random graphs by defining weights ρ_v for each node v, and trying to give approximately ρ_v random neighbors to each node v (for a precise definition, see Section 8.1). On these models, as far as we know, it was only proved [144] that the hyperbolicity of a graph generated through the Chung-Lu model tends to infinity if the maximum and minimum degree are "close to each other" (meaning that their ratio is smaller than $2^{\frac{1}{3}}$). Other models were analyzed in [49]: also in that paper, the estimation of the hyperbolicity of random graphs of different kind is mentioned as an open problem.

Following suggestions in [49], we use our algorithm to shed some light on the behavior of these random graphs, at least experimentally, in order to help formulating sensible conjectures on possible asymptotics. In particular, we have restricted our attention to four examples, chosen among the models where exact asymptotics have not been proved: Erdös-Renyi random graphs with $m = 3n$ and $m = 5n$, and graphs generated through the Chung-Lu and the Configuration Model, with power-law degree distribution with exponent 2.5 (similar to the degree distribution of several real-world networks [125]). For each number of nodes $n = k \cdot 10^i$ where $k < 10$ and $i \geq 2$, we have generated 10 graphs and we have computed their hyperbolicity. More precisely, we have computed the value $\frac{2\delta}{D}$, where D is the diameter, which is always between 0 and 1 because of Lemma 7.1: this value might be more interesting than the plain hyperbolicity value, since, for all these models, asymptotics for the diameter are known. Figure 7.9 shows the average value of $\frac{2\delta}{D}$ and the corresponding standard error over the 10 measures performed.

We have been able to compute the hyperbolicity of Erdös-Renyi graphs with up to 60 000 nodes, and graphs generated with the Configuration Model or the Chung-Lu model with up to 200 000 nodes. In all models considered, it is quite evident that the ratio $\frac{2\delta}{D}$ does not tend to 0, and consequently $\delta = \Theta(D)$. Furthermore, the ratio in Erdös-Renyi graphs is not very far from 1, even if the results are not precise enough to discriminate between $\delta = \frac{D}{2}$ or $\delta = cD$ for some $c < \frac{1}{2}$. Instead, in graphs generated through the Configuration Model or the Chung-Lu model, this ratio seems to tend to a value between 0.5 and 0.7.

7.6 Bibliographic Notes

The algorithm described in this chapter is original, and it was published in [30], but it is strongly based on the existing algorithm in [58] (Section 7.1 is based on [58]). The analysis of the hyperbolicity of real-world graph is original, as well, and it was published in [29].

Chapter 8

Probabilistic Analysis of Algorithms

Abstract

In this thesis, we proposed several algorithms that compute metric quantities of real-world complex networks, and that are very efficient in practice, although there is no worst-case guarantee.

In this chapter, we propose an axiomatic framework to analyze the performances of these algorithms, by proving that they are efficient on the class of graphs satisfying certain axioms. Furthermore, we prove that the axioms are satisfied asymptotically almost surely by several probabilistic models that generate power law random graphs, such as the Configuration Model, the Chung-Lu model, and the Norros-Reittu model. Thus, our results imply average-case analyses in these models.

For example, in our framework, our algorithms can compute the diameter and the radius of a graph in subquadratic time, and sometimes even in time $n^{1+o(1)}$. Moreover, in some regimes, it is possible to compute the k most central nodes according to closeness centrality in subquadratic time, and to design a distance oracle with sublinear query time and subquadratic space occupancy.

As we saw in the first chapter, comparable results cannot be obtained in the worst-case, unless widely-believed conjectures are false.

In this thesis, we have developed several algorithms that achieve surprisingly good results in practice, despite they provide no worst-case guarantee (note that, probably, it is impossible to design algorithms with better worst-case guarantees, as we saw in Chapter 3).

Following the standard approach [23, 83, 141, 50, 114, 62, 154, 165, 63, 155, 57, 61, 66, 67, 8, 129, 58], we have validated our algorithms empirically, by showing that they achieve good performances on large datasets of real-world networks. However, this kind of analysis might be biased by the choice of the specific dataset used for the evaluation, and it might not be satisfactory from a theoretical point of view: indeed, it provides no insight into the reasons why these algorithms are so efficient.

In this chapter, we provide a more theoretical framework where the performances of these algorithms can be evaluated and compared. Our framework is axiomatic: we define some axioms, we experimentally show that these axioms hold in most real-world graphs, and we perform a worst-case analysis on the class of graphs satisfying these axioms. This axiomatic approach to the analysis of complex networks follows the research agenda proposed in [85], where a similar analysis was performed for a different property, related to the number of triangles in a graph. Similarly to [85], our approach offers three main advantages: we validate the efficiency of the algorithms considered, we highlight the properties of the input graphs that are exploited, and we perform a comparison that does not depend on the specific dataset used for the evaluation. A further confirmation of the validity of this approach comes from the results obtained, that are very similar to existing empirical results.

Furthermore, again following [85], we show that these axioms are satisfied on some models of random graphs, asymptotically almost surely (a.a.s.), that is, with probability that tends to 1 as the number of nodes n goes to infinity: as a consequence, all results can be turned into average-case analyses on these models, with no modification. This modular approach to average-case complexity has two advantages: since our axioms are satisfied by different models, we can prove results in all these models with a single worst-case analysis. Furthermore, we clearly highlight which properties of random graphs we are using: this way, we can experimentally validate the choice of the probabilistic model, by showing that these properties are reflected by real-world graphs.

In the past, most average-case analyses were performed on the Erdös-Renyi model, which is defined by fixing the number n of nodes, and connecting each pair of nodes with probability p [79, 140, 165, 115]. However many algorithms that work well in practice have poor average-case running time on this model.[1] Indeed, these algorithms are efficient if there are some nodes with very high degree, and such nodes are not present in Erdös-Renyi graphs. Conversely, most real-world graphs contain such nodes, because their degree distribution is power law [15], that is, the number of nodes with degree d is proportional to $\frac{n}{d^\beta}$ for some $\beta > 1$. For this reason, we only consider models that generate power law random graphs. Our framework encompasses almost all values of β, and many of these models: the Configuration Model [26], and Rank-1 Inhomogeneous Random Graph models ([159], Chapter 3), such as the Chung-Lu [54] and the Norros-Reittu model [127].

This approach is based on three axioms that study the behavior of $\boldsymbol{\tau}_s\left(n^x\right)$, which is defined as the smallest integer ℓ such that the number of nodes at distance ℓ from s is at least n^x. The first axiom describes the typical and extremal behavior of $\boldsymbol{\tau}_s\left(n^x\right)$, where s ranges over all nodes in the graph. The other two axioms link the distance between two nodes s and t with $\boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right)$: informally, for each x between 0 and 1, $\operatorname{dist}(s,t)$ is close to $\boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^{1-x}\right)$. Then, we need a fourth axiom, that simply says that the degree distribution is power law (however, we can easily generalize the results to any distribution with finite mean, by changing this axiom). We prove that these axioms are satisfied in the aforementioned graph models.

Using these axioms, we analyze the distance distribution of a graph $G = (V, E)$: we start by estimating the eccentricity of a given node s, and consequently the diameter $D = \max_{s \in V} \operatorname{ecc}(s)$. Similarly, we estimate the *farness* $f(s)$ of s, and consequently the *closeness centrality* of s and the average distance between two nodes. By specializing these results to the random graph models considered, we retrieve known asymptotics for these quantities, and we prove some new asymptotics in the regime $1 < \beta < 2$.

After proving these results, we turn our attention to the analysis of many heuristics and algorithms, by proving all the results in Table 8.1 (a plot of the results is available in Figure 8.1).[2] For approximation algorithms, we usually know the running time and we analyze the error; conversely, for exact algorithms, we bound the running time. All algorithms analyzed are exactly the algorithms published in the original papers or in this thesis, apart from the SumSweepHeuristic and the SumSweep, where we need a small variation to make the analysis work.

In many regimes, our results improve the corresponding worst-case bounds that we proved in Chapter 3: for example, there is no algorithm that computes a $(\frac{3}{2} - \varepsilon)$-approximation of the diameter or the radius in $\mathcal{O}\left(n^{2-\varepsilon}\right)$ (see also [138, 31, 3]), and the complexity of computing

[1] The poor performances of some of these algorithms in the Erdös-Renyi model were empirically shown in [61], and they can be proved with a simple adaptation of the analysis in this chapter.

[2] Some of the results contain a value $o(1)$: this value comes from the axioms, which depend on a parameter ε. In random graphs, this notation is formally correct: indeed, we can let ε tend to 0, since the axioms are satisfied a.a.s. for each ε. In real-world graphs, we experimentally show that these axioms are satisfied for small values of ε, and with abuse of notation we write $o(1)$ to denote a function bounded by $c\varepsilon$, for some constant c.

Table 8.1. A summary of the results of our probabilistic analyses. The constant C is an abbreviation of $\frac{-\log \eta(1)}{\log M_1(\mu)}$, where $\eta(1)$ and $M_1(\mu)$ are defined in Section 8.1; in the paper, we also prove that $C \approx \frac{2\,\mathrm{dist}_{\mathrm{avg}}}{D-\mathrm{dist}_{\mathrm{avg}}}$, where D is the diameter of the graph, $\mathrm{dist}_{\mathrm{avg}}$ is the average distance. The values marked with $(*)$ are proved using further characteristics of the probabilistic models.

Quantity	Algorithm	Proof	Running time		
			$\beta > 3$	$2 < \beta < 3$	$1 < \beta < 2$,
Diameter (lower bound)	BFS from n^γ random nodes	Section 8.4.1	$\Theta(n^{1+\gamma})$ $\varepsilon_{rel} = \frac{1-\gamma+o(1)}{2+C}$	$\Theta(n^{1+\gamma})$ $\varepsilon_{rel} = \frac{1-\gamma+o(1)}{2}$	$\Theta(mn^\gamma)$ $\varepsilon_{abs} = \left\lfloor \frac{2(\beta-1)}{2-\beta} \right\rfloor - \left\lfloor \frac{(\gamma+1)(\beta-1)}{2-\beta} \right\rfloor$
Diameter (lower bound)	2-Sweep [114]	Section 8.5.2	$\Theta(n)$ $\varepsilon_{rel} = o(1)$	$\Theta(n)$ $\varepsilon_{rel} = o(1)$	$\Theta(m)$ $\varepsilon_{abs} \leq \begin{cases} 1 & D \text{ even} \\ 2 & D \text{ odd} \end{cases}$
Diameter (lower bound)	RW [138]	Section 8.7.2	$\Theta(n^{\frac{3}{2}} \log n)$ $\varepsilon_{rel} = o(1)$	$\Theta(n^{\frac{3}{2}} \log n)$ $\varepsilon_{rel} = o(1)$	$\Theta(m\sqrt{n} \log n)$ $\varepsilon_{abs} \leq \begin{cases} 1 & D \text{ even} \\ 2 & D \text{ odd} \end{cases}$
All eccentricities (lower bound)	SS [33]	Section 8.8.1	$n^{1+o(1)}$ $\varepsilon_{abs} = 0$	$n^{1+o(1)}$ $\varepsilon_{abs} = 0$	$\leq mn^{1-\frac{2-\beta}{\beta-1}\left(\left\lfloor \frac{\beta-1}{2-\beta} - \frac{3}{2} \right\rfloor - \frac{1}{2}\right)}$ $\varepsilon_{abs} = 0$
Diameter	iFub [61]	Section 8.9.2	$\leq n^{1+\left(\frac{1}{2}-\frac{1}{\beta-1}\right)C+o(1)}$	$n^{1+o(1)}$	$\leq mn^{1-\frac{2-\beta}{\beta-1}\left\lfloor \frac{\beta-1}{2-\beta} - \frac{1}{2} \right\rfloor + o(1)}$
Diameter	ESS [33, 32]	Section 8.10.1	$\leq n^{1+\frac{C}{C+\frac{\beta-1}{\beta-3}}}$ $(*)$	$n^{1+o(1)}$	$\leq mn^{1-\frac{2-\beta}{\beta-1}\left(\left\lfloor \frac{\beta-1}{2-\beta} - \frac{3}{2} \right\rfloor - \frac{1}{2}\right)}$
Radius	ESS [33, 32]	Section 8.10.1	$n^{1+o(1)}$	$n^{1+o(1)}$	$\leq mn^{1-\frac{2-\beta}{\beta-1}\left(\left\lfloor \frac{\beta-1}{2-\beta} - \frac{3}{2} \right\rfloor - \frac{1}{2}\right)}$
Top-k closeness	BCM [20]	Section 8.11	$n^{2-\frac{1}{\beta-1}}$ $(*)$	$n^{2-o(1)}$	$m^{1+o(1)}$
Distance oracle (query time) (space needed)	AIY [8]	Section 8.12.2	$n^{1-o(1)}$ $n^{2-o(1)}$	$\leq n^{f(\beta)}$ $(*)$ (no closed form) $\leq n^{1+f(\beta)}$ $(*)$	$\leq n^{\frac{1}{2}+o(1)}$ $\leq n^{\frac{3}{2}+o(1)}$
Distance between two nodes	BBBFS [36] (folklore)	Section 8.13.2	$n^{\frac{1}{2}+o(1)}$ $(*)$	$n^{\frac{4-\beta}{2}}$ $(*)$	$n^{1-o(1)}$

the most closeness central node is $\Omega\left(n^{2-\varepsilon}\right)$ (see also [3]). Furthermore, there are hardness results on the possible tradeoffs between space needed and query time in distance oracles [157, 149].

Furthermore, we observe that our results strongly depend on the exponent β: in particular, there are two phase transitions corresponding to $\beta = 2$ and $\beta = 3$. This is due to the fact that, if $1 < \beta < 2$, the average degree is unbounded, if $2 < \beta < 3$, the average degree is finite, but the variance is unbounded, while if $\beta > 3$ also the variance is finite. Furthermore, all the results with $\beta > 3$ can be easily generalized to any degree distribution with finite variance, but the results become more cumbersome and dependent on specific characteristics of the distribution, such as the maximum degree of a node in the graph: for this reason, we focus on the power law case. Conversely, in the case $\beta < 3$, our results strongly depend on the degree distribution to be power law, because random graphs generated with different degree distributions can have very different behaviors. The only open cases are $\beta = 2$ and $\beta = 3$,

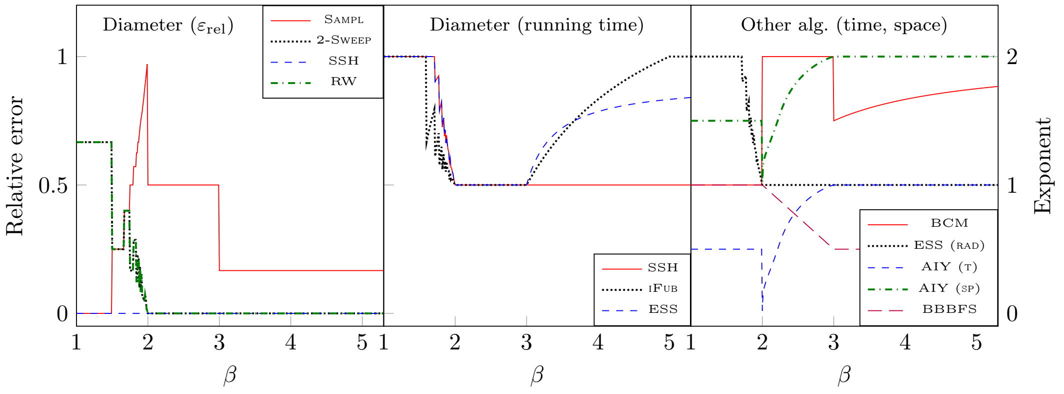


Figure 8.1. Plot of the running time and relative errors of the heuristics and algorithms considered. The constant C was set to 3, and the $o(1)$ were ignored.

which are left for future work (note that, if $\beta \leq 1$, the degree distribution is not well defined). Let us discuss a bit more the results obtained.

Approximating the diameter. We confirm the empirical results in [114], proving that the 2-Sweep heuristic is significantly better than the basic sampling algorithm, which returns the maximum eccentricity of a random set of nodes. Furthermore, we prove that the SumSweep-Heuristic that we developed in Chapter 4 outperforms the 2-Sweep heuristic, confirming our experimental results. Finally, we analyze the well-known RW algorithm, which provides a guaranteed $\frac{3}{2}$-approximation of the diameter in time $\Theta(m\sqrt{n})$. In our framework, it does not improve the 2-Sweep algorithm (which is much faster): this might theoretically explain why many graph libraries implement (variations of) the 2-Sweep heuristic, but not the RW algorithm (for instance, Sagemath [152], WebGraph [23], NetworKit [151]).

Computing the diameter. Some of the aforementioned heuristics can be turned into exact algorithms, that always provide the correct result, but that can be inefficient in the worst-case. We analyze two of these algorithms, proving that, for small values of β, both the iFub and the new ExactSumSweep algorithm are very efficient; for big values of β, the ExactSumSweep algorithm is usually better, because it is always subquadratic. These results explain the surprisingly small running time on most graphs, and the reason why the ExactSumSweep algorithm is usually faster on "hard" instances, as observed in Chapter 4 (see also [33, 32]). It is interesting to note that all the running times for $\beta > 3$ depend on the same constant C, which is close to $\frac{2\,\mathrm{dist}_{\mathrm{avg}}}{D-\mathrm{dist}_{\mathrm{avg}}}$, where D is the diameter and $\mathrm{dist}_{\mathrm{avg}}$ is the average distance in the input graph. Intuitively, if this ratio is small, it means that there are "few far nodes", and the algorithms are quite efficient because they only need to analyze these nodes (the only exception is the sampling algorithm, which is not able to find these nodes, and hence achieves better performances when C is large). For $2 < \beta < 3$, a very similar argument applies, but in this case $C = 0$, because $D = \mathcal{O}(\log n)$ and $\mathrm{dist}_{\mathrm{avg}} = \mathcal{O}(\log \log n)$.

Other algorithms. Our framework lets us also analyze algorithms for computing other quantities. For example, our ExactSumSweep algorithm is also able to compute the radius: in Chapter 4, we experimentally showed that the algorithm is almost linear in practice, since it needs at most 10 BFSes on all inputs but one, and on the last input it needs 18 BFSes. In this chapter, we confirm this result through our probabilistic analysis, proving that in most regimes it has running time $\mathcal{O}(n^{1+o(1)})$. We also analyze the BCM algorithm, defined in Chapter 5, that computes the k most central nodes according to closeness centrality: we show significant improvements with respect to the worst-case in the regime $1 < \beta < 2$ and $\beta > 3$ (assuming k is constant). We also show that the algorithm is not efficient if $2 < \beta < 3$: this is the *only* probabilistic analysis which is not reflected in practice. The problem is that our analysis relies on the fact that $\mathrm{dist}_{\mathrm{avg}}(n) = \Theta(\log \log n)$ tends to infinity, but the experiments were performed on graphs where $n < 10\,000\,000$, and consequently $\log\log(n) < 4$. Furthermore, we analyze the efficiency of the distance oracle AIY: we show that, if $\beta < 3$, the expected time needed to compute the distance between two random nodes is sublinear, and the space occupied is subquadratic. Finally, we analyze a folklore algorithm, used to compute the distance and/or the shortest path between two nodes [148, 65, 95]: the balanced bidirectional BFS, which we described and formalized in Section 6.5. The idea of this algorithm is very simple: in order to compute $\mathrm{dist}(s,t)$, instead of performing a BFS from s until it hits the node t, we perform two BFSs, one from s and one from t, until they "touch each other" (at each step, we extend the "smallest" BFS). We show that, in the regime $\beta > 2$, the time needed by this approach to compute a shortest path is sublinear, and, if $\beta > 3$, it is $\mathcal{O}(n^{\frac{1}{2}+\varepsilon})$.

In Section 8.1, we define the random graph models considered; in Section 8.2 we precisely define the axioms and in Section 8.3 we apply the axioms to prove known asymptotics for metric properties of random graphs (diameter, closeness centrality, average distance, etc.). In Sections 8.4 to 8.13, we provide the axiomatic analysis of all the algorithms considered: for algorithms developed before this thesis, we also include a summary of the main empirical results obtained (while, for algorithms developed in this thesis, we refer to the corresponding chapters). The remainder of this chapter analyzes the validity of the axioms: in Section 8.14 we experimentally show that they hold in most real-world graphs, with good approximation. In Section 8.15, we sketch the main ideas of the proof of the validity of the axioms in random graphs. Since the actual proof is rather long and technical, we moved it to Appendix A.

8.1 The Model

Before defining the model, let us start with some notation. We say that an event E holds asymptotically almost surely or a.a.s. if, when n tends to infinity, $\mathbb{P}(E) = 1 - o(1)$; it holds with high probability or w.h.p. if $\mathbb{P}(E) = 1 - o\left(n^{-k}\right)$ for each $k \in \mathbb{N}$.

Since, in the literature, there is little agreement on which are the best models for directed graphs, in this chapter we only focus on the undirected case, and we tacitly assume that all graphs mentioned are undirected. Our proofs work on different random graph models: the Configuration Model (CM, [26]), and Rank-1 Inhomogeneous Random Graph models (IRG, [159], Chapter 3), such as the Chung-Lu model [54], and the Norros-Reittu model [127]. All these models are defined by fixing in advance the number n of nodes, and n weights ρ_v, one for each node. Then, edges are created at random, trying to give ρ_v outgoing edges to each node v. We assume that the weights ρ_v are chosen according to a power law distribution λ, which is the degree distribution of many real-world graphs [125]: more specifically, we assume that, for each d, the number of nodes with weight bigger than d is $\Theta(\frac{n}{d^{\beta-1}})$, for some constant β.[3] In any case, our results can be easily extended to any degree distribution with finite variance.

After defining the weights, we have to define how we generate the edges:

- in the CM, we give ρ_v half-edges, or stubs to a node v; edges are created by pairing these $M = \sum_{v \in V} \rho_v$ stubs at random (we assume the number of stubs to be even, by adding a stub to a random node if necessary).
- in IRG, an edge between a node v and node w is created independently with probability $f(\frac{\rho_v \rho_w}{M})$, where $M = \sum_{v \in V} \rho_v$, and
  - in general, we assume the following:
    - f is derivable at least twice in 0;
    - f is increasing;
    - $f'(0) = 1$;
    - $f(x) = 1 - o(x^k)$ for each k, when x tends to infinity.
  - in the Chung-Lu model, $f(x) = \min(x, 1)$;
  - in the Norros-Reittu model, $f(x) = 1 - e^{-x}$.

Remark 8.1. The first two assumptions in IRG are needed to exclude pathological cases. The third assumption is just needed to simplify notation, but it can be easily lifted by modifying the weights ρ_v: for instance, if $f'(0) = c$, we may multiply all ρ_vs by $\sqrt{c}$, and redefine $f_1(x) = f\left(\frac{x}{c}\right)$, obtaining the same graph with a function satisfying $f_1'(0) = 1$. The fourth assumption is less natural, and there are models where it is not satisfied, like the Generalized Random Graph model ([160], Chapter 6). However, if the average degree is finite (that is, $\beta > 2$), the proofs do not need this assumption (in this work, we have chosen to use this assumption in order to simplify the statements).

In order to prove our results, we further need some technical assumptions, to avoid pathological cases. In particular, we exclude from our analysis the values of β corresponding to the phase transitions: $\beta = 2$, and $\beta = 3$. Furthermore, in the regime

[3] In some cases, a stronger definition of power law is used, that is, it is assumed that there are $\Theta(\frac{n}{d^{\beta}})$ nodes with degree d, for each d. However, our proofs still work with the weaker definition.

$1 < \beta < 2$, we have other phase transitions related to the diameter of the graph, which is $\left\lfloor 3+\frac{\beta-1}{2-\beta}\right\rfloor$: we assume that $\frac{\beta-1}{2-\beta}$ is not an integer, and, with abuse of notation, we often write $\left\lfloor \frac{\beta-1}{2-\beta}-\varepsilon\right\rfloor = \left\lfloor \frac{\beta-1}{2-\beta}\right\rfloor = \left\lceil \frac{\beta-1}{2-\beta}\right\rceil - 1 = \left\lceil \frac{\beta-1}{2-\beta}+\varepsilon\right\rceil - 1$, with obvious meaning.

Finally, we need a last assumption on the degree distribution λ: all our metric quantities make sense only if the graph is connected. Hence, we need to assume that λ does not contain "too many nodes" of small degree, so that a.a.s. there is a unique connected component of size $\Theta(n)$, named giant component. All our results hold in the giant component of the graph considered.

In the remainder of this section, we define precisely this assumption, and we further define some more constants that appear in the main theorems. A reader who is not interested in these technicalities might just skip this part, assuming that the graph is connected, and that the main axioms hold (our probabilistic analyses do not depend on the definition of these constants).

The first definition is the *residual* distribution μ [76, 159, 160]: intuitively, if we choose a random node v, and we choose a random neighbor w of v, the degree of w is μ-distributed (see Theorem A.21, in the case $\ell = 1$). This distribution is defined as follows.

Definition 8.2. Given a distribution λ, its first moment $M_1(\lambda)$ is the expected value of a λ-distributed random variable. The residual distribution μ of the distribution λ is:

- in the CM, $\mu(i) = \frac{(i+1)\lambda(i+1)}{M_1(\lambda)}$;
- in IRG, let $\mu'(i) = \frac{i\lambda(i)}{M_1(\lambda)}$: $\mu(i)$ is a Poisson distribution with random parameter μ'.

In Appendix A.3, we show that the number of nodes at distance ℓ from a given node v is very close to a μ-distributed branching process $\boldsymbol{Z}^\ell$ (for more background on branching processes, we refer to [12]). If $M_1(\mu) < 1$, this branching process dies a.a.s.: in terms of graphs, it means that the biggest component has size $\mathcal{O}(\log n)$, and there is no giant component. Conversely, if $M_1(\mu)$ is bigger than 1, then the branching process has an infinite number of descendants with positive probability p: in terms of graphs, it means that there is a connected component of size close to pn (see [160] for a proof). Hence, we assume $M_1(\mu) > 1$ and we ignore all the nodes that are not in the giant component.

Finally, given a μ-distributed branching process, we may consider only the branches that have an infinite number of descendant (see [12], I.D.12): we obtain another branching process with offspring distribution η depending on μ. In particular, our results depend on $\eta(1)$, that is, the probability that an η-distributed random variable has value 1 (for estimating the value of $\eta(1)$, we refer to [76]). In the following, we also assume that $\eta(1) > 0$: this is true if and only if $\mu(0) \neq 0$ or $\mu(1) \neq 0$. In IRG, this is automatically implied by the definition of μ, while in the CM this is an additional technical assumption.

8.2 The Axioms

In order to state our axioms, we need the following definitions. In Figure 8.2, we give a visualization of the quantities appearing in these definitions.

Definition 8.3. Given a graph $G = (V, E)$, if $s \in V$, we denote by $\boldsymbol{\Gamma}^\ell(s)$ the set of nodes at distance exactly ℓ from s, and we let $\boldsymbol{\gamma}^\ell(s) = |\boldsymbol{\Gamma}^\ell(s)|$. Similarly, we denote by $\boldsymbol{N}^\ell(s)$ the set of nodes at distance at most ℓ from s, and we let $\boldsymbol{n}^\ell(s) = |\boldsymbol{N}^\ell(s)|$.

Definition 8.4. Let x be a real number such that $0 < x < 1$: we denote by $\boldsymbol{\tau}_s(n^x) = \min\{\ell \in \mathbb{N} : \boldsymbol{\gamma}^\ell(s) > n^x\}$, and by $T(d \to n^x)$ the average number of steps for a node of degree d to obtain a neighborhood of n^x nodes (more formally, $T(d \to n^x)$ is the average of $\boldsymbol{\tau}_s(n^x)$ over all nodes s of degree d).

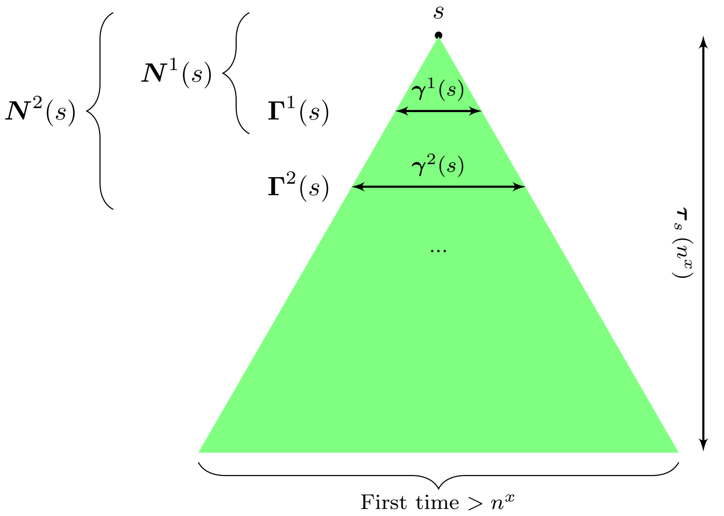


Figure 8.2. A visualization of the quantities appearing in Definitions 8.3 and 8.4. The green triangle represents the tree obtained by performing a BFS from s, stopped as soon as we hit a neighbor with size at least n^x.

Remark 8.5. In general, $\boldsymbol{\tau}_s(n^x)$ could not be defined, if no neighborhood of s has size at least n^x. In random graphs, we solve this issue by showing that $\boldsymbol{\tau}_s(n^x)$ is defined for each node in the giant component, because it contains $\Theta(n)$ nodes and has diameter $\mathcal{O}(\log n)$. In real-world graphs, since the diameter is usually very small, $\boldsymbol{\tau}_s(n^x)$ is defined for each value of x considered in our experiments.

Our axioms depend on a parameter ε: for instance, the first axiom bounds the number of nodes s such that $\boldsymbol{\tau}_s(n^x) \geq (1+\varepsilon)T(d \to n^x)$. Intuitively, one can think of ε as a constant which is smaller than any other constant appearing in the proofs, but bigger than $\frac{1}{n}$, or any other infinitesimal function of n. Indeed, in random graphs, we prove that if we fix $\varepsilon, \delta > 0$, we can find $n_{\varepsilon,\delta}$ such that the axioms hold for each $n > n_{\varepsilon,\delta}$, with probability at least $1-\delta$. In real-world graphs, we experimentally show that the axioms are satisfied with good approximation for $\varepsilon = 0.2$. In our analyses, the time bounds are of the form $n^{c+\mathcal{O}(\varepsilon)}$, and the constants in the $\mathcal{O}$ are quite small. Since, in our dataset, $n^{0.2}$ is between 6 and 19, we can safely consider $n^{c+\mathcal{O}(\varepsilon)}$ close to n^c.

The first axiom analyzes the typical and extremal values of $\boldsymbol{\tau}_s(n^x)$, where s is any node.

Axiom 1. *There exists a constant c such that:*

- *for each node s with degree $d > n^\varepsilon$, $\boldsymbol{\tau}_s(n^x) \leq (1+\varepsilon)\left(T(d \to n^x) + 1\right)$;*
- *the number of nodes satisfying $\boldsymbol{\tau}_s(n^x) \geq (1+\varepsilon)\left(T(d \to n^x) + \alpha\right)$ is $\mathcal{O}\left(nc^{\alpha - x}\right)$;*
- *the number of nodes satisfying $\boldsymbol{\tau}_s(n^x) \geq (1-\varepsilon)\left(T(1 \to n^x) + \alpha\right)$ is $\Omega\left(nc^{\alpha - x}\right)$.*

In random graphs, the values of $T(d \to n^x)$ depend on the exponent β (see Table 8.2). In many of our analyses, we do not use the actual values of $T(d \to n^x)$, but we use the following properties:

- $T\left(d \to n^{x+\varepsilon}\right) \leq T(d \to n^x)(1+\mathcal{O}(\varepsilon))$;
- $\sum_{d=1}^{\infty} |\{v \in V : \deg(v) = d\}| T(d \to n^x) = (1+o(1))nT(1 \to n^x)$;
- $T(1 \to n^x) + T\left(1 \to n^{1-x}\right) - 1 = (1+o(1))\,\mathrm{dist}_{\mathrm{avg}}(n)$, where $\mathrm{dist}_{\mathrm{avg}}(n)$ is a function not depending on x (this function is very close to the average distance $\mathrm{dist}_{\mathrm{avg}}$, as we prove in Corollary 8.16).

Table 8.2. The values of $T\left(d \to n^x\right)$, $\text{dist}_{\text{avg}}\left(n\right)$ and c, depending on the value of β. All these values should be multiplied by $(1 + o(1))$, which is omitted to increase readability.

Regime	$T\left(d \to n^x\right)$	$\text{dist}_{\text{avg}}\left(n\right)$	c
$1 < \beta < 2$	1 if $d \geq n^x$, 2 otherwise	3	$n^{-\frac{2-\beta}{\beta-1}+o(1)}$
$2 < \beta < 3$	$\log_{\frac{1}{\beta-2}} \frac{\log n^x}{\log d}$ if $n^x < n^{\frac{1}{\beta-1}}$ $\log_{\frac{1}{\beta-2}} \frac{\log n^x}{\log d} + \mathcal{O}(1)$ if $n^x > n^{\frac{1}{\beta-1}}$	$2 \log_{\frac{1}{\beta-2}} \log n$	$\eta(1)$
$\beta > 3$	$\log_{M_1(\mu)} \frac{n^x}{d}$	$\log_{M_1(\mu)} n$	$\eta(1)$

The other two axioms relate the distance between two nodes s, t with the values of $\boldsymbol{\tau}_s\left(n^x\right)$, $\boldsymbol{\tau}_t\left(n^y\right)$, where x, y are two reals between 0 and 1. The idea behind these axioms is to apply the "birthday paradox", assuming that $\mathbf{\Gamma}^{\boldsymbol{\tau}_s(n^x)}(s)$ and $\mathbf{\Gamma}^{\boldsymbol{\tau}_t(n^y)}(t)$ are random sets of n^x and n^y nodes. In this idealized setting, if $x + y > 1$, there is a node that is common to both, and $\text{dist}(s,t) \leq \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right)$; conversely, if $x + y < 1$, $\text{dist}(s,t)$ is likely to be bigger than $\boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right)$. Let us start with the simplest axiom, which deals with the case $x + y > 1$.

Axiom 2. *Let us fix two real numbers $0 < x, y < 1$ such that $x + y > 1 + \varepsilon$. For each pair of nodes s, t, $\text{dist}(s,t) < \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right)$.*

The third axiom is a sort of converse: the main idea is that, if the product of the size of two neighborhoods is smaller than n, then the two neighborhoods are usually not connected. The simplest way to formalize this is to state that, for each pair of nodes s, t, $\text{dist}(s,t) \geq \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right)$. However, there are two problems with this statement: first, in random graphs, if we fix s and t, $\text{dist}(s,t) \geq \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right)$ a.a.s., not w.h.p., and hence there might be $o(n)$ nodes t such that $\text{dist}(s,t) < \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right)$ (for example, if s and t are neighbors, they do not satisfy $\text{dist}(s,t) \geq \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right)$). To solve this, our theorem bounds the number of nodes t satisfying $\text{dist}(s,t) \geq \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right)$. The second problem is more subtle: for example, if s has degree 1, and its only neighbor has degree $n^{\frac{1}{2}}$, $\boldsymbol{\tau}_s\left(n^{\frac{1}{4}}\right) = \boldsymbol{\tau}_s\left(n^{\frac{1}{2}}\right) = 2$, and the previous statement cannot hold for $x = \frac{1}{4}$. However, this problem does not occur if $x \geq y$: the intuitive idea is that we can "ignore" nodes with degree bigger than n^x. Indeed, if a shortest path from s to t passes through a node v with degree bigger than n^x, then $\boldsymbol{\tau}_s\left(n^x\right) \leq \text{dist}(s,v) + 1$, $\boldsymbol{\tau}_t\left(n^y\right) \leq \text{dist}(t,v) + 1$, and hence $\text{dist}(s,t) = \text{dist}(s,v) + \text{dist}(v,t) \geq \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right) - 2$.

Axiom 3. *Let $0 < z \leq y < x < 1$, let $x + y \geq 1 + \varepsilon$, and let α, ω be integers. If $T_{\alpha,\omega,z}$ is the set of nodes t such that $\boldsymbol{\tau}_t\left(n^z\right)$ is between α and ω, there are at most $|T_{\alpha,\omega,z}|\frac{n^{x+y+\varepsilon}}{n}$ nodes $t \in T$ such that $\text{dist}(s,t) < \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right) - 2$.*

Finally, in some analyses, we also need to use the fact that the degree distribution is power law. To this purpose, we add a further axiom (in random graphs, this result is well-known [159, 160]).

Axiom 4. *The number of nodes with degree bigger than d is $\Theta\left(\frac{n}{d^{\max(1,\beta-1)}}\right)$.*

8.3 Consequences of the Axioms

Before showing the validity of the axioms, we provide some consequences of the axioms, that describe the metric structure of the graphs considered. These results are fundamental in all our probabilistic analyses, and they are repeatedly used in the subsequent chapters.

Furthermore, by specializing these results to random graphs, we obtain a new proof of known asymptotics, and we prove new asymptotics in the case $1 < \beta < 2$. In all the following lemmas, with abuse of notation, we write $\mathcal{O}(\varepsilon)$ even if ε is a constant, in order to indicate a function bounded by $k\varepsilon$ for some constant k.

Lemma 8.6. *All nodes s with degree d satisfy*

$$\boldsymbol{\tau}_s\left(n^x\right) \leq \left\lfloor (1+\mathcal{O}(\varepsilon))\left(T\left(d \to n^x\right)+\frac{\log n}{-\log c}+x\right)\right\rfloor .$$

Moreover, for each $\delta>0$, there are $\Omega\left(n^\delta\right)$ nodes s with degree 1 satisfying

$$\boldsymbol{\tau}_s\left(n^x\right) \geq \left\lceil (1-\varepsilon-\delta)\left(T\left(1 \to n^x\right)+\frac{\log n}{-\log c}-1+x\right)\right\rceil .$$

Proof. By Axiom 1 applied with $\alpha=(1+\varepsilon)\frac{\log n}{-\log c}+x$, there are

$$\mathcal{O}\left(nc^{\alpha-x}\right)=\mathcal{O}\left(nc^{(1+\varepsilon)\frac{\log n}{-\log c}}\right) \leq \mathcal{O}\left(n^{-\varepsilon}\right)<1$$

nodes s such that

$$\begin{aligned}\boldsymbol{\tau}_s\left(n^x\right) &\geq (1+\varepsilon)\left(T\left(d \to n^x\right)+\alpha\right)\\ &=(1+\varepsilon)\left(T\left(d \to n^x\right)+(1+\varepsilon)\frac{\log n}{-\log c}+x\right).\end{aligned}$$

By observing that $\boldsymbol{\tau}_s\left(n^x\right)$ is an integer, we obtain the first claim.

For the other inequality, let us apply Axiom 1 with $\alpha=(1-\delta)\frac{\log n}{-\log c}-1+x$: we obtain that there are $\Omega\left(nc^{\alpha+1-x}\right)=\Omega\left(nc^{(1-\delta)\frac{\log n}{-\log c}}\right)=\Omega\left(n^\delta\right)$ nodes s such that $\boldsymbol{\tau}_s\left(n^x\right) \geq (1-\varepsilon)\left(T\left(1 \to n^x\right)+\alpha\right)=(1-\varepsilon)\left(T\left(1 \to n^x\right)+(1-\delta)\frac{\log n}{-\log c}-1+x\right) \geq (1-\varepsilon-\delta)\left(T\left(1 \to n^x\right)+\frac{\log n}{-\log c}-1+x\right)$. By observing that $\boldsymbol{\tau}_s\left(n^x\right)$ is an integer, the second claim is proved.

□

By combining the previous lemma with Axioms 2 and 3, we can estimate the eccentricity of each node.

Definition 8.7. The eccentricity $\operatorname{ecc}(s)$ of a node s is $\max_{t \in V} \operatorname{dist}(s,t)$.

Theorem 8.8. *For each node s and for each $0<x<1$,*

$$\operatorname{ecc}(s) \leq \boldsymbol{\tau}_s\left(n^x\right)+\left\lfloor (1+\mathcal{O}(\varepsilon))\left(T\left(1 \to n^{1-x}\right)+\frac{\log n}{-\log c}-x\right)\right\rfloor .$$

Furthermore, for each s and for each $x \geq \frac{1}{2}$:

$$\operatorname{ecc}(s) \geq \boldsymbol{\tau}_s\left(n^x\right)+\left\lceil (1+\mathcal{O}(\varepsilon))\left(T\left(1 \to n^{1-x}\right)+\frac{\log n}{-\log c}-x\right)\right\rceil-2.$$

Proof. By Axiom 2, for each node t, $\operatorname{dist}(s,t) \leq \boldsymbol{\tau}_s\left(n^x\right)+\boldsymbol{\tau}_t\left(n^{1-x+\varepsilon}\right)-1$. By Lemma 8.6, for each t,

$$\boldsymbol{\tau}_t\left(n^{1-x+\varepsilon}\right) \leq \left\lfloor (1+\mathcal{O}(\varepsilon))\left(T\left(\deg(t) \to n^{1-x+\varepsilon}\right)+\frac{\log n}{-\log c}+1-x+\varepsilon\right)\right\rfloor ,$$

and consequently

$$\operatorname{ecc}(s)=\max_{t \in V} \operatorname{dist}(s,t)$$

$$
\begin{aligned}
&\leq \boldsymbol{\tau}_s\left(n^x\right) + \left\lfloor (1+\mathcal{O}(\varepsilon)) \left(T\left(1 \to n^{1-x+\varepsilon}\right) + \frac{\log n}{-\log c} + 1 - x + \varepsilon\right)\right\rfloor - 1 \\
&= \left\lfloor (1+\mathcal{O}(\varepsilon)) \left(T\left(1 \to n^{1-x}\right) + \frac{\log n}{-\log c} - x\right)\right\rfloor .
\end{aligned}
$$

For the other inequality, if $x \geq \frac{1}{2}$, let $y = 1 - x - \varepsilon < x$, and let T be the set of nodes t such that $\boldsymbol{\tau}_t\left(n^y\right) \geq \left\lceil (1-3\varepsilon)\left(T\left(1 \to n^y\right) + \frac{\log n}{-\log c} - 1 + y\right)\right\rceil$ (by Lemma 8.6, $|T| \geq n^{2\varepsilon}$). By Axiom 3, there is at least a node $t \in T$ satisfying

$$
\begin{aligned}
\operatorname{dist}(s,t) &\geq \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right) - 2 \\
&\geq \boldsymbol{\tau}_s\left(n^x\right) + \left\lceil (1-3\varepsilon)\left(T\left(1 \to n^y\right) + \frac{\log n}{-\log c} - 1 + y\right) - 2\right\rceil .
\end{aligned}
$$

The second claim follows. □

Thanks to this lemma, we can compute the diameter of a graph as the maximum eccentricity.

Definition 8.9. The diameter D of a connected graph is $\max_{s,t \in V} \operatorname{dist}(s,t) = \max_{s \in V} \operatorname{ecc}(s)$.

Theorem 8.10. *For each x, the diameter of our graph is*

$$
D = \left\lfloor (1+\mathcal{O}(\varepsilon)) \left(\operatorname{dist}_{\text{avg}}(n) + \frac{2\log n}{-\log c}\right)\right\rfloor .
$$

Proof. By combining the upper bounds in Theorem 8.8 and Lemma 8.6, we can prove that

$$
\begin{aligned}
D \leq &\left\lfloor (1+\mathcal{O}(\varepsilon)) \left(T\left(1 \to n^x\right) + \frac{\log n}{-\log c} + x\right)\right\rfloor \\
&\qquad + \left\lfloor (1+\mathcal{O}(\varepsilon)) \left(T\left(1 \to n^{1-x}\right) + \frac{\log n}{-\log c} - x\right)\right\rfloor .
\end{aligned}
$$

If we choose x such that $(1+\mathcal{O}(\varepsilon))\left(T\left(1 \to n^x\right) + \frac{\log n}{-\log c} + x\right) = i - \varepsilon$, we obtain that

$$
\begin{aligned}
D &\leq i - 1 + \left\lfloor (1+\mathcal{O}(\varepsilon)) \left(T\left(1 \to n^{1-x}\right) + \frac{\log n}{-\log c} - x\right)\right\rfloor \\
&\leq \left\lfloor i + (1+\mathcal{O}(\varepsilon)) \left(T\left(1 \to n^{1-x}\right) + \frac{\log n}{-\log c} - x - 1\right)\right\rfloor \\
&\leq \left\lfloor (1+\mathcal{O}(\varepsilon)) \left(\operatorname{dist}_{\text{avg}}(n) + \frac{2\log n}{-\log c}\right)\right\rfloor .
\end{aligned}
$$

Let us combine the lower bounds in Theorem 8.8 and Lemma 8.6: we obtain that

$$
\begin{aligned}
D \geq &\left\lceil (1-\mathcal{O}(\varepsilon)) \left(T\left(1 \to n^x\right) + \frac{\log n}{-\log c} + x - 1\right)\right\rceil \\
&\qquad + \left\lceil (1-\mathcal{O}(\varepsilon)) \left(T\left(1 \to n^{1-x}\right) + \frac{\log n}{-\log c} - x - 1\right)\right\rceil - 1.
\end{aligned}
$$

For all but a constant number of values of x, we obtain that

$$D \geq \left\lfloor (1-\mathcal{O}(\varepsilon)) \left(T\left(1 \to n^x\right) + \frac{\log n}{-\log c} + x \right) \right\rfloor \\ + \left\lfloor (1-\mathcal{O}(\varepsilon)) \left(T\left(1 \to n^{1-x}\right) + \frac{\log n}{-\log c} - x \right) \right\rfloor - 1.$$

Furthermore, if $(1-\mathcal{O}(\varepsilon)) \left(T\left(1 \to n^x\right) + \frac{\log n}{-\log c} + x \right) = i + \varepsilon$ for some integer i, this value is $\left\lfloor i + (1-\mathcal{O}(\varepsilon)) \left(T\left(1 \to n^{1-x}\right) + \frac{\log n}{-\log c} - x \right) \right\rfloor - 1 \geq \left\lfloor (1-\mathcal{O}(\varepsilon)) \left(\mathrm{dist}_{\mathrm{avg}}(n) + \frac{2\log n}{-\log c} \right) \right\rfloor$. A similar argument can be applied if the second term is $i+\varepsilon$: hence, it only remains to prove that we can find a value of x between $\frac{1}{2}$ and 1 such that one of the two parts is close to an integer. This is true because $T\left(1 \to n^x\right) + x$ is continuous and increasing with respect to x, and $T\left(1 \to n^{1-x}\right) - x$ is continuous and decreasing. Since the increase and the decrease are at least $\frac{1}{2}$, the sum of the two is at least 1. □

Given Theorem 8.32, and given the values in Table 8.2, Theorem 8.10 gives diameter bounds for power law graphs generated through the models considered (since the axioms hold for each ε, we can safely let ε tend to 0, and transform $\mathcal{O}(\varepsilon)$ into $o(1)$). As far as we know, the bound for $1 < \beta < 2$ is new, while the other bounds are already known [76, 27].

Corollary 8.11. *If λ is a power law degree distribution with exponent β, the diameter of a random graph with degree distribution λ is:*

- *if $1 < \beta < 2$, $D = \left\lfloor 3 + \frac{\beta-2}{\beta-1} \right\rfloor$;*
- *if $2 < \beta < 3$, $D = (1+o(1)) \left(\frac{2}{-\log \eta(1)} \right) \log n$;*
- *if $\beta > 3$, $D = (1+o(1)) \left(\frac{2}{-\log \eta(1)} + \frac{1}{\log M_1(\mu)} \right) \log n$;*

All the previous results deal with "extremal" properties of the distance distribution. Instead, the next results deal with properties that hold on average. Let us start by estimating the farness of a node s, that is, $\frac{1}{n-1} \sum_{t \in V} \mathrm{dist}(s,t)$.

Theorem 8.12. *For each node s and for each $0 < x < 1$, the* farness $f(s)$ *of s satisfies*

$$f(s) \leq (1+\mathcal{O}(\varepsilon)) \left(\boldsymbol{\tau}_s\left(n^x\right) - T\left(1 \to n^x\right) + \mathrm{dist}_{\mathrm{avg}}(n) \right) - \frac{\deg(s)}{n}.$$

Proof. By Axiom 2, for each node t, $\mathrm{dist}(s,t) \leq \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^{1-x+\varepsilon}\right) - 1$, and hence

$$\begin{aligned} \sum_{t \in V} \mathrm{dist}(s,t) &\leq n\left(\boldsymbol{\tau}_s\left(n^x\right) - 1\right) + \sum_{d=1}^{+\infty} \sum_{\deg(t)=d} \boldsymbol{\tau}_t\left(n^{1-x+\varepsilon}\right) \\ &= n\left(\boldsymbol{\tau}_s\left(n^x\right) - 1\right) + \sum_{d=1}^{+\infty} |\{t \in V : \deg(t) = d\}| T\left(d \to n^{1-x+\varepsilon}\right) - n \\ &\leq n\left(\boldsymbol{\tau}_s\left(n^x\right) - 1\right) + n(1+o(1)) T\left(1 \to n^{1-x+\varepsilon}\right) + nT\left(1 \to n^{x-\varepsilon}\right) \\ &\quad - nT\left(1 \to n^{x-\varepsilon}\right) \\ &\leq n(1+\mathcal{O}(\varepsilon)) \left(\boldsymbol{\tau}_s\left(n^x\right) - T\left(1 \to n^x\right) + \mathrm{dist}_{\mathrm{avg}}(n) \right). \end{aligned}$$

We need to subtract $\frac{\deg(s)}{n}$ from this result. To this purpose, we observe that in the first estimate, all neighbors of s with degree at most n^{1-x} were given a distance $\boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^{1-x+\varepsilon}\right) -$

$1 = \boldsymbol{\tau}_s(n^x) + 2 - 1 \geq 2$, and consequently the other estimates remain correct if we subtract the number of neighbors of s with degree at most n^{1-x}, or equivalently if we subtract $\deg(s)$ and we sum the number of neighbors of s with degree at least n^{1-x}. Since $|E| \leq n^{1+\varepsilon}$ by Axiom 4, the number of nodes with degree at least n^{1-x} is at most $n^{x+\varepsilon}$, and the latter contribution is negligible. □

Theorem 8.13. *For each node s and for each $\frac{1}{2} \leq x < 1$,*

$$f(s) \geq (1 - \mathcal{O}(\varepsilon)) \left(\boldsymbol{\tau}_s(n^x) - T(1 \to n^x) + \mathrm{dist}_{\mathrm{avg}}(n) - 1\right).$$

Proof. Let s be any node, and let us apply Axiom 3 with $T = V$: there are at most $\mathcal{O}\left(n^{1-\varepsilon}\right)$ nodes $t \in V$ such that $\mathrm{dist}(s,t) < \boldsymbol{\tau}_s(n^x) + \boldsymbol{\tau}_t\left(n^{1-x-2\varepsilon}\right) - 2$. Let $T' := \{t \in V : \mathrm{dist}(s,t) \geq \boldsymbol{\tau}_s(n^x) + \boldsymbol{\tau}_t\left(n^{1-x-2\varepsilon}\right) - 2\}$.

$$\begin{aligned}
f(s) &= \frac{1}{n-1}\sum_{t \in V} \mathrm{dist}(s,t) \\
&\geq \frac{1}{n-1}\sum_{t \in V'} \boldsymbol{\tau}_s(n^x) + \boldsymbol{\tau}_t\left(n^{1-x-2\varepsilon}\right) - 2 \\
&= (1 - o(1))\left(\boldsymbol{\tau}_s(n^x) - 2\right) + \frac{1}{n-1}\sum_{t \in V} \boldsymbol{\tau}_t\left(n^{1-x-2\varepsilon}\right) - \boldsymbol{\tau}_t\left(n^{1-x-2\varepsilon}\right) \\
&= (1 - \mathcal{O}(\varepsilon))(\boldsymbol{\tau}_s(n^x) - T(1 \to n^x) + \mathrm{dist}_{\mathrm{avg}}(n) - 1) - \frac{|V - V'|}{n-1}\mathcal{O}(\log n) \\
&= (1 - \mathcal{O}(\varepsilon))(\boldsymbol{\tau}_s(n^x) - T(1 \to n^x) + \mathrm{dist}_{\mathrm{avg}}(n) - 1).
\end{aligned}$$

□

By computing the inverse of the farness, we can compute the closeness centrality of a node.

Definition 8.14. The closeness centrality of a node s in a connected graph is $\frac{1}{f(s)} = \frac{1}{\sum_{t \in V} \mathrm{dist}(s,t)}$.

Corollary 8.15. *For each x such that $\frac{1}{2} \leq x < 1$, the closeness centrality of a node s satisfies*

$$\begin{aligned}
\frac{1 - \mathcal{O}(\varepsilon)}{\boldsymbol{\tau}_s(n^x) - T(1 \to n^x) + \mathrm{dist}_{\mathrm{avg}}(n) - \deg(s)/n} &\leq c(s) \\
&\leq \frac{1 + \mathcal{O}(\varepsilon)}{\boldsymbol{\tau}_s(n^x) - T(1 \to n^x) + \mathrm{dist}_{\mathrm{avg}}(n) - 1}.
\end{aligned}$$

Corollary 8.16. *The average distance* $\mathrm{dist}_{\mathrm{avg}}$ *is between* $(1 - \mathcal{O}(\varepsilon))\,\mathrm{dist}_{\mathrm{avg}}(n) - 1$ *and* $(1 + \mathcal{O}(\varepsilon))\,\mathrm{dist}_{\mathrm{avg}}(n)$*.*

Proof. The average distance is the average of the farness of all nodes. By the two previous theorems, for each $x \geq \frac{1}{2}$,

$$\begin{aligned}
(1 + \mathcal{O}(\varepsilon))\left(\boldsymbol{\tau}_s(n^x) - T(1 \to n^x) + \mathrm{dist}_{\mathrm{avg}}(n) - 1\right) &\leq f(s) \\
&\leq (1 + \mathcal{O}(\varepsilon))\left(\boldsymbol{\tau}_s(n^x) - T(1 \to n^x) + \mathrm{dist}_{\mathrm{avg}}(n)\right).
\end{aligned}$$

Let us compute $\sum_{s \in V} \boldsymbol{\tau}_s(n^x) = \sum_{d=1}^{+\infty} \sum_{\deg(s)=d} \boldsymbol{\tau}_s(n^x) = \sum_{d=1}^{\infty} |\{s : \deg(s) = d\}| T(d \to n^x) = n(1 + o(1))T(1 \to n^x)$. Combining this estimate with the previous equation, we obtain:

$$(1-\mathcal{O}(\varepsilon))\,\mathrm{dist}_{\mathrm{avg}}(n)-1 \leq \frac{1}{n}\sum_{s\in V} f(s) \leq (1+\mathcal{O}(\varepsilon))\,\mathrm{dist}_{\mathrm{avg}}(n).$$

□

Again, assuming Theorem 8.32, and given the values in Table 8.2, we have proved the following asymptotics for the average distance in random graphs.

Corollary 8.17. *If λ is a power law degree distribution with exponent β, the average distance in a random graph with degree distribution λ is:*

- *if* $1<\beta<2$, $2 \leq \mathrm{dist}_{\mathrm{avg}} \leq 3$*;*
- *if* $2<\beta<3$, $\mathrm{dist}_{\mathrm{avg}} = (2+o(1))\left(\log_{\frac{1}{\beta-1}}\log n\right)$*;*
- *if* $\beta>3$, $\mathrm{dist}_{\mathrm{avg}} = (1+o(1))\frac{\log n}{\log M_1(\mu)}$*.*

8.4 The Sampling Algorithm to Lower Bound the Diameter

The first algorithm we analyze is very simple: it lower bounds the diameter of a graph by performing k BFSs from random nodes $s_1,\dots,s_k$, and returning $\max_{i=1,\dots,k}\mathrm{ecc}(s_i) = \max_{i=1,\dots,k}\max_{t\in V}\mathrm{dist}(s_i,t) \leq D$.

This algorithm was used in [118] for a large-scale analysis of social networks, it is implemented in the SNAP graph library [108], where it was used to estimate the diameter of all the graphs in the well-known dataset which is shipped with the library itself.

Clearly, the running time of this algorithm is $\Theta(mk)$: we want to analyze the error of this method on graphs that satisfy our assumptions. The main idea behind this analysis is that $\mathrm{ecc}(s)$ is strongly correlated with $\boldsymbol{\tau}_s(n^x)$, and the number of nodes satisfying $\boldsymbol{\tau}_s(n^x)>\alpha$ decreases exponentially with respect to α. This means that the number of nodes with high eccentricity is very small, and it is difficult to find them by sampling: as a consequence, we expect the error to be quite big. Indeed, it was proved in [62, 61] that the values computed by this algorithm are not tight on many graphs.

8.4.1 Probabilistic Analysis

Let us formalize the aforementioned intuition. By Theorem 8.8, the eccentricity of a node s satisfies

$$\mathrm{ecc}(s) \leq \boldsymbol{\tau}_s(n^x) + \left\lfloor (1+\mathcal{O}(\varepsilon))\left(T\left(1\to n^{1-x}\right)+\frac{\log n}{-\log c}-x\right)\right\rfloor,$$

and consequently the output is at most

$$\max_{i=1,\dots,k}\boldsymbol{\tau}_{s_i}(n^x) + \left\lfloor (1+\mathcal{O}(\varepsilon))\left(T\left(1\to n^{1-x}\right)+\frac{\log n}{-\log c}-x\right)\right\rfloor.$$

We want to estimate $\max_{i=1,\dots,k}\boldsymbol{\tau}_{s_i}(n^x)$ through Axiom 1: the number of nodes s satisfying $\boldsymbol{\tau}_s(n^x) \geq (1+\mathcal{O}(\varepsilon))\big(T(1\to n^x)+\alpha\big)$ is at most $nc^{\alpha-x}$, and consequently a random set of k nodes does not contain any such node, a.a.s., if $k \leq \frac{n^{1-\varepsilon}}{nc^{\alpha-x}} \ll \frac{n}{nc^{\alpha-x}}$. Solving the first inequality with respect to α, we obtain $\alpha \geq x+\frac{\varepsilon\log n+\log k}{-\log c}$.

We conclude that, a.a.s., if $\alpha = x + \frac{\varepsilon \log n + \log k}{-\log c}$, we do not perform any BFS from a node s such that $\boldsymbol{\tau}_s\left(n^x\right) \geq (1+\varepsilon)\left(T\left(1 \to n^x\right) + \alpha\right) = (1+\mathcal{O}(\varepsilon))\left(T\left(1 \to n^x\right) + \frac{\gamma \log n}{-\log c} + x\right)$. This means that, for a suitable choice of x, the output is smaller than:

$$\begin{aligned}
&\max_{s \in X} \boldsymbol{\tau}_s\left(n^x\right) + \left\lfloor (1+\mathcal{O}(\varepsilon))\left(T\left(1 \to n^{1-x}\right) + \frac{\log n}{-\log c} - x\right)\right\rfloor \\
&\leq \left\lfloor (1+\mathcal{O}(\varepsilon))\left(T\left(1 \to n^x\right) + \frac{\gamma \log n}{-\log c} + x\right)\right\rfloor + \\
&\quad + \left\lfloor (1+\mathcal{O}(\varepsilon))\left(T\left(1 \to n^{1-x}\right) + \frac{\log n}{-\log c} - x\right)\right\rfloor \\
&\leq \left\lfloor (1+\mathcal{O}(\varepsilon))\left(T\left(1 \to n^x\right) + \frac{\gamma \log n}{-\log c} + T\left(1 \to n^{1-x}\right) + \frac{\log n}{-\log c} - 1\right)\right\rfloor \\
&\leq \left\lfloor (1+\mathcal{O}(\varepsilon))\left(\mathrm{dist}_{\mathrm{avg}}\left(n\right) + \frac{(1+\gamma)\log n}{-\log c} - 1\right)\right\rfloor.
\end{aligned}$$

By replacing the values in Table 8.2, we obtain the desired results. In order to obtain a lower bound on the error, it is enough to perform similar computations after replacing ε with $-\varepsilon$.

8.5 The 2-Sweep Heuristic

The 2-Sweep heuristic [114] finds a lower bound on the diameter, by performing a BFS from a node s, finding a node t that maximizes the distance from s, and returning the eccentricity of t. The running time is linear in the input size, because the algorithm performs 2 BFSs: the interesting part is analyzing the error obtained, both in real-world graphs and in the graphs in our framework.

8.5.1 Experimental Results

The performances of the 2-Sweep heuristic on real-world graphs were analyzed in [114]. The authors used four real-world networks for their experiments:

- the `as-skitter` network available in the SNAP dataset;
- a traffic graph (`ip`) obtained from MetroSec, which captured each IP packet header routed by a given router during 24 hours, two IP addresses being linked if they appear in a packet as sender and destination, leading to 2 250 498 nodes and 19 394 216 edges.
- a peer-to-peer graph (`p2p`) in which two peers are linked if one of them provided a file to the other in a measurement conducted on a large eDonkey server for a period of 47 hours in 2004, leading to 5 792 297 nodes and 142 038 401 edges;
- the `uk-2004` network available in the WebGraph dataset [23].

The graphs used in this analysis are publicly available on the webpage `https://www.complexnetworks.lip6.fr/~magnien/Diameter/`, managed by Clémence Magnien, one of the authors of [114].

In the analysis, the authors run the 2-Sweep heuristic for 2 000 iterations in the first two graphs, 5 000 iterations in the third graph, and 10 000 iterations in the fourth graph. We report in Table 8.3 the results obtained, together with the exact diameter, computed with more modern approaches that were not available when the paper [114] was published.

Table 8.3. For all the graphs considered, the maximum lower obtained by the 2-Sweep heuristic in all the iterations, the percentage of starting nodes achieving that bound, and the first iteration when that bound was obtained.

Network	Diameter	Lower bound	Percentage	First iteration
as-skitter	31	31	0.49	2
ip	9	9	0.989	1
p2p	9	9	0.7005	1
uk-2004	32	32	0.985	1

From the table, it is clear that the 2-Sweep heuristic provides the correct value of the diameter in most cases: hence, by performing very few iterations, one can obtain a bound which is almost always tight.

8.5.2 Probabilistic Analysis

Let us start by giving an intuition of this probabilistic analysis. First, we estimate the eccentricity of any node s: if we ignore ε and the additive constants, by Axiom 2, $\mathrm{ecc}(s) = \max_{t \in V} \mathrm{dist}(s,t) \leq \boldsymbol{\tau}_s\left(\sqrt{n}\right) + \max_{t \in V} \boldsymbol{\tau}_t\left(\sqrt{n}\right)$. Furthermore, by Axiom 3, we can find a node t maximizing $\boldsymbol{\tau}_t\left(\sqrt{n}\right)$, such that $\mathrm{dist}(s,t) \geq \boldsymbol{\tau}_s\left(\sqrt{n}\right) + \boldsymbol{\tau}_t\left(\sqrt{n}\right)$. This means that $\mathrm{ecc}(s) \approx \boldsymbol{\tau}_s\left(\sqrt{n}\right) + \max_{t \in V} \boldsymbol{\tau}_t\left(\sqrt{n}\right)$, and that the node t farthest from s maximizes $\boldsymbol{\tau}_t\left(\sqrt{n}\right)$. Using these results, we can prove that the diameter is $D = \max_{s \in V} \mathrm{ecc}(s) \approx \max_{s \in V} \boldsymbol{\tau}_s\left(\sqrt{n}\right) + \max_{t \in V} \boldsymbol{\tau}_t\left(\sqrt{n}\right) = 2 \max_{t \in V} \boldsymbol{\tau}_t\left(\sqrt{n}\right)$. Furthermore, if t is the node farthest from s, $\mathrm{ecc}(t) \approx \boldsymbol{\tau}_t\left(\sqrt{n}\right) + \max_{u \in V} \boldsymbol{\tau}_u\left(\sqrt{n}\right) \approx 2 \max_{t \in V} \boldsymbol{\tau}_t\left(\sqrt{n}\right) \approx D$. This shows that, if we ignore ε and the additive constants, we obtain that the output of the SumSweep heuristic is the exact diameter.

Let us formalize this intuition: first of all, we need to compute the eccentricity of the node t farthest from s.

Lemma 8.18. *For each node s, let t be a node maximizing the distance from s. Then, for each $x \geq \frac{1}{2}$,*

$$\boldsymbol{\tau}_t\left(n^{1-x}\right) \geq \left\lceil (1 - \mathcal{O}(\varepsilon)) \left(T\left(1 \to n^{1-x}\right) + \frac{\log n}{-\log c} - 1 - x \right) \right\rceil .$$

Proof. By Axiom 2 and Theorem 8.8,

$$\begin{aligned} &\boldsymbol{\tau}_s\left(n^{x+\varepsilon}\right) + \left\lceil (1 - \mathcal{O}(\varepsilon)) \left(T\left(1 \to n^{1-x-2\varepsilon}\right) + \frac{\log n}{-\log c} - 2 - x \right) \right\rceil \\ &\leq \mathrm{ecc}(s) \\ &= \mathrm{dist}(s,t) \\ &\leq \boldsymbol{\tau}_s\left(n^{x+\varepsilon}\right) + \boldsymbol{\tau}_t\left(n^{1-x}\right) - 1. \end{aligned}$$

From this inequality, we obtain that

$$\boldsymbol{\tau}_t\left(n^{1-x}\right) \geq \left\lceil (1 - \mathcal{O}(\varepsilon)) \left(T\left(1 \to n^{1-x}\right) + \frac{\log n}{-\log c} - 1 - x \right) \right\rceil .$$

□

By Theorems 8.8 and 8.10, if t is the node maximizing the distance from s:

$$\begin{aligned}
\mathrm{ecc}(t) &\geq \boldsymbol{\tau}_t\left(n^{\frac{1}{2}}\right) + \left\lceil (1-\mathcal{O}(\varepsilon))\left(T\left(1 \to n^{\frac{1}{2}}\right) + \frac{\log n}{-\log c} - \frac{1}{2}\right)\right\rceil - 2 \\
&\geq \left\lceil (1-\mathcal{O}(\varepsilon))\left(T\left(1 \to n^{\frac{1}{2}}\right) + \frac{\log n}{-\log c} - \frac{3}{2}\right)\right\rceil \\
&\quad + \left\lceil (1-\mathcal{O}(\varepsilon))\left(T\left(1 \to n^{\frac{1}{2}}\right) + \frac{\log n}{-\log c} - \frac{5}{2}\right)\right\rceil \\
&\geq 2\left\lfloor (1-\mathcal{O}(\varepsilon))\left(T\left(1 \to n^{\frac{1}{2}}\right) + \frac{\log n}{-\log c} - \frac{1}{2}\right)\right\rfloor - 1 \\
&\geq 2\left\lfloor (1-\mathcal{O}(\varepsilon))\frac{D}{2}\right\rfloor - 1
\end{aligned}$$

(in this analysis, we used that $T\left(1 \to n^{\frac{1}{2}}\right) + \frac{\log n}{-\log c}$ is not an integer, as stated in Section 8.1, and that ε is small enough).

We conclude that the output of the 2-Sweep heuristic is $2\left\lfloor (1-\mathcal{O}(\varepsilon))\frac{D}{2}\right\rfloor - 1$.

8.6 The 4-Sweep Heuristic

The 4-Sweep heuristic [62, 61] is a generalization of the 2-Sweep heuristic, which performs 4 BFSs instead of 2, and it should provide better approximations. It works as follows: first, we perform a 2-Sweep heuristic starting from a node s, finding the node t farthest from s, and the node u farthest from t. Then, it chooses a node s' in the middle of a tu-shortest path, and it performs another 2-Sweep heuristic from s'. At the end, it returns the maximum eccentricity found.

8.6.1 Experimental Results

In [61], the authors perform a thorough analysis of the 4-Sweep heuristic, by testing its performances on a dataset made by 196 real-world and synthetic graphs. They show that, on all but three graphs, 10 rounds of the 4-Sweep heuristic provide tight values on the diameter, and very often even a smaller number of iterations is needed. However, in [61], there is no comparison between the 2-Sweep and the 4-Sweep. Here, we perform this comparison, and consequently we further validate the efficiency of these heuristic. In order to perform a fair comparison, we chose to perform the 2-Sweep heuristic twice, so that the running time is approximately the same as the 4-Sweep heuristic (we name this procedure 2×2-Sweep).

We run 10 times each of the two heuristics, and we show in Table 8.4 the average of the relative errors obtained, on each graph in a dataset made by 26 real-world undirected networks taken from SNAP [108].

From the result, we see that both heuristics are very efficient: indeed, very often there is no error, and the error is always below 1%. However, we do not find any big difference between the behavior of the two algorithms: this seems to suggest that the choice of the starting node for the second BFS does not yield better approximations.

8.6.2 Probabilistic Analysis

In the previous section, we saw that, in practice, it is quite difficult to distinguish the performances of the 2-Sweep and of the 4-Sweep heuristic. Also in the probabilistic framework,

Table 8.4. Average of the relative error obtained by the 2×2-Sweep and the 4-Sweep heuristics, over 10 runs.

Network	2×2-Sweep	4-Sweep
as20000102	0.00%	0.00%
CA-AstroPh	0.00%	0.00%
CA-CondMat	0.00%	0.00%
ca-GrQc	0.00%	0.00%
ca-HepPh	0.00%	0.00%
ca-HepTh	0.00%	0.00%
com-amazon.all.cmty	0.00%	0.00%
com-amazon.ungraph	0.00%	0.00%
com-dblp.ungraph	0.00%	0.00%
com-lj.all.cmty	0.00%	0.31%
com-youtube.ungraph	0.00%	0.00%
email-Enron	0.00%	0.00%
facebook_combined	0.00%	0.00%
flickrEdges	0.00%	0.00%
gowalla_edges	0.00%	0.00%
loc-brightkite_edges	0.00%	0.00%
oregon1_010519	0.00%	0.00%
oregon1_010526	0.00%	0.00%
oregon2_010519	0.00%	0.00%
oregon2_010526	0.00%	0.00%
p2p-Gnutella09	0.00%	0.00%
roadNet-CA	0.72%	0.24%
roadNet-PA	0.33%	0.40%
roadNet-TX	0.08%	0.72%
soc-pokec-relationships	0.71%	0.00%
youtube-u-growth	0.00%	0.00%

there is little we can do to exploit the choice of the starting node: indeed, the only result about this heuristic is that it is at least as efficient as the 2-Sweep heuristic. The proof of lower bounds on the error obtained with this heuristic is left as a challenging open problem.

8.7 The RW Algorithm

The RW algorithm [138] is a randomized algorithm that computes a $\frac{2}{3}$-approximation of the diameter of a graph, in time $\Theta(m\sqrt{n}\log n)$. The algorithm works as follows: we choose a set $S = \{s_1, \dots, s_k\}$ made by $k = c\sqrt{n}\log n$ nodes, and we perform a BFS from each of these nodes. Then, we compute the node t maximizing $\min_{i=1,\dots,k} \operatorname{dist}(s_i, t)$, and we compute the set T made by the $h = \sqrt{n}$ nodes closest to t. If $S \cap T \neq \emptyset$, the algorithm terminates and returns the approximation $\tilde{D} = \max_{v \in S \cup T} \operatorname{ecc}(v)$. Otherwise, the algorithm fails, and, for instance, we can run the algorithm again until it succeeds (we prove that the latter case occurs with probability n^{1-c}, which can be made small by a suitable choice of c).

8.7.1 Worst-Case Analysis

In order to provide a worst-case analysis of the RW algorithm, we need to prove the following facts:

- the probability that the algorithm succeeds is at least $1 - n^{1-c}$;
- the output of the algorithm satisfies $\frac{2}{3}D \leq \tilde{D} \leq D$.

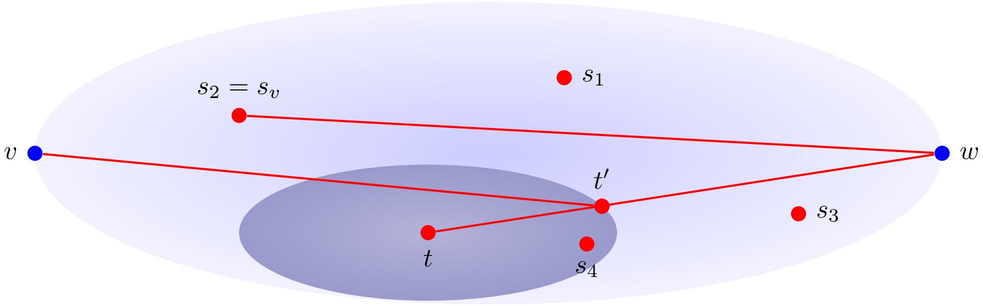


Figure 8.3. A visualization of the analysis of the RW algorithm. The red lines denote the three paths in Lemma 8.20.

Let us start with the first statement: the algorithm fails if $S \cap T = \emptyset$, that is, if there is no node in S among the h nodes closest to t. Since we do not know the node t, we prove something stronger: we bound the probability that there exists a node $v \in V$ such that there is no node in S among the h nodes closest to v.

Lemma 8.19 ([138])**.** *For each node v, let $N_h(v)$ be the set containing the h nodes closest to v. Then, the probability that a random set of k nodes $s_1, \dots, s_k$ touches $N_h(v)$ for each v is at least $1 - ne^{\frac{hk}{n}}$.*

Proof. Let us fix a node v. For each node s_i, the probability that s_i is not one of the h nodes closest to v is $1 - \frac{h}{n}$. The probability that none of the nodes s_i is among the h nodes closest to v is $\left(1 - \frac{h}{n}\right)^k \leq e^{-\frac{hk}{n}}$. A union bound over all nodes v lets us conclude. □

If we apply this lemma with $h = \sqrt{n}$, $k = c\sqrt{n}\log n$, we obtain that the probability that the algorithm succeeds is at least the probability that, for each v, $N_h(v)$ contains a node in S, which is at least $1 - e^{-\frac{hk}{n}} = 1 - ne^{-c\log n} = n^{1-c}$. This proves the first item.

For the second item, let v, w be a diametral pair (that is, v and w are nodes such that $\mathrm{dist}(v, w) = D$), let s_v be the node in S closest to v, and let t' be the last node in a shortest path from t to w that belongs to $N_h(v)$ (see Figure 8.3). We claim that either $\mathrm{dist}(s_v, w) \geq \frac{2D}{3}$, or $\mathrm{dist}(v, t') \geq \frac{2D}{3}$ (red paths in Figure 8.3).

Lemma 8.20 ([138])**.** *In the previous construction, one of the following holds:*

- $\mathrm{dist}(s_v, w) \geq \frac{2D}{3}$;
- $\mathrm{dist}(t, w) \geq \frac{2D}{3}$;
- $\mathrm{dist}(v, t') \geq \frac{2D}{3}$.

Proof. By the triangular inequality, $D = \mathrm{dist}(v, w) \leq \mathrm{dist}(v, s_v) + \mathrm{dist}(s_v, w)$. Hence, either $\mathrm{dist}(s_v, w) \geq \frac{2D}{3}$, and the result follows, or we can assume that $\mathrm{dist}(v, s_v) \geq \lceil \frac{D+1}{3} \rceil$. As a consequence, since t maximizes $\min_{i=1,\dots,k} \mathrm{dist}(t, s_i)$, we obtain that $\mathrm{dist}(t, s_i) \geq \mathrm{dist}(v, s_v) \geq \lceil \frac{D+1}{3} \rceil$ for each i, and consequently T contains all nodes at distance at least $\lceil \frac{D+1}{3} \rceil - 1$ from t, because it contains at least a node in S by assumption (green circle in Figure 8.3). If t' is the last node in a tw-shortest path, either $\mathrm{dist}(t, w) \geq \frac{2D}{3}$, and we conclude, or $\mathrm{dist}(t', w) = \mathrm{dist}(t, w) - \mathrm{dist}(t, t') \leq \lfloor \frac{2D-1}{3} \rfloor - (\lceil \frac{D+1}{3} \rceil - 1)$, and $\mathrm{dist}(t', v) \geq \mathrm{dist}(v, w) - \mathrm{dist}(t', w) \geq D - \lfloor \frac{2D-1}{3} \rfloor + \lceil \frac{D+1}{3} \rceil - 1 \geq \frac{2D}{3}$. □

This lemma implies that the value $\tilde{D} = \max_{v \in S \cup T} \mathrm{ecc}(v)$ returned by the algorithm is at least $\max(\mathrm{ecc}(s_v), \mathrm{ecc}(t), \mathrm{ecc}(t')) \geq \frac{2}{3}D$.

8.7.2 Probabilistic Analysis

If we analyze the RW algorithm on the graphs in our framework, we obtain that the running time is still $\Theta(m\sqrt{n}\log n)$, since the algorithm requires $\Theta(\sqrt{n}\log n)$ BFSs. Let us study the approximation factor. Intuitively, this algorithm is quite similar to the 2-Sweep heuristic (if k and h were 1, the algorithm would be the 2-Sweep heuristic), and we conjecture that also its behavior should be similar.

In this section, we prove an upper bound on the error obtained which matches the upper bound obtained with the 2-Sweep heuristic. Finding a lower bound on the behavior of this algorithm is left as a challenging open problem. Let v be any node: since the nodes s_i are random,

$$\mathbb{P}\left(\forall i, s_i \in \boldsymbol{N}^{\boldsymbol{\tau}_v(n^x)}(v)\right) \geq \left(1-\frac{n^x}{n}\right)^{\sqrt{n}\log n} = e^{n^{x-\frac{1}{2}+o(1)}},$$

and similarly

$$\mathbb{P}\left(\forall i, s_i \notin \boldsymbol{N}^{\boldsymbol{\tau}_v(n^x)-1}(v)\right) \leq \left(1-\frac{\mathcal{O}\left(n^x\log n\right)}{n}\right)^{\sqrt{n}\log n} = e^{n^{x-\frac{1}{2}+o(1)}},$$

because $\boldsymbol{n}^{\boldsymbol{\tau}_v(n^x)-1}(v) \leq n^x\boldsymbol{\tau}_v\left(n^x\right) = \mathcal{O}(n^x\log n)$. This means that

$$\min_{i=1,\dots,k}\operatorname{dist}(s_i,v) \leq \boldsymbol{\tau}_v\left(n^{\frac{1+\varepsilon}{2}}\right) \text{ w.h.p.,}$$

and

$$\min_{i=1,\dots,k}\operatorname{dist}(s_i,v) \geq \boldsymbol{\tau}_v\left(n^{\frac{1-\varepsilon}{2}}\right)-1 \text{ a.a.s..}$$

Hence, if v is one of the nodes maximizing $\boldsymbol{\tau}_v\left(n^{\frac{1}{2}}\right)$, a.a.s., by Lemma 8.6,

$$\begin{aligned}\min_{i=1,\dots,k}\operatorname{dist}(s_i,v) &\geq \boldsymbol{\tau}_v\left(n^{\frac{1-\varepsilon}{2}}\right)-1\\ &\geq \left\lceil(1-\mathcal{O}(\varepsilon))\left(T\left(1\to n^{\frac{1}{2}}\right)+\frac{\log n}{-\log c}-\frac{1}{2}\right)\right\rceil-1.\end{aligned}$$

This means that the node t maximizing $\min_{i=1,\dots,k}\operatorname{dist}(s_i,t)$ satisfies

$$\begin{aligned}\boldsymbol{\tau}_t\left(n^{\frac{1+\varepsilon}{2}}\right) &\geq \min_{i=1,\dots,k}\operatorname{dist}(s_i,t)\\ &\geq \min_{i=1,\dots,k}\operatorname{dist}(s_i,v)\\ &\geq \left\lceil(1-\mathcal{O}(\varepsilon))\left(T\left(1\to n^{\frac{1}{2}}\right)+\frac{\log n}{-\log c}-\frac{1}{2}\right)\right\rceil.\end{aligned}$$

We conclude that

$$\begin{aligned}\operatorname{ecc}(t) &\geq \boldsymbol{\tau}_t\left(n^{\frac{1}{2}}\right)+\left\lceil(1-\mathcal{O}(\varepsilon))\left(T\left(1\to n^{\frac{1}{2}}\right)+\frac{\log n}{-\log c}-\frac{1}{2}\right)\right\rceil-2\\ &\geq \left\lceil(1-\mathcal{O}(\varepsilon))\left(T\left(1\to n^{\frac{1}{2}}\right)+\frac{\log n}{-\log c}-\frac{1}{2}\right)\right\rceil-1\\ &\quad+\left\lceil(1-\mathcal{O}(\varepsilon))\left(T\left(1\to n^{\frac{1}{2}-\varepsilon}\right)+\frac{\log n}{-\log c}-\frac{1}{2}\right)\right\rceil-2\\ &=2\left\lceil(1-\mathcal{O}(\varepsilon))\left(T\left(1\to n^{\frac{1}{2}}\right)+\frac{\log n}{-\log c}-\frac{1}{2}\right)\right\rceil-3.\end{aligned}$$

This value is exactly the same value we obtained for the 2-Sweep heuristic.

8.8 The SumSweepHeuristic

The 2-Sweep and the 4-Sweep heuristics were further improved in this work, by defining the SumSweepHeuristic. For the experimental results, we refer to Section 4.5, where we showed that the SumSweepHeuristic outperforms in accuracy both the 2-Sweep and the 4-Sweep heuristics (and, of course, the sampling algorithm). In this section, we provide the probabilistic analysis of the SumSweepHeuristic (actually, we need a small variation to make all the proofs work).

8.8.1 Probabilistic Analysis

In order to perform our probabilistic analysis of the SumSweepHeuristic, we have to consider a small variation in the choice of the nodes $t_1, \dots, t_k$ from which the BFSs are performed. The heuristic explained before chooses t_i as the node v maximizing $S^F(v) := \sum_{t \in \{t_1, \dots, t_{i-1}\}} \operatorname{dist}(v,t)$: now, we start from a random node s_1, then we choose t_1 as the node v maximizing $\operatorname{dist}(v, s_1)$. Then, we choose again s_2 as a random node, and we choose t_2 as the node in $V - \{t_1\}$ maximizing $\operatorname{dist}(v,s_1) + \operatorname{dist}(v,s_2)$. In general, after $2i$ BFSs are performed, we choose a random node s_{i+1}, we perform a BFS from s_{i+1}, and we choose t_{i+1} as the node v in $V - \{t_1, \dots, t_i\}$ maximizing $\sum_{j=1}^{i+1} \operatorname{dist}(s_j, v)$.

The idea behind the analysis of the 2-Sweep heuristic and the RW algorithm is to exploit the existence of few nodes t with big values of $\boldsymbol{\tau}_t(n^x)$: both algorithms find a single node t such that $\boldsymbol{\tau}_t(n^x)$ is high, and they lower bound the diameter with the eccentricity of this node, which is peripheral (see Sections 8.5.2 and 8.7.2). Instead, the SumSweep heuristic highlights all nodes t_i having big values of $\boldsymbol{\tau}_t(n^x)$, and it performs a BFS from each of these nodes. Then, for each node v, if t is the node farthest from v, $\boldsymbol{\tau}_t(n^x)$ is big, and this means that $t = t_i$ for some small i (we recall that $\boldsymbol{\tau}_t(n^x)$ is the first neighborhood of t having size at least n^x, as in Definition 8.4). Consequently, if we lower bound $\operatorname{ecc}(s) \geq \max_{i=1,\dots,k} \operatorname{ecc}(t_i)$, the lower bounds obtained are tight after few steps. Let us formalize this intuition: first, we need to prove that the nodes t with high value of $\boldsymbol{\tau}_t\left(n^{1-x}\right)$ are chosen soon by this procedure.

Lemma 8.21. *Let S be a random set of nodes, let $\boldsymbol{\tau}_S(n^y) = \frac{1}{|S|}\sum_{s \in S} \boldsymbol{\tau}_s(n^y)$, and let t be any node in the graph. Then, $\frac{\sum_{s\in S}\operatorname{dist}(s,t)}{|S|} \leq \boldsymbol{\tau}_t(n^x) + \boldsymbol{\tau}_S\left(n^{1-x+\varepsilon}\right) - 1$. Furthermore, if $|S| > n^{3\varepsilon}$, $x \geq \frac{1}{2}$, $\frac{\sum_{s\in S}\operatorname{dist}(s,t)}{|S|} \geq (1-o(1))\left(\boldsymbol{\tau}_t(n^x) + \boldsymbol{\tau}_S\left(n^{1-x-2\varepsilon}\right) - 1\right)$, w.h.p..*

Proof. For the upper bound, by Axiom 2, $\sum_{s\in S}\operatorname{dist}(s,t) \leq \sum_{s \in S} \boldsymbol{\tau}_t(n^x) + \boldsymbol{\tau}_s\left(n^{1-x+\varepsilon}\right) - 1 = |S|\left(\boldsymbol{\tau}_t(n^x) + \boldsymbol{\tau}_S\left(n^{1-x+\varepsilon}\right) - 1\right)$.

For the lower bound, by Axiom 3, for each node t, the number of nodes $s \in V$ satisfying $\operatorname{dist}(s,t) < \boldsymbol{\tau}_t(n^x) + \boldsymbol{\tau}_s\left(n^{1-x-2\varepsilon}\right) - 2$ is at most $n^{1-\varepsilon}$. Let $S' \subseteq S$ be the set of nodes satisfying $\operatorname{dist}(s,t) \geq \boldsymbol{\tau}_t(n^x) + \boldsymbol{\tau}_s\left(n^{1-x-2\varepsilon}\right) - 2$: since S is random, the probability that a node $s \in S$ does not belong to S' is at least $n^{-\varepsilon}$. From this bound, we want to prove that $|S'| \geq (1 - \mathcal{O}(n^{-\varepsilon}))|S|$, using Hoeffding's inequality (Lemma A.2). For each $s \in S$, let $\boldsymbol{X}_s = 1$ if $\operatorname{dist}(s,t) \geq \boldsymbol{\tau}_t(n^x) + \boldsymbol{\tau}_s\left(n^{1-x-2\varepsilon}\right) - 2$, 0 otherwise: clearly, $|S'| = \sum_{s \in S}\boldsymbol{X}_s$, the variables $\boldsymbol{X}_s$ are independent, and $\mathbb{P}(\boldsymbol{X}_s = 1) \geq 1 - n^{-\varepsilon}$. By Hoeffding's inequality, $\mathbb{P}\left(\sum_{s\in S}\boldsymbol{X}_s < \mathbb{E}\left[\sum_{s \in S}\boldsymbol{X}_s\right] - \lambda\right) \leq e^{-\frac{\lambda^2}{|S|}}$. Since $\mathbb{E}\left[\sum_{s\in S}\boldsymbol{X}_s\right] \geq |S|(1-n^{-\varepsilon})$, if we choose $\lambda = |S|n^{-\varepsilon}$, we obtain that $\mathbb{P}\left(|S'| < (1-2n^{-\varepsilon})|S|\right) \leq e^{-|S|n^{-2\varepsilon}}$. We proved that, w.h.p., $|S'| \geq |S|(1 - \mathcal{O}(n^{-\varepsilon}))$. As a consequence:

$$\begin{aligned}\sum_{s \in S}\operatorname{dist}(s,t) &\geq \sum_{s\in S'}\operatorname{dist}(s,t) \\ &\geq \sum_{s \in S'} \boldsymbol{\tau}_t(n^x) + \boldsymbol{\tau}_s\left(n^{1-x-2\varepsilon}\right) - 2\end{aligned}$$

$$
\begin{aligned}
&\geq |S'| \left(\boldsymbol{\tau}_t\left(n^x\right) + \boldsymbol{\tau}_S\left(n^{1-x-2\varepsilon}\right) - 2\right) - \sum_{s \in S-S'} \mathcal{O}(\log n) \\
&\geq |S'| \left(\boldsymbol{\tau}_t\left(n^x\right) + \boldsymbol{\tau}_S\left(n^{1-x-2\varepsilon}\right) - 2\right) - \mathcal{O}\left(n^{-\varepsilon}|S|\log n\right) \\
&\geq \left(1 - o\left(1\right)\right)|S| \left(\boldsymbol{\tau}_t\left(n^x\right) + \boldsymbol{\tau}_S\left(n^{1-x-2\varepsilon}\right) - 2\right).
\end{aligned}
$$

□

By Lemma 8.18, if t maximizes $\mathrm{dist}(u,t)$ for some $u \in V$, then for each $y \geq \frac{1}{2}$,

$$\boldsymbol{\tau}_t\left(n^{1-y-2\varepsilon}\right) \geq \left\lceil (1-\mathcal{O}(\varepsilon))\left(T\left(1 \to n^{1-y}\right) + \frac{\log n}{-\log c} - 1 - y\right)\right\rceil.$$

If we choose $x = y = \frac{1}{2}$, the previous lemma proves that, if a node v is chosen before t in this procedure, then

$$
\begin{aligned}
\boldsymbol{\tau}_v\left(n^{\frac{1}{2}+3\varepsilon}\right) + \boldsymbol{\tau}_S\left(n^{\frac{1}{2}-2\varepsilon}\right) - 1 &\geq \frac{\sum_{s\in S}\mathrm{dist}(s,v)}{|S|} \\
&\geq \frac{\sum_{s\in S}\mathrm{dist}(s,t)}{|S|} \\
&\geq \left(1 - o\left(1\right)\right)\left(\boldsymbol{\tau}_t\left(n^{\frac{1}{2}}\right) + \boldsymbol{\tau}_S\left(n^{\frac{1}{2}-2\varepsilon}\right) - 2\right).
\end{aligned}
$$

Rearranging this inequality, we obtain

$$
\begin{aligned}
\boldsymbol{\tau}_v\left(n^{\frac{1}{2}+3\varepsilon}\right) &\geq (1-o(1))\left(\boldsymbol{\tau}_t\left(n^{\frac{1}{2}}\right) - 1\right) \\
&\geq \left\lceil (1-\mathcal{O}(\varepsilon))\left(T\left(1 \to n^{\frac{1}{2}}\right) + \frac{\log n}{-\log c} - \frac{5}{2}\right)\right\rceil \\
&= \left\lfloor (1-\mathcal{O}(\varepsilon))\left(T\left(1 \to n^{\frac{1}{2}}\right) + \frac{\log n}{-\log c} - \frac{3}{2}\right)\right\rfloor.
\end{aligned}
$$

If we apply Axiom 1 with the value of α satisfying

$$(1+\varepsilon)\left\lfloor T\left(1 \to n^{\frac{1}{2}+3\varepsilon}\right) + \alpha\right\rfloor = \left\lfloor (1-\mathcal{O}(\varepsilon))\left(T\left(1 \to n^{\frac{1}{2}}\right) + \frac{\log n}{-\log c} - \frac{3}{2}\right)\right\rfloor,$$

we obtain that the number of nodes v satisfying the latter equation is $\mathcal{O}(nc^{\alpha-x}) = nc^{\left\lfloor (1-\mathcal{O}(\varepsilon))\left(T\left(1\to n^{\frac{1}{2}}\right)+\frac{\log n}{-\log c}-\frac{3}{2}\right)\right\rfloor - T\left(1\to n^{\frac{1}{2}+3\varepsilon}\right)-\frac{1}{2}}$. If $\beta > 2$, this value is simply $\mathcal{O}(\varepsilon)$, while if $1 < \beta < 2$, this value is $n^{1-\frac{2-\beta}{\beta-1}\left(\left\lfloor\frac{\beta-1}{2-\beta}-\frac{3}{2}\right\rfloor-\frac{1}{2}\right)}$, if ε is small enough.

We conclude that, after $n^{3\varepsilon} + \mathcal{O}(nc^{\alpha-x})$ BFSes, we have performed a BFS from all nodes t that maximize $\mathrm{dist}(u,t)$ for some $u \in V$: this means that the lower bounds on all eccentricities are tight.

8.9 The iFub Algorithm

The iFub algorithm [61] is used to compute the exact value of the diameter of a graph: the time needed is $\mathcal{O}(mn)$ in the worst-case, but it can be much smaller in practice.

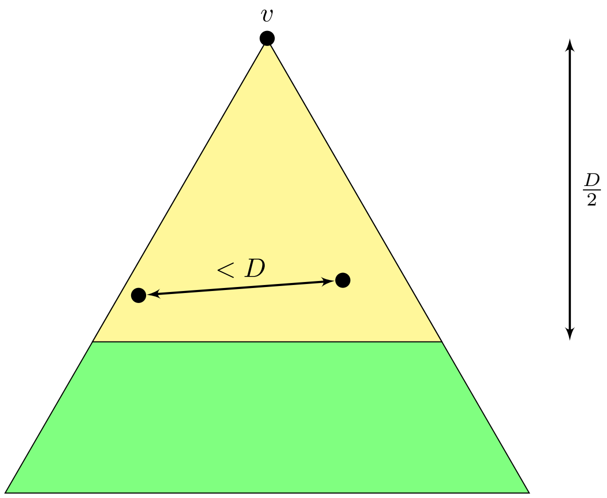


Figure 8.4. An intuition of the proof of the correctness of the iFub algorithm.

It works as follows: it performs a BFS from a node v, and it uses the fact that, if $D = \text{dist}(s,t)$, either $\text{dist}(s,v) \geq \frac{D}{2}$, or $\text{dist}(v,t) \geq \frac{D}{2}$ (that is, in Figure 8.4, at least one of s and t should be not be in the top yellow triangle). To exploit this fact, after the first BFS from v, the iFub algorithm computes the eccentricity of all the other nodes, in decreasing order of distance from v. During this process, it keeps track of the maximum eccentricity found D_L, which is a lower bound on the diameter. As soon as we are processing a node s such that $\text{dist}(v,s) \leq \frac{D_L}{2}$, we know that, for each pair (s,t) of unprocessed nodes, $\text{dist}(s,t) \leq 2\frac{D_L}{2} = D_L$: this means that we have processed at least one of the nodes in a diametral pair, and $D_L = D$.

Lemma 8.22. *The running time of the* iFub *algorithm is*

$$\mathcal{O}\left(m\left(n - \boldsymbol{n}^{\lfloor \frac{D}{2}-1 \rfloor}(v)\right)\right),$$

where $\boldsymbol{n}^{\ell}(v)$ is the number of nodes at distance at most ℓ from v.

Proof. The algorithm performs a BFS from each node at distance at least $\frac{D}{2}$ from v. □

This algorithm works only on undirected graphs, but similar counterparts were developed for directed, strongly connected graphs, and weighted graphs [63, 116]. However, in this work, we focus on the undirected case, where we can perform our probabilistic analysis.

8.9.1 Experimental Results

In [61], the authors make a very detailed analysis of the iFub algorithm, by testing it on a dataset made by 196 real-world graphs, including several real-world graphs of different kind and some synthetic graphs, all available in the LASAGNE dataset. Since giving a precise account of all these experiments goes far beyond the scope of this thesis, in this section we summarize the main results, and we refer to [61] for more details.

In order to test the iFub algorithm, Crescenzi et al. study its *performance ratio*, that is defined as the number of BFSs performed by the algorithm, divided by the total number of nodes. In Table 8.5, we provide the results obtained in [61] where the node v is chosen by selecting the node in the middle of an st-shortest path, where s and t are chosen as an approximate diametral pair computed using the 4-Sweep heuristic.

From the table, it is clear that the algorithm achieves very good performance ratios: indeed, in 81 graphs on a total of 196 graphs, it is more than 1000 times faster than the textbook algorithm, which performs a BFS from each node. Moreover, in most of the graph, the iFub algorithm is 100 times faster than the trivial algorithm.

Table 8.5. The performance ratio of the iFub algorithm on several real-world networks.

	Number	Number of networks with performance ratio p			
Network type	of networks	$p \leq 0.001$	$0.001 < p \leq 0.01$	$0.01 \leq p \leq 0.1$	$0.1 \leq p \leq 1$
Autonomous system	2	2	0	0	0
Biological	48	5	31	12	0
Citation	5	3	2	0	0
Collaboration	13	5	6	1	1
Communication	38	26	8	3	1
Peer to peer	1	0	0	0	1
Meshes and circuits	34	8	7	9	10
Product co-purchasing	4	3	1	0	0
Road	3	1	0	2	0
Social	11	9	0	2	0
Synthetic	18	7	5	2	4
Web crawls	9	0	0	0	0
Words adjacency	4	2	2	0	0
Others	6	1	2	2	1
Total	196	81	64	33	18

8.9.2 Probabilistic Analysis

In order to perform a probabilistic analysis, we only need to estimate $\boldsymbol{n}^{\left\lfloor \frac{D}{2}-1 \right\rfloor}(v)$. Intuitively, the diameter is the sum of two contributions: one is $\operatorname{dist}_{\text{avg}}(n)$, which is close to the average distance between two nodes, and the other is twice the maximum deviation from this value, that is, $2\frac{\log n}{-\log c}$. Hence, $\boldsymbol{n}^{\left\lfloor \frac{D}{2}-1 \right\rfloor}(v)$ is the number of nodes at distance at most $\frac{\operatorname{dist}_{\text{avg}}(n)}{2}+\frac{\log n}{-\log c}$ from v: if the second term is dominant (for instance, if $2 < \beta < 3$), this neighborhood covers a high percentage of nodes, and the number of nodes from which a BFS is performed is much smaller than n. Conversely, if the deviation is smaller than $\frac{\operatorname{dist}_{\text{avg}}(n)}{2}$, we expect that $n - \boldsymbol{n}^{\left\lfloor \frac{D}{2}-1 \right\rfloor}(v) = \Theta(n)$ (for instance, if $\beta > 3$ and $\eta(1)$ is small).

Let us formalize this intuition. By Theorem 8.10,

$$\frac{D}{2} \geq \frac{1}{2}\left\lfloor \left(1+\mathcal{O}(\varepsilon)\right)\left(\operatorname{dist}_{\text{avg}}(n)+\frac{2\log n}{-\log c}\right)\right\rfloor.$$

In order to estimate $n - \boldsymbol{n}^{\left\lfloor \frac{D}{2}-1 \right\rfloor}(v)$, we use the fact that if $x + y \geq 1 + \varepsilon$, $\operatorname{dist}(v,w) \leq \boldsymbol{\tau}_v(n^x) + \boldsymbol{\tau}_w(n^y) - 1$, and consequently, if $\operatorname{dist}(v,w) \geq \frac{D}{2}$, then

$$\begin{aligned}\boldsymbol{\tau}_w(n^y) &\geq \left\lceil \frac{D}{2} + 1 - \boldsymbol{\tau}_v(n^x)\right\rceil \\ &\geq \left\lfloor \frac{1}{2}\left(1+\mathcal{O}(\varepsilon)\right)\left(\operatorname{dist}_{\text{avg}}(n)+\frac{2\log n}{-\log c}\right)\right\rfloor + 1 - \boldsymbol{\tau}_v(n^x).\end{aligned}$$

Let us apply Axiom 1 with α chosen in a way that

$$(1+\varepsilon)\left(T(d \to n^y)+\alpha\right) = \left\lfloor \frac{1}{2}\left(1+\mathcal{O}(\varepsilon)\right)\left(\operatorname{dist}_{\text{avg}}(n)+\frac{2\log n}{-\log c}\right)\right\rfloor + 1 - \boldsymbol{\tau}_v(n^x).$$

We obtain that $n - \boldsymbol{n}^{\left\lfloor \frac{D}{2}-1 \right\rfloor}(v)$ is at most

$$\begin{aligned} nc^{\alpha-y} &= n^{1-(1+\mathcal{O}(\varepsilon))\frac{-\log c}{\log n}\left(\left\lfloor\frac{1}{2}\left(1+\mathcal{O}(\varepsilon)\right)\left(\operatorname{dist}_{\text{avg}}(n)+\frac{2\log n}{-\log c}\right)\right\rfloor+1-\boldsymbol{\tau}_v(n^x)-T(d\to n^y)\right)} \\ &\leq n^{1-(1+\mathcal{O}(\varepsilon))\frac{-\log c}{\log n}\left(\left\lfloor\frac{1}{2}\left(\operatorname{dist}_{\text{avg}}(n)+\frac{2\log n}{-\log c}\right)\right\rfloor-\operatorname{dist}_{\text{avg}}(n)+T\left(1\to n^{\max\left(1,\frac{1}{\beta-1}\right)}\right)\right)}.\end{aligned}$$

For $\beta > 3$, this value is

$$\begin{aligned}
&n^{1-(1+\mathcal{O}(\varepsilon))\frac{-\log c}{\log n}\left(\frac{1}{2}\operatorname{dist}_{\text{avg}}(n)+\frac{\log n}{-\log c}-\operatorname{dist}_{\text{avg}}(n)+T\left(1\to n^{\frac{1}{\beta-1}}\right)\right)}\\
&=n^{(1+\mathcal{O}(\varepsilon))\frac{-\log c}{\log n}\left(\frac{1}{2}\operatorname{dist}_{\text{avg}}(n)-T\left(1\to n^{\frac{1}{\beta-1}}\right)\right)}\\
&=n^{(1+\mathcal{O}(\varepsilon))\frac{-\log \eta(1)}{\log n}\left(\frac{1}{2}-\frac{1}{\beta-1}\right)\frac{\log n}{\log M_1(\mu)}}\\
&=n^{\left(\frac{1}{2}-\frac{1}{\beta-1}+\mathcal{O}(\varepsilon)\right)\frac{-\log \eta(1)}{\log M_1(\mu)}}.
\end{aligned}$$

Hence, the running time is

$$\mathcal{O}\left(m\left(n-\boldsymbol{n}^{\left\lfloor\frac{D}{2}-1\right\rfloor}(v)\right)\right)=n^{1+\left(\frac{1}{2}-\frac{1}{\beta-1}+\mathcal{O}(\varepsilon)\right)\frac{-\log \eta(1)}{\log M_1(\mu)}}.$$

For $2 < \beta < 3$, the computation is similar, but $M_1(\mu)$ is infinite: the running time is $n^{1+\mathcal{O}(\varepsilon)}$.

Finally, for $1 < \beta < 2$, if v is the maximum degree node, this value is at most

$$n^{1-(1+\mathcal{O}(\varepsilon))\frac{2-\beta}{\beta-1}\left\lfloor\frac{3}{2}+\frac{\beta-1}{2-\beta}-2\right\rfloor}=n^{1+\mathcal{O}(\varepsilon)-\frac{2-\beta}{\beta-1}\left\lfloor\frac{\beta-1}{2-\beta}-\frac{1}{2}\right\rfloor}.$$

The running time is $mn^{1-\frac{2-\beta}{\beta-1}\left\lfloor\frac{\beta-1}{2-\beta}-\frac{1}{2}\right\rfloor+\mathcal{O}(\varepsilon)}$.

8.10 The ExactSumSweep Algorithm

In this thesis, we provided a new algorithm to compute the diameter and the radius of a graph: the ExactSumSweep algorithm. We experimentally showed in Section 4.5 that the ExactSumSweep algorithm outperforms the iFub algorithm, and we also observed that the ExactSumSweep algorithm is more robust, because it performs better on the "hard" instances. In this section, we show that this behavior can be analytically proved through our axioms.

8.10.1 Probabilistic Analysis

Also in the analysis of the ExactSumSweep algorithm, we need two slight modifications: first, we assume that the initial SumSweepHeuristic is performed as in Section 8.8. Furthermore, only in the case $\beta > 3$, we need to assume that we perform a BFS from a node maximizing the degree every k steps, for some constant k.

The analysis for the radius computation is very simple: after the initial SumSweepHeuristic, all lower bounds are tight w.h.p. by the results in Section 8.8, and consequently it is enough to perform a further BFS from a node minimizing L to obtain the final value of R_U. Then, the running time is the same as the running time of the SumSweepHeuristic. For the diameter, the analysis is more complicated, because we have to check when all upper bounds are below the diameter, and the upper bounds are not tight, in general.

Intuitively, if $\beta < 3$, the radius is very close to half the diameter, and the first BFS is performed from a radial node s: consequently, after the first BFS, the upper bound of a node v becomes $\operatorname{ecc}(s) + \operatorname{dist}(s, v) \leq D$ if $\operatorname{dist}(s, v) \leq D - \operatorname{ecc}(s) = D - R \approx R = \operatorname{ecc}(s)$. This means that, after this BFS, we have to perform a BFS from each node whose distance from s is approximately the eccentricity of s, and there are not many such nodes, as shown by Lemma 8.18. Hence, we obtain that, in this regime, the running time of the ExactSumSweep algorithm is the same as the running time of the initial SumSweepHeuristic. Conversely, if $\beta > 3$, a BFS from a node s sets upper bounds smaller than D to all nodes closer to s than $D - \operatorname{ecc}(s)$, and the number of such nodes is close to $M_1(\mu)^{D-\operatorname{ecc}(s)}$. Since $D - \operatorname{ecc}(s)$ is usually $\mathcal{O}(\log n)$, a BFS sets correct bounds to $M_1(\mu)^{\mathcal{O}(\log n)} = n^{\mathcal{O}(1)}$ nodes: hence, we expect the number of BFSes needed to be subquadratic.

The Case $\beta < 3$

As we said before, the first BFS is performed from a radial node s: by Theorem 8.8, if $\overline{s}$ is a node maximizing $\boldsymbol{\tau}_s\left(n^x\right)$,

$$\operatorname{ecc}(s) \leq \operatorname{ecc}(\overline{s}) \leq \boldsymbol{\tau}_{\overline{s}}\left(n^x\right) + \left\lfloor (1+\varepsilon)\left(T\left(1 \to n^{1-x}\right) + \frac{\log n}{-\log c} - x\right)\right\rfloor.$$

Let $x := 1-\varepsilon$, if $1 < \beta < 2$, $x := \frac{1}{\beta-1} - \varepsilon$ if $2 < \beta < 3$: this value is at most

$$\left\lfloor (1+2\varepsilon)\left(2 + \frac{\log n}{-\log c}\right)\right\rfloor.$$

As a consequence, after the first BFS, the algorithm sets upper bounds smaller than D to any node closer to s than

$$\begin{aligned} D - \operatorname{ecc}(s) &\geq \left\lfloor (1-\mathcal{O}(\varepsilon))\left(\operatorname{dist}_{\text{avg}}(n) + \frac{2\log n}{-\log c}\right)\right\rfloor - \left\lfloor (1+2\varepsilon)\left(2 + \frac{\log n}{-\log c}\right)\right\rfloor \\ &\geq (1-\mathcal{O}(\varepsilon))\left(\left\lfloor \frac{2\log n}{-\log c}\right\rfloor - \left\lfloor \frac{\log n}{-\log c}\right\rfloor + 1\right). \end{aligned}$$

This means that we only have to analyze nodes v such that $D - \operatorname{ecc}(s) \leq \operatorname{dist}(s,v) \leq \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_v\left(n^{1-x+\varepsilon}\right) - 1$, that is:

$$\begin{aligned} \boldsymbol{\tau}_v\left(n^{1-x+\varepsilon}\right) &\geq D - \operatorname{ecc}(s) - \boldsymbol{\tau}_s\left(n^x\right) + 1 \\ &\geq (1-\mathcal{O}(\varepsilon))\left(\left\lfloor \frac{2\log n}{-\log c}\right\rfloor - \left\lfloor \frac{\log n}{-\log c}\right\rfloor + 1\right). \end{aligned}$$

Let us apply Axiom 1 with α such that

$$(1+\varepsilon)\left(T\left(1 \to n^{1-x+\varepsilon}\right) + \alpha\right) = (1-\mathcal{O}(\varepsilon))\left(\left\lfloor \frac{2\log n}{-\log c}\right\rfloor - \left\lfloor \frac{\log n}{-\log c}\right\rfloor + 1\right).$$

We obtain that the number of nodes v that do not receive bounds smaller than D is at most

$$\begin{aligned} nc^{\alpha-1+x-\varepsilon} &= nc^{(1-\mathcal{O}(\varepsilon))\left(\left\lfloor \frac{2\log n}{-\log c}\right\rfloor - \left\lfloor \frac{\log n}{-\log c}\right\rfloor + 1\right) - T\left(1\to n^{1-x+\varepsilon}\right) - 1 + x - \varepsilon} \\ &= nc^{\alpha-1+x-\varepsilon} \\ &= nc^{(1-\mathcal{O}(\varepsilon))\left(\left\lfloor \frac{2\log n}{-\log c}\right\rfloor - \left\lfloor \frac{\log n}{-\log c}\right\rfloor\right) - T\left(1\to n^{1-x+\varepsilon}\right) - 1}, \end{aligned}$$

which is smaller than the number of iteration of the SumSweepHeuristic. Hence, the total running time is bounded by the time needed to perform the initial SumSweepHeuristic.

The Case $\beta > 3$

In the case $\beta > 3$, the previous argument does not work, because $\operatorname{dist}_{\text{avg}}(n)$ can be small. We need a different technique: we prove that, for each node v and for some x, either $\boldsymbol{\tau}_v\left(n^x\right)$ is quite large, or there is a node s with high degree that is "not far" from s. After $\mathcal{O}(l)$ steps, we have performed a BFS from the l nodes with highest degree, and consequently all nodes which are quite close to one of these nodes have bounds smaller than D: this means that there are few nodes with upper bound bigger then D. Then, since every $\mathcal{O}(1)$ steps, the number of nodes with upper bound bigger than D decreases by 1, after few more BFSes, all upper bounds are smaller than or equal to D.

More formally, let $s_1, \dots, s_k$ be all the nodes with degree bigger than n^x: by Axiom 4, $k = \frac{n^{1 \pm \mathcal{O}(\varepsilon)}}{n^{x(\beta-1)}}$, and after at most $\frac{n^{1+\mathcal{O}(\varepsilon)}}{n^{x(\beta-1)}}$ BFSes (apart from the initial SumSweepHeuristic), we have performed a BFS from each of these nodes.

We start by estimating $\mathrm{ecc}(s_i)$, because, after the BFS from s_i, for each node v, $U(v) \leq \mathrm{dist}(v, s_i) + \mathrm{ecc}(s_i)$. By Theorem 8.8,

$$\begin{aligned} \mathrm{ecc}(s_i) &\leq \boldsymbol{\tau}_{s_i}(n^x) + \left\lfloor (1+\varepsilon)\left(T\left(1 \to n^{1-x}\right) + \frac{\log n}{-\log c} - x \right) \right\rfloor \\ &\leq 1 + (1+\varepsilon)\left((1-x)\frac{\log n}{\log M_1(\mu)} + \frac{\log n}{-\log c} \right) \\ &\leq (1+2\varepsilon)\left((1-x)\frac{\log n}{\log M_1(\mu)} + \frac{\log n}{-\log c} \right). \end{aligned}$$

Hence, after the BFS from node s_i, the upper bound of any node v is smaller than $\mathrm{dist}(v, s_i) + (1+2\varepsilon)\left((1-x)\frac{\log n}{\log M_1(\mu)} + \frac{\log n}{-\log c} \right)$, which is smaller than D if $\mathrm{dist}(v, s_i) \leq D - (1+2\varepsilon)\left((1-x)\frac{\log n}{\log M_1(\mu)} + \frac{\log n}{-\log c} \right) = (1+4\varepsilon)\left(\frac{x \log n}{\log M_1(\mu)} + \frac{\log n}{-\log c} \right)$ by Theorem 8.10.

Now, we want to compute the number of nodes that are at distance at least $(1+4\varepsilon)\left(\frac{x \log n}{\log M_1(\mu)} + \frac{\log n}{-\log c} \right)$ from each s_i. To estimate this quantity, we use Lemma A.54, which does not follow directly from the main axioms. This lemma says that, for each node v, $\min_{i=1,\dots,k} \mathrm{dist}(v, s_i) \leq \boldsymbol{\tau}_v\left(n^{x(\beta-2)+\varepsilon}\right)$: hence, after a BFS from each node s_i has been performed, the upper bound of v is at most D if $\boldsymbol{\tau}_v\left(n^{x(\beta-2)+\varepsilon}\right) \leq (1+4\varepsilon)\left(\frac{x \log n}{\log M_1(\mu)} + \frac{\log n}{-\log c} \right)$. We conclude that, after $n^{1+\varepsilon-x(\beta-1)}$ BFSes, only nodes satisfying $\boldsymbol{\tau}_v\left(n^{x(\beta-2)+\varepsilon}\right) > (1+4\varepsilon)\left(\frac{x \log n}{\log M_1(\mu)} + \frac{\log n}{-\log c} \right)$ have upper bounds bigger than D.

By Axiom 1, the number of nodes that satisfy the latter inequality is at most

$$\begin{aligned} \mathcal{O}\left(nc^{(1-\mathcal{O}(\varepsilon))\left(\frac{x \log n}{\log M_1(\mu)} + \frac{\log n}{-\log c} - \frac{x(\beta-2)\log n}{\log M_1(\mu)} \right)} \right) &= n^{1 - \frac{-\log c}{\log n}\left(\frac{\log n}{-\log c} - \frac{x(\beta-3)\log n}{\log M_1(\mu)} \right) + \mathcal{O}(\varepsilon)} \\ &= n^{\frac{x(\beta-3)(-\log c)}{\log M_1(\mu)} + \mathcal{O}(\varepsilon)} \end{aligned}$$

Hence, by performing $\mathcal{O}\left(n^{\frac{x(\beta-3)(-\log c)}{\log M_1(\mu)} + \mathcal{O}(\varepsilon)} \right)$ more BFSes, the algorithm terminates.

We conclude that the total number of BFSes is at most

$$\max\left(n^{\frac{x(\beta-3)(-\log c)}{\log M_1(\mu)} + \mathcal{O}(\varepsilon)}, n^{1-x(\beta-1)} \right).$$

If we substitute

$$x = \frac{1}{\beta - 1 + (\beta-3)\frac{-\log c}{\log M_1(\mu)}},$$

we obtain

$$n^{\frac{1}{1+\frac{\beta-1}{\beta-3}\frac{\log M_1(\mu)}{\log c}} + \mathcal{O}(\varepsilon)}.$$

Then, the running time is at most

$$n^{1+\frac{1}{1+\frac{\beta-1}{\beta-3}\frac{\log M_1(\mu)}{\log c}} + \mathcal{O}(\varepsilon)}.$$

8.11 The BCM Algorithm

In Chapter 5, we designed a new algorithm to compute the k most central nodes according to closeness centrality. The algorithm came in four different variations: in this section, we consider the simplest variation, that analyzes the nodes in decreasing order of degree, and, for each node s, performs a `updateBoundsBFSCut` function, that is, a BFS which is stopped as soon as we can guarantee that s is not in the top-k. In Section 5.7, we showed that this algorithm is much more efficient than the *textbook* algorithm that performs a BFS from each node. In this section, we partially confirm this result, by showing that the new algorithm is efficient in the regimes $1 < \beta < 2$ and $\beta > 3$. In the regime $2 < \beta < 3$, a strange phenomenon happens: we can prove that the algorithm is not efficient, because it needs quadratic time in the input size. However, the proof is based on the fact that $\log\log n$ tends to infinity, where n is the number of nodes: this behavior is not reflected in practice, because n is always less than 1 billion, and consequently $\log\log n$ is less than 4.

8.11.1 Probabilistic Analysis

First, let us provide a deterministic bound on the running time. The idea behind this bound is that a BFS from s visits all nodes at distance at most $f_k - 2$, then it might find a lower bound bigger than f_k, where f_k is the k-th smallest farness. In particular, if the number of nodes visited at distance at most $f_k - 2$ is much smaller than n, the bound is likely to be sufficient to stop the visit. Otherwise, the BFS from s has already visited $\Theta(n)$ nodes: in both cases, the number of visited nodes is close to the number of nodes at distance at most $f_k - 2$ from s.

Lemma 8.23. *Let f_k be the k-th smallest farness among the k nodes with highest degree, and let $\ell \geq (1+\alpha)f_k - 2$ be an integer. Then, the running time of the algorithm is at most*

$$\mathcal{O}\left(m + \frac{1}{\alpha}\sum_{s \in V}\sum_{v \in \boldsymbol{N}^\ell(s)} \deg(v)\right).$$

Proof. The first k BFSes need time $\mathcal{O}(m)$, because they cannot be cut. For all subsequent BFSes, the k-th smallest farness found is at least f_k. Let us consider a BFS from s, and let us assume that we have visited all nodes up to distance ℓ: our lower bound on the farness of s is

$$\begin{aligned} f(s) &\geq \frac{1}{n-1}\left(\sum_{v \in \boldsymbol{N}^\ell(s)} \operatorname{dist}(s,v) + (\ell+1)\gamma_U^\ell(s) + (\ell+2)\left(n - |\boldsymbol{N}^\ell(s)| - \gamma_U^\ell(s)\right)\right) \\ &\geq \frac{\ell+2}{n-1}\left(n - |\boldsymbol{N}^\ell(s)| - \gamma_U^\ell(s)\right) \\ &\geq \frac{\ell+2}{n-1}\left(n - \sum_{v \in \boldsymbol{N}^\ell(s)} \deg(v)\right) \\ &\geq (1+\alpha)\frac{f_k}{n}\left(n - \sum_{v \in \boldsymbol{N}^\ell(s)} \deg(v)\right). \end{aligned}$$

We claim that the BFS from s visits at most $\frac{m}{n\alpha}\sum_{v \in \boldsymbol{N}^\ell(s)} \deg(v)$ edges: this is trivially true if $\sum_{v \in \boldsymbol{N}^\ell(s)} \deg(v) > \frac{\alpha}{2}n$, because the value becomes $\mathcal{O}(m)$, while if $\sum_{v \in \boldsymbol{N}^\ell(s)} \deg(v) < \frac{\alpha}{2}n$, the lower bound is at least $(1+\alpha)\, f_k \left(1 - \frac{\alpha}{2}\right) \geq f_k$, and the BFS is stopped after ℓ steps. □

Lemma 8.24. *Let f_k be the k-th smallest farness, and let $\ell \leq f_k - 2$. Then, the running time of the algorithm is $\Omega\left(\sum_{s \in V} \boldsymbol{n}^\ell(s)\right)$.*

Proof. Clearly, at any step, the k-th minimum farness found is at least f_k. We want to prove that the BFS from s reaches level ℓ: indeed, the lower bound on the farness of s is

$$\frac{1}{n-1}\left(\sum_{v \in \boldsymbol{N}^\ell(s)} \mathrm{dist}(s,v) + (\ell+1)\gamma_U^\ell(v) + (\ell+2)\left(n - |\boldsymbol{N}^\ell(s)| - \gamma_U^\ell(s)\right)\right) \leq \ell + 2 \leq f_k.$$

Hence, the BFS is not cut until level ℓ, and at least $\Omega\left(\sum_{s \in V} \boldsymbol{n}^\ell(s)\right)$ nodes are visited. □

We now need to compute these values in graphs in our framework. We analyze separately the running time in the case $1 < \beta < 2$, in the case $2 < \beta < 3$, and in the case $\beta > 3$.

The Case $1 < \beta < 2$

By Axiom 4, if s is one of the k nodes with maximum degree, then $\deg(s) > 2cn$ for some c only depending on k, and by Theorem 8.12, the farness of s is at most $(2+o(1))n - \deg(s) \leq (2-c)n$ if n is big enough. By Lemma 8.23 applied with $\ell = 0$ and $\alpha = c$, the running time is at most

$$\mathcal{O}\left(m + \frac{1}{c}\sum_{s \in V}\sum_{t \in \boldsymbol{N}^0(s)} \deg(t)\right) = \mathcal{O}\left(m + \sum_{s \in V} \deg(s)\right) = \mathcal{O}(m).$$

The Case $2 < \beta < 3$

In this case, we know by Theorem 8.13 applied with $x = \frac{1}{2}$ that the k-th minimum farness is at least $(1+o(1))\left(\boldsymbol{\tau}_s(n^x) - T(1 \to n^x) + \mathrm{dist}_{\mathrm{avg}}(n) - 1\right) = n\left(\frac{1}{2} + o(1)\right)\mathrm{dist}_{\mathrm{avg}}(n) = (1 + o(1))\log_{\frac{1}{\beta-2}} \log n = \Theta(\log\log n)$.

We claim that all nodes with degree at least n^x are at distance $\mathcal{O}(1)$ from each other: indeed, if $\deg(s), \deg(t) > n^x$, by Axiom 1,

$$\boldsymbol{\tau}_s\left(n^{\frac{2}{3}}\right), \boldsymbol{\tau}_t\left(n^{\frac{2}{3}}\right) = \mathcal{O}(1),$$

and by Axiom 2,

$$\mathrm{dist}(s,t) < \boldsymbol{\tau}_s\left(n^{\frac{2}{3}}\right) + \boldsymbol{\tau}_t\left(n^{\frac{2}{3}}\right) = \mathcal{O}(1).$$

By Lemma 8.24, if $\ell + 2$ is smaller than f_k, the running time is

$$\begin{aligned}
\Omega\left(\sum_{s \in V} \boldsymbol{n}^\ell(v)\right) &= \Omega\left(\sum_{\deg(s) > n^x} \boldsymbol{n}^{\Theta(\log\log n)}(s)\right)\\
&= \Omega\left(\sum_{\deg(s) > n^x} |\{t : \deg(t) > n^x\}|\right)\\
&\geq \Omega\left(\left(n^{1-x(\beta-1)}\right)^2\right)\\
&= \Omega\left(n^{2-2x(\beta-1)}\right).
\end{aligned}$$

If we choose $x = \varepsilon$, the running time is $\Omega\left(n^{2-\mathcal{O}(\varepsilon)}\right)$.

The Case $\beta > 3$

Let us estimate the farness of the k nodes with highest degree. By Axiom 4, the k maximum degrees are

$$\Theta\left(n^{\frac{1}{\beta-1}}\right),$$

and their farness is

$$\begin{aligned}&(1+\mathcal{O}(\varepsilon))\left(\boldsymbol{\tau}_s\left(n^{\frac{1}{\beta-1}}\right)-T\left(1\to n^{\frac{1}{\beta-1}}\right)+\mathrm{dist}_{\mathrm{avg}}(n)\right)\leq\\&n(1+\mathcal{O}(\varepsilon))\log_{M_1(\mu)}\left(n^{1-\frac{1}{\beta-1}}\right).\end{aligned}$$

Hence, by Lemma 8.24 applied with

$$\ell=(1+\mathcal{O}(\varepsilon))\log_{M_1(\mu)}\left(n^{1-\frac{1}{\beta-1}}\right),$$

we obtain that the running time is

$$\mathcal{O}\left(\sum_{s\in V}\sum_{v\in\boldsymbol{N}^{\ell}(s)}\deg(v)\right)=\mathcal{O}\left(\sum_{s\in V}\sum_{v\in\boldsymbol{N}^{\ell}(s)}\rho_v\right)=\mathcal{O}\left(n^{\varepsilon}\sum_{s\in V}\boldsymbol{n}^{\ell+1}(v)\right),$$

because $\deg(v)\leq n^{\varepsilon}\rho_v$, and $\boldsymbol{\gamma}^{\ell+1}(v)>\rho_{\boldsymbol{\Gamma}^{\ell}(v)}$ in random graphs, as shown in Lemmas A.11 and A.16. For the lower bound, we use two partial results that we obtain in the proof of the main theorems: Corollaries A.15 and A.20. These corollaries say that $\boldsymbol{\tau}_s(n^y)-\boldsymbol{\tau}_s(n^x)\geq(1-\varepsilon)(y-x)\log_{M_1(\mu)}n$, for each $\frac{1}{\beta-1}<x<y<1$. Hence, for each node s,

$$\begin{aligned}f(s)&\geq\frac{n-n^{1-\varepsilon}}{n-1}\boldsymbol{\tau}_s\left(n^{1-\varepsilon}\right)\\&\geq\left(1-\frac{1}{\beta-1}-\mathcal{O}(\varepsilon)\right)\log_{M_1(\mu)}n\\&\geq(1-\mathcal{O}(\varepsilon))\log_{M_1(\mu)}\left(n^{1-\frac{1}{\beta-1}}\right).\end{aligned}$$

We conclude that, by Lemmas 8.23 and 8.24, the running time of the algorithm is $\sum_{s\in V}\boldsymbol{n}^{\ell}(s)$, where

$$\ell=(1\pm\mathcal{O}(\varepsilon))\log_{M_1(\mu)}\left(n^{1-\frac{1}{\beta-1}}\right).$$

To estimate this value, we use Theorem A.9: for each node s,

$$\boldsymbol{\tau}_s(n^y)\leq\boldsymbol{\tau}_s\left(n^{\varepsilon}\deg(s)\right)+(1+\varepsilon)\log_{M_1(\mu)}n^{y-x}\leq(1+\mathcal{O}(\varepsilon))\log_{M_1(\mu)}\frac{n^y}{\deg(s)}\leq\ell$$

if $y=1-\frac{1}{\beta-1}+\frac{\log\deg(s)}{\log n}+\mathcal{O}(\varepsilon)$, and consequently $\boldsymbol{\gamma}^i(s)<n^y$ for each $i<\ell$. Hence, the running time is smaller than

$$n^{\mathcal{O}(\varepsilon)}\sum_{s\in V}\boldsymbol{n}^{\ell}(s)\leq n^{\mathcal{O}(\varepsilon)}\sum_{s\in V}\ell n^y\leq n^{1-\frac{1}{\beta-1}+\mathcal{O}(\varepsilon)}\sum_{s\in V}\deg(s)\leq n^{2-\frac{1}{\beta-1}+\mathcal{O}(\varepsilon)}.$$

An analogous argument proves that the running time is at least $n^{2-\frac{1}{\beta-1}-\mathcal{O}(\varepsilon)}$.

8.12 The AIY Distance Oracle

In [8], Akiba et al. describe a distance oracle, that is, an algorithm to speed-up the computation of distances in a graph. The algorithm consists of two separate phases: a preprocessing phase, where the algorithm computes additional quantities that describe distances in the graph, and a query phase, where the user asks for a distance, and the oracle should answer as fast as possible.

There are two trivial approaches to the design of a distance oracle: the first simply skips the preprocessing phase, and performs a complete BFS to answer each query; the second approach precomputes the distance between all pairs of nodes.

Both these trivial approaches have some disadvantages: the first approach does not need any space apart from storing the graph, but the query time is $\mathcal{O}(m)$, which is quite high; the second approach has $\mathcal{O}(1)$ query time, but it needs $\mathcal{O}(n^2)$ space, which might be too large. For this reason, researchers developed several *distance oracles*, that is, algorithms that offer different trade-offs between query time and space occupancy.

Since it is quite hard to design an exact distance oracle, a large amount of research has focused on designing approximate distance oracles, starting from the seminal paper [157]. In this paper, Thorup and Zwick design a distance oracle with $2k-1$ *stretch*, that is, such that, when the user asks for $\operatorname{dist}(s,t)$, the value ℓ returned satisfies $\operatorname{dist}(s,t) \leq \ell \leq (2k-1)\operatorname{dist}(s,t)$. The preprocessing time is $\mathcal{O}(kmn^{\frac{1}{k}})$, the space needed is $\mathcal{O}(kn^{1+\frac{1}{k}})$ space, and the query time is $\mathcal{O}(k)$. They also prove that the space requirement is almost optimal, assuming a girth conjecture by Erdös [74]. As a corollary, if this conjecture is true, then every distance oracle with stretch at most 3 needs at space $\Omega(n^2)$ in the worst-case (note that the trivial approach of performing a BFS at each query needs space $\Omega(n^2)$ if the graph has $\Omega(n^2)$ edges). For more worst-case results, we refer to [130] for the dense case and to [149] for the sparse case.

A different approach is to design distance oracles that work well on real-world graphs, even if there is no guarantee in the worst-case. Among these approaches, a lot of works have focused on the computation of shortest paths in road networks, because of practical applications [141, 5, 67, 66]. Among these works, one of the most successful approaches is hub-labeling: we assign a label $L(s)$ to each node s, consisting of a set of pairs $(v, \operatorname{dist}(s,v))$, where v is a node in the graph, and $\operatorname{dist}(s,v)$ is the distance between s and v. The construction of these labels enforces the so-called 2-hop cover property: for each pair of nodes s,t, there is a node v in a shortest path between s and t that belongs both to $L(s)$ and to $L(t)$. Using the 2-hop cover property, it is possible to compute $\operatorname{dist}(s,t) = \min_{v \in L(s) \cap L(t)} \operatorname{dist}(s,v) + \operatorname{dist}(v,t)$, in time $\mathcal{O}(|L(s)| + |L(t)|)$. The space required is $\Theta\left(\sum_{s \in V} |L(s)|\right)$.

In this framework, the main challenge is to design a preprocessing phase that generates small labels: in [5, 67, 66], several strategies are proposed to create small labels in road networks. As far as we know, the only work that addresses this problem on real-world networks is [8], which suggests a very simple but powerful strategy to generate small labels that satisfy the 2-hop cover property: in this section, we define and analyze this algorithm.

First, we sort all nodes s obtaining $s_1 < s_2 < \cdots < s_n$ (any order is fine), and we perform a BFS from each node, following this order. During the BFS from s_i, as soon as we visit a node t, we add $(s_i, \operatorname{dist}(s_i,t))$ to the label of t (in this case, we say with abuse of notation that s_i belongs to the label of t). Furthermore, we prune each BFS at each node t such that $\operatorname{dist}(s,t) = \min_{v \in L(s) \cap L(t)} \operatorname{dist}(s,v) + \operatorname{dist}(v,t)$, where $L(s)$ and $L(t)$ are the current labels. The 2-hop cover property is ensured by the following lemma and its corollary.

Lemma 8.25 ([8])**.** *Let L be the labels generated by this procedure. Then, $s \in L(t)$ if and only if there is no shortest path between s and t containing a node $v < s$ in the initial ordering of nodes.*

Proof. Let L_s be the labels obtained when we are performing the BFS from s, in the preprocessing phase. We prove by induction that, after we perform a BFS from a node s, for each $s' < s$ and for each $t \in V$, $s' \in L(t)$ if and only if there is no shortest path between s and t

Table 8.6. Performance comparison between the AIY algorithm and the trivial algorithm of performing a BFS at each query.

Graph	n	m	Preproc.	Space	Av. label	Query	BFS query
			s	MB		μs	μs
p2p-Gnutella31	63 K	148 K	54	209	660	5.2	3 200
soc-Epinions1	76 K	509 K	1.7	32	49	0.5	7 400
soc-sign-Slashdot	82 K	948 K	6.0	64	68	0.8	12 000
web-NotreDame	326 K	1.5 M	4.5	138	50	0.5	17 000
wiki-Talk	2.4 M	4.7 M	61	1 000	50	0.6	197 000
as-Skitter	1.7 M	11 M	359	2 700	187	2.3	190 000
in-2004	1.4 M	17 M	173	2 300	197	1.6	150 000
MetroSec	2.3 M	22 M	108	2 500	83	0.7	150 000
Flickr	1.8 M	23 M	866	4 000	311	2.6	361 000
holliwood2009	1.1 M	114 M	15 164	12 000	2 162	15.6	1 200 000
indochina2004	7.4 M	194 M	6 068	22 000	479	4.1	1 500 000

containing a node $v < s$. The base case is trivial, since there is no $s' < s$. For induction step, let us prove the two directions separately.

- If there is a shortest path from s to t containing a node $v < s$, we can suppose without loss of generality that v is the smallest such node: by inductive hypothesis, $v \in L(s) \cap L(t)$, because there is no node smaller than v in a shortest path from v to s and from v to t. Hence, since $\text{dist}(s,t) = \text{dist}(s,v) + \text{dist}(v,t)$, s is not added to the label of t.

- Otherwise, let us assume for a contradiction that $s \notin L(t)$ and there is no node $v < s$ in an st-shortest path. Let t' be the first node in an st-shortest path such that $s \notin L(t')$. Since $s \notin L(t')$, but t' was visited by the BFS, it means that $\min_{v \in L_s(s) \cap L_s(t')} \text{dist}(s,v) + \text{dist}(v,t') = \text{dist}(s,t')$, and consequently there is a node $v < s$ in a shortest path from s to t', against the assumptions.

□

Corollary 8.26 ([8]). *The labels generated by this procedure satisfy the 2-hop cover property.*

Proof. Let us consider two nodes s and t, and let v be the smallest node in an st-shortest path (v might also be s or t). By definition of v, all the nodes in a shortest path from s to v and all the nodes in a shortest path from v to t are bigger than v: by Lemma 8.25, v belongs to the labels of s and t. □

It remains to define how nodes are sorted: in [8], it is suggested to sort them in order of degree (tie-breaks are solved arbitrarily).

8.12.1 Experimental Results

In this section, we report the experimental results performed in [8]. The authors analyzed a dataset made by 11 real-world networks, taken from the SNAP dataset [108] and from the WebGraph dataset [23] . We report the tradeoff between query time and space occupancy in Table 8.6, and the label size of some graphs in Figure 8.5. The data are taken from [8].

From the results, it is clear that the average label size is very small, compared to the number of nodes, with improvements up to a factor 48 000 in the graphs `wiki-Talk`. Furthermore, they show that most of the labels have similar sizes, and this means that the algorithm is quite robust.

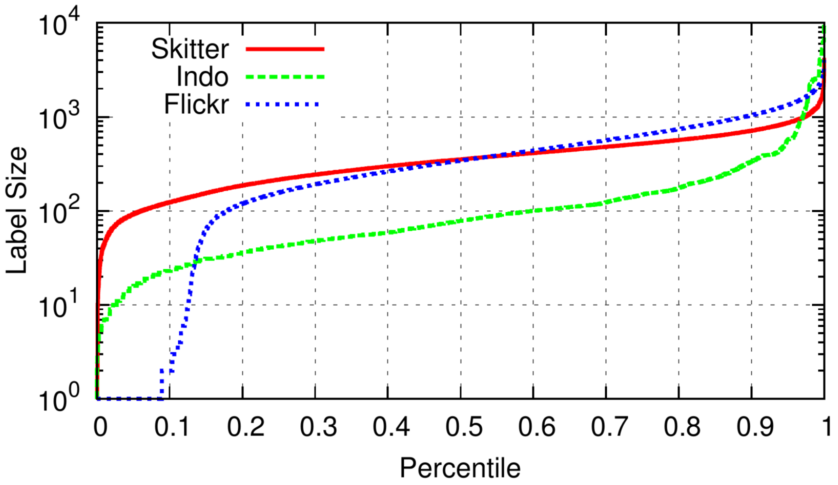


Figure 8.5. Distribution of the sizes of labels for three graphs in the dataset (taken from [7]).

8.12.2 Probabilistic Analysis

First, as we did in the analysis of the iFub algorithm, we compute a deterministic bound on the expected time of a distance query between two random nodes.

Lemma 8.27. *For each s_i, let N_{s_i} be the number of nodes $t \in V$ such that no st-shortest path passes from a node s_j with $j < i$. Then, the average query time is $\mathcal{O}\left(\frac{1}{n}\sum_{s \in V} N(s)\right)$, and the space used is $\Theta\left(\sum_{s \in V} N(s)\right)$.*

Proof. If the labels are sorted, in order to intersect $L(s)$ and $L(t)$, the time needed is $\mathcal{O}\left(|L(s)| + |L(t)|\right)$. Hence, the expected time of a distance query between two random nodes is

$$\begin{aligned}\frac{1}{n^2}\sum_{s,t \in V} \mathcal{O}\left(|L(s)| + |L(t)|\right) &= \mathcal{O}\left(\frac{1}{n}\sum_{t \in V}|L(t)|\right)\\ &= \mathcal{O}\left(\frac{1}{n}\sum_{s,t \in V} X_{st}\right)\\ &= \mathcal{O}\left(\frac{1}{n}\sum_{s \in V} N(s)\right),\end{aligned}$$

where $X_{st} = 1$ if $s \in L(t)$, 0 otherwise. Similarly, the space used is $\Theta\left(\frac{1}{n}\sum_{t \in V}|L(t)|\right) = \Theta\left(\sum_{s \in V} N(s)\right)$. □

The Case $1 < \beta < 2$

In the case $1 < \beta < 2$, let us fix $\varepsilon > 0$, and let us consider nodes s with small degree (at most, $n^{2\varepsilon}$): the number of nodes reachable from s at distance k passing only through nodes of degree smaller than $\deg(s)$ is at most $\deg(s)^k$: hence, $N(s) \leq D \deg(s)^D \leq Dn^{2\varepsilon D} = n^{\mathcal{O}(\varepsilon)}$ (because the diameter D is constant).

Let us consider nodes s such that $\deg(s) > n^{2\varepsilon}$: by Axiom 2, all these nodes are connected to each node with degree at least $n^{1-\varepsilon}$, and no node t with degree bigger than n^{ε} can contain

s in their label, unless t is a neighbor of s. Consequently, the nodes that contain s in their label are at most $\deg(s)$ nodes at distance 1 from s, $N_2(s)$ nodes at distance 2 from s, and at most $\left(n^{2\varepsilon}\right)^D N_2(s) = n^{\mathcal{O}(\varepsilon)}N_2(s)$ nodes at a bigger distance. Summing these values, we obtain that $N(s) \leq \deg(s) + N_2(s)n^{\mathcal{O}(\varepsilon)}$: summing over all nodes s, the average query time is $\frac{1}{n}\mathcal{O}\left(n^{1+\varepsilon} + \sum_{s\in V,\deg(s)>n^{2\varepsilon}} \deg(s) + N_2(s)n^{2\varepsilon}\right) = n^{\mathcal{O}(\varepsilon)}\left(1 + \frac{1}{n}\sum_{s\in V} N_2(s)\right)$. Since $t \in N_2(s)$ implies that $\deg(t) < n^{2\varepsilon}$, and since the number of nodes with degree bigger than n^x is $n^{1-x+o(1)}$ by Axiom 4,

$$\begin{aligned}
\sum_{s\in V} N_2(s) &= \sum_{s\in V}\ \sum_{v\in\mathbf{\Gamma}^1(s),\deg(v)<\deg(s)}\ \sum_{t\in\mathbf{\Gamma}^1(v),\deg(t)<n^{2\varepsilon}} 1\\
&= \sum_{v\in V}\left|\left\{t\in\mathbf{\Gamma}^1(v):\deg(t)<n^{2\varepsilon}\right\}\right|\left|\left\{s\in\mathbf{\Gamma}^1(v):\deg(s)>\deg(v)\right\}\right|\\
&\leq \sum_{v\in V}\deg(v)\max\left(\deg(v),\frac{n^{1+\varepsilon}}{\deg(v)}\right)\\
&= \sum_{d=1}^{n}\left|\left\{v:\deg(v)=d\right\}\right| d\max\left(d,\frac{n^{1+\varepsilon}}{d}\right)\\
&\leq \sum_{d=1}^{n^{\frac{1}{2}}}\left|\left\{v:\deg(v)=d\right\}\right| d^2 + \sum_{d=n^{\frac{1}{2}}}^{n}\left|\left\{v:\deg(v)=d\right\}\right| n^{1+\varepsilon}\\
&\leq S_1 + n^{\frac{3}{2}+\varepsilon}.
\end{aligned}$$

Let us estimate S_1 using Abel's summation technique:

$$\begin{aligned}
S_1 &= \sum_{d=1}^{n^{\frac{1}{2}}}\left|\left\{v:\deg(v)=d\right\}\right| d^2\\
&= \sum_{d=1}^{n^{\frac{1}{2}}}\left|\left\{v:\deg(v)\geq d\right\}\right| d^2 - \sum_{d=1}^{n^{\frac{1}{2}}}\left|\left\{v:\deg(v)\geq d+1\right\}\right| d^2\\
&\leq n + \sum_{d=1}^{n^{\frac{1}{2}}}\left|\left\{v:\deg(v)\geq d\right\}\right|(d^2-(d-1)^2)\\
&\leq n + \sum_{d=1}^{n^{\frac{1}{2}}}\frac{n}{d}2d\\
&= \mathcal{O}\left(n^{\frac{3}{2}}\right).
\end{aligned}$$

Then, the average query time is $n^{\mathcal{O}(\varepsilon)}\left(1+\frac{1}{n}\sum_{s\in V}N_2(s)\right) = n^{\frac{1}{2}+\mathcal{O}(\varepsilon)}$. The space occupied is n multiplied by the average query time, that is, $n^{\frac{3}{2}+\mathcal{O}(\varepsilon)}$, by Lemma 8.27.

The Case $2 < \beta < 3$

In the case $2 < \beta < 3$, let us consider the number $N_\ell(s)$ of nodes t such that $s \in L(t)$ and $\mathrm{dist}(s,t) = \ell$. For small values of ℓ, we estimate $N_\ell(s) \leq N_{\ell-1}(s)\deg(s)$, while for bigger values of ℓ, we prove that all nodes with high degree are at distance at most $\ell - 2$ from s, obtaining that $N_\ell(s) \leq f(\ell, \deg(s))N_{k-1}(s)$, for some function f.

More formally, let s, t be two nodes with degree at least $\log^2 n$: the distance between s and t is at most

$$\boldsymbol{\tau}_s\left(n^{\frac{1+\varepsilon}{2}}\right)+\boldsymbol{\tau}_t\left(n^{\frac{1+\varepsilon}{2}}\right) \leq (1+\varepsilon)\left(\log_{\frac{1}{\beta-2}}\left(\frac{\log^2 n^{\frac{1+\varepsilon}{2}}}{\log\deg(s)\log\deg(t)}\right)\right)$$

(our axioms say that this approximation holds for $\deg(s), \deg(t) > n^{\varepsilon}$, but it is easy to extend the results in Appendix A.2 to the case $\deg(s) > \log^2 n$). Consequently there is no node at distance k from s with degree bigger than

$$\max\left(\log^2 n, e^{(\beta-2)^{\frac{k}{1+\varepsilon}}\frac{\log^2 n^{\frac{1+\varepsilon}{2}}}{\log(\deg(s))}}\right)$$

.

As we said before, for

$$k < k_0 = (1+\varepsilon)\left(\log_{\frac{1}{\beta-2}}\left(\frac{\log^2 n^{\frac{1+\varepsilon}{2}}}{\log^2\deg(s)}\right)\right),$$

we estimate $N_{k+1}(s) \leq N_k(s)\deg(s)$, and consequently

$$N_k(s) \leq \deg(s)^k \leq \deg(s)^{(1+\varepsilon)\left(\log_{\frac{1}{\beta-2}}\left(\frac{\log^2 n^{\frac{1+\varepsilon}{2}}}{\log^2\deg(s)}\right)\right)}.$$

For bigger values of k, since $\boldsymbol{\Gamma}^{k+1}(v)$ does not contain any node with degree bigger than

$$\max\left(\log^2 n, e^{(\beta-2)^{\frac{k}{1+\varepsilon}}\frac{\log^2 n^{\frac{1+\varepsilon}{2}}}{\log(\deg(s))}} + \log^2 n\right),$$

$$N_{k+1}(S) \leq N_k(S)\max\left(\log^2 n, e^{(\beta-2)^{\frac{k-1}{1+\varepsilon}}\frac{\log^2 n^{\frac{1+\varepsilon}{2}}}{\log(\deg(s))}}\right).$$

As a consequence, we can prove by induction that, if $k = \mathcal{O}(\log\log n)$:

$$\begin{aligned}
N_k(s) &\leq N_{k_0}(s)\prod_{i=k_0}^{k} e^{(\beta-2)^{\frac{i-1}{1+\varepsilon}}\frac{\log^2 n^{\frac{1+\varepsilon}{2}}}{\log^2\deg(s)}}\log^{2k} n \\
&\leq N_{k_0}(s)n^{\varepsilon}\prod_{i=k_0}^{k} e^{(\beta-2)^{\frac{i-k_0}{1+\varepsilon}}(\beta-2)^{\frac{k_0-1}{1+\varepsilon}}\frac{\log^2 n^{\frac{1+\varepsilon}{2}}}{\log(\deg(s))}} \\
&\leq N_{k_0}(s)n^{\varepsilon}e^{\sum_{i=k_0}^{k}(\beta-2)^{\frac{i-k_0}{1+\varepsilon}}\log\deg(s)} \\
&\leq N_{k_0}(s)n^{\varepsilon}\deg(s)^{\frac{1}{1-(\beta-2)^{\frac{1}{1+\varepsilon}}}}. \\
&\leq n^{2\varepsilon}\deg(s)^{(2+2\varepsilon)\left(\log_{\frac{1}{\beta-2}}\left(\frac{\log n^{\frac{1+\varepsilon}{2}}}{\log\deg(s)}\right)\right)+\frac{1}{3-\beta}} = f(\deg(s)).
\end{aligned}$$

For bigger values of k, there are very few nodes at distance k from s, and their contribution is negligible. Hence, we know that $N(s) \leq \min(n, n^{\varepsilon}f(\deg(s))) = g(\deg(s))$: we want to compute $\sum_{s\in V} g(\deg(s))$.

Lemma 8.28. *If G is a random graph with power law degree distribution,*

$$\sum_{s \in V, \deg(s) > \log^2 n} g(d) = \mathcal{O}\left(n \int_{\log^2 n}^{n^{\frac{1}{\beta-1}+\varepsilon}} \frac{g(x)}{x^\beta} dx + n^{1+\varepsilon} \right).$$

Proof. We use Abel's summation technique twice:

$$\begin{aligned}
\sum_{s \in V, \deg(s) > \log^2 n} g(d) &= \sum_{d=\log^2 n}^{n^{\frac{1+\varepsilon}{\beta-1}}} \left| \left\{ s \in V : \deg(s) = d \right\} \right| g(d) \\
&= \sum_{d=\log^2 n}^{n^{\frac{1+\varepsilon}{\beta-1}}} (\left| \left\{ s \in V : \deg(s) \geq d \right\} \right| \\
&\quad - \left| \left\{ s \in V : \deg(s) \geq d+1 \right\} \right|) g(d) \\
&\leq \sum_{d=\log^2 n}^{n^{\frac{1+\varepsilon}{\beta-1}}} \frac{n}{d^{\beta-1}} (g(d) - g(d-1)) + \mathcal{O}(ng(\log^2 n)) \\
&\leq \mathcal{O}(ng(\log^2 n)) + \sum_{d=\log^2 n}^{n^{\frac{1+\varepsilon}{\beta-1}}} ng(d) \left(\frac{1}{d^{\beta-1}} - \frac{1}{(d+1)^{\beta-1}} \right) \\
&\leq \mathcal{O}(ng(\log^2 n)) + \sum_{d=\log^2 n}^{n^{\frac{1+\varepsilon}{\beta-1}}} ng(d) \left(\frac{1}{d^{\beta-1}} - \frac{1}{(d+1)^{\beta-1}} \right) \\
&= \mathcal{O}\left(n^{1+\varepsilon} + \sum_{d=\log^2 n}^{n^{\frac{1+\varepsilon}{\beta-1}}} \frac{ng(d)}{d^\beta} \right).
\end{aligned}$$

We have to transform this sum into an integral: to this purpose, we observe that $g(d+\varepsilon) = \mathcal{O}(g(d))$ for each $d > \log^2 n - 1$, and for each $\varepsilon \leq 1$. Hence, $\sum_{d=\log^2 n}^{n^{\frac{1+\varepsilon}{\beta-1}}} \frac{g(d)}{d^\beta} = \int_{\log^2 n - 1}^{n^{\frac{1}{\beta-1}+\varepsilon}} \frac{g(\lceil x \rceil)}{\lceil x \rceil^\beta} dx = \mathcal{O}\left(\int_{\log^2 n - 1}^{n^{\frac{1}{\beta-1}+\varepsilon}} \frac{g(x)}{x^\beta} dx \right)$. □

It remains to estimate this integral.

$$\begin{aligned}
& n \int_{\log^2 n}^{n^{\frac{1}{\beta-1}}} \frac{1}{x^\beta} \min\left(n, x^{(2+2\varepsilon) \log_{\frac{1}{\beta-2}} \left(\frac{\log n^{\frac{1+\varepsilon}{2}}}{\log x} \right) + \frac{1}{3-\beta}} \right) dx = \\
& \leq n^{1+\mathcal{O}(\varepsilon)} \int_0^{\frac{1}{\beta-1}} n^{-\beta t} \min\left(n, n^{t\left(2 \log_{\frac{1}{\beta-2}} \left(\frac{1}{2t} \right) + \frac{1}{3-\beta} \right)} \right) n^t \log n \, dt \\
& = n^{1+\mathcal{O}(\varepsilon)} \int_0^{\frac{1}{\beta-1}} n^{t\left(\min\left(1, 2 \log_{\frac{1}{\beta-2}} \left(\frac{1}{2t} \right) + \frac{1}{3-\beta} \right) - \beta + 1 \right)} dt \\
& = n^{1+\mathcal{O}(\varepsilon) + \max_{t \in \left[0, \frac{1}{\beta-1}\right]} t\left(\min\left(1, 2 \log_{\frac{1}{\beta-2}} \left(\frac{1}{2t} \right) + \frac{1}{3-\beta} \right) - \beta + 1 \right)}.
\end{aligned}$$

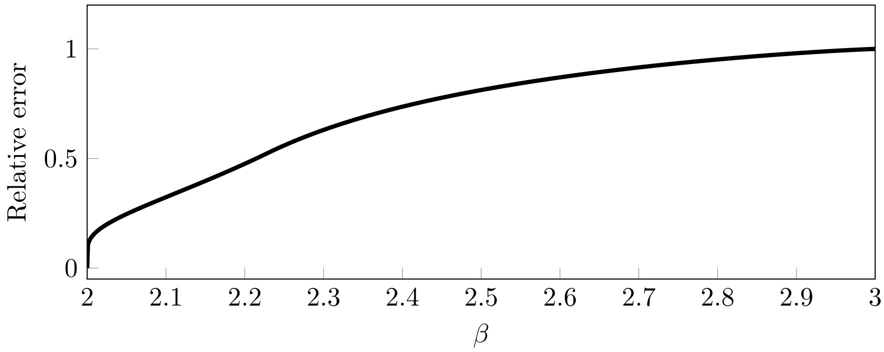


Figure 8.6. An upper bound on the exponent in the average distance query time of the distance oracle for $2 < \beta < 3$.

Then, the average query time is

$$n^{\max_{t \in \left[0, \frac{1}{\beta-1}\right]} t\left(\min\left(1, 2\log_{\frac{1}{\beta-2}}\left(\frac{1}{2t}\right) + \frac{1}{3-\beta}\right) - \beta + 1\right) + \mathcal{O}(\varepsilon)}.$$

We are not able to find an analytic form for this function, but the result is plotted in Figure 8.6. The exponent of the total space occupancy is this function, plus one.

The Case $\beta > 3$

In this case, we prove that the algorithm does not provide a significant improvement: indeed, the time needed for a distance query is $n^{1-\mathcal{O}(\varepsilon)}$. To prove this result, it is enough to show that, for a set of graphs that satisfy the axioms, the algorithm is not efficient on this set of graph. The set of graphs we choose is the set of random graphs generated through the CM, or through IRG: for this reason, we are allowed to use theorems that are specific of these models.

In this proof, we show that, if S is the set of nodes with degree between n^α and $n^{\alpha+\varepsilon^2}$, then the number of pairs $(s,t) \in S^2$ such that $s \in L(t)$ is $\Omega\left(|S|^2\right)$, and hence the average label size is big, because, by Axiom 4, $|S| = n^{1-\alpha(\beta-1)+o(1)} = n^{1-o(1)}$ if α tends to 0.

More formally, by Axioms 1 and 2, for each pair of nodes $s,t \in S$, $\mathrm{dist}(s,t) \leq \boldsymbol{\tau}_s\left(n^{\frac{1}{2}+\varepsilon}\right) + \boldsymbol{\tau}_t\left(n^{\frac{1}{2}+\varepsilon}\right) \leq (2+2\varepsilon)T\left(n^\alpha \to n^{\frac{1}{2}+\varepsilon}\right) \leq (2+2\varepsilon)\frac{\log n^{\frac{1}{2}+\varepsilon-\alpha}}{\log M_1(\mu)} \leq (1-2\alpha+4\varepsilon)\frac{\log n}{\log M_1(\mu)} =: D_S$. We want to prove that, a.a.s., if $s,t \in S$ and $\deg(t) < \deg(s)$, there is a high chance that $s \in L(t)$. To this purpose, we consider all nodes v with degree bigger than $\deg(s)$, and we count the number of pairs $s,t \in S$ such that $\mathrm{dist}(s,v) + \mathrm{dist}(v,t) \leq D_S$. Then, we sum this contribution over all nodes v: if this sum is $o\left(|S|\right)$, it means that $\Omega(|S|)$ nodes in S have s in their label.

More formally, we start by estimating, for each node v, the number of nodes $s,t \in S$ such that $\mathrm{dist}(s,v) + \mathrm{dist}(v,t) \leq D_S$.

Lemma 8.29 (for a proof, see Lemma A.56)**.** *Let v be a node with degree $\omega(1)$. Then, the number of pairs of nodes $s,t \in S$ such that $\mathrm{dist}(s,v) + \mathrm{dist}(v,t) \leq D_S$, and $\mathrm{dist}(s,w) + \mathrm{dist}(w,t) > D_S$ for each w such that $\deg(w) > \deg(v)$, is at most $\deg(v)^2|S|^2 n^{-1+\mathcal{O}(\varepsilon)}$.*

Let us consider the ordering of all nodes $s_1, \ldots, s_n$, and let us estimate:

$$\begin{aligned}
&|\{(s_i,s_j) \in S^2 : i < j, s_i \notin L(s_j)\}| \\
&\leq |\{(s_i,s_j) \in S^2 : i < j, \exists k < i, \mathrm{dist}(s_i,s_j) = \mathrm{dist}(s_i,s_k) + \mathrm{dist}(s_k,s_j)\}| \\
&\leq |\{(s_i,s_j) \in S^2 : i < j, \exists k < i, \mathrm{dist}(s_i,s_k) + \mathrm{dist}(s_k,s_j) \leq D_S\}|
\end{aligned}$$

$$\begin{aligned}
&\le \sum_{k<i} \deg(s_k)^2 |S|^2 n^{-1+\mathcal{O}(\varepsilon)} \\
&\le n^{1-\alpha(\beta-3)+\varepsilon} |S|^2 n^{-1+\mathcal{O}(\varepsilon)} \\
&= o(|S|^2).
\end{aligned}$$

We used the fact that $\sum_{k<i} \deg(s_k)^2 \le n^{1-\alpha(\beta-3)+\varepsilon}$: let us prove it formally, using Abel's summation technique and Axiom 4.

$$\begin{aligned}
\sum_{k<i} \deg(s_k)^2 &= \sum_{d=n^\alpha}^{+\infty} d^2 |\{v : \deg(v) = n^\alpha\}| \\
&= \sum_{d=n^\alpha}^{+\infty} d^2 |\{v : \deg(v) \ge d\}| - \sum_{d=n^\alpha+1}^{+\infty} (d-1)^2 |\{v : \deg(v) \ge d\}| \\
&\le n^{2\alpha} |\{v : \deg(v) \ge n^\alpha\}| + \sum_{d=n^\alpha}^{+\infty} 2d |\{v : \deg(v) \ge d\}| \\
&\le n^{2\alpha} \frac{n}{n^{\alpha(\beta-1)}} + \sum_{d=n^\alpha}^{+\infty} 2d \frac{n}{d^{\beta-1}} = \mathcal{O}\left(n^{1-\alpha(\beta-3)}\right).
\end{aligned}$$

We have proved that $|\{(s_i, s_j) \in S^2 : i < j, s_i \in L(s_j)\}| = \Omega\left(|S|^2\right)$, and consequently the total label size is at least $\Omega\left(|S|^2\right) \ge n^{2-3\alpha(\beta-1)}$. We claim that this means that there are many labels with size bigger than $n^{1-4\alpha(\beta-1)}$, because no label has size bigger than n. Indeed, if ℓ_i is the size of label i, $n^{2-3\alpha(\beta-1)} \le \sum_{i=1}^n \ell_i \le n^{1-4\alpha(\beta-1)}|\{i : \ell_i \le n^{1-4\alpha(\beta-1)}\}| + n|\{i : \ell_i > n^{1-4\alpha(\beta-1)}\}| \le n^{2-4\alpha(\beta-1)} + n|\{i : \ell_i > n^{1-4\alpha(\beta-1)}\}|$, and hence $|\{i : \ell_i > n^{1-4\alpha(\beta-1)}\}| \ge n^{1-3\alpha(\beta-1)} - n^{1-4\alpha(\beta-1)} \ge n^{1-4\alpha(\beta-1)}$.

We have proved that, for each $\alpha' = 4\alpha(\beta-1)$, there are at least $n^{1-\alpha'}$ labels of size $n^{1-\alpha'}$: consequently, the expected time to perform a distance query is at least $\frac{n^{2-2\alpha'}}{n^2} n^{1-\alpha'}$, because the probability that we hit two nodes s, t whose labels are bigger than $n^{1-\alpha'}$ is at least $\frac{n^{2-2\alpha'}}{n^2}$. If we let $\alpha' = \mathcal{O}(\varepsilon)$, the average time for a distance query becomes at least $n^{1-\mathcal{O}(\varepsilon)}$. Similarly, the space occupied is $n^{2-\mathcal{O}(\varepsilon)}$.

8.13 BBBFS

In Chapter 6, we designed a new algorithm to approximate the betweenness centrality of all the nodes in a network. The main subroutine of this algorithm was to sample a random shortest path between two nodes s and t.

All previous approaches used simply a BFS from s, that needs time $\mathcal{O}(m)$, while our approach used a BBBFS, that is, a balanced bidirectional BFS (see Section 6.5 for the exact definition). The idea of using bidirectional search heuristics has been proposed several years ago [134, 95], but it has never been very successful, probably because of the lack of theoretical guarantees. In this section, we provide this kind of guarantees, using our axioms: we prove that it improves the $\mathcal{O}(m)$ time needed by a standard BFS. Before, let us confirm experimentally the efficiency of this technique.

8.13.1 Experimental Results

In this section, we experimentally show that the bidirectional BFS can yield significant improvements in the running time. To this purpose, we have collected a dataset of 16 undirected networks and 19 directed network taken from the taken from the datasets SNAP (`snap.stanford.edu/`), LASAGNE (`piluc.dsi.unifi.it/lasagne`), and KONECT

Table 8.7. The balanced bidirectional BFS: experimental results.

Undirected networks

Network	n	m	m_{vis}	$\alpha := \frac{\log(m_{\text{vis}})}{\log m}$
HC-BIOGRID	4039	20642	429.127	0.610
Mus_musculus	4696	11494	139.545	0.528
Caenorhabditis_elegans	4748	19684	105.892	0.472
ca-GrQc	5241	28968	200.369	0.516
as20000102	6474	25144	47.253	0.381
advogato	7420	90929	46.549	0.336
p2p-Gnutella09	8114	52026	162.258	0.469
hprd_pp	9617	74078	139.346	0.440
Drosophila_melanogaster	10644	81562	167.717	0.453
oregon1_010526	11174	46818	64.894	0.388
oregon2_010526	11461	65460	86.307	0.402
Homo_sapiens	13935	122260	163.974	0.435
dip20090126_MAX	19928	82404	12496.746	0.833
email-Enron	36692	367662	294.589	0.444
ca-HepTh	68746	51971	5.600	0.159
com-amazon.all.cmty	134386	198866	21.356	0.251

Directed networks

Network	n	m	m_{vis}	$\alpha := \frac{\log(m_{\text{vis}})}{\log m}$
polblogs	1490	19025	63.270	0.421
opsahl-openflights	2940	30501	94.562	0.441
ca-GrQc	5241	28968	200.369	0.516
subelj_jung-j_jung-j	6121	50535	42.859	0.347
subelj_jdk_jdk	6435	53892	47.935	0.355
wiki-Vote	7115	103689	43.452	0.327
ca-HepTh	9875	51946	220.181	0.497
freeassoc	10617	72176	104.390	0.415
ca-HepPh	12006	236978	344.083	0.472
lasagne-spanishbook	12644	57451	60.368	0.374
cfinder-google	15764	170335	286.943	0.470
ca-CondMat	23133	186878	363.456	0.486
subelj_cora_cora	23167	91500	377.655	0.519
ego-twitter	23371	33101	2.200	0.076
ego-gplus	23629	39242	2.003	0.066
as-caida20071105	26475	106762	109.520	0.406
cit-HepTh	27769	352768	4985.010	0.667
p2p-Gnutella31	62586	147892	110.557	0.395
soc-Epinions1	75879	508837	124.459	0.367

(`http://konect.uni-koblenz.de/networks/`). For each of these networks, we have computed the distance of 100 000 random pairs of nodes, using the balanced bidirectional BFS. We have evaluated the performances of the algorithm by comparing the number of visited edges m_{vis} with the total number of edges m (which is the number of edges visited in the worst-case). Furthermore, we have computed the exponent $\alpha := \frac{\log(m_{\text{vis}})}{\log m}$: this value is such that $m_{\text{vis}} = m^{\alpha}$. The results are available in Table 8.7, both for undirected and for directed networks (in any case, our probabilistic analysis only holds in the undirected case).

From the table, we see that the number of visited edges is much smaller than m: indeed, most of the times it is even smaller than $\sqrt{m}$, both in the undirected and in the directed case. This yields a significant improvement in the running time, when this algorithm is applied to very large networks.

8.13.2 Probabilistic Analysis

The Case $1 < \beta < 2$

In this case, we prove that the time needed to compute a shortest path from s to t is $n^{1-o(1)}$ for $\Omega(n^2)$ pairs (s,t), on graphs that satisfy our assumptions. Indeed, let v_1, v_2 be the two nodes with maximum degree: since the weights are chosen according to a power law degree distribution $\rho_{v_1}, \rho_{v_2} = \mathcal{O}(M)$. Let W be the set of nodes with weight 1: for each $w \in W$, the probability that w has degree 1 and it is a neighbor of v_1 or v_2 is $f\left(\Theta\left(\frac{m \cdot 1}{M}\right)\right) = \Theta(1)$. Through simple concentration inequalities, it is quite straightforward to prove there are $\Theta(n)$ neighbors s_i of v_1, with degree 1, and $\Theta(n)$ neighbors t_j of v_2, with degree 1. If a shortest path from s_i to t_j is required, we have to process either v_1 or v_2, and this takes time $\Theta(n)$. Hence, the time needed to perform one of these queries is $\Theta(n)$.

The Case $\beta > 2$

The idea of the probabilistic analysis is that the time needed by a bidirectional BFS is proportional to the number of visited edges, which is the sum of the degrees of the visited nodes, which are very close to their weights. Indeed, the following lemma shows that it is enough to analyze the weights of the processed nodes.

Lemma 8.30. *For each node v and for each ε, $\rho_v n^{-\varepsilon} \leq \deg(v) \leq \rho_v n^{\varepsilon}$ w.h.p..*

Proof. If v is a node with weight $\rho_w > n^{\varepsilon}$, by Lemmas A.11 and A.16, $\deg(v) = \Theta(d)$, and the conclusion follows. If $\rho_v \leq n^{\varepsilon}$, the first inequality is trivial, and the second inequality follows from the fact that decreasing the weight can only decrease the degree, and every node with weight n^{ε} has degree at most $\Theta(n^{\varepsilon}) \leq n^{2\varepsilon}$. □

Due to the importance of the weights of sets, we use the notation $\boldsymbol{\delta}^{\ell}(s)$ to denote the sum of the weights of all the nodes at distance ℓ from s, as in Chapter 8.

Our visit proceeds by "levels" in the BFS trees from s and t: if we never process a level with total weight at least $n^{\frac{1}{2}+\varepsilon}$, since the diameter is $\mathcal{O}(\log n)$, the volume of the set of processed nodes is $\mathcal{O}(n^{\frac{1}{2}+\varepsilon} \log n)$, and the number of visited edges cannot be much bigger. Otherwise, assume that, at some point, we process a level ℓ_s in the BFS from s with total weight $n^{\frac{1}{2}+\varepsilon}$: then, the corresponding level ℓ_t in the BFS from t has also weight $n^{\frac{1}{2}+\varepsilon}$ (otherwise, we would have expanded from t, because weights and degrees are strongly correlated). By Axiom 2 and Lemmas A.11 and A.16, since $\operatorname{dist}(s,t) \leq \boldsymbol{\tau}_s\left(n^{\frac{1}{2}+\varepsilon}\right) + \boldsymbol{\tau}_t\left(n^{\frac{1}{2}+\varepsilon}\right) - 1$, there is an edge from level ℓ_s to level ℓ_t, and the visit terminates when level ℓ_s is visited. This means that the time needed by the bidirectional BFS is proportional to the volume of all levels in the BFS tree from s, until ℓ_s, plus the volume of all levels in the BFS tree from t, until ℓ_t (note that we do not expand levels $\ell_s + 1$ and $\ell_t + 1$). All levels except the last one have volume at most $n^{\frac{1}{2}+\varepsilon}$, and there are $\mathcal{O}(\log n)$ such levels because the diameter is $\mathcal{O}(\log n)$ by Theorem 8.10: it only remains to estimate the volume of the last level.

Lemma 8.31. *With high probability, for each $s \in V$ and for each ℓ such that $\sum_{i=0}^{\ell-1} \boldsymbol{\delta}^i(s) < n^{\frac{1}{2}+\varepsilon}$, $\boldsymbol{\delta}^{\ell}(s) < n^{\frac{1}{2}+3\varepsilon}$ if $\beta > 3$, $\boldsymbol{\delta}^{\ell}(s) < n^{\frac{4-\beta}{2}+3\varepsilon}$ if the weights are chosen to a power law distribution with $2 < \beta < 3$.*

Proof. We consider separately nodes with weight at most $n^{\frac{1}{2}-2\varepsilon}$ from nodes with bigger weights: in the former case, we bound the number of such nodes that are in $\boldsymbol{\Gamma}^{\ell}(s)$, while in the latter case we bound the total number of nodes with weight at least $n^{\frac{1}{2}-2\varepsilon}$. Let us start with the latter case.

Claim: for each ε, $\sum_{\rho_v \geq n^{\frac{1}{2}-\varepsilon}} \rho_v$ is smaller than $n^{\frac{1}{2}+3\varepsilon}$ if λ has finite second moment, and it is smaller than $n^{\frac{4-\beta}{2}+3\varepsilon}$ if λ is power law with $2 < \beta < 3$.

Proof of claim. If λ has finite second moment, by Chebyshev inequality, for each α,

$$\mathbb{P}\left(\lambda > n^{\frac{1}{2}+\alpha}\right) \le \frac{\operatorname{Var}(\lambda)}{n^{1+2\alpha}} \le \frac{M_2(\lambda)}{n^{1+2\alpha}} = \mathcal{O}\left(\frac{M_2(\lambda)}{n^{1+2\alpha}}\right) = \mathcal{O}\left(n^{-1-2\alpha}\right).$$

For $\alpha = \varepsilon$, this means that no node has weight bigger than $n^{\frac{1}{2}+\varepsilon}$, and for $\alpha = -\varepsilon$, this means that the number of nodes with weight bigger than $n^{\frac{1}{2}-\varepsilon}$ is at most $n^{2\varepsilon}$. We conclude that $\sum_{\rho_v \ge n^{\frac{1}{2}-\varepsilon}} \rho_v \le \sum_{\rho_v \ge n^{\frac{1}{2}-\varepsilon}} n^{\frac{1}{2}+\varepsilon} \le n^{\frac{1}{2}+3\varepsilon}$.

If λ is power law with $2 < \beta < 3$, by Axiom 4 the number of nodes with weight at least d is at most $Cnd^{-\beta+1}$. Consequently, using Abel's summation technique,

$$\begin{aligned}
\sum_{\rho_v \ge n^{\frac{1}{2}-\varepsilon}} \rho_v &= \sum_{d=\rho_v}^{+\infty} d|\{v : \rho_v = d\}| \\
&= \sum_{d=n^{\frac{1}{2}-\varepsilon}}^{+\infty} d(|\{v : \rho_v \ge d\}| - |\{v : \rho_v \ge d+1\}|) \\
&= \sum_{d=n^{\frac{1}{2}-\varepsilon}}^{+\infty} d|\{v : \rho_v \ge d\}| - \sum_{d=n^{\frac{1}{2}-\varepsilon}+1}^{+\infty} (d-1)|\{v : \rho_v \ge d\}| \\
&= n^{\frac{1}{2}-\varepsilon}|\{v : \rho_v \ge n^{\frac{1}{2}-\varepsilon}\}| + \sum_{d=n^{\frac{1}{2}-\varepsilon}+1}^{+\infty} |\{v : \rho_v \ge d\}| \\
&\le Cn^{\frac{1}{2}-\varepsilon} n^{1-\left(\frac{1}{2}-\varepsilon\right)(\beta-1)} + \sum_{d=n^{\frac{1}{2}-\varepsilon}+1}^{+\infty} Cnd^{-\beta+1} \\
&= \mathcal{O}\left(n^{\frac{4-\beta}{2}+\varepsilon\beta} + n^{1-\left(\frac{1}{2}-\varepsilon\right)(\beta-2)}\right) = \mathcal{O}\left(n^{\frac{4-\beta}{2}+\varepsilon\beta}\right).
\end{aligned}$$

□

By this claim, $\sum_{v \in \mathbf{\Gamma}^{l+1}(s), \rho_v \ge n^{\frac{1}{2}-2\varepsilon}} \rho_v$ is smaller than $n^{\frac{1}{2}+6\varepsilon}$ if λ has finite second moment, and it is smaller than $n^{\frac{4-\beta}{2}+6\varepsilon}$ if λ is power law with $2 < \beta < 3$. To conclude the proof, we only have to bound $\sum_{v \in \mathbf{\Gamma}^{l+1}(s), \rho_v < n^{\frac{1}{2}-2\varepsilon}} \rho_v$.

Claim: with high probability, $\sum_{v \in \mathbf{\Gamma}^{l+1}(s), \rho_v < n^{\frac{1}{2}-2\varepsilon}} \rho_v < n^{\frac{1}{2}+\varepsilon}$ if λ has finite second moment, $\sum_{v \in \mathbf{\Gamma}^{l+1}(s), \rho_v < n^{\frac{1}{2}-2\varepsilon}} \rho_v < n^{\frac{4-\beta}{2}+\varepsilon}$ if λ is power law with $2 < \beta < 3$.

Proof. By Lemma A.38, we can remove all nodes with weight at least $n^{\frac{1}{2}-2\varepsilon}$ from the graph. We conclude by Lemmas A.13 and A.18. □

This claim lets us conclude the proof of the lemma. □

All these ingredients let us conclude our probabilistic analysis: indeed, the number of visited edges is at most the sum of the degree of all visited nodes, which is at most the sum of their weights, multiplied by n^{ε}, by Lemma 8.30. As we argued before, all visited levels except the last have weight at most $n^{\frac{1}{2}+\varepsilon}$, and, by Lemma 8.31, the last level has weight at most $n^{\frac{1}{2}+\mathcal{O}(\varepsilon)}$ if $\beta > 3$, $n^{\frac{4-\beta}{2}+\mathcal{O}(\varepsilon)}$ if $2 < \beta < 3$. This result concludes our probabilistic analysis.

8.14 Validity of the Axioms in Real-World Graphs: Experimental Evaluation

In this section, we experimentally show that the axioms hold in real-world graphs, with good approximation. To this purpose, we consider a dataset made by 18 real-world networks of different kinds (social networks, citation networks, technological networks, and so on), taken from the well-known datasets SNAP (`snap.stanford.edu/`) and KONECT (`http://konect.uni-koblenz.de/networks/`). Then, for each of the axioms, we compute the quantities considered, on all graphs in the dataset, and we show that the actual behavior reflects the predictions.

We start with Axiom 1: to verify the first claim, we consider all nodes with degree at least $n^{0.2}$, which is between 6 and 19 in our inputs. For each of these nodes, we compute $\boldsymbol{\tau}_s\left(n^{\frac{1}{2}}\right) - T\left(\deg(s) \to n^{\frac{1}{2}}\right)$ (in this paper, we show the results for $x = \frac{1}{2}$, but very similar results hold for all values of x). The results obtained are represented in Table 8.8.

Table 8.8. The percentage of nodes with degree at least $n^{0.2}$ that satisfy $\boldsymbol{\tau}_s\left(n^{\frac{1}{2}}\right) - \left\lceil T\left(d \to n^{\frac{1}{2}}\right)\right\rceil = k$ (the other values are 0, for each graph in the dataset).

Network	$n^{0.2}$	Vert.	$k=-2$	$k=-1$	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$
p2p-Gnutella09	6.1	2811	0.00%	61.37%	38.63%	0.00%	0.00%	0.00%	0.00%
oregon1-010526	6.5	640	0.00%	58.75%	41.25%	0.00%	0.00%	0.00%	0.00%
ego-gplus	7.5	348	0.00%	2.87%	97.13%	0.00%	0.00%	0.00%	0.00%
oregon2-010526	6.5	1113	0.00%	55.17%	44.83%	0.00%	0.00%	0.00%	0.00%
ca-HepTh	6.1	1987	2.21%	48.97%	43.48%	4.98%	0.25%	0.00%	0.10%
ca-CondMat	7.3	6519	0.00%	45.25%	51.20%	3.27%	0.23%	0.05%	0.00%
ca-HepPh	6.5	4644	0.00%	46.32%	50.39%	2.84%	0.45%	0.00%	0.00%
email-Enron	8.0	6354	0.00%	69.00%	30.33%	0.66%	0.02%	0.00%	0.00%
loc-brightkite	8.9	9929	0.00%	69.45%	29.94%	0.42%	0.18%	0.00%	0.00%
email-EuAll	11.8	2654	0.00%	59.08%	40.66%	0.23%	0.00%	0.00%	0.04%
ca-AstroPh	7.1	9812	0.00%	58.55%	41.10%	0.18%	0.16%	0.00%	0.00%
gowalla-edges	11.5	33263	0.00%	65.69%	34.07%	0.23%	0.01%	0.00%	0.00%
munmun-twitter	13.6	6670	0.00%	70.57%	29.43%	0.00%	0.00%	0.00%	0.00%
com-dblp	12.6	33363	1.65%	63.03%	32.41%	2.57%	0.32%	0.01%	0.00%
com-lj.all.cmty	12.5	5258	0.51%	65.96%	32.98%	0.53%	0.02%	0.00%	0.00%
enron	9.7	7792	0.00%	77.71%	21.79%	0.37%	0.13%	0.00%	0.00%
com-youtube	16.3	46471	0.00%	79.01%	20.32%	0.45%	0.15%	0.04%	0.02%
wiki-Talk	18.9	27536	0.00%	62.63%	37.37%	0.00%	0.00%	0.00%	0.00%

The table shows that in all the graphs considered, the first statement of Axiom 1 is satisfied with good approximation: almost all nodes with degree at least $n^{0.2}$ satisfy $\boldsymbol{\tau}_s\left(n^{\frac{1}{2}}\right) - \left\lceil T\left(\deg(s) \to n^{\frac{1}{2}}\right)\right\rceil \leq 2$; the percentage of nodes satisfying $\boldsymbol{\tau}_s\left(n^{\frac{1}{2}}\right) - \left\lceil T\left(\deg(s) \to n^{\frac{1}{2}}\right)\right\rceil = 2$ is always below 0.5%, and the percentage of nodes satisfying $\boldsymbol{\tau}_s\left(n^{\frac{1}{2}}\right) - \left\lceil T\left(\deg(s) \to n^{\frac{1}{2}}\right)\right\rceil = 1$ is always below 5%.

For the other two points of Axiom 1, for each node s, we have computed $\boldsymbol{\tau}_s\left(n^{\frac{1}{2}}\right) - T\left(\deg(s) \to n^{\frac{1}{2}}\right)$. We want to prove that the number of nodes that satisfy $\boldsymbol{\tau}_s\left(n^{\frac{1}{2}}\right) - T\left(\deg(s) \to n^{\frac{1}{2}}\right) \geq k$ is close to nc^k, for some constant c smaller than 1. For this reason, we have plotted the fraction of nodes satisfying this inequality in logarithmic scale, in Figure 8.7.

This plot confirms the last two points of Axiom 1: indeed, in logarithmic scale, the number of nodes satisfying $\boldsymbol{\tau}_s\left(n^{\frac{1}{2}}\right) - T\left(\deg(s) \to n^{\frac{1}{2}}\right) \geq k$ decreases almost linearly with k, when $k > 0$.

Then, let us validate Axiom 2, which says that, whenever $x + y > 1 + \varepsilon$, for each pair of nodes s, t, $\mathrm{dist}(s,t) < \boldsymbol{\tau}_s(n^x) + \boldsymbol{\tau}_t(n^y)$: we have tested this condition with $(x,y) = (0.3, 0.9), (0.4, 0.8), (0.5, 0.7), (0.6, 0.6)$. For each graph $G = (V,E)$ in the dataset, and for each of the aforementioned pairs (x,y), we have chosen a set $T \subseteq V$ made by 10 000 random

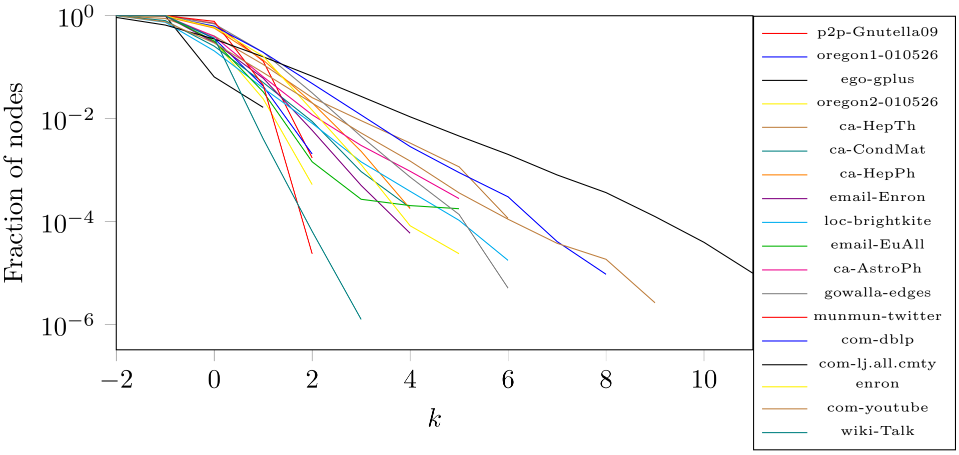


Figure 8.7. The percentage of nodes satisfying $\boldsymbol{\tau}_s\left(n^{\frac{1}{2}}\right) - T\left(\deg(s) \to n^{\frac{1}{2}}\right) \geq k$, in all the graphs in our dataset.

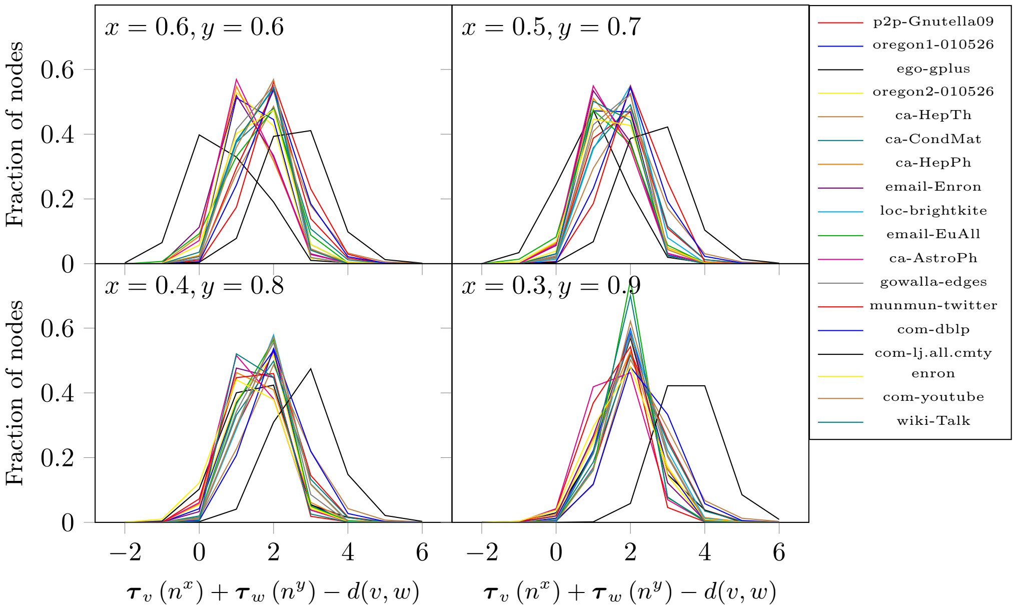


Figure 8.8. The values of $\boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right) - \text{dist}(s,t)$ for 10 000 pairs of nodes in each graph.

nodes (or the whole V if $|V| < 10\,000$), and for each i we have plotted the percentages of pairs $(s,t) \in T^2$ such that $\boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right) - \text{dist}(s,t) = i$. The plots are shown in Figure 8.8.

From the figure, it is clear that $\boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right)$ is almost always at least $\text{dist}(s,t)$, as predicted by Axiom 2. However, in some cases, $\text{dist}(s,t) = \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right)$: we think that this is due to the fact that, in our random graph models, the guarantee is $\mathcal{O}\left(e^{-n^{\varepsilon}}\right)$, and for $\varepsilon = 0.2$, this value is not very small (for instance, if $n = 10\,000$, $e^{-n^{\varepsilon}} = 0.012$). However, this value tends to 0 when n tends to infinity, and this is reflected in practice: indeed, the fit is better when the number of nodes is larger. Overall, we conclude that Axiom 2 is valid with good approximation on the networks in the dataset, and we conjecture that the correspondance is even stronger for bigger values of n.

Finally, we need to validate Axiom 3, which says that, given a node s, for "many" sets of nodes T, $|\{t \in T : \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right) < \text{dist}(s,t) + 2\}| \leq |T|n^{1-x-y+\varepsilon}$. Hence, we have chosen

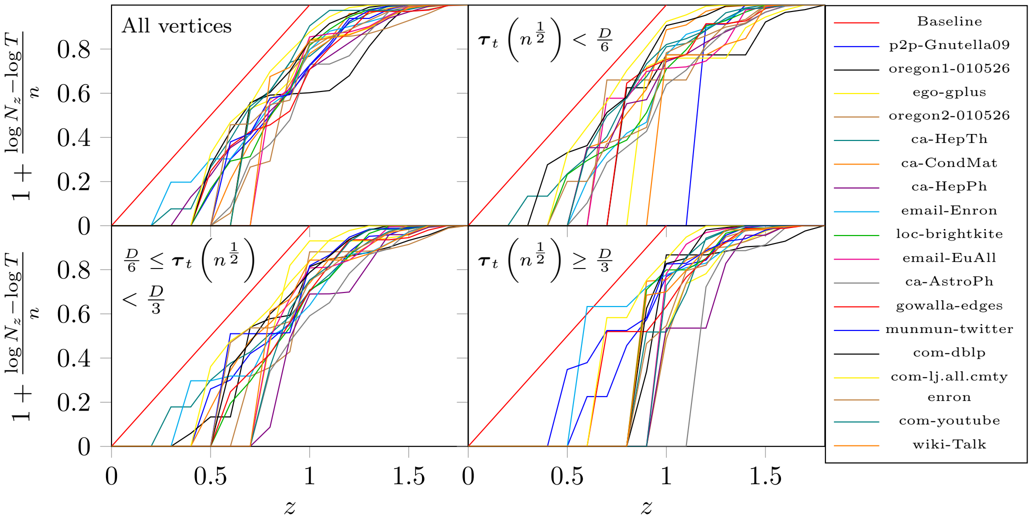


Figure 8.9. The values of $1 + \frac{\log \frac{N_z}{|T|}}{\log n}$, as a function of z.

a random node s and a random set T made by 10 000 nodes, and for each $t \in T$, we have computed $z_t = \min\{x + y : x > y, \boldsymbol{\tau}_s(n^x) + \boldsymbol{\tau}_t(n^y) < \text{dist}(s,t) + 2\}$. If the number N_z of nodes t such that $z_t < z$ is at most $|W|n^{-1+z+\varepsilon}$, then we can guarantee that the theorem holds for each x and y. Solving with respect to z, we want that $N_z \le |T|n^{-1+z+\varepsilon}$, that is, $\log \frac{N_z}{|T|} \le (-1 + z + \varepsilon) \log n$, that is, $z \ge 1 - \varepsilon + \frac{\log \frac{N_z}{|T|}}{\log n}$. Hence, Figure 8.9 shows the values of the function $1 + \frac{\log \frac{N_z}{|T|}}{\log n}$, for each graph in our dataset. Furthermore, since Axiom 3 also deals with sets T defined depending on $\boldsymbol{\tau}_t(n^x)$, we have also repeated the experiment on sets T containing only nodes t satisfying $0 \le \boldsymbol{\tau}_t\left(n^{\frac{1}{2}}\right) < \frac{D}{6}$, $\frac{D}{6} \le \boldsymbol{\tau}_t\left(n^{\frac{1}{2}}\right) < \frac{D}{3}$, $\boldsymbol{\tau}_t\left(n^{\frac{1}{2}}\right) > \frac{D}{3}$, where D is the diameter of the graph.

From the plot, it is clear the claim is satisfied even with $\varepsilon = 0$, by all but one case. Also the latter case is satisfied with a very small value of ε.

For the validation of Axiom 4, we rely on extensive studies that show that the degree distribution of many real-world graphs is power law (for more information, we refer to [125] and the references therein).

8.15 Validity of the Axioms in Random Graphs: Proof Sketch

In order to transform the axiomatic worst-case analyses into average-case analyses on random graphs, we use the following theorem, which is proved in Appendix A.

Theorem 8.32. *For each fixed $\varepsilon > 0$, Axioms 1 to 4 are satisfied in the random graphs defined in Section 8.1, a.a.s..*

In other words, for each $\varepsilon, \delta > 0$, there exists $n_{\varepsilon,\delta}$ such that the probability that a random graph with $n > n_{\varepsilon,\delta}$ nodes does not satisfy the axioms is at most $1 - \delta$. The remainder of this section sketches the proof of Theorem 8.32. It is quite easy to prove that Axiom 4 holds: indeed, it is enough to show that the degree of a node v is close to its weight ρ_v. For the other axioms, we study the size of neighbors of a given node s. We use three different techniques.

1. When $\boldsymbol{\gamma}^\ell(s) = |\boldsymbol{\Gamma}^\ell(s)|$ is small (say, smaller than n^ε), we show that the behavior of $\boldsymbol{\gamma}^\ell(s)$ is well approximated by a μ-distributed branching process, where μ is the residual

distribution of λ (see Definition 8.2). Furthermore, if s and t are two different nodes, and if $\boldsymbol{\gamma}^{\ell}(s)$ and $\boldsymbol{\gamma}^{\ell'}(t)$ are small, the behavior of $\boldsymbol{\Gamma}^{\ell}(s)$ and the behavior of $\boldsymbol{\Gamma}^{\ell'}(t)$ are "almost" independent.

2. When $\boldsymbol{\gamma}^{\ell}(s)$ is large, the branching process approximation and the independence do not hold anymore. We need a different technique: since $\boldsymbol{\gamma}^{\ell}(s) > n^{\varepsilon}$, a Chernoff-type probability bound gives guarantees of the form $e^{-n^{\varepsilon}}$, which is bigger than any polynomial in n. This way, we can prove very precise bounds on the size of $\boldsymbol{\gamma}^{\ell+1}(s)$ given the size of $\boldsymbol{\gamma}^{\ell}(s)$, and through a union bound we can show that these bounds hold for any node s.

3. When λ is a power law distribution with $1 < \beta < 2$, none of the previous results hold anymore (indeed, the residual distribution μ is not even defined). In this case, we directly prove that almost all shortest paths pass through a small set of nodes with degree $\Theta(n)$, and we only have to analyze the minimum ℓ such that $\boldsymbol{\gamma}^{\ell}(s)$ contains one of these nodes.

We provide some more details of the case $\beta > 2$. For small neighborhoods, we start by considering the residual distribution μ of λ (Definition 8.2), and by defining a μ-distributed branching process $\boldsymbol{\delta}^{\ell}(s)$ coupled with $\boldsymbol{\gamma}^{\ell}(s)$ (that is, $\boldsymbol{\gamma}^{\ell}(s)$ and $\boldsymbol{\delta}^{\ell}(s)$ are defined on the same probability space, and the probability that they are equal is high). Then, we analyze the size of $\boldsymbol{\delta}^{\ell}(s)$: if $M_1(\mu)$ is finite (or, equivalently, if $M_2(\lambda)$ is finite), it is well known [12] that the expected size of $\boldsymbol{\delta}^{\ell}(s)$ is $\boldsymbol{\delta}^{1}(s)M_1(\mu)^{\ell-1} = \deg(s)M_1(\mu)^{\ell-1}$. If λ is a power law distribution with $2 < \beta < 3$, and consequently μ is a power law distribution with $1 < \beta < 2$, the expected size of $\boldsymbol{\delta}^{\ell}(s)$ is infinite, but the typical size of $\boldsymbol{\delta}^{\ell}(s)$ is close to $\boldsymbol{\delta}^{1}(s)^{\left(\frac{1}{\beta-2}\right)^{\ell-1}} = \deg(s)^{\left(\frac{1}{\beta-2}\right)^{\ell-1}}$ (by typical size, we mean that size of $\boldsymbol{\delta}^{\ell}(s)$ is close to $\deg(s)^{\left(\frac{1}{\beta-2}\right)^{\ell-1}}$ a.a.s.). Hence, heuristically, we can estimate $\boldsymbol{\tau}_s\left(n^x\right)$, that is, the smallest ℓ such that $\boldsymbol{\delta}^{\ell}(s) > n^x$, by setting $\deg(s)M_1(\mu)^{\ell-1} = n^x$ if $M_1(\mu)$ is finite and strictly bigger than 1, and $\deg(s)^{\left(\frac{1}{\beta-2}\right)^{\ell-1}} = n^x$ if μ is power law with exponent $1 < \beta < 2$. Solving with respect to ℓ, $\boldsymbol{\tau}_s\left(n^x\right) \approx \log_{M_1(\mu)}\left(\frac{n^x}{\deg(s)}\right) + 1 \approx \log_{M_1(\mu)}\left(\frac{n^x}{\deg(s)}\right)$ in the first case, and $\boldsymbol{\tau}_s\left(n^x\right) \approx \log_{\frac{1}{\beta-2}}\left(\log_{\deg(s)} n^x\right) + 1 \approx \log_{\frac{1}{\beta-2}}\left(\frac{\log n^x}{\log(\deg(s))}\right)$, in line with the values in Table 8.2.

The branching process approximation lets us also approximate the deviations from these value. First of all, the probability that $\boldsymbol{\tau}_s\left(n^x\right)$ is much smaller than expected is not very interesting: for example, a node of degree 1 can be a neighbor of the maximum degree node, and in this case $\boldsymbol{\tau}_s\left(n^x\right) = 2$, if $x < \frac{1}{\beta-1}$ (in any case, we do not need a result in this direction for our analysis). Conversely, the probability that $\boldsymbol{\tau}_s\left(n^x\right)$ is bigger than expected is very interesting (Axiom 1): to bound it, we show that the "worst" that can happen is that $\boldsymbol{\delta}^{\ell}(s)$ is 1 for a long time, and then it grows as expected: under this assumption, the probability that $\boldsymbol{\tau}_s\left(n^x\right) > T\left(\deg(s) \rightarrow n^x\right) + k$ should be close to $\mu(1)^k$, that is, the probability that for k steps we find a node of degree 1. However, the probability we obtain is slightly different: for instance, in the first k steps, we can find with positive probability a node v with degree 2, such that the first child of v has degree 0 (hence, we still have one branch that "continues"). The probability of this event is $\mu(2)\mu(0)$, and consequently the probability that $\boldsymbol{\tau}_s\left(n^x\right) > T\left(\deg(s) \rightarrow n^x\right) + k$ should be at least $(\mu(1)+\mu(2)\mu(0))^k$. To take this possibility into account, we use the decomposition of the supercritical branching process [12, 1.D.12]: basically, we remove from the branching process all nodes that have a finite number of descendants. If the whole branching process is finite, then the starting node is not in the *giant component* of the graph, and we ignore it. Otherwise, after eliminating all finite branches, we obtain another branching process, with distribution η that depends on μ, and such that $\eta(0) = 0$. In this new process, we prove that the probability that $\boldsymbol{\tau}_s\left(n^x\right) > T\left(\deg(s) \rightarrow n^x\right) + k$ is close to $\eta(1)^k$, and we transfer this result to the μ distributed process.

Summarizing, we sketched the proof that the values appearing in Table 8.2 are correct, and that Axiom 1 holds, at least when x is small. For big values of x, the branching process approximation does not hold anymore: however, as soon as $\boldsymbol{\gamma}^{\ell}(s)$ is large enough, we can prove directly that $\boldsymbol{\gamma}^{\ell+1}(s) \approx \boldsymbol{\gamma}^{\ell}(s) M_1(\mu)$ if $M_1(\mu)$ is finite, and $\boldsymbol{\gamma}^{\ell+1}(s) \approx \boldsymbol{\gamma}^{\ell}(s)^{\frac{1}{\beta-2}}$ if λ is power law with exponent $2 < \beta < 3$, w.h.p.. This way, we can prove results on $\boldsymbol{\tau}_s\left(n^x\right)$ by proving the same results for $\boldsymbol{\tau}_s\left(n^y\right)$ for some small y, and extending the result to $\boldsymbol{\tau}_s\left(n^x\right)$ using this argument. This concludes the proof that the values appearing in Table 8.2 are correct, and that Axiom 1 holds.

Then, we need to prove that Axioms 2 and 3 hold: these axioms bound $\operatorname{dist}(s,t)$ with $\boldsymbol{\tau}_s\left(n^x\right)+\boldsymbol{\tau}_t\left(n^y\right)$. Let us assume that $\boldsymbol{\gamma}^{\ell}(s) = n^x$, and $\boldsymbol{\gamma}^{\ell'}(t) = n^y$: if all nodes are in $\boldsymbol{\Gamma}^{\ell}(s)$ with the same probability, $\boldsymbol{\Gamma}^{\ell}(s)$ will be a random subset of the set of nodes, and the probability that a node in $\boldsymbol{\Gamma}^{\ell}(s)$ is also in $\boldsymbol{\Gamma}^{\ell'}(t)$ is close to $\frac{\boldsymbol{\gamma}^{\ell'}(t)}{n} = \frac{1}{n^{1-y}}$. Hence, the probability that $\operatorname{dist}(s,t) \geq \ell + \ell'$ is related to the probability that $\boldsymbol{\Gamma}^{\ell}(s)$ does not intersect $\boldsymbol{\Gamma}^{\ell'}(t)$, which is close to $\left(1 - \frac{1}{n^{1-y}}\right)^{n^x} \approx e^{-n^{x+y-1}}$. For $x + y > 1$, this means that $\operatorname{dist}(s,t) \leq \ell + \ell'$ w.h.p., and this is very close to the statement of Axiom 2. For $x + y < 1$, $\boldsymbol{\Gamma}^{\ell}(s)$ does not intersect $\boldsymbol{\Gamma}^{\ell'}(t)$ with probability $e^{-n^{x+y-1}} \approx 1 - n^{x+y-1}$, and hence $\operatorname{dist}(s,t) \leq \ell + \ell'$ with probability close to n^{x+y-1}. The proof that Axiom 3 holds is then concluded by applying concentration inequalities, exploiting the fact that T is "enough random". Finally, Axiom 4 is well-known, and we follow existing proof techniques, based on concentration inequalities.

8.16 Bibliographic Notes

All the results in this section are published in our paper [35]. The models considered are well-known, and the definitions used in Section 8.1 are taken from [159]. The axioms in Section 8.2 are original, but similar results are available in previous papers (see e.g., Theorem 3.4 and Lemma 5.2 in [76] for the CM, [127] for the Norros-Reittu model, and [27] for similar related results). The results in Section 8.3 are collect and generalize to our framework several existing results on random graphs [158, 161, 127, 54, 162, 76, 27, 159, 160], and they prove some original results. Finally, in Section 8.15, we sketch the proof that the axioms are satisfied by the random graph models considered (full proof is available in Appendix A). The proof techniques take inspiration from existing results: for "big neighborhoods", we follow the approach in [54, 127] (see also [159, 160]), while the branching process approximation for "small neighborhoods" was already used in [127, 27] in IRG; similar techniques were used in [76] for the CM. The statement of the main theorem about branching processes is original (Theorem A.21), and it formalizes ideas that previously were only considered as intuitions [159, 160]. The techniques used and the results obtained in the case $1 < \beta < 2$ are original: the only known result is the average distance between two nodes [158].

Chapter 9

Conclusions and Open Problems

In this thesis, we have addressed the problem of designing subquadratic algorithms to compute metric quantities in graphs.

In the first part, we have proved that such problems do not admit subquadratic solutions, unless widely believed conjectures are false. In other words, we cannot significantly outperform trivial approaches in the worst-case.

In order to overcome these difficulties, at least in practice, we have developed several algorithms that are very efficient when tested on real-world networks, even if they provide no guarantee on the time-complexity. We have implemented these algorithms in widely-used graph libraries, and we have experimentally validated their performances.

In the last part of the thesis, we have provided a probabilistic framework where many of these algorithms can be evaluated and compared in a rigorous way. This framework does not only allow us to validate the performances of the new algorithms: it also shows the main properties of the input graph that we are using, and it can be used to prove average-case results on realistic graph models.

All these results open the way to a number of new research directions. In the field of worst-case reductions, an open problem is to find new relations between the Strong Exponential Time Hypothesis, the Orthogonal Vector conjecture, and the Hitting Set conjecture: in particular, it would be very interesting to find more evidences for the validity of the Hitting Set conjecture. However, obtaining such results might be very difficult: the problem is widely studied, and there are even some irreducibility results [46]. Among simpler open problem, one might be interested in adding new problem to the reduction network in Figure 3.1: for example, it would be very interesting to prove stronger bounds on the hardness of computing the hyperbolicity. Finally, in this work we have only considered exact algorithms, even if there are also several inapproximability results: it could also be interesting to extend our results in terms of hardness of approximations, especially the ones dealing with betweenness centrality.

In the field of designing practical algorithms, a standard open problem is to refine the existing algorithms, or design new algorithms that outperform the existing ones. In particular, for closeness centrality, it would be very interesting to try probabilistic algorithms: currently, all such algorithms are not suited to the ranking of nodes, because the number of iterations needed to obtain sensible guarantees is very large. However, with adaptive approaches, one might obtain good guarantees much faster, similarly to what we have done with betweenness centrality. Conversely, in the case of betweenness centrality, one might be interested in designing exact algorithms to rank the k most central nodes. The main difficulty of this task is that computing the betweenness of a single node is already hard, meaning that we have both to find upper bounds on the betweenness of all nodes, and lower bounds on the betweenness of single nodes (conversely, for closeness centrality, we only had to find lower bounds). Furthermore, it would be interesting to overcome the $\mathcal{O}(n^2)$ space needed by the HYP algorithm to compute the hyperbolicity (in practice, this is the main bottleneck of this algorithm). Finally, it would be interesting to extend the techniques developed in this thesis to address the computation of other metric quantities: other variations of closeness and betweenness centrality, new centrality measures, computation or approximation of the distance distribution, and so on.

Several problems remain open also in the field of probabilistic analyses, which is very recent. One of the possible research directions is to prove the axioms for more models, so that all our probabilistic analyses are valid in these new models, as well. A first and simple extension would be to lift some of the technical assumptions in Section 8.1: we believe that this task should not be difficult, since we already know what happens in more general cases [159, 160], and our proofs should be easily adaptable.

A more complicated task is to prove the axioms for new models. For example, we conjecture that our axioms hold in all Inhomogeneous Random Graphs [27], and not only Rank-1 Inhomogeneous Random Graphs: this is a much larger class of graphs, including, among other, all stochastic block models. We believe that this extension should be possible, by combining arguments in [27] with our proof techniques, also because the asymptotics for various graph quantities in Inhomogeneous Random Graphs are very similar to the asymptotics obtained in our framework. For the extension to other models, we believe that the task could be harder, because, in most existing models, the asymptotics for the metric quantities studied in this thesis are not known with enough precision, yet, despite a large amount of research. This means that, probably, the existing techniques are not sufficient to prove our axioms, and new techniques are needed.

Furthermore, one might be interested in extending this kind of analysis to directed graphs. We believe that, with few modifications, it is possible to adapt the axioms, and to analyze many of the algorithms discussed in this thesis. We also believe that it is not very difficult to generalize the current models to the directed case, in order to prove the new axioms. The main difficulty is that, currently, the generalizations of the models we have considered are not well-established in the literature: in particular, there is no standard way to choose the distribution of the in-degrees and the out-degrees (for example, it is not known whether only one of the two is usually power law, or they both are; also the dependency of in-degrees and out-degrees is not established). This means that, in order to extend this work, one has to propose new models, and show that the graphs generated by these models represent well real-world directed networks.

Another possible extension of our work is to provide new probabilistic analyses. A very interesting result would be to analyze our hyperbolicity algorithm, which seems to be harder to analyze than the other ones, because the current asymptotics for the hyperbolicity of random graphs are not precise enough. However, we believe that these asymptotics can be obtained by combining our axioms with known results on the "local" structure of the random graphs considered, which resembles a tree [159].

Building on this idea, one might be interested in improving the set of axioms, for example by adding an axiom saying that the number of edges in a (small enough) neighborhood is not much larger than the number of nodes. This would not only help us with the hyperbolicity, but it would also allow us to perform more probabilistic analyses only using the axioms, without relying on specific properties of the models considered (in other words, we would be able to remove some asterisks from Table 8.1).

Finally, an interesting open problem is to try different axioms, and study the differences: for example, we could use different neighborhood growth functions, in order to study the behavior of our algorithms on other kinds of networks, such as road networks. In this case, the neighborhood growth is not exponential, and we believe that we could make it linear. Furthermore, two neighborhoods are likely to touch each other only if they have size $\mathcal{O}(n)$, and not $\mathcal{O}(\sqrt{n})$ as in the case of complex networks.

In conclusion, this thesis opens several possible research directions, and we believe that, in the future, more light will be shed on the validity of these conjectures, and maybe some of them will be proved!

Appendix A

Proof of the Validity of the Axioms in Real-World Graphs

In this appendix, we prove Theorem 8.32, that states that the four axioms in Section 8.2 are a.a.s. satisfied if a graph is generated with the Configuration Model, or with Rank-1 Inhomogeneous Random Graph models. We follow the sketch in Section 8.15. In Appendix A.1 we state some basic lemmas that are used throughout this section, while in Appendix A.2 we analyze the size of $\boldsymbol{\gamma}^{\ell}(s)$ when $\boldsymbol{\gamma}^{\ell}(s)$ is "big" (at least n^{ε}). Then, Appendix A.3 completes Appendix A.2 by analyzing the size of $\boldsymbol{\gamma}^{\ell}(s)$ when $\boldsymbol{\gamma}^{\ell}(s)$ is small, using branching process approximation. Then, in Appendix A.4 we analyze separately the case $1 < \beta < 2$, which has a different behavior. Appendix A.5 develop tools to convert probabilistic results into results on the number of nodes satisfying a certain property. Finally, Appendix A.6 proves Theorem 8.32, relying on the results of all previous sections, and Appendix A.7 proves other results that are used in some analyses.

A.1 Probabilistic Preliminaries

In this section, we state some basic probabilistic theorems that are used in the proof of our main results. For a more thorough discussion and for their proof, we refer to [54].

Lemma A.1 (Multiplicative form of Chernoff bound). *Let $\boldsymbol{X}_1, \dots, \boldsymbol{X}_k$ be independent Bernoulli random variables, and let $\boldsymbol{S} = \sum_{i=1}^{k} \boldsymbol{X}_i$. Then,*

$$\mathbb{P}\left(\boldsymbol{S} < (1-\varepsilon)\mathbb{E}[\boldsymbol{S}]\right) \leq \left(\frac{e^{-\varepsilon}}{(1-\varepsilon)^{1-\varepsilon}}\right)^{\mathbb{E}[\boldsymbol{S}]}$$

$$\mathbb{P}\left(\boldsymbol{S} > (1+\varepsilon)\mathbb{E}[\boldsymbol{S}]\right) \leq \left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{1+\varepsilon}}\right)^{\mathbb{E}[\boldsymbol{S}]}.$$

Lemma A.2 (Hoeffding inequality). *Let $\boldsymbol{X}_1, \dots, \boldsymbol{X}_k$ be independent random variables such that $a_i < \boldsymbol{X}_i < b_i$ almost surely, and let $\boldsymbol{S} = \sum_{i=1}^{k} \boldsymbol{X}_i$. Then,*

$$\mathbb{P}\left(|\boldsymbol{S} - \mathbb{E}[\boldsymbol{S}]| > \lambda\right) \leq 2e^{-\frac{2\lambda^2}{\sum_{i=1}^{k} |b_i - a_i|^2}}$$

The next lemmas deal with supermartingales and submartingales, which are defined as follows.

Definition A.3. Let $\boldsymbol{X}_1, \dots, \boldsymbol{X}_k$ be a sequence of random variables, let $\mathcal{F}_1, \dots, \mathcal{F}_k$ be a sequence of σ-fields such that $\boldsymbol{X}_1, \dots, \boldsymbol{X}_i$ are $\mathcal{F}_i$-measurable. The sequence is a martingale if the conditional expectation $\mathbb{E}[\boldsymbol{X}_{i+1}|\mathcal{F}_i]$ is equal to $\boldsymbol{X}_i$, it is a supermartingale if $\mathbb{E}[\boldsymbol{X}_{i+1}|\mathcal{F}_i] \leq \boldsymbol{X}_i$, and it is a submartingale if $\mathbb{E}[\boldsymbol{X}_{i+1}|\mathcal{F}_i] \geq \boldsymbol{X}_i$.

The terms "submartingale" and "supermartingale" have not been used consistently in the literature, since in some works a supermartingale satisfies $\mathbb{E}[\boldsymbol{X}_{i+1}|\mathcal{F}_i] \geq \boldsymbol{X}_i$, and a submartingale satisfies $\mathbb{E}[\boldsymbol{X}_{i+1}|\mathcal{F}_i] \leq \boldsymbol{X}_i$ [54]. In this work, we use the most common definition.

Lemma A.4 (Azuma inequality for supermartingales)**.** *Let $\boldsymbol{X}_k$ be a supermartingale, and let us assume that $|\boldsymbol{X}_k - \boldsymbol{X}_{k+1}| < M$ almost surely. Then, $\mathbb{P}(\boldsymbol{X}_n - \boldsymbol{X}_0 \geq \lambda) \leq e^{-\frac{\lambda^2}{2nM^2}}$.*

Lemma A.5 (Azuma inequality for submartingales)**.** *Let $\boldsymbol{X}_k$ be a submartingale, and let us assume that $|\boldsymbol{X}_k - \boldsymbol{X}_{k+1}| < M$ almost surely. Then, $\mathbb{P}(\boldsymbol{X}_n - \boldsymbol{X}_0 \leq -\lambda) \leq e^{-\frac{\lambda^2}{2nM^2}}$.*

Lemma A.6 (strengthened version of Azuma inequality)**.** *Let $\boldsymbol{X}_k$ be a supermartingale associated with a filter $\mathcal{F}$, and assume that $\mathrm{Var}(\boldsymbol{X}_k|\mathcal{F}_{k-1}) \leq \sigma^2$, and $\boldsymbol{X}_k - \mathbb{E}(\boldsymbol{X}_k|\mathcal{F}_{k-1}) \leq M$. Then,*

$$\mathbb{P}\left(\boldsymbol{X}_k \geq \boldsymbol{X}_0 + \lambda\right) \leq e^{\frac{-\lambda^2}{2k\sigma^2 + \frac{M\lambda}{3}}}.$$

Finally, we need a technical lemma on the sum of power law random variables.

Lemma A.7. *Let $\boldsymbol{X} = \sum_{i=1}^k \boldsymbol{X}_i$, where k tends to infinity and the $\boldsymbol{X}_i$s are power law random variables with exponent $1 < \beta < 2$. Then, for each $c > 0$, $\mathbb{P}\left(\boldsymbol{X} > k^{\frac{1+c}{\beta-1}}\right) = \mathcal{O}(k^{-c})$.*

Proof. For each i, $\mathbb{P}\left(\boldsymbol{X}_i > k^{\frac{1+c}{\beta-1}}\right) = \mathcal{O}\left(\left(k^{-\frac{1+c}{\beta-1}}\right)^{\beta-1}\right) = \mathcal{O}\left(\frac{1}{k^{1+c}}\right)$, and consequently the probability that there exists i such that $X_i > k^{\frac{1+c}{\beta-1}}$ is $\mathcal{O}\left(k^{-c}\right)$.

Conditioned on $\boldsymbol{X}_i \leq k^{\frac{1+c}{\beta-1}}$ for each i,

$$\begin{aligned}
\mathbb{E}\left[\boldsymbol{X}\right] &= \mathbb{E}\left[\sum_{\ell=1}^{\infty} |\{i : \boldsymbol{X}_i > \ell\}|\right] \\
&= \mathbb{E}\left[\sum_{\ell=1}^{k^{\frac{1+c}{\beta-1}}} |\{i : \boldsymbol{X}_i > \ell\}|\right] \\
&= \sum_{\ell=1}^{k^{\frac{1+c}{\beta-1}}} \mathbb{E}\left[|\{i : \boldsymbol{X}_i > \ell\}|\right] \\
&= \sum_{\ell=1}^{k^{\frac{1+c}{\beta-1}}} \mathcal{O}(k\ell^{-\beta+1}) \\
&= \mathcal{O}\left(k^{1+\frac{(1+c)(2-\beta)}{\beta-1}}\right) \\
&= k^{\frac{1+c(2-\beta)}{\beta-1}}.
\end{aligned}$$

We conclude that

$$\begin{aligned}
\mathbb{P}\left(\boldsymbol{X} > k^{\frac{1+c}{\beta-1}}\right) &= \mathbb{P}\left(\boldsymbol{X} > k^{\frac{1+c}{\beta-1}} \middle| \exists i, \boldsymbol{X}_i > k^{\frac{1+c}{\beta-1}}\right) \mathbb{P}\left(\exists i, \boldsymbol{X}_i > k^{\frac{1+c}{\beta-1}}\right) \\
&\quad + \mathbb{P}\left(\boldsymbol{X} > k^{\frac{1+c}{\beta-1}} \middle| \forall i, \boldsymbol{X}_i < k^{\frac{1+c}{\beta-1}}\right) \mathbb{P}\left(\forall i, \boldsymbol{X}_i < k^{\frac{1+c}{\beta-1}}\right) \\
&\leq \mathcal{O}\left(k^{-c}\right) + \frac{1}{k^{\frac{1+c}{\beta-1}}} \mathbb{E}\left(\boldsymbol{X} > k^{\frac{1+c}{\beta-1}} \middle| \forall i, \boldsymbol{X}_i < k^{\frac{1+c}{\beta-1}}\right)
\end{aligned}$$

$$= \mathcal{O}\left(k^{-c} + k^{\frac{1+c(2-\beta)}{\beta-1} - \frac{1+c}{\beta-1}}\right)$$
$$= \mathcal{O}\left(k^{-c} + k^{-c}\right) = \mathcal{O}\left(k^{-c}\right)$$

by Markov inequality. □

A.2 Big Neighborhoods

First of all, let us define precisely the typical time needed by a node of degree d to reach size n^x. In Section 8.2, we defined $T(d \to n^x)$ as the smallest ℓ such that $\boldsymbol{\gamma}^\ell(s) > n^x$, and then we stated which are the typical values of $T(d \to n^x)$ in different regimes. In this section, we do the converse: we define $F(d \to S)$ as a function of the degree distributions, and we show that there is a high chance that $\boldsymbol{\gamma}^{(1-\varepsilon)F(d \to S)}(s) < S < \boldsymbol{\gamma}^{(1+\varepsilon)F(d \to S)}(s)$, if s is a node of degree d in the giant component.

Definition A.8. In the following for any $0 < d < S$, we denote by

$$F(d \to S) = \begin{cases} \log_{M_1(\mu)}\left(\frac{S}{d}\right) & \text{if } M_1(\mu) \text{ is finite.} \\ \log_{\frac{1}{\beta-2}}\left(\frac{\log S}{\log d}\right) & \text{if } \lambda \text{ is power law with } 2 < \beta < 3. \end{cases}$$

Following the intuitive proof, in this section we fix x, y bigger than ε, and we bound $\boldsymbol{\tau}_s(n^y) - \boldsymbol{\tau}_s(n^x)$. The main technique used is to prove that, w.h.p., each neighbor which is big enough satisfies some constraints, and these constraints imply bounds on $\boldsymbol{\tau}_s(n^y) - \boldsymbol{\tau}_s(n^x)$. More formally, we prove the following theorem.

Theorem A.9. *For each* $0 < x < y < 1$*,* $\boldsymbol{\tau}_s(n^y) - \boldsymbol{\tau}_s(n^x) \geq (1-\varepsilon)F(n^x \to n^y)$ *a.a.s., and* $\boldsymbol{\tau}_s(n^y) - \boldsymbol{\tau}_s(n^x) \leq (1+\varepsilon)F(n^x \to n^y)$ *w.h.p..*

The proof of this theorem is different for the CM and for IRG. In particular, the main tool used to prove this theorem is an estimate on $\boldsymbol{\gamma}^{\ell+1}(s)$ knowing $\boldsymbol{\gamma}^\ell(s)$: intuitively, in the CM, for each node in $\boldsymbol{\Gamma}^\ell(s)$ we count how many neighbors it has in $\boldsymbol{\Gamma}^{\ell+1}(s)$, while in IRG we count how many nodes outside $\boldsymbol{N}^\ell(s)$ have a neighbor in $\boldsymbol{\Gamma}^\ell(s)$.

A.2.1 Configuration Model

Let us assume that we know the structure of $\boldsymbol{N}^\ell(s)$ (that is, we consider all possible events $\boldsymbol{E}_i$ that describe the structure of $\boldsymbol{N}^\ell(s)$, and we prove bounds conditioned on $\boldsymbol{E}_i$; finally, though a union bound, we remove the conditioning). Let us define a random variable $\boldsymbol{\Delta}^\ell(s)$, which measures “how big a neighbor is”.

Definition A.10. Given a graph $G = (V, E)$ generated through the CM, we denote by $\boldsymbol{\Delta}^\ell(s)$ the set of stubs of nodes in $\boldsymbol{\Gamma}^\ell(s)$, not paired with stubs of nodes in $\boldsymbol{\Gamma}^{\ell-1}(s)$. We denote $\boldsymbol{\delta}^\ell(s) = |\boldsymbol{\Delta}^\ell(s)|$.

In order to make this analysis work, we need to assume that $\rho_{\boldsymbol{N}^\ell(s)} < n^{1-\varepsilon}$ and $\boldsymbol{\Delta}^\ell(s) > n^\varepsilon$.

Let us consider the following process: we sort all the stubs in $\boldsymbol{\Delta}^\ell(s)$, obtaining $a_1, \dots, a_{\boldsymbol{\delta}^\ell(s)}$, and, starting from a_1, we choose uniformly at random the “companion” of a_i among all free stubs (if a_i is already paired with a stub a_j for some $j < i$, we do not do anything). The companion of a_i can be one of the following:

1. a stub of a node v that already belongs to $\boldsymbol{\Gamma}^{\ell+1}(s)$ (because another stub of v was already paired with a stub in $\boldsymbol{\gamma}^\ell(s)$);

2. a stub of a “new” node;

3. another unpaired stub in $\boldsymbol{\Delta}^\ell(s)$.

Let us prove that the number of stubs in $\boldsymbol{\Delta}^{\ell}(s)$ that are paired with other stubs in $\boldsymbol{\Delta}^{\ell}(s)$ is small (Item 3): at each step, the probability that we choose one of these stubs is the ratio between the number of unpaired stubs in $\boldsymbol{\Delta}^{\ell}(s)$ with respect to the total number of unpaired stubs. Since $\rho_{\boldsymbol{N}^{\ell}(s)} < n^{1-\varepsilon}$, the number of unpaired stubs in $\boldsymbol{\Delta}^{\ell}(s)$ is at most $n^{1-\varepsilon}$, and the total number of unpaired stubs is at least $M - n^{1-\varepsilon} = M(1-o(1))$. Hence, the probability that we choose one of these stubs is at most $\frac{n^{1-\varepsilon}}{M} < n^{-\varepsilon}$. Let $\boldsymbol{X}_a$ be a Bernoulli random variable which is 1 if we pair a with another stub inside $\boldsymbol{\Delta}^{\ell}(s)$, 0 if a is already paired when we process it, or if it is paired outside $\boldsymbol{\Delta}^{\ell}(s)$ (observe that the number of nodes paired inside $\boldsymbol{\Delta}^{\ell}(s)$ is $2\sum_{a \in \boldsymbol{\Delta}^{\ell}(s)} \boldsymbol{X}_a$). We want to apply Azuma's inequality: first, we sort the stubs in $\boldsymbol{\Delta}^{\ell}(s)$, obtaining $a_1, \dots, a_{\boldsymbol{\delta}^{\ell}(s)}$. By the previous argument, $\boldsymbol{S}_k = \sum_{i=1}^{k} \boldsymbol{X}_{a_i} - kn^{-\varepsilon}$ is a supermartingale, and hence $\mathbb{P}(\boldsymbol{X}_k > \varepsilon k) \leq e^{-\frac{\varepsilon^2 k^2}{2k}}$: for $k = \boldsymbol{\Delta}^{\ell}(s)$, this probability is at most $e^{-\varepsilon^3 n^{\varepsilon}}$. In conclusion, w.h.p., at most $2\varepsilon\boldsymbol{\Delta}^{\ell}(s)$ stubs in $\boldsymbol{\Delta}^{\ell}(s)$ are paired to other stubs in $\boldsymbol{\Delta}^{\ell}(s)$.

Let us consider a stub a paired outside $\boldsymbol{\Delta}^{\ell}(s)$ with a random stub $\boldsymbol{a}'$: if the number of stubs that are already in $\boldsymbol{\Delta}^{\ell+1}(s)$ is at most $n^{1-\varepsilon^2}$, then the probability that $\boldsymbol{a}'$ is already in $\boldsymbol{\Delta}^{\ell+1}(s)$ is at most $n^{-\varepsilon^2}$. In order to solve the case where $\rho_{\boldsymbol{\Gamma}^{\ell+1}(s)} > n^{1-\varepsilon^2}$, let us assume that $\boldsymbol{\Delta}^{\ell}(s) < n^{1-\varepsilon}$: in this case, since the number of elements in $\boldsymbol{\Delta}^{\ell+1}(s)$ decreases at most by 1 at each step, $\boldsymbol{\Delta}^{\ell+1}(s) \geq n^{1-\varepsilon^2} - n^{1-\varepsilon} \geq n^{1-\varepsilon}$.

Hence, the case that "almost always" occurs is that the new stub $\boldsymbol{a}'$ belongs to a "new" node. Relying on this, we can lower bound $\boldsymbol{\gamma}^{\ell+1}(s)$: by definition, $\boldsymbol{\gamma}^{\ell+1}(s) \leq \boldsymbol{\delta}^{\ell}(s)$, and we want to prove that $\boldsymbol{\gamma}^{\ell+1}(s) \geq (1-\varepsilon)\boldsymbol{\delta}^{\ell}(s)$. Since the number of stubs in $\boldsymbol{\Delta}^{\ell}(s)$ paired with other stubs in $\boldsymbol{\Delta}^{\ell}(s)$ is negligible w.h.p., we can write $\boldsymbol{\gamma}^{\ell+1}(s) = \sum_{i=1}^{(1-\varepsilon)\boldsymbol{\delta}^{\ell}(s)} \boldsymbol{X}_i$, where $\boldsymbol{X}_i = 1$ with probability at least $1-n^{-\varepsilon^2}$, 0 otherwise (note that the $\boldsymbol{X}_i$s are not independent, but if $\boldsymbol{\Delta}^{\ell+1}(s) < n^{1-\varepsilon}$, then $\mathbb{P}(\boldsymbol{X}_i = 1) \geq 1 - n^{-\varepsilon^2}$, as before). We want to apply Azuma's inequality: $\boldsymbol{S}_k = \sum_{i=1}^{k} \boldsymbol{X}_i - k(1-n^{-\varepsilon^2})$ is a submartingale, and hence $\mathbb{P}(\boldsymbol{S}_k < -\varepsilon k) \leq e^{-\frac{\varepsilon^2 k^2}{2k}}$: for $k = \boldsymbol{\delta}^{\ell}(s)$, this probability is at most $e^{-\varepsilon^3 n^{\varepsilon}}$. Hence, w.h.p., $\boldsymbol{X}_i \geq k(1-n^{-\varepsilon^2}) - \varepsilon k \geq (1-2\varepsilon)k$, and for $k = \boldsymbol{\delta}^{\ell}(s)$ we have proved the following lemma.

Lemma A.11. *Given a random graph $G = (V, E)$ generated through the CM and a node $s \in V$, if $\boldsymbol{\delta}^{\ell}(s) > n^{\varepsilon}$ and $\rho_{\boldsymbol{N}^{\ell}(s)} < n^{1-\varepsilon}$, then $(1-2\varepsilon)\boldsymbol{\delta}^{\ell}(s) \leq \boldsymbol{\gamma}^{\ell+1}(s) \leq \boldsymbol{\delta}^{\ell}(s)$ w.h.p..*

Corollary A.12. *For each node s, let $\boldsymbol{\tau}'_s(S)$ be the smallest integer such that $\boldsymbol{\delta}^{\ell}(s) > S$. Then, for each $0 < x < 1$, $\boldsymbol{\tau}'_s(n^x) + 1 \leq \boldsymbol{\tau}_s(n^x) \leq \boldsymbol{\tau}'_s\left(\frac{n^x}{1-\varepsilon}\right) + 1$ w.h.p..*

Proof. For the first inequality, if $\ell = \boldsymbol{\tau}_s(n^x)$, $\boldsymbol{\delta}^{\ell-1}(x) \geq \boldsymbol{\gamma}^{\ell}(s) \geq n^x$. For the second inequality, for each $i < \ell - 1$, $n^x > \boldsymbol{\gamma}^{i+1}(s) \geq (1-\varepsilon)\boldsymbol{\delta}^{i}(s)$ by the previous lemma. Hence, $\boldsymbol{\tau}'_s\left(\frac{n^x}{1-\varepsilon}\right)$ cannot be smaller than $\ell - 1$. □

Hence, in order to understand $\boldsymbol{\tau}_s(n^x) - \boldsymbol{\tau}_s(n^y)$, we may as well understand $\boldsymbol{\tau}'_s(n^x) - \boldsymbol{\tau}'_s(n^y)$, and we do it by estimating $\boldsymbol{\delta}^{\ell+1}(s)$ from $\boldsymbol{\delta}^{\ell}(s)$. As before, $\boldsymbol{\delta}^{\ell+1}(s) = \sum_{a \in \boldsymbol{\Delta}^{\ell}(s)} \boldsymbol{Y}_a$, where $\boldsymbol{Y}_a$ is 0 if the stub $\boldsymbol{a}$ paired with a is in $\boldsymbol{\Delta}^{\ell}(s)$, -1 if $\boldsymbol{a}$ is in $\boldsymbol{\Gamma}^{\ell+1}(s)$, otherwise it the number of stubs of the node of $\boldsymbol{a}$, minus one (because $\boldsymbol{a}$ is not in $\boldsymbol{\Delta}^{\ell+1}(s)$). By definition, the distribution of $\boldsymbol{Y}_a$ is very close to μ (more specifically, $\sum_{k=0}^{\infty} |\mu(k) - \mathbb{P}(\boldsymbol{Y}_a = k)| < \frac{1}{n^{\varepsilon}}$).

It remains to estimate this sum: we need to do it differently for upper and lower bounds, and for different regimes of β.

Lower bound, $2 < \beta < 3$**.**The probability that at least one of the $\boldsymbol{Y}_a$ is at least $\boldsymbol{\delta}^{\ell}(s)^{\frac{1-\varepsilon}{\beta-2}}$ is close to $\mathbb{P}\left(\mu > \boldsymbol{\delta}^{\ell}(s)^{\frac{1-\varepsilon}{\beta-2}}\right)$, because, w.h.p., no visited node can have weight bigger than $\boldsymbol{\delta}^{\ell}(s)^{\frac{1-\varepsilon}{\beta-2}}$ (otherwise, there would be a $\ell' < \ell$ such that $\boldsymbol{\delta}^{\ell'}(s) \geq \boldsymbol{\delta}^{\ell}(s)^{\frac{1-\varepsilon}{\beta-2}}$). Hence, the probability that one of the $\boldsymbol{Y}_a$s is at least $\boldsymbol{\delta}^{\ell}(s)^{\frac{1-\varepsilon}{\beta-2}}$ is $\Theta\left(\frac{1}{\boldsymbol{\delta}^{\ell}(s)^{\frac{1-\varepsilon}{\beta-2}(\beta-2)}}\right) = \Theta\left(\boldsymbol{\delta}^{\ell}(s)^{-1+\varepsilon}\right)$. We want to apply Azuma's inequality to prove that at least one of $\boldsymbol{Y}_a$s is bigger than $\boldsymbol{\delta}^\ell(s)^{\frac{1-\varepsilon}{\beta-2}}$. Let us number the stubs in $\boldsymbol{\Delta}^\ell(s)$, obtaining $a_1, \dots, a_{\boldsymbol{\delta}^\ell(s)}$, and let $\boldsymbol{S}_k = \sum_{i=0}^k \boldsymbol{Y}'_{a_i} - ck\boldsymbol{\delta}^\ell(s)^{-1+\varepsilon}$, where $\boldsymbol{Y}'_{a_i} = 1$ if $\boldsymbol{Y}_{a_i} > \boldsymbol{\delta}^\ell(s)^{\frac{1-\varepsilon}{\beta-2}}$, 0 otherwise, and c is a small enough constant, so that $\boldsymbol{S}_k$ is a submartingale. Furthermore, $\mathrm{Var}(\boldsymbol{Y}'_{a_i}) \leq \mathbb{E}[(\boldsymbol{Y}'_{a_i})^2] = \mathbb{E}[\boldsymbol{Y}'_{a_i}]) = \mathcal{O}\left(\boldsymbol{\delta}^\ell(s)^{\frac{1-\varepsilon}{\beta-2}}\right)$. Then, by the strengthened version of Azuma's inequality (Lemma A.6), if $k = \boldsymbol{\delta}^\ell(s)$, $\mathbb{P}\left(\boldsymbol{S}_k \leq \frac{c}{2}k\boldsymbol{\delta}^\ell(s)^{-1+\varepsilon}\right) \leq e^{-\Omega\left(\frac{k^2\boldsymbol{\delta}^\ell(s)^2}{2k\boldsymbol{\delta}^\ell(s)+k\boldsymbol{\delta}^\ell(s)}\right)} \leq e^{-\Omega(\boldsymbol{\delta}^\ell(s)^\varepsilon)} \leq e^{-n^{\varepsilon^3}}$. Hence, w.h.p., $\boldsymbol{S}_{\boldsymbol{\delta}^\ell(s)} \geq \frac{c}{2}k\boldsymbol{\delta}^\ell(s)^{-1+\varepsilon} > 0$, and consequently there is i such that $\boldsymbol{Y}'_{a_i} \neq 0$. This means that, for each i, $\boldsymbol{\delta}^{\ell+i}(s) \geq \boldsymbol{\delta}^\ell(s)^{\left(\frac{1-\varepsilon}{\beta-2}\right)^i}$.

Upper bound, $2 < \beta < 3$.By Lemma A.7, since μ is a power law with exponent $\beta - 1$, and $2 < \beta < 3$, the probability that $\sum_{a \in \boldsymbol{\Delta}^\ell(s)} \boldsymbol{Y}_a$ is bigger than $k^{\frac{1+\varepsilon}{\beta-2}}$ is at most $\mathcal{O}(k^{-\varepsilon}) = \mathcal{O}\left(n^{-\varepsilon^2}\right)$. Consequently, by a union bound, $\boldsymbol{\delta}^{\ell+i}(s) \leq \boldsymbol{\delta}^\ell(s)^{\left(\frac{1+\varepsilon}{\beta-2}\right)^i}$ for each $i < n^{\varepsilon^3}$, with probability $1 - o(1)$.

Lower bound, $\beta > 3$.We cannot apply directly Azuma's inequality to say that $\boldsymbol{\delta}^{\ell+1}(s)$ is close to $\mathbb{E}[\boldsymbol{\delta}^{\ell+1}(s)] = (1 + o(1))M_1(\mu)\boldsymbol{\delta}^\ell(s)$, because $\boldsymbol{Y}_a$ can assume very large values. However, we can "cut the distribution", by defining $\boldsymbol{Y}'_a = \boldsymbol{Y}_a$ if $\boldsymbol{Y}_a < N$, 0 otherwise. If N is big enough, $\mathbb{E}[\boldsymbol{Y}'_a] > M_1(\mu) - \varepsilon$. By a straightforward application of Azuma's inequality (Lemma A.5), $\boldsymbol{\delta}^{\ell+1}(s) \geq \sum_{a \in \boldsymbol{\Delta}^\ell(s)} \boldsymbol{Y}_a \geq (1 - \varepsilon)(M_1(\mu) - \varepsilon)\boldsymbol{\delta}^\ell(s)$ w.h.p.. Consequently, $\boldsymbol{\delta}^{\ell+i}(s) \geq (M_1(\mu) - \mathcal{O}(\varepsilon))^i\boldsymbol{\delta}^\ell(s)$, w.h.p..

Upper bound, $\beta > 3$.The expected value of $\boldsymbol{\delta}^{\ell+i}(s)$ is at most $(M_1(\mu) + \varepsilon)^i\boldsymbol{\delta}^\ell(s)$: by a straightforward application of Markov inequality, we conclude that

$$\mathbb{P}\left(\boldsymbol{\delta}^{\ell+i}(s) > (M_1(\mu) + \varepsilon)^i\boldsymbol{\delta}^\ell(s)n^\varepsilon\right) \leq n^{-\varepsilon}.$$

Proof of Theorem A.9, CM. By Corollary A.12, $\boldsymbol{\tau}'_s(n^x) + 1 \leq \boldsymbol{\tau}_s(n^x) \leq \boldsymbol{\tau}'_s\left((1+\varepsilon)n^x\right) + 1$. Hence, $\boldsymbol{\tau}'_s(n^y) - \boldsymbol{\tau}'_s\left((1+\varepsilon)n^x\right) \leq \boldsymbol{\tau}_s(n^y) - \boldsymbol{\tau}_s(n^x) \leq \boldsymbol{\tau}'_s\left(n^y(1+\varepsilon)\right) - \boldsymbol{\tau}'_s(n^x)$.

If we apply the lower bounds with $i = F\left(Z^0 \to S\right)(1 + \varepsilon')$, $\ell = \boldsymbol{\tau}'_s(n^x)$, we obtain the following.

- If $2 < \beta < 3$, either $\boldsymbol{n}^{\ell+j}(s) > n^{1-\varepsilon}$ for some $j < i$, or, w.h.p., $\boldsymbol{\delta}^{\ell+i}(s) \geq \boldsymbol{\delta}^\ell(s)^{\left(\frac{1-\varepsilon}{\beta-2}\right)^i} \geq n^{x\left(\frac{1-\varepsilon}{\beta-2}\right)^{(1+\varepsilon')\log_{\frac{1}{\beta-2}}\frac{y}{x}}} = n^{xe^{\log\left(\frac{y}{x}\right)(1+\varepsilon')\frac{\log\frac{1-\varepsilon}{\beta-2}}{\log\frac{1}{\beta-2}}}} \geq (1+\varepsilon)n^y$ if ε is small enough with respect to ε'. In both cases, $\boldsymbol{\tau}_s(n^y) - \boldsymbol{\tau}_s(n^x) \leq \boldsymbol{\tau}'_s\left((1+\varepsilon)n^y\right) - \boldsymbol{\tau}'_x(n^x) \leq F\left(Z^0 \to S\right)(1 + \varepsilon')$. With a very similar computation, one can conclude that $\boldsymbol{\tau}_s(n^y) - \boldsymbol{\tau}_s(n^x) \geq F\left(Z^0 \to S\right)(1 - \varepsilon')$ a.a.s. (the only difference is how to handle the case where $\boldsymbol{n}^{\ell+j}(s) > n^{1-\varepsilon}$: to this purpose, it is enough to observe that if $n^x < n^y < n^{1-\varepsilon}$, for the whole process $\boldsymbol{n}^{\ell+j}(s) < n^y < n^{1-\varepsilon}$).

- If $\beta > 3$, as before, either $\boldsymbol{n}^{\ell+j}(s) > n^{1-\varepsilon}$ for some $j < i$, or, w.h.p., $\boldsymbol{\delta}^{\ell+i}(s) \geq \boldsymbol{\delta}^\ell(s)(M_1(\mu) - \varepsilon)^i \geq \boldsymbol{\delta}^\ell(s)(M_1(\mu) - \varepsilon)^{(1+\varepsilon)\log_{M_1(\mu)} n^{y-x}} = n^x e^{\log(n^{y-x})(1+\varepsilon')\frac{M_1(\mu)-\varepsilon}{M_1(\mu)}} \geq n^y(1+\varepsilon)$ if ε is small enough with respect to ε'. We conclude that $\boldsymbol{\tau}_s(n^y) - \boldsymbol{\tau}_s(n^x) \leq \boldsymbol{\tau}'_s\left((1+\varepsilon)n^y\right) - \boldsymbol{\tau}'_x(n^x) \leq F\left(Z^0 \to S\right)(1 + \varepsilon')$ w.h.p.. A similar computation yields $\boldsymbol{\tau}_s(n^y) - \boldsymbol{\tau}_s(n^x) \geq \boldsymbol{\tau}'_s(n^y) - \boldsymbol{\tau}'_x\left((1+\varepsilon)n^x\right) \leq F\left(Z^0 \to S\right)(1 - \varepsilon')$ a.a.s..

□

To conclude this section, we prove a stronger upper bound in the case $\beta > 3$, which is used in three of our probabilistic analyses.

Lemma A.13. *Assume that $\boldsymbol{\delta}^{\ell}(s) > d_{\max} n^{\varepsilon}$, where $d_{\max}$ is the maximum degree in the graph, and that $M_1(\mu)$ is finite. Then, w.h.p., $\boldsymbol{\delta}^{\ell+1}(s) \leq \boldsymbol{\delta}^{\ell}(s)(M_1(\mu)+\varepsilon)$.*

Proof. We want to apply Azuma's inequality as in the lower bound. More precisely, $\boldsymbol{\delta}^{\ell+1}(s) \leq \sum_{i=1}^{\boldsymbol{\delta}^{\ell}(s)} \boldsymbol{Y}_{a_i}$, where $\mathbb{E}\left[\boldsymbol{Y}_{a_i}\right]$ is at most $M_1(\mu)+\varepsilon$, conditioned on the values of $\boldsymbol{Y}_{a_j}$ for each $j<i$. Hence, $\sum_{i=1}^{k} \boldsymbol{Y}_{a_i} - k(M_1(\mu)+\varepsilon)$ is a supermartingale, and $\boldsymbol{Y}_{a_i} < n^{\frac{1}{\beta-1}} < n^{\frac{1}{2}-\varepsilon}$ for ε small enough. By Azuma's inequality (Lemma A.4), $\mathbb{P}\left(\boldsymbol{Y}_k \leq \varepsilon k\right) \leq e^{-\frac{\varepsilon^2 k^2}{2 k d_{\max}}} \leq e^{-\varepsilon^3 n^{\varepsilon}}$ for $k = \boldsymbol{\delta}^{\ell}(s) > d_{\max} n^{\varepsilon}$. We proved that, w.h.p., $\boldsymbol{\delta}^{\ell+1}(s) - \boldsymbol{\delta}^{\ell}(s)(M_1(\mu)+\varepsilon) = \sum_{i=1}^{\boldsymbol{\delta}^{\ell}(s)} \boldsymbol{Y}_{a_i} - \boldsymbol{\delta}^{\ell}(s)(M_1(\mu)+\varepsilon) \leq \varepsilon \boldsymbol{\delta}^{\ell}(s)$, and consequently $\boldsymbol{\delta}^{\ell+1}(s) \leq \boldsymbol{\delta}^{\ell}(s)(M_1(\mu)+2\varepsilon)$. □

Combining this lemma with Lemma A.11, we obtain the following corollary.

Corollary A.14. *Assume that $d_{\max} n^{\varepsilon} < \boldsymbol{\gamma}^{\ell}(s) < n^{1-\varepsilon}$, where $d_{\max}$ is the maximum degree in the graph, and that $M_1(\mu)$ is finite. Then, w.h.p., $\boldsymbol{\gamma}^{\ell+1}(s) \leq \boldsymbol{\gamma}^{\ell}(s)(M_1(\mu)+\varepsilon)$.*

Corollary A.15. *For each node v, and for each $0<x<y<1$ such that $d_{\max} < n^{x-\varepsilon}$, $\boldsymbol{\tau}_v\left(n^y\right) - \boldsymbol{\tau}_v\left(n^x\right) \geq (1-\varepsilon)\log_{M_1(\mu)} n^{y-x}$, w.h.p..*

A.2.2 Inhomogeneous Random Graphs

Let us assume that we know the structure of $\boldsymbol{N}^{\ell}(s)$. Following the proof for the CM, we define the auxiliary quantity $\boldsymbol{\delta}^{\ell}(s) = \rho_{\boldsymbol{\gamma}^{\ell}(s)}$. Again, we need to assume that $\rho_{\boldsymbol{N}^{\ell}(s)} < n^{1-\varepsilon}$ and that $\boldsymbol{\delta}^{\ell}(s) > n^{\varepsilon}$.

Let w be a node with weight at most $\frac{n^{1-\varepsilon}}{\boldsymbol{\delta}^{\ell}(s)}$, outside $\boldsymbol{N}^{\ell}(s)$:

$$\begin{aligned}
P\left(w \notin \boldsymbol{\gamma}^{\ell+1}(s)\right) &= \prod_{v \in \boldsymbol{\gamma}^{\ell}(s)} \left(1 - f\left(\frac{\rho_v \rho_w}{M}\right)\right) \\
&= \prod_{v \in \boldsymbol{\gamma}^{\ell}(s)} \left(1 - (1+o(1))\left(\frac{\rho_v \rho_w}{M}\right)\right) \\
&= e^{-\sum_{v \in \boldsymbol{\gamma}^{\ell}(s)} (1+o(1))\left(\frac{\rho_v \rho_w}{M}\right)} \\
&= e^{-(1+o(1))\left(\frac{\boldsymbol{\delta}^{\ell}(s)\rho_w}{M}\right)} \\
&= 1 - (1+o(1))\left(\frac{\boldsymbol{\delta}^{\ell}(s)\rho_w}{M}\right).
\end{aligned}$$

Hence, $\mathbb{P}\left(w \in \boldsymbol{\gamma}^{\ell+1}(s)\right) = (1+o(1))\left(\frac{\boldsymbol{\delta}^{\ell}(s)\rho_w}{M}\right)$, and $\boldsymbol{\gamma}^{\ell+1}(s) = \sum_{w \notin \boldsymbol{N}^{\ell}(s)} \boldsymbol{X}_w$, where the $\boldsymbol{X}_w$s are independent Bernoulli random variables with success probability $(1+o(1))\left(\frac{\boldsymbol{\delta}^{\ell}(s)\rho_w}{M}\right)$ if this quantity is much smaller than 1, otherwise $\mathcal{O}(1)$. We want to compute the number of nodes in $\boldsymbol{\Gamma}^{\ell+1}(s)$, knowing $\boldsymbol{\delta}^{\ell}(s)$: first, we observe that the number of nodes with weight at least $\frac{n^{1-\varepsilon}}{\boldsymbol{\delta}^{\ell}(s)}$ is $\mathcal{O}\left(n\left(\frac{\boldsymbol{\delta}^{\ell}(s)}{n^{1-\varepsilon}}\right)^{\beta-1}\right) = \mathcal{O}\left(\boldsymbol{\delta}^{\ell}(s)\frac{n^{\varepsilon(\beta-1)}\boldsymbol{\delta}^{\ell}(s)^{\beta-2}}{n^{\beta-2}}\right) = \mathcal{O}\left(\boldsymbol{\delta}^{\ell}(s) n^{\varepsilon(\beta-1)-\sqrt{\varepsilon}(\beta-2)}\right) = o(\boldsymbol{\delta}^{\ell}(s))$, assuming $\boldsymbol{\delta}^{\ell}(s) < n^{1-\sqrt{\varepsilon}}$, and we can safely ignore these nodes. By the multiplicative form of Chernoff bound (Lemma A.1), if $\boldsymbol{S} = \sum_{w \notin \boldsymbol{N}^{\ell}(s), \rho_w < \frac{n^{1-\varepsilon}}{\boldsymbol{\delta}^{\ell}(s)}} \boldsymbol{X}_w$:

$$\begin{aligned}
\mathbb{P}\left(\boldsymbol{S} < (1-\varepsilon)\mathbb{E}[\boldsymbol{S}]\right) &\leq \left(\frac{e^{-\varepsilon}}{(1-\varepsilon)^{1-\varepsilon}}\right)^{\mathbb{E}[\boldsymbol{S}]} \\
&\leq e^{(-\varepsilon-(1-\varepsilon)\log(1-\varepsilon))n^{\varepsilon}}
\end{aligned}$$

$$\le e^{-\varepsilon^3 n^\varepsilon}$$

$$\begin{aligned}\mathbb{P}\left(\boldsymbol{S} > (1+\varepsilon)\mathbb{E}[\boldsymbol{S}]\right) &\le \left(\frac{e^\varepsilon}{(1+\varepsilon)^{1+\varepsilon}}\right)^{\mathbb{E}[\boldsymbol{S}]}\\ &\le e^{(\varepsilon+(1+\varepsilon)\log(1+\varepsilon))n^\varepsilon}\\ &\le e^{-\varepsilon^3 n^\varepsilon}\end{aligned}$$

if ε is small enough. By changing the value of ε with $\sqrt{\varepsilon}$, we have proved the following lemma.

Lemma A.16. *Assume that* $\rho_{\boldsymbol{N}^\ell(s)} < n^{1-\varepsilon}$ *and that* $\boldsymbol{\delta}^\ell(s) > n^\varepsilon$. *Then,* $(1-\varepsilon)\boldsymbol{\delta}^\ell(s) \le \boldsymbol{\gamma}^{\ell+1}(s) \le (1+\varepsilon)\boldsymbol{\delta}^\ell(s)$ *w.h.p..*

Corollary A.17. *For each node* s*, let* $\boldsymbol{\tau}'_s(S)$ *be the smallest integer such that* $\boldsymbol{\delta}^\ell(s) > S$*. Then, for each* $0 < x < 1$, $\boldsymbol{\tau}'_s\left((1-\varepsilon)n^x\right) + 1 \le \boldsymbol{\tau}_s(n^x) \le \boldsymbol{\tau}'_s\left((1+\varepsilon)n^x\right) + 1$.

As in the CM, we need to estimate $\boldsymbol{\tau}'_s(n^x) - \boldsymbol{\tau}'_s(n^y)$. To this purpose, we compute $\rho_{\boldsymbol{\Gamma}^{\ell+1}(s)}$ knowing $\boldsymbol{\delta}^\ell(s)$. Using the previous notations, $\rho_{\boldsymbol{\Gamma}^{\ell+1}(s)} = \sum_{w \notin \boldsymbol{N}^\ell(s)} \rho_w \boldsymbol{X}_w$, and we estimate this sum by considering separately upper and lower bounds, and the different possible values of β.

Lower bound, $2 < \beta < 3$**.**Let us consider all nodes with weight at least $\boldsymbol{\delta}^\ell(s)^{\frac{1-\varepsilon}{\beta-2}}$. The probability that one of these nodes is connected to a node in $\boldsymbol{\Gamma}^\ell(s)$ is $\Theta\left(\boldsymbol{\delta}^\ell(s)^{\frac{1-\varepsilon}{\beta-2}+1} n^{-1}\right) = \Theta\left(\boldsymbol{\delta}^\ell(s)^{\frac{\beta-1-\varepsilon}{\beta-2}} n^{-1}\right)$. Through a straigthforward application of the multiplicative for of Chernoff bound (Lemma A.1), since there are $\Theta\left(n\boldsymbol{\delta}^\ell(s)^{-\frac{1-\varepsilon}{\beta-2}}\right)$ such nodes, we can prove that there is at least one node with weight at least $\boldsymbol{\delta}^\ell(s)^{\frac{1-\varepsilon}{\beta-2}}$ which is connected to $\boldsymbol{\Gamma}^\ell(s)$, w.h.p.. This means that $\rho_{\boldsymbol{\gamma}^{\ell+1}(s)} \ge \boldsymbol{\delta}^\ell(s)^{\frac{1-\varepsilon}{\beta-2}}$, and consequently, by a union bound, $\rho_{\boldsymbol{\Gamma}^{\ell+i}(s)} \ge \rho_{\boldsymbol{\Gamma}^\ell(s)}^{\left(\frac{1-\varepsilon}{\beta-2}\right)^i}$ for each i such that $\rho_{\boldsymbol{\Gamma}^{\ell+i}(s)} < n^{1-\varepsilon}$.

Upper bound, $2 < \beta < 3$**.**Let us consider all nodes with weight at least $\boldsymbol{\delta}^\ell(s)^{\frac{1+\varepsilon}{\beta-2}}$. The probability that one of these nodes is connected to a node in $\boldsymbol{\Gamma}^\ell(s)$ is

$$\Theta\left(\boldsymbol{\delta}^\ell(s)^{\frac{1+\varepsilon}{\beta-2}+1} n^{-1}\right) = \Theta\left(\boldsymbol{\delta}^\ell(s)^{\frac{\beta-1+\varepsilon}{\beta-2}} n^{-1}\right).$$

Since there are $\Theta\left(n\boldsymbol{\delta}^\ell(s)^{-\frac{(1+\varepsilon)(\beta-1)}{\beta-2}}\right)$ such nodes, by a union bound, the probability that none of these nodes is connected to a node in $\boldsymbol{\Gamma}^\ell(s)$ is $1 - \Theta\left(\boldsymbol{\delta}^\ell(s)^{-\frac{\beta-1+\varepsilon-(1+\varepsilon)(\beta-1)}{\beta-2}}\right) = 1 - \Theta\left(n^{-\varepsilon^2}\right)$. Conditioned on this event, the expected value of $\rho_{\boldsymbol{\Gamma}^{\ell+1}(s)}$ is at most

$$\boldsymbol{\delta}^\ell(s) \sum_{\rho_v < \boldsymbol{\delta}^\ell(s)^{\frac{1+\varepsilon}{\beta-2}}} \frac{\rho_v^2}{n} = \Theta\left(\boldsymbol{\delta}^\ell(s)^{1+\frac{(1+\varepsilon)(3-\beta)}{\beta-2}}\right) = \Theta\left(\boldsymbol{\delta}^\ell(s)^{\frac{1+\varepsilon(3-\beta)}{\beta-2}}\right).$$

By Markov inequality, with probability at least $1 - n^{-\varepsilon^3}$, $\rho_{\boldsymbol{\Gamma}^{\ell+1}(s)} \le \boldsymbol{\delta}^\ell(s)^{\frac{1+\varepsilon(3-\beta)}{\beta-2}+\varepsilon} = \boldsymbol{\delta}^\ell(s)^{\frac{1}{\beta-2}+\varepsilon}$. As a consequence, $\rho_{\boldsymbol{\Gamma}^{\ell+i}(s)} \le \boldsymbol{\delta}^\ell(s)^{\left(\frac{1}{\beta-2}\right)^i}$ for each $i < n^{\varepsilon^4}$, a.a.s..

Lower bound, $\beta > 3$**.**We want to apply Hoeffding's inequality to prove that, if $\boldsymbol{\delta}^\ell(s) > n^\varepsilon$, $\boldsymbol{\delta}^{\ell+1}(s) \ge (1-\varepsilon)\mathbb{E}[\boldsymbol{\delta}^{\ell+1}(s)] \ge (1-2\varepsilon)M_1(\mu)\boldsymbol{\delta}^\ell(s)$. To this purpose, let N be a big constant (to be chosen later): $\boldsymbol{\delta}^{\ell+1}(s) = (1+o(1))\sum_{w \in V} \rho_w \boldsymbol{X}_w \ge (1+o(1))\sum_{\rho_w < N} \rho_w \boldsymbol{X}_w$. By Hoeffding's inequality (Lemma A.2), w.h.p., $\sum_{\rho_w < N} \rho_w \boldsymbol{X}_w$ is at least $(1-\varepsilon)\mathbb{E}\left[\sum_{\rho_w < N} \rho_w \boldsymbol{X}_w\right]$: if we choose N big enough, the latter value is at least $(1-2\varepsilon)M_1(\mu)$. This means that $\boldsymbol{\delta}^{\ell+1}(s) \ge (1-2\varepsilon)M_1(\mu)\boldsymbol{\delta}^\ell(s)$ w.h.p., and by a union bound $\boldsymbol{\delta}^{\ell+i}(s) \ge (1-2\varepsilon)^i M_1(\mu)^i \boldsymbol{\delta}^\ell(s)$.

Upper bound, $\beta > 3$.Conditioned on $\boldsymbol{\delta}^\ell(s)$, the expected value of $\boldsymbol{\delta}^{\ell+1}(s)$ is at most

$$\begin{aligned}\sum_{w \notin \boldsymbol{N}^\ell(s)} \rho_w \mathbb{E}[\boldsymbol{X}_w] &= (1+o(1)) \sum_{w \in V} \rho_w \frac{\boldsymbol{\delta}^\ell(s)\rho_w}{M} \\ &= (1+o(1))\boldsymbol{\delta}^\ell(s)\frac{M_2(\lambda)}{M_1(\lambda)} \\ &= (1+o(1))\boldsymbol{\delta}^\ell(s)M_1(\mu) \leq (M_1(\mu)+\varepsilon)\boldsymbol{\delta}^\ell(s).\end{aligned}$$

By Markov inequality, $\mathbb{P}\left(\boldsymbol{\delta}^{\ell+i}(s) > (M_1(\mu)+\varepsilon)^i \boldsymbol{\delta}^\ell(s) n^\varepsilon\right) \leq n^{-\varepsilon}$.

Proof of Theorem A.9, IRG. By Corollary A.17,

$$\boldsymbol{\tau}'_s\left((1-\varepsilon)n^x\right) + 1 \leq \boldsymbol{\tau}_s\left(n^x\right) \leq \boldsymbol{\tau}'_s\left((1+\varepsilon)n^x\right) + 1.$$

Hence, $\boldsymbol{\tau}'_s\left((1-\varepsilon)n^y\right) - \boldsymbol{\tau}'_s\left((1+\varepsilon)n^x\right) \leq \boldsymbol{\tau}_s\left(n^y\right) - \boldsymbol{\tau}_s\left(n^x\right) \leq \boldsymbol{\tau}'_s\left(n^y(1+\varepsilon)\right) - \boldsymbol{\tau}'_s\left(n^x(1-\varepsilon)\right)$. By the aforementioned lower bounds on $\boldsymbol{\delta}^{d+i}(s)$, the conclusion follows. □

As before, we conclude this section by proving a stronger upper bound in the case $\beta > 3$, which is used in two of our probabilistic analyses.

Lemma A.18. *Assume that $\boldsymbol{\delta}^\ell(s) > d_{\max} n^\varepsilon$, where $d_{\max}$ is the maximum degree in the graph, and assume that $\beta > 3$. Then, w.h.p., $\boldsymbol{\delta}^{\ell+1}(s) \leq \boldsymbol{\delta}^\ell(s)(M_1(\mu)+\varepsilon)$.*

Proof. As in the lower bound, we write $\boldsymbol{\delta}^{\ell+1}(s) = \rho_{\boldsymbol{\Gamma}^{\ell+1}(s)} \leq \sum_{w \in V} \rho_w \boldsymbol{X}_w$, where $\boldsymbol{X}_w$ is a Bernoulli random variable with success probability $1 - \prod_{v \in \boldsymbol{\Gamma}^\ell(s)} \left(1 - f\left(\frac{\rho_v \rho_w}{n}\right)\right) = 1 - \prod_{v \in \boldsymbol{\Gamma}^\ell(s)} e^{-(1+o(1))\frac{\rho_v\rho_w}{n}} = 1 - e^{-(1+o(1))\frac{\rho_{\boldsymbol{\Gamma}^\ell(s)}\rho_w}{n}} = (1+o(1))\left(\frac{\rho_{\boldsymbol{\Gamma}^\ell(s)}\rho_w}{n}\right)$. Hence, $\mathbb{E}\left[\sum_{w \in V} \rho_w \boldsymbol{X}_w\right] = (1+o(1))\left(\sum_{w \in V} \frac{\rho_w^2 \rho_{\boldsymbol{\Gamma}^\ell(s)}}{n}\right) \leq \left(M_1(\mu)+\varepsilon\right)\boldsymbol{\delta}^\ell(s)$. A simple application of Hoeffding's inequality (Lemma A.2) lets us conclude, since $\rho_w < d_{\max}$ for each w. □

Combining this lemma with Lemma A.11, we obtain the following corollary.

Corollary A.19. *Assume that $d_{\max} n^\varepsilon < \boldsymbol{\Gamma}^\ell(s) < n^{1-\varepsilon}$, where $d_{\max}$ is the maximum degree in the graph, and that $M_1(\mu)$ is finite. Then, w.h.p., $\boldsymbol{\gamma}^{\ell+1}(s) \leq \boldsymbol{\gamma}^\ell(s)(M_1(\mu)+\varepsilon)$.*

Corollary A.20. *For each node v, and for each $0 < x < y < 1$ such that $d_{\max} < n^{x-\varepsilon}$, $\boldsymbol{\tau}_v\left(n^y\right) - \boldsymbol{\tau}_v\left(n^x\right) \geq (1-\varepsilon)\log_{M_1(\mu)} n^{y-x}$, w.h.p..*

A.3 Small Neighborhoods

Using Theorem A.9, we reduced ourselves to prove Axiom 1 and the bounds in Table 8.2 for some small values of x. We start the proof by formalizing Item 1 in Section 8.15: the main tool is the relationship between the size $\boldsymbol{\gamma}^\ell(s)$ of a neighbor of a node s and a μ-distributed branching process $\boldsymbol{\delta}^\ell(s)$.

Theorem A.21. *Let $\boldsymbol{G} = (V, \boldsymbol{E})$ be a random graph with degree distribution λ, let μ be the corresponding residual distribution, and let $s \in V$. There are multisets $\boldsymbol{\Delta}^\ell(s)$ of nodes such that:*

1. *the cardinality $\boldsymbol{\delta}^\ell(s)$ of $\boldsymbol{\Delta}^\ell(s)$ is a μ-distributed branching process;*
2. *if $\boldsymbol{\Theta}^\ell(s) = \cup_{i=0}^\ell \boldsymbol{\Delta}^i(s)$, then*

$$\mathbb{P}\left(\boldsymbol{\Gamma}^{\ell+1}(s) = \boldsymbol{\Delta}^{\ell+1}(s) \middle| \boldsymbol{\Theta}^\ell(s) = \boldsymbol{N}^\ell(s)\right) = \mathcal{O}\left(\rho^2_{\boldsymbol{\Theta}^\ell(s)}\frac{M_2(\lambda)}{n}\right).$$

A.3.1 Proof for the Configuration Model

For each node v, let us fix a set of stubs $a_{v,1}, a_{v,\rho_v}$ attached to v (let A the set of all stubs).

We define a procedure that generates a random pairing of stubs (and, hence, a graph), by fixing a node s and pairing stubs "in increasing order of distance from s", obtaining something similar to a breadth-first search (BFS). This way, in order to understand the structure of $\boldsymbol{n}^\ell(s)$, we only have to consider the first ℓ levels of this BFS, and we may ignore how all other stubs are paired.

The procedure keeps the following information:

- a partial function $\boldsymbol{\alpha} : A \to A$, that represent a partial pairing of stubs;
- for each ℓ, a set $\boldsymbol{I}^\ell$ of all stubs at distance ℓ from s;
- for each ℓ, a set $\boldsymbol{\Delta}^\ell(s)$ of all nodes at distance ℓ from s.

The random part of our procedure is given by a set of random variables $\{\boldsymbol{b}_{a,i}\}_{a \in A, i \in \mathbb{N}}$, whose range is the set of stubs. Informally, stub a "wants to be paired" with stub $\boldsymbol{b}_{a,0}$, if available, otherwise $\boldsymbol{b}_{a,1}$, and so on. We assume that, for each a, $A = \{b \in A : \boldsymbol{b}_{a,i} = b \text{ for some } i\}$ is infinite (this event occurs with probability 1).

Definition A.22. The procedure `P1` starts with $\boldsymbol{\alpha}_1$ as the empty function, $\boldsymbol{\Delta}_1^0(s) = \{s\}$, $\boldsymbol{\Delta}_1^\ell(s) = \emptyset$, $\boldsymbol{I}_1^0 = \{a_{s,1}, \ldots, a_{s,\rho_s}\}, \boldsymbol{I}_1^\ell = \emptyset$ for each $\ell > 0$. Then, for increasing values of ℓ, for each stub a in $\boldsymbol{I}_1^\ell$ (any order is fine):

1. it sets $\boldsymbol{b}$ as the first $\boldsymbol{b}_{a,i}$ such that $\boldsymbol{\alpha}_1(\boldsymbol{b}_{a,i})$ is undefined;
2. it defines $\boldsymbol{\alpha}_1(a) = \boldsymbol{b}$, $\boldsymbol{\alpha}_1(\boldsymbol{b}) = a$;
3. if $\boldsymbol{b}$ is not in $\boldsymbol{I}_1^\ell$ for any ℓ:
   (a) it adds to $\boldsymbol{I}_1^{\ell+1}$ all stubs of $V(\boldsymbol{b})$ except $\boldsymbol{b}$;
   (b) it adds $V(\boldsymbol{b})$ to $\boldsymbol{\Delta}_1^{\ell+1}(s)$, and it sets $V(a)$ as the father of $V(\boldsymbol{b})$;
4. else:
   (a) it removes $\boldsymbol{b}$ from $\boldsymbol{I}_1^\ell$.

The procedure ends when $\boldsymbol{I}_1^\ell$ is empty, and all remaining stubs are paired uniformly at random (so that $\boldsymbol{\alpha}_1$ becomes a total function).

At the end, the pairing $\boldsymbol{\alpha}_1$ is uniformly distributed, because, at each step, we choose the "companion" of a stub uniformly among all unpaired stubs. Furthermore, if we consider the graph obtained with the pairing $\boldsymbol{\alpha}_1$, the set $\boldsymbol{\Delta}_1^\ell(s)$ is the set of nodes at distance ℓ from s.

Now, we define another similar, simplified procedure. This time, we let $\boldsymbol{b}$ be any stub, and we do not test if $\boldsymbol{b}$ is not in $\boldsymbol{I}_2^\ell$ for some ℓ, and we add it anyway to $\boldsymbol{I}_2^{\ell+1}$. This way, $\boldsymbol{I}_2^\ell$ and $\boldsymbol{\Delta}^\ell(s)$ become multisets (that is, repetitions are allowed).

Definition A.23. The procedure `P2` starts with $\boldsymbol{\Delta}^0(s) = \{s\}$, $\boldsymbol{\Delta}^\ell(s) = \emptyset$ for each $d > 0$, $\boldsymbol{I}_2^0 = \{a_{s,1}, \ldots, a_{s,\rho_s}\}, \boldsymbol{I}_2^\ell = \emptyset$ for each $d > 0$. Then, for increasing values of ℓ, for each stub a in $\boldsymbol{I}_2^\ell$ (any order is fine):

1. it sets $\boldsymbol{b} = \boldsymbol{b}_{a,0}$, and it shifts the $\boldsymbol{b}_{a,i}$s by one;
2. it defines $\boldsymbol{\alpha}_2(a) = \boldsymbol{b}$, $\boldsymbol{\alpha}_2(\boldsymbol{b}) = a$ (in case, it replaces its value);
3. in any case:
   (a) it adds to $\boldsymbol{I}_2^{\ell+1}$ all stubs of $V(\boldsymbol{b})$ except $\boldsymbol{b}$;
   (b) it adds $V(\boldsymbol{b})$ to $\boldsymbol{\Delta}^{\ell+1}(s)$, and it sets $V(a)$ as the father of $V(\boldsymbol{b})$;

The procedure ends when $\boldsymbol{I}_2^\ell$ is empty, or continues indefinitely.

Thanks to these simplifications, we are able to prove that, in procedure P2, the cardinality $\boldsymbol{\delta}^\ell(s)$ of $\boldsymbol{\Delta}^\ell(s)$ is a branching process, starting from $\boldsymbol{\delta}^1(s) = \rho_s$.

Lemma A.24. *In procedure* P2*, the stochastic process $\boldsymbol{\delta}^\ell(s)$ is a μ-distributed branching process, starting from $\boldsymbol{\delta}^1(s) = \rho_s$.*

Proof. First of all, we observe that $\boldsymbol{\delta}^{\ell+1}(s) = |\boldsymbol{I}_2^\ell|$, so it is enough to prove that $\boldsymbol{I}_2^\ell$ is a branching process. It is clear that $\boldsymbol{I}_2^0 s = \rho_s$. Moreover, $\boldsymbol{I}_2^{\ell+1} = \sum_{a \in \boldsymbol{I}_2^\ell} \rho_{V(\boldsymbol{b}_{a,0})} - 1$. Let $\boldsymbol{X}_i : \rho_{V(\boldsymbol{b}_{a,0})} - 1$: since the $\boldsymbol{b}_{a,0}$s are independent, also the $\boldsymbol{X}_i$s are independent, and $\mathbb{P}(\boldsymbol{X}_i = k) = \mathbb{P}(\rho_{V(\boldsymbol{b}_i)}) = k + 1 = \frac{(k+1)n\lambda(k+1)}{\sum_{j=0}^\infty jn\lambda(j)} = \mu(k)$. □

With this choice of $\boldsymbol{\delta}^\ell(s)$, we have proved the first part of Theorem A.21. Now, we have to bound the probability that $\boldsymbol{\Delta}^{\ell+1}(s) \neq \boldsymbol{\Gamma}^{\ell+1}(s)$, assuming $\boldsymbol{\Theta}^\ell(s) = \boldsymbol{N}^\ell(s)$: the next lemma gives a bound which is stronger than the bound in Theorem A.21, and it concludes the proof.

Lemma A.25. *Let $\boldsymbol{N}^\ell(s) = \bigsqcup_{i=0}^\ell \boldsymbol{\Gamma}^\ell(s)$, $\boldsymbol{\Theta}^\ell(s) = \bigsqcup_{i=0}^\ell \boldsymbol{\Delta}^\ell(s)$, where $\bigsqcup$ denotes the disjoint union, and let $\boldsymbol{n}^\ell(s) = |\boldsymbol{N}^\ell(s)|, \boldsymbol{\vartheta}^\ell(s) = |\boldsymbol{\Theta}^\ell(s)|$. Assuming $\boldsymbol{\Theta}^\ell(s) = \boldsymbol{N}^\ell(s)$, the probability that $\boldsymbol{\Delta}^{\ell+1}(s) \neq \boldsymbol{\Gamma}^{\ell+1}(s)$ is at most $\frac{1}{n} 2\rho^2_{\boldsymbol{\Theta}^\ell(s)}$.*

Proof. Let us pair the stubs in $\boldsymbol{\Delta}^\ell(s)$ one by one, and try to bound the probability that a stub is paired "differently". More formally, for each stub a paired by the procedures P1, P2, we bound the probability that a is the first stub that was paired differently (so that we can assume that the pairing of all other stubs was the same). In particular, both procedures choose the companion $\boldsymbol{b}$ of a as $\boldsymbol{b}_{a,0}$, if $\boldsymbol{b}_{a,0}$ is not already in $\boldsymbol{\Theta}^\ell(s)$, and it is not already paired with a stub in $\boldsymbol{\Theta}^\ell(s)$. Hence, the probability that the companion of a is the same in the two procedures is at most the probability that $\boldsymbol{b}_{a,0}$ is not in $\boldsymbol{\Theta}^\ell(s)$, and it is not already paired with a stub in $\boldsymbol{\Theta}^\ell(s)$. This probability is at most $\frac{2\rho_{\boldsymbol{\Theta}^\ell(s)}}{M}$. By a union bound, we can estimate that the probability that at least a stub is paired differently in the two procedures is at most $\rho_{\boldsymbol{\Delta}^\ell(s)} \frac{2\rho_{\boldsymbol{\Theta}^\ell(s)}}{M} \leq \frac{2\rho^2_{\boldsymbol{\Theta}^\ell(s)}}{M} \leq \frac{2\rho^2_{\boldsymbol{\Theta}^\ell(s)}}{n}$. □

A.3.2 Proof for Rank-1 Inhomogeneous Random Graphs

In this section, we prove Theorem A.21 in IRG. Let us fix a set V of nodes, let us fix the expected degree ρ_v of each node $v \in V$, and let us fix $M = \sum_{v \in V} \rho_v$.

Let s be any node, and let us define a procedure that considers edges "in increasing order of distance from s", obtaining something similar to a BFS. This way, in order to understand the structure of $\boldsymbol{n}^\ell(s)$, we only have to consider the first d levels of this BFS, and we may ignore all other edges.

We denote by $\{\boldsymbol{X}_{v,w}\}$ a random variable that has value 1 if the edge (v, w) exists, 0 otherwise. Note that the $\boldsymbol{X}_{v,w}$s are independent Bernoulli random variables with success probability $f\left(\frac{\rho_v \rho_w}{M}\right)$.

Definition A.26. The procedure P1 starts with $\boldsymbol{\Delta}_1^0(s) = \{s\}$, $\boldsymbol{\Delta}_1^\ell(s) = \emptyset$. Then, for increasing values of d, for each node $v \in \boldsymbol{\Delta}_1^\ell(s)$:

1. for each node w such that $\boldsymbol{X}_{v,w} = 1$:
   (a) if w is not in $\bigcup_{i=0}^\infty \boldsymbol{\Delta}_1^i(s)$:
      i. add w to $\boldsymbol{\Delta}_1^{\ell+1}(s)$.

The procedure ends when $\boldsymbol{\Delta}_1^\ell(s)$ is empty.

There are two reasons why this procedure is not a branching process. The first and simplest problem, that occurred also in the CM, is that we need to check that w is “a new node”, and hence there is dependance between the number of children of different nodes. However, there is also a more subtle problem: if we assume that there is no dependency, we can informally write $\boldsymbol{\delta}_1^{\ell+1}(s) = \sum_{v \in \boldsymbol{\Delta}_1^{\ell}(s)} \sum_{w \in V} \boldsymbol{X}_{v,w}$. To turn this into a branching process, we have to link $\boldsymbol{\delta}_1^{\ell}(s)$ with $\boldsymbol{\delta}_1^{\ell+1}(s)$, but the previous formula also depends on which nodes are in $\boldsymbol{\Delta}_1^{\ell}(s)$. If we condition on which nodes we find in $\boldsymbol{\Delta}_1^{\ell}(s)$, then the random variables $\sum_{w \in V} \boldsymbol{X}_{v,w}$ are not identically distributed. So, we have to fix $\boldsymbol{\delta}_1^{\ell}(s)$, write $\boldsymbol{\Delta}_1^{\ell}(s) = \{\boldsymbol{v}_1, \dots, \boldsymbol{v}_k\}$, where $\boldsymbol{v}_i$ is a random variable taking values in V, and then set $\boldsymbol{\delta}_1^{\ell+1}(s) = \sum_{i=1}^{\boldsymbol{\Delta}_1^{\ell}(s)} \sum_{w \in V} \boldsymbol{X}_{\boldsymbol{v}_i,w}$. Now, the random variables $\sum_{w \in V} \boldsymbol{X}_{\boldsymbol{v}_i,w}$ are i.i.d., but the distribution of the weight of $\boldsymbol{v}_i$ (and hence the distribution of the sum) depends on $\boldsymbol{\gamma}^{\ell}(s)$ in general, so we do not obtain a branching process. Summarizing, the second problem is that we need somehow to choose $\boldsymbol{\delta}^{\ell}(s)$ before choosing which nodes are in $\boldsymbol{\Delta}^{\ell}(s)$, then choose $\boldsymbol{v}_i$ in a way that is independent from $\boldsymbol{\delta}^{\ell}(s)$. It turns out that, if the random variables $\boldsymbol{X}_{v,w}$ are Poisson-distributed, actually the two choices can be made independent. So, in the second procedure we define, we do not only ignore already visited nodes, but we also define new Poisson random variables $\boldsymbol{Y}_{v,w}$ such that $\boldsymbol{Y}_{v,w} = \boldsymbol{X}_{v,w}$ with probability $1 - \mathcal{O}\left(\left(\frac{\rho_v \rho_w}{M}\right)^2\right)$, and we work with $\boldsymbol{Y}_{v,w}$.

Lemma A.27. *Given a Bernoulli random variable $\boldsymbol{X}$ with success probability $f(p)$, it is possible to define a random variable $\boldsymbol{Y} = \text{Poisson}(p)$ such that $\boldsymbol{X} = \boldsymbol{Y}$ with probability $1 - \mathcal{O}\left(p^2\right)$.*

Proof. Let $\boldsymbol{E}_0, \boldsymbol{E}_1$ be the events $\boldsymbol{X} = 0$, $\boldsymbol{X} = 1$. Let $\boldsymbol{E}_0'$ be an event such that $\boldsymbol{E}_0' \subseteq \boldsymbol{E}_0$ or $\boldsymbol{E}_0' \supseteq \boldsymbol{E}_0$, and $\mathbb{P}(\boldsymbol{E}_0') = \mathbb{P}(\text{Poisson}(p) = 0)$: we define $\boldsymbol{Y} = 0$ in $\boldsymbol{E}_0'$. Similarly, let $\boldsymbol{E}_1'$ be an event such that $\boldsymbol{E}_1' \subseteq \boldsymbol{E}_1$ or $\boldsymbol{E}_1' \supseteq \boldsymbol{E}_1$, $\boldsymbol{E}_1' \cap \boldsymbol{E}_0' = \emptyset$, and $\mathbb{P}(\boldsymbol{E}_1') = \mathbb{P}(\text{Poisson}(p) = 1)$. We define $\boldsymbol{Y} = 1$ in $\boldsymbol{E}_1'$. Then, we cover the rest of the space as we wish.

We know that $\boldsymbol{Y} = \boldsymbol{X}$ on $\boldsymbol{E}_0 \cap \boldsymbol{E}_0'$ and on $\boldsymbol{E}_1 \cap \boldsymbol{E}_1'$: let us prove that the probability of these events is $1 - \mathcal{O}\left(p^2\right)$. Indeed, the probability of $\boldsymbol{E}_0$ is $1 - f(p) = 1 - p + \mathcal{O}(p^2)$, the probability of $\boldsymbol{E}_1$ is $p + \mathcal{O}(p^2)$, the probability of $\boldsymbol{E}_0'$ is $e^{-p} = 1 - p + \mathcal{O}(p^2)$, and the probability of $\boldsymbol{E}_1'$ is $pe^{-p} = p + \mathcal{O}(p^2)$. In any case, $\mathbb{P}((\boldsymbol{E}_0 \cap \boldsymbol{E}_0') \cup (\boldsymbol{E}_1 \cap \boldsymbol{E}_1')) = \min(\mathbb{P}(\boldsymbol{E}_0), \mathbb{P}(\boldsymbol{E}_0')) + \min(\mathbb{P}(\boldsymbol{E}_1), \mathbb{P}(\boldsymbol{E}_1')) = 1 - p + \mathcal{O}(p^2) + p + \mathcal{O}(p^2) = 1 + \mathcal{O}(p^2)$. □

Definition A.28. The procedure `P2` starts with $\boldsymbol{\Delta}^0(s) = \{s\}$, $\boldsymbol{\Delta}^{\ell}(s) = \emptyset$. Then, for increasing values of ℓ, for each node $v \in \boldsymbol{\Delta}^{\ell}(s)$:

1. for each node w:
   (a) in any case:
      i. add $\boldsymbol{Y}_{v,w}$ times w to $\boldsymbol{\Delta}^{\ell+1}(s)$;
      ii. replace $\boldsymbol{Y}_{v,w}$ with another Poisson $\left(\frac{\rho_v \rho_w}{M}\right)$ random variable, independent from all previous events.

The procedure ends when $\boldsymbol{\Delta}^{\ell}(s)$ is empty, or it continues forever.

Theorem A.29. *The cardinality $\boldsymbol{\delta}^{\ell}(s)$ of $\boldsymbol{\Delta}^{\ell}(s)$ is a μ-distributed branching process, starting from $\boldsymbol{\Delta}^1(s) = \deg(s)$.*

Proof. In this procedure, we got rid of the dependencies between different zones of the branching tree. Hence, if $\boldsymbol{\Delta}^{\ell}(s) = \{v_1, \dots, v_{\boldsymbol{\delta}^{\ell}(s)}\}$, we formalize the previous computation by saying that $\boldsymbol{\delta}^{\ell+1}(s) = \sum_{i=1}^{\boldsymbol{\delta}^{\ell}(s)} \sum_{w \in V} \boldsymbol{Y}_{\boldsymbol{v}_i,w} = \sum_{i=1}^{\boldsymbol{\delta}^{\ell}(s)} \sum_{w \in V} \text{Poisson}\left(\frac{\rho_{\boldsymbol{v}_i} \rho_w}{M}\right) = \sum_{i=1}^{\boldsymbol{\delta}^{\ell}(s)} \text{Poisson}\left(\rho_{\boldsymbol{v}_i}\right)$.

It only remains to prove that the probability that $\mathbb{P}\left(\rho_{\boldsymbol{v}_i} = k\right) = \frac{k\lambda(k)}{M_1(\lambda)}$, independently from $\boldsymbol{\Delta}^{\ell}(s)$. We need the following facts:

- $\boldsymbol{\delta}^{\ell}(s) = \sum_{v \in \boldsymbol{\delta}^{\ell-1}(s)} \sum_{w \in V} \text{Poisson}\left(\frac{\rho_v \rho_w}{M}\right) = \text{Poisson}\left(\rho_{\boldsymbol{\delta}^{\ell-1}(s)}\right)$ (in the following, we denote $\eta = \rho_{\boldsymbol{\delta}^{\ell-1}(s)}$);

- if $\boldsymbol{T}_u$ is the number of times that node u appears in $\boldsymbol{\Delta}^\ell(s)$, and $\vartheta = \frac{\rho_u \rho_{\boldsymbol{\Delta}^{\ell-1}(s)}}{M}$,

$$\begin{aligned}\boldsymbol{T}_u &= \sum_{v \in \boldsymbol{\delta}^{\ell-1}(s)} \text{Poisson}\left(\frac{\rho_u \rho_v}{M}\right) \\ &= \text{Poisson}\left(\frac{\rho_u \rho_{\boldsymbol{\Delta}^{\ell-1}(s)}}{M}\right) \\ &= \text{Poisson}(\vartheta);\end{aligned}$$

- conditioned on $\boldsymbol{T}_u = k$,

$$\begin{aligned}\boldsymbol{\delta}^\ell(s) - k &= \sum_{v \in \boldsymbol{\Delta}^{\ell-1}(s)} \sum_{w \in V - \{u\}} \text{Poisson}\left(\frac{\rho_v \rho_w}{M}\right) \\ &= \text{Poisson}\left(\rho_{\boldsymbol{\Delta}^{\ell-1}(s)} \frac{M - \rho_u}{M}\right) \\ &= \text{Poisson}(\eta - \vartheta).\end{aligned}$$

Using these three results, we can prove that:

$$\begin{aligned}\mathbb{P}\left(\boldsymbol{T}_u = k | \boldsymbol{\delta}^\ell(s) = h\right) &= \frac{\mathbb{P}(\boldsymbol{\delta}^\ell(s) = h | \boldsymbol{T}_u = k)\mathbb{P}(\boldsymbol{T}_u = k)}{\mathbb{P}(\boldsymbol{\delta}^\ell(s) = h)} \\ &= \frac{\mathbb{P}(\text{Poisson}(\eta - \vartheta) = h - k)\mathbb{P}(\text{Poisson}(\vartheta) = k)}{\text{Poisson}(\eta) = h} \\ &= \frac{e^{-(\eta-\vartheta)} \frac{(\eta-\vartheta)^{h-k}}{(h-k)!} e^{-\vartheta} \frac{\vartheta^k}{k!}}{e^{-\eta} \frac{\eta^h}{h!}} \\ &= \frac{h!}{k!(h-k)!} \left(\frac{\vartheta}{\eta}\right)^k \left(1 - \frac{\vartheta}{\eta}\right)^{h-k} \\ &= \binom{h}{k} \left(\frac{\rho_u}{M}\right)^k \left(1 - \frac{\rho_u}{M}\right)^{h-k}\end{aligned}$$

Hence, the probability that u appears k times in our process is exactly the probability that u appears k times if we select $\boldsymbol{\delta}^\ell(s)$ nodes, by picking u with probability $\frac{\rho_u}{M}$. Summing over all nodes u with weight k, $\mathbb{P}\left(\rho_{\boldsymbol{v}_i} = k\right) = n\lambda(k)\frac{k}{M} = \frac{k\lambda(k)}{M_1(\lambda)}$. This concludes the proof: indeed, $\boldsymbol{\delta}^{\ell+1}(s) = \sum_{i=1}^{\boldsymbol{\delta}^\ell(s)} \text{Poisson}(\rho_{\boldsymbol{v}_i})$, and $\mathbb{P}\left(\rho_{\boldsymbol{v}_i} = k\right) = \frac{k\lambda(k)}{M_1(\lambda)}$: hence, Poisson$(\rho_{\boldsymbol{v}_i})$ is μ-distributed. □

With this choice of $\boldsymbol{\Delta}^\ell(s)$, we have proved the first part of Theorem A.21. Now, we have to bound the probability that $\boldsymbol{\Delta}^{\ell+1}(s) \neq \boldsymbol{\Gamma}^{\ell+1}(s)$, assuming $\boldsymbol{\Delta}^\ell(s) = \boldsymbol{\Gamma}^\ell(s)$: the next lemma gives a bound which is stronger than the bound in Theorem A.21, and it concludes the proof.

Lemma A.30. *Let $\boldsymbol{\Theta}^\ell(s) = \bigsqcup_{i=0}^\ell \boldsymbol{\Delta}^i(s)$, where $\bigsqcup$ denotes the disjoint union, and let $\boldsymbol{\vartheta}^\ell(s) = |\boldsymbol{\Theta}^\ell(s)|$. Assuming $\boldsymbol{\Theta}^\ell(s) = \boldsymbol{N}^\ell(s)$, $\mathbb{P}\left(\boldsymbol{\Delta}^{\ell+1}(s) \neq \boldsymbol{\Gamma}^{\ell+1}(s)\right) \leq \frac{1}{n} 2\rho^2_{\boldsymbol{\Theta}^\ell(s)}$.*

Proof. The procedures `P1` and `P2` behave differently only if one of the following holds:

1. $\boldsymbol{Y}_{v,w} > 0$ for some $v \in \boldsymbol{\Delta}^\ell(s), w \in \boldsymbol{\Theta}^\ell(s)$;

2. $\boldsymbol{Y}_{v,w} \neq \boldsymbol{X}_{v,w}$ for some $v \in \boldsymbol{\Delta}^\ell(s), w \in V$

The probability that the first case occurs is $\sum_{v\in\boldsymbol{\Delta}^\ell(s)}\sum_{w\in\boldsymbol{\Theta}^\ell(s)}\frac{\rho_v\rho_w}{M} \le \frac{\rho^2_{\boldsymbol{\Theta}^\ell(s)}}{n}$. The probability that the second case occurs is

$$\sum_{v\in\boldsymbol{\Delta}^\ell(s)}\sum_{w\in V}\mathcal{O}\left(\frac{\rho_v^2\rho_w^2}{M^2}\right)=\mathcal{O}\left(\rho^2_{\boldsymbol{\Delta}^\ell(s)}\frac{nM_2(\lambda)}{n^2M_1(\lambda)^2}\right)=\mathcal{O}\left(\rho^2_{\boldsymbol{\Delta}^\ell(s)}\frac{M_2(\lambda)}{n}\right).$$

□

A.3.3 Bounds for Branching Processes

In order to analyze the neighborhood sizes, we need to better understand the behavior of branching processes. For this reason, we need the following lemmas.

Lemma A.31. *Let $\boldsymbol{Z}$ be a μ-distributed branching process, let ℓ, S be integers such that $S \le \log^2 \ell$. Then, for ℓ tending to infinity, $\mathbb{P}\left(0 < \boldsymbol{Z}^\ell < S\right) \le (\eta(1)+o(1))^\ell$.*

Proof. We divide the proof in two different cases: in the first case, we condition on the fact that $\boldsymbol{Z}^\ell$ eventually dies (for more background on branching processes conditioned on death/survival, we refer to [12]). Conditioned on death, the expected number of descendants after ℓ steps is $Z^0\eta(1)^\ell \le e^{-\ell(-\log\eta(1))}$: by Markov inequality, the probability that $\boldsymbol{Z}^\ell \ge 1$ is at most $\mathbb{E}[\boldsymbol{Z}^\ell] = e^{-\ell(-\log\eta(1))}$.

In the second case, let $\tilde{\boldsymbol{Z}}^\ell$ be the process $\boldsymbol{Z}^\ell$ conditioned on survival: since $\boldsymbol{Z}^\ell \ge \tilde{\boldsymbol{Z}}^\ell$, it is enough to prove the claim for $\tilde{\boldsymbol{Z}}$. We name "bad" a step of this process in which $\tilde{\boldsymbol{Z}}^{\ell+1} = \tilde{\boldsymbol{Z}}^\ell$ (note that $\tilde{\boldsymbol{Z}}^{\ell+1} \ge \tilde{\boldsymbol{Z}}^\ell$): let us perform $h = \ell - S$ steps, trying to find S good steps. A step is bad with probability $\eta(1)$, and the probability that at least $\ell - S$ steps are bad is

$$\sum_{i=\ell-S}^{\ell}\binom{\ell}{i}\eta(1)^i(1-\eta(1))^{\ell-i} \le S\ell^S\eta(1)^{\ell-S} = e^{-(1+o(1))\ell(-\log\eta(1))}.$$

If i is a good step, $\tilde{\boldsymbol{Z}}^i \ge \tilde{\boldsymbol{Z}}^{i-1}+1$, otherwise $\tilde{\boldsymbol{Z}}^i \ge \boldsymbol{Z}^{i-1}$: hence, if there are at least S good steps, $\tilde{\boldsymbol{Z}}^\ell \ge S$. □

Lemma A.32. *Let $\boldsymbol{Z}$ be a μ-distributed branching process with $Z^0 = \omega(1)$, and let $S > Z^0$. Then, for each $\ell > (1+\varepsilon)F\left(Z^0 \to S\right)$, $\mathbb{P}\left(\boldsymbol{Z}^\ell < S\right) \le e^{-\Omega(Z^0)} + o(1)^{\ell - F\left(Z^0\to S\right)}$. If $\mu(0) = 0$, $\mathbb{P}\left(\boldsymbol{Z}^\ell < S\right) \le o(1)^{\ell-F\left(Z^0\to S\right)}$.*

Proof. We can view the branching process as the sum of Z^0 different branching processes. A standard theorem in the theory of branching processes [12, 1.A.5, Theorem 1] says that the probability that one of this branching processes dies is z_μ, where z_μ is the only integer between 0 and 1 such that $z_\mu = \sum_{i\in\mathbb{N}}\mu(i)z_\mu^i$. Since the different processes are independent, by Chernoff bound, the probability that at least $\frac{Z^0\mu}{2}$ processes survive is at least $e^{-\frac{Z^0 z_\mu}{8}} = e^{-\Omega(Z^0)}$. Hence, if $\tilde{\boldsymbol{Z}}$ is the process $\boldsymbol{Z}$ conditioned on survival, $\tilde{\boldsymbol{Z}}^0 = \Omega(Z^0)$ with probability $e^{-\Omega(Z^0)}$. Furthermore, if $\mu(0) = 0$, $\tilde{\boldsymbol{Z}}^0 = Z^0$ by definition.

Then, let us perform ℓ steps, and let us estimate $\tilde{\boldsymbol{Z}}^\ell$: a step is "bad" if $\tilde{\boldsymbol{Z}}^{i+1} \le \tilde{\boldsymbol{Z}}^i M_1(\mu)^{1-\varepsilon}$, if $M_1(\mu)$ is finite, or $\tilde{\boldsymbol{Z}}^{i+1} \le \left(\tilde{\boldsymbol{Z}}^i\right)^{\frac{1-\varepsilon}{\beta-1}}$ if μ is a power law with exponent β: it is simple to prove that a step is bad with probability at most $o(1)$, if $\tilde{\boldsymbol{Z}}^\ell \ge \tilde{\boldsymbol{Z}}^0 = \Omega(Z^0)$ tends to infinity. If the number of good steps is at least $(1+3\varepsilon)F\left(Z^0 \to S\right)$:

- if $M_1(\mu)$ is finite, $\tilde{\boldsymbol{Z}}^\ell \ge \tilde{\boldsymbol{Z}}^0 M_1(\mu)^{(1-\varepsilon)(1+3\varepsilon)F\left(Z^0\to S\right)} \ge \tilde{\boldsymbol{Z}}^0\frac{S}{Z^0}\omega(1) \ge S$;

- if μ is power law with $1 < \beta < 2$,

$$\begin{aligned}
\tilde{\boldsymbol{Z}}^{\ell} &\geq \left(\tilde{\boldsymbol{Z}}^{0}\right)^{\left(\frac{1-\varepsilon}{\beta-1}\right)^{(1+3\varepsilon)F\left(Z^0 \to S\right)}} \\
&\geq \left(\tilde{\boldsymbol{Z}}^{0}\right)^{\left(\frac{1}{\beta-1}\right)^{(1+\varepsilon)F\left(Z^0 \to S\right)}} \\
&\geq e^{\log\left(\tilde{\boldsymbol{Z}}^{0}\right)\frac{\log(S)}{\log\left(Z^0\right)}\Omega(1)} \\
&\geq S.
\end{aligned}$$

Let $\ell' = \ell - (1+3\varepsilon)F\left(Z^0 \to S\right)$ be the maximum number of bad steps: by changing the value of ε in the statement, we can assume that $\ell(1-\varepsilon) \geq (1+3\varepsilon)F\left(Z^0 \to S\right)$, and hence $\ell' = \ell - (1+3\varepsilon)F\left(Z^0 \to S\right) \geq \varepsilon\ell$. We need to bind the probability that at least ℓ' steps are bad: this is equal to the probability that the sum of ℓ Bernoulli variables with success probability $o(1)$ is at least ℓ'. This probability is

$$\sum_{i=\ell'}^{\ell} \binom{\ell}{i} (o(1))^{i}(1-o(1))^{n-i} \leq \ell 2^{\ell}(o(1))^{\ell'} \leq 2^{\mathcal{O}(\ell')}(o(1))^{\ell'} = o(1)^{\ell'}.$$

□

Corollary A.33. *Let $\boldsymbol{Z}$ be a μ-distributed branching process, and let S be an integer. If $\ell = \omega(1)$, and $\ell > (1+\varepsilon)F\left(Z^0 \to S\right)$, then $\mathbb{P}\left(0 < \boldsymbol{Z}^{(1+\varepsilon)\ell} < S\right) \leq \eta(1)^{\ell - F\left(Z^0 \to S\right)}$.*

Proof. If the process dies, by Lemma A.31 it dies before performing ℓ steps with probability smaller than $\eta(1)^{\ell}$. Otherwise, by Lemma A.31, $\tilde{\boldsymbol{Z}}^{(1+\varepsilon)\ell - F\left(Z^0 \to S\right)} \geq \log \ell = \omega(1)$ with probability $1 - \frac{\eta(1)^{\ell - F\left(Z^0 \to S\right)}}{2}$, if ℓ is big enough. Conditioned on this event, by Lemma A.32, $\tilde{\boldsymbol{Z}}^{(1+\varepsilon)\ell - F\left(Z^0 \to S\right) + (1+2\varepsilon)F\left(Z^0 \to S\right) + \varepsilon\ell} \geq S$ with probability at least $1 - o(1)^{\varepsilon\ell + 2\varepsilon F(\log \ell \to S))} \geq 1 - o(1)^{\varepsilon\ell} \geq 1 - \frac{\eta(1)^{\ell}}{2}$. Summing the two probabilities, we obtain $\mathbb{P}\left(0 < \boldsymbol{Z}^{(1+2\varepsilon)\ell} < S\right) \leq \eta(1)^{\ell - F\left(Z^0 \to S\right)}$. □

Now, let us prove upper bounds on neighborhood sizes, that correspond to lower bounds on $\boldsymbol{\tau}_s\left(n^x\right)$.

Lemma A.34. *Let us fix $\varepsilon > 0$, and a μ-distributed branching process $\boldsymbol{Z}^{\ell}$. Given a value $S = \omega(1)$, $\mathbb{P}\left(\forall \ell < (1-\varepsilon)F\left(Z^0 \to S\right), \boldsymbol{Z}^{\ell} < S\right) \geq 1 - o(1)$.*

Proof. Assume that $M_1(\mu)$ is finite:

$$\begin{aligned}
E\left[\boldsymbol{Z}^{\ell}\right] &\leq \mathbb{E}\left[\boldsymbol{Z}^{(1-\varepsilon)F\left(Z^0 \to S\right)}\right] \\
&= Z^0 M_1(\mu)^{(1-\varepsilon)F\left(Z^0 \to S\right)} \\
&= Z^0 M_1(\mu)^{(1-\varepsilon)\log_{M_1(\mu)}\frac{S}{Z^0}} \\
&= Z^0 \left(\frac{S}{Z^0}\right)^{1-\varepsilon} \\
&= S^{1-\varepsilon} Z_0^{\varepsilon} \\
&= S\left(\frac{Z^0}{S}\right)^{\varepsilon}.
\end{aligned}$$

We conclude by Markov inequality.

Let us consider the case where μ is a power law distribution with $1 < \beta < 2$. By Lemma A.7 applied with $k = \boldsymbol{Z}^i$, with probability at least $\left(\frac{1}{\boldsymbol{Z}^i}\right)^{\varepsilon}$, $\boldsymbol{Z}^{i+1} < \left(\boldsymbol{Z}^i\right)^{\frac{1+\varepsilon}{\beta-1}}$. Let us assume $\boldsymbol{Z}^i > \log S$ for each i (increasing the number of elements, we can only increase the number of descendants). Consequently, the probability that $\boldsymbol{Z}^\ell$ is bigger than $\left(Z^0\right)^{\left(\frac{1+\varepsilon}{\beta-1}\right)^\ell}$ is at most $\sum_{i=1}^{\ell}\left(\frac{1}{\boldsymbol{Z}^i}\right)^{\varepsilon} \geq \ell\left(\frac{1}{\log S}\right)^{\varepsilon} = o(1)$ if $\ell = (1-\varepsilon)F\left(Z^0 \to S\right)$. With probability $1-o(1)$, $\boldsymbol{Z}^k < \left(Z^0\right)^{\left(\frac{1+\varepsilon}{\beta-1}\right)^\ell}$: since $\ell < (1-\varepsilon')F\left(Z^0 \to S\right)$, the claim follows. □

Now, we need to prove a corresponding bound for tail probabilities.

Lemma A.35. *Let us fix $\varepsilon > 0$, and a μ-distributed branching process $\boldsymbol{Z}^\ell$ such that $\boldsymbol{Z}^0 = 1$. Given integers $\ell = \omega(1)$, S such that $F\left(1 \to \ell^2\right) \leq \varepsilon F\left(\ell^2 \to S\right)$,*

$$\mathbb{P}\left(\forall i < (1-\varepsilon)(F\left(Z^0 \to S\right) + \ell), 0 < \boldsymbol{Z}^i < S\right) \geq \eta(1)^\ell.$$

Proof. First of all, let $\tilde{\boldsymbol{Z}}^\ell$ be the corresponding branching process conditioned on survival ($\tilde{\boldsymbol{Z}}^0 = 1$ with probability $\Omega(1)$). Assuming $\tilde{\boldsymbol{Z}}^0 = 1$, the probability that $\tilde{\boldsymbol{Z}}^\ell$ starts with a path of length ℓ is $\eta(1)^\ell$. Now, let us estimate $\mathbb{P}\left(\boldsymbol{Z}^\ell = k \wedge \tilde{\boldsymbol{Z}}^\ell = 1\right) \leq \mathbb{P}\left(\tilde{\boldsymbol{Z}}^\ell = 1 \middle| \boldsymbol{Z}^\ell = k\right) \leq k z_\mu^k$, where z_μ is the probability that a μ-distributed branching process has an infinite number of descendants. Hence, $\mathbb{P}(\tilde{\boldsymbol{Z}}^\ell = 1 \wedge \boldsymbol{Z}^\ell < k) \geq \eta(1)^\ell - \sum_{i=k}^{\infty} i z_\mu^{i-1} = \eta(1)^\ell - \frac{k z_\mu^{k-1}(1-z_\mu)+z_\mu^k}{(1-z_\mu)^2} = \eta(1)^\ell - \mathcal{O}\left(k z_\mu^{k-1}\right)$.

For $k = \ell^2$,

$$\begin{aligned}\mathbb{P}(\boldsymbol{Z}^\ell < \ell^2) &\geq \mathbb{P}\left(\boldsymbol{Z}^\ell < \ell^2 \wedge \tilde{\boldsymbol{Z}}^\ell = 1\right) \\ &= \eta(1)^\ell - \mathcal{O}\left(\ell^2 z_\mu^{\ell^2-1}\right) \\ &= \eta(1)^\ell(1-o(1))\end{aligned}$$

. Then, let us consider the process $\boldsymbol{Z}_1$ defined by $\boldsymbol{Z}_1^k = \boldsymbol{Z}^{\ell+k}$: since $\boldsymbol{Z}_1^0 < \ell^2$, we know by Lemma A.34 that $\mathbb{P}\left(\forall k < (1-\varepsilon)F\left(\ell^2 \to S\right) + \ell, \boldsymbol{Z}_1^k < S\right) \geq 1 - o(1)$. Since the behavior of $\boldsymbol{Z}_1$ is independent from the behavior of $\boldsymbol{Z}$, and since $F\left(\ell^2 \to S\right) = F\left(1 \to S\right) - F\left(1 \to \ell^2\right) \geq (1-\varepsilon)F\left(1 \to S\right)$,

$$\mathbb{P}\left(\forall k < (1-2\varepsilon)F\left(1 \to S\right) + \ell, \boldsymbol{Z}^k < S\right) = \Omega\left(\eta(1)^\ell(1-o(1))\right) = \Omega\left(\eta(1)^\ell\right).$$

We conclude by replacing ℓ with $(1-\varepsilon)\ell$. □

A.3.4 Bounds on Neighborhood Sizes

Now, we need to translate the results in the previous section from the realm of branching processes to the realm of random graphs, using Theorem A.21. First of all, the following corollary of Theorem A.21 gives us a simpler bound to decide when the branching process approximation works.

Corollary A.36. *Let G be a random graph, $s \in G$. There exists a constant c_λ only depending on λ such that, for each $\ell < n^{c_\lambda}$ $\mathbb{P}\left(\boldsymbol{\Theta}^\ell(v) \neq \boldsymbol{N}^\ell(v)\right) = \mathcal{O}\left(n^{-c_\lambda}\right)$, assuming $\rho_{\boldsymbol{\Delta}^i(s)} < n^{c_\lambda}$ for each $i < \ell$. The same is true if we condition on the size of $\boldsymbol{\Delta}^\ell(v)$.*

Proof. By Theorem A.21,

$$\begin{aligned}
\mathbb{P}\left(\boldsymbol{\Theta}^{\ell}(v) \neq \boldsymbol{N}^{\ell}(v)\right) &= \sum_{i=1}^{\ell} \mathbb{P}\left(\boldsymbol{\Theta}^{i}(v) \neq \boldsymbol{N}^{i}(v) \wedge \boldsymbol{\Theta}^{i-1}(v) = \boldsymbol{N}^{i-1}(v)\right) \\
&= \sum_{i=1}^{\ell} \mathbb{P}\left(\boldsymbol{\Delta}^{i}(v) \neq \boldsymbol{\Gamma}^{i}(v) \wedge \boldsymbol{\Theta}^{i-1}(v) = \boldsymbol{N}^{i-1}(v)\right) \\
&\leq \sum_{i=1}^{\ell} \mathbb{P}\left(\boldsymbol{\Delta}^{i}(v) \neq \boldsymbol{\Gamma}^{i}(v) \middle| \boldsymbol{\Theta}^{i-1}(v) = \boldsymbol{N}^{i-1}(v)\right) \\
&= \mathcal{O}\left(\sum_{i=1}^{\ell} \rho_{\boldsymbol{\Theta}^{i}(s)}^{2} \frac{M_2(\lambda)}{n}\right) \\
&= \mathcal{O}\left(\ell^3 n^{2c_\lambda} \frac{M_2(\lambda)}{n}\right) \\
&= \mathcal{O}\left(\frac{M_2(\lambda)}{n^{1-5c_\lambda}}\right).
\end{aligned}$$

We conclude because $M_2(\lambda) = \mathcal{O}(1)$ if λ has finite variance, $M_2(\lambda) = n^{3-\beta}$ if λ is power law with exponent $2 < \beta < 3$: this means that it is enough to choose c_λ such that $n^{\max(0,3-\beta)-1+5c_\lambda} < n^{-c_\lambda}$, that is, $c_\lambda < \frac{1-\max(0,3-\beta)}{6}$. □

From this corollary, it is easy to translate Lemmas A.34 and A.35 in terms of random graphs, at least for values of x smaller than c_λ.

Corollary A.37. *Let $G = (V, E)$ be a random graph, let $s \in V$ be in the giant component, and let $x < c_\lambda$ be a fixed, small enough constant. Then, $\mathbb{P}\left(\boldsymbol{\tau}_s\left(n^x\right) > (1-\varepsilon)F\left(\deg(s) \to n^x\right)\right) \geq 1 - o(1)$. Furthermore, if s has degree 1, $\mathbb{P}\left(\boldsymbol{\tau}_s\left(n^x\right) > (1-\varepsilon)(F\left(1 \to n^x\right) + \alpha)\right) \geq \eta(1)^{\alpha}$ for $\alpha = \omega(1)$.*

Proof. By Corollary A.36, assuming $x < c_\lambda$, $\boldsymbol{\Delta}^{\ell}(s) = \boldsymbol{\gamma}^{\ell}(s)$ with probability $1 - o(1)$; furthermore, by Lemma A.34, $\boldsymbol{\Delta}^{\ell}(s) < n^x$ for each $\ell < (1-\varepsilon)F\left(\deg(s) \to n^x\right)$ with probability $1 - o(1)$.

Similarly, by Lemma A.35, $\mathbb{P}(\forall i < (1-\varepsilon)(F\left(\deg(s) \to n^x\right) + \alpha), \boldsymbol{\delta}^{i}(s) < n^x) \geq \eta(1)^{\alpha}$, and since $\boldsymbol{\gamma}^{i}(s) = \boldsymbol{\delta}^{i}(s)$ with probability $1 - o(1)$ for each $i < \boldsymbol{\tau}_s\left(n^x\right)$, we conclude that $\mathbb{P}\left(\boldsymbol{\tau}_s\left(n^x\right) > (1-\varepsilon)(F\left(1 \to n^x\right) + \alpha)\right) \geq (1 - o(1))\eta(1)^{\alpha}$. □

The translation of the lower bounds is more complicated: the main problem is that, when $\boldsymbol{\Theta}^{\ell}(s) \neq \boldsymbol{N}^{\ell}(s)$, we know very little on the size of $\boldsymbol{\Theta}^{\ell}(s)$. In order to deal also with this case, as soon as $\boldsymbol{\Theta}^{\ell}(s) \neq \boldsymbol{N}^{\ell}(s)$, we remove the whole $\boldsymbol{N}^{\ell}(s)$ from the graph, and we consider the neighborhood growth of a new node s' which was in $\boldsymbol{\Gamma}^{\ell+1}(s)$ in the previous graph. We prove that the behavior of the neighbors of s' in the new graph is “very similar” to the behavior of the neighbors of s in the old graph: basically, the only difference is that we re-start from size 1 instead of $\boldsymbol{\delta}^{\ell}(s)$. However, this difference is compensated by the fact that the probability that $\boldsymbol{\Theta}^{\ell}(s) \neq \boldsymbol{N}^{\ell}(s)$ is small. In other words, it is more likely that $\boldsymbol{\delta}^{\ell}(s)$ remains 1 for ℓ steps, rather than that $\boldsymbol{\Theta}^{\ell}(s) \neq \boldsymbol{N}^{\ell}(s)$.

Let us formalize this intuitive proof. First of all, we need to understand what happens when we remove a neighbor from the graph.

Lemma A.38. *Let $G = (V, E)$ be a random graph, let $s \in V$, let $\ell \in \mathbb{N}$, and let us assume that $\rho_{\boldsymbol{N}^{\ell}(s)} < n^{1-\varepsilon}$. Then, conditioned on the structure of $\boldsymbol{N}^{\ell}(s)$, the subgraph induced by $V - \boldsymbol{N}^{\ell}(s)$ is again a random graph, and the values of $\eta(1)$, $M_1(\mu)$ change by $\mathcal{O}\left(\frac{1}{n^{\varepsilon}}\right)$.*

Proof for the CM. Let us consider the graph obtained from G by removing all the stubs in $\boldsymbol{N}^{\ell}(s)$, and all the stubs paired with stubs in $\boldsymbol{N}^{\ell}(s)$. The pairing on the remaining stubs is clearly a random pairing, and the number of stubs removed is at most $n^{1-\varepsilon}$, and if λ' is the degree distribution of $G-\boldsymbol{N}^{\ell}(s)$, $\sum_{i\in\mathbb{N}}|\lambda(i)-\lambda'(i)|<\frac{1}{n^{\varepsilon}}$. From this condition, it is easy to prove that $\eta(1)$ and $M_1(\mu)$ cannot change by more than $\mathcal{O}(n^{\varepsilon})$, if n is big enough. □

Proof for IRG. In this case, let us remove $\boldsymbol{N}^{\ell}(s)$, and let us consider the probability that two nodes outside $\boldsymbol{N}^{\ell}(s)$ are connected: $\mathbb{P}(E(v,w))=f\left(\frac{\rho_v\rho_w}{M}\right)=f\left(\frac{\rho_v\rho_w}{M-\rho_{\boldsymbol{N}^{\ell}(s)}}\frac{M-\rho_{\boldsymbol{N}^{\ell}(s)}}{M}\right)$. Let $\rho'_v=\rho_v\sqrt{\frac{M-\rho_{\boldsymbol{N}^{\ell}(s)}}{M}}$: clearly, $\rho'_v=\rho_v(1+o(1))$, and $G-\boldsymbol{N}^{\ell}(s)$ is a random graph with weights ρ'_v. Furthermore, if λ' is the degree distribution of $G-\boldsymbol{N}^{\ell}(s)$, it is clear that the required conditions are satisfied, because the dependency between λ, μ, and η is continuous. □

Using this lemma, we may translate Corollary A.33 to the context of random graphs.

Lemma A.39. *Let G be a graph with a power law degree distribution λ with exponent β, let μ, η be as before. There exists a positive constant c_λ only depending on λ such that, for each ℓ, S such that $\ell=\mathcal{O}(\log n)$, $n^{\varepsilon}<S<n^{c_\lambda}$, $\mathbb{P}\left(\forall\ell'<\ell(1+\varepsilon), 0<\boldsymbol{\gamma}^{\ell'}(s)<S\right)=\mathcal{O}\left(\eta(1)^{\ell-F\left(Z^0\to S\right)}\right)$.*

Proof. First of all, we may assume that $\ell-F\left(Z^0\to S\right)=\omega(1)$, otherwise the probabilistic bound is trivial. By Corollary A.36, the three following cases are possible:

- $\boldsymbol{N}^{\ell}(s)=\boldsymbol{\Theta}^{\ell}(s)$;
- $\rho_{\boldsymbol{\Gamma}^i(s)}\geq 4Sn^{\varepsilon}$ for some $i<\ell$;
- none of the two cases above applies.

In the first case, the result follows directly by Corollary A.33. In the second case, let i be the smallest integer such that $\rho_{\boldsymbol{\Gamma}^i(s)}\geq 4Sn^{\varepsilon}$: in IRG, by Lemma A.16, $\boldsymbol{\Gamma}^{i+1}(s)\geq(1-\varepsilon)\rho_{\boldsymbol{\Gamma}^i(s)}$ w.h.p., and $\boldsymbol{\tau}_s(n^x)<i+1<\ell$. In the CM, $\boldsymbol{\delta}^{i-1}(s)+\boldsymbol{\delta}^{i}(s)\geq 4Sn^{\varepsilon}$, and as a consequence either $\boldsymbol{\delta}^{i-1}(s)\geq 2Sn^{\varepsilon}$ or $\boldsymbol{\delta}^{i}(s)\geq 2Sn^{\varepsilon}$: by Lemma A.11, $\boldsymbol{\tau}_s(n^x)<i+1<\ell$.

It only remains to solve the third case. The probability that this case occurs is $\mathcal{O}\left(n^{-c_\lambda}\right)$ by Corollary A.36. However, n^{-c_λ} is not sufficient for our purposes, because $\eta(1)^{\ell-F\left(Z^0\to S\right)}$ can be much smaller. Let us consider the following process: we explore neighbors of v of increasing size, until we hit a neighbor i satisfying $\boldsymbol{\Delta}^i(s)\neq\boldsymbol{\Gamma}^i(s)$. If $\boldsymbol{\Delta}^i(s)\neq\boldsymbol{\Gamma}^i(s)$, either all nodes in $\boldsymbol{\Gamma}^i(s)$ have all edges directed inside $\boldsymbol{\Gamma}^i(s)$, and $\boldsymbol{\Gamma}^{i+1}(s)$ is empty, or there is at least a node v with an edge directed outside $\boldsymbol{\Gamma}^i(s)$. In the former case, we know that $\boldsymbol{\gamma}^{i+1}(s)=0$, and the conclusion follows. In the latter case, we remove $\boldsymbol{\Gamma}^i(s)$ from the graph: the size of $\boldsymbol{\Gamma}^{i+j}(s)$ is at least the size of $\boldsymbol{\Gamma}^j(v')$, where v' is the neighbor of v outside $\boldsymbol{\Gamma}^i(s)$. Furthermore, by Lemma A.38, $G-\boldsymbol{\Gamma}^i(s)$ is a random graph, with degree distribution very similar to the degree distribution of G: indeed, $i<\ell=\mathcal{O}(\log n)$, and the volume of nodes removed is at most $S\log n\leq n^{1-\varepsilon}$. Moreover, the size of the neighbors of v' is independent from all previous events, because all we knew about v' has been removed from the graph. Then, we can restart the exploration from v', in the new graph: if $\boldsymbol{\Delta}^j(v')\neq\boldsymbol{\Gamma}^j(v')$, we proceed again as before.

More formally, let us fix ℓ, and let $P(\ell,h)$ be the probability that $\boldsymbol{\Gamma}^{\ell}(s)<S$, and that $\boldsymbol{\Delta}^j(s)\neq\boldsymbol{\Gamma}^j(s)$ happened h times in the aforementioned process. We prove by induction on h that $P(\ell,h)\leq e^{-(1+\varepsilon)\left(\ell-F\left(Z^0\to S\right)\right)(-\log\eta(1))}$. The base case follows by our initial argument. For inductive step, let $\boldsymbol{\ell}'$ be the smallest integer such that $\boldsymbol{\Gamma}^{\boldsymbol{\ell}'}(s)\neq\boldsymbol{\Delta}^{\boldsymbol{\ell}'}(s)$: note that $\mathbb{P}\left(\boldsymbol{\ell}'=i\right)\leq\mathbb{P}\left(\boldsymbol{\ell}'<\ell\right)\leq\frac{n^{-k}}{\ell}\leq n^{-k+\varepsilon}$, and that, by inductive hypothesis, $P(\ell-\boldsymbol{\ell}',h)\leq e^{-(-\log\eta(1)+\varepsilon)(\ell-\boldsymbol{\ell}')}$ if $\ell-\boldsymbol{\ell}'\geq\log S$, and consequently $P(\ell-\boldsymbol{\ell}',h)\leq e^{-(-\log\eta(1)+\varepsilon)(\ell-\boldsymbol{\ell}'-\log S)}$.

$$\mathbb{P}(\ell,h+1)\leq\sum_{i=0}^{\ell}\mathbb{P}(\boldsymbol{\ell}'=i)P(i,0)P(\ell-i,S,h)$$

$$\begin{aligned}
&\leq \sum_{i=0}^{\ell} n^{-k+\varepsilon} e^{-(-\log \eta(1)+\varepsilon)(i-F\left(Z^0 \to S\right))} e^{-(-\log \eta(1)+\varepsilon)\left(\ell-i-\log S-F\left(Z^0 \to S\right)\right)} \\
&\leq n^{-k+\varepsilon} e^{-(-\log \eta(1)+\varepsilon)(\ell-2F\left(Z^0 \to S\right)-\log S)} \\
&\leq e^{-(-\log \eta(1)+\varepsilon)(\ell-F\left(Z^0 \to S\right))} e^{-(k-\varepsilon)\log n+F\left(Z^0 \to S\right)+\log S}.
\end{aligned}$$

The inductive step is proved, if $e^{-(k-\varepsilon)\log n+F\left(Z^0\to S\right)+\log S} < 1$, that is, $(\min(1,\beta-2)-2\log_n S-2\varepsilon)\log n > F\left(Z^0 \to S\right)+\log S$, which is implied by $(\min(1,\beta-2)-2\varepsilon)\log n > \log_{M_1(\mu)} S + 3\log S$, that is, $S\left(3+\frac{1}{\log M_1(\mu)}\right) < n^{\min(1,\beta-2)-2\varepsilon}$. The lemma follows by choosing the right value of c_λ. □

By combining this lemma with Theorem A.9, we have proved the following theorem.

Theorem A.40. *Let $G=(V,E)$ be a random graph, let λ be the degree distribution of G, let μ, η be as before, and let $0<x<1$. Then, if $s \in V$, $\deg(v)=d$, the following hold:*

- $\boldsymbol{\tau}_s\left(n^x\right) \geq (1-\varepsilon)F\left(d \to n^x\right)$ *a.a.s.;*
- $\mathbb{P}\left(\boldsymbol{\tau}_s\left(n^x\right) \geq (1+\varepsilon)\left(\alpha+F\left(d\to n^x\right)\right)\right) = \mathcal{O}\left(\eta(1)^\alpha\right)$*;*
- $\mathbb{P}\left(\boldsymbol{\tau}_s\left(n^x\right) \geq (1-\varepsilon)\left(\alpha+F\left(d\to n^x\right)\right)\right) = \Omega\left(\eta(1)^\alpha\right)$.

A.4 The Case $1<\beta<2$

In the case $1<\beta<2$, the branching process approximation does not work: indeed, the distribution μ is not even defined, because $M_1(\lambda)$ is infinite. For this reason, we need a different analysis, which looks similar to the "big neighbors" analysis in the case $\beta>2$. We prove that the graph which is generated from this distribution has a very dense core, which is made by all nodes whose degree is big enough: almost all the other nodes are either connected to the core, or isolated, so that the average distance between two nodes is 2 or 3. There are also some paths of length $\mathcal{O}(1)$ leaving the core, whose length depends on the value β of the distribution.

In our analysis, in order to avoid pathological cases, we have to assume that $\rho_v < (1-\varepsilon)M$ for each v in the Chung-Lu model (otherwise, all nodes with weight at least $1+\varepsilon$ would be connected to the maximum degree node). Note that this event holds with probability $\mathcal{O}(1)$.

Before entering the details of our analysis, we need some probabilistic lemmas that describe the relationship between the weight and the degree of a node.

Lemma A.41 ([158], Equation A.1.7)**.** *For each $\varepsilon>0$, there exists N_ε and C_ε not depending on n such that the following hold a.a.s.:*

- $M=(1+\varepsilon)\sum_{i=1}^{N_\varepsilon}\rho_i$*;*
- $\rho_1 \leq C_\varepsilon \rho_{N_\varepsilon}$.

Corollary A.42. *The node with maximum weight has weight $\Theta\left(n^{\frac{1}{\beta-1}}\right)$ a.a.s., and $M=\Theta\left(n^{\frac{1}{\beta-1}}\right)$.*

In this regime, we still need the definitions of $\boldsymbol{\Delta}^\ell(s)$ as in the case $\beta>2$, but in this case we will not prove that $\boldsymbol{\gamma}^{\ell+1}(s)$ is close to $\boldsymbol{\delta}^\ell(s)$ w.h.p.: for example, let s be a node with weight $M=\Theta\left(n^{\frac{1}{\beta-1}}\right)$: clearly, $\boldsymbol{\gamma}^{\ell+1}(s)$ cannot be M, which is bigger than n. Indeed, we prove that $\boldsymbol{\gamma}^{\ell+1}(s)$ is close to $\boldsymbol{\Delta}^\ell(s)^{\beta-1}$: this way, the number of neighbors of a node s with weight $\Theta(M)$ is close to n, which makes sense. In order to prove this result, we need a technical lemma on the volume of some nodes.

Lemma A.43. *Given a random graph with a degree distribution λ which is power law with exponent $1 < \beta < 2$, $\sum_{\rho_w \leq d} \rho_w = \mathcal{O}\left(nd^{2-\beta}\right)$.*

Proof. This result is a simple application of Abel's trick to estimate a sum: $\sum_{\rho_w \leq d} \rho_w = \sum_{i=1}^{d} i(|\{w : \rho_w \geq i\}| - |\{w : \rho_w \geq i+1\}|) = \sum_{i=1}^{d} i|\{w : \rho_w \geq i\}| - \sum_{i=2}^{d+1} |(i-1)\{w : \rho_w \geq i\}|) \leq \sum_{i=1}^{d} |\{w : \rho_w \geq i\}| = \sum_{i=1}^{d} \mathcal{O}\left(\frac{n}{i^{\beta-1}}\right) = \mathcal{O}\left(n \int_{\frac{1}{2}}^{d} x^{1-\beta} dx\right) = \mathcal{O}\left(nd^{2-\beta}\right)$. □

Using this lemma, we can formally prove the relation between $\boldsymbol{\delta}^{\ell}(s)$ and $\boldsymbol{\gamma}^{\ell+1}(s)$.

Lemma A.44. *For each $\varepsilon > 0$, and for each ℓ such that $\boldsymbol{n}^{\ell}(s) < \boldsymbol{\delta}^{\ell}(s)^{\beta-1} n^{-\varepsilon}$, and $\boldsymbol{\delta}^{\ell}(s) > n^{\varepsilon}$, $\boldsymbol{\gamma}^{\ell+1}(s) = \Theta\left(\boldsymbol{\delta}^{\ell}(s)^{\beta-1}\right)$.*

Proof. Let us fix $\varepsilon > 0$, and let us prove that $\boldsymbol{\delta}^{\ell+1}(v) \geq \boldsymbol{\gamma}^{\ell}(v)^{\beta-1}$. Let us consider the set W made by all nodes with weight at least $\frac{M}{\boldsymbol{\delta}^{\ell}(s)}$, not in $\boldsymbol{N}^{\ell}(s)$: there are $\Theta\left(n\left(\frac{\boldsymbol{\delta}^{\ell}(s)}{M}\right)^{\beta-1}\right) = \Theta\left(\boldsymbol{\delta}^{\ell}(s)^{\beta-1}\right)$ such nodes, because $\boldsymbol{n}^{\ell}(s) < \boldsymbol{\delta}^{\ell}(s)^{\beta-1}$. We want to apply concentration inequalities to prove that there are $\mathcal{O}(|W|)$ nodes in W that are in $\boldsymbol{\Delta}^{\ell+1}(s)$. First of all, let us assume without loss of generality that $\boldsymbol{\delta}^{\ell}(s) > n^{\varepsilon}$, otherwise this inequality is empty. In the Configuration Model, let us sort the nodes in W, obtaining $w_1, \dots, w_k$, and let us consider a procedure where we pair stubs of w_i until we find a connection to $\boldsymbol{\Delta}^{\ell}(s)$. Since, at each step, the number of stubs in $\boldsymbol{\Delta}^{\ell}(s)$ that are not paired with a node in W is $\mathcal{O}(\boldsymbol{\Delta}^{\ell}(s))$, and w_i has $\frac{Mn^{\varepsilon}}{\boldsymbol{\Delta}^{\ell}(s)}$ stubs, at each step there is probability $\mathcal{O}(1)$ that w_i is connected to a node in $\boldsymbol{\Gamma}^{\ell}(s)$. A simple application of Azuma's inequality lets us conclude. In IRG, the probability that a node $w \in W$ is linked to a node in $\boldsymbol{\Gamma}^{\ell+1}(s)$ is at least $\sum_{v \in \boldsymbol{\Gamma}^{\ell}(s)} f\left(\frac{\rho_v}{\boldsymbol{\delta}^{\ell}(s)}\right) = \mathcal{O}(1)$: a simple application of the multiplicative form of Chernoff bound (Lemma A.1) lets us conclude.

For an upper bound, we can divide the nodes in $\boldsymbol{\Gamma}^{\ell+1}(s)$ in two sets W, W', where W is the set of nodes with weight at most $\frac{M}{\rho_v}$, $W' = W^C$. For W', the number of nodes with weight at least $\frac{M}{\rho_v}$ is $\mathcal{O}\left(n\left(\frac{\rho_v}{M}\right)^{\beta-1}\right) = \mathcal{O}\left(\rho_v^{\beta-1}\right)$, and hence the number of neighbors of v in W' is at most $\rho_v^{\beta-1}$. For the set W, we have to consider separately IRG and the CM. In the first case, let $\boldsymbol{X}_w = 1$ if $w \in \boldsymbol{\Gamma}^{\ell+1}(s)$, 0 otherwise: we want to estimate $\sum_{w \in W} \boldsymbol{X}_w$. Through the previous lemma, the expected value of this sum is:

$$
\begin{aligned}
\mathbb{E}\left[\sum_{w \in W} \boldsymbol{X}_w\right] &= \sum_{w \in W} \sum_{v \in \boldsymbol{\Gamma}^{\ell}(s)} f\left(\frac{\rho_v \rho_w}{M}\right) \\
&= \sum_{w \in W} \sum_{v \in \boldsymbol{\Gamma}^{\ell}(s)} (1+o(1)) \frac{\rho_v \rho_w}{M} \\
&= (1+o(1)) \frac{\boldsymbol{\delta}^{\ell}(s)}{M} \sum_{w \in W} \rho_w \\
&= \mathcal{O}\left(\frac{\boldsymbol{\delta}^{\ell}(s)}{M} n \left(\frac{M}{\boldsymbol{\delta}^{\ell}(s)}\right)^{2-\beta}\right) \\
&\leq \boldsymbol{\delta}^{\ell}(s)^{\beta-1}.
\end{aligned}
$$

Since these random variables are independent, we can apply Chernoff bound to prove that $\sum_{w \in W} \boldsymbol{X}_w \leq \mathbb{E}\left[\sum_{w \in W} \boldsymbol{X}_w\right]$.

In the CM, let $a_1, \dots, a_{\boldsymbol{\delta}^{\ell}(s)}$ be the stubs in $\boldsymbol{\Delta}^{\ell}(s)$. By the previous lemma, the number of stubs in W is $\sum_{w \in W} \rho_w = \mathcal{O}\left(\frac{nM^{2-\beta}}{\boldsymbol{\delta}^{\ell}(s)^{2-\beta}}\right) = \mathcal{O}\left(\frac{M}{\boldsymbol{\delta}^{\ell}(s)^{2-\beta}}\right)$. The number of nodes in $W \cap \boldsymbol{\gamma}^{\ell+1}(s)$ is at most the number of stubs in $\boldsymbol{\delta}^{\ell}(s)$ which are paired with stubs in W: let us pair stubs in $\boldsymbol{\delta}^{\ell}(s)$ in order: at each step, the probability that we hit a stub in W is $\mathcal{O}\left(\frac{1}{M}\frac{M}{\boldsymbol{\delta}^{\ell}(s)^{2-\beta}}\right)$, because there are still $\mathcal{O}(M)$ stubs outside W. A simple application of Azuma's inequality proves that $\boldsymbol{\gamma}^{\ell+1}(s) \leq \mathcal{O}\left(\frac{\boldsymbol{\delta}^{\ell}(s)}{\boldsymbol{\delta}^{\ell}(s)^{2-\beta}}\right) = \mathcal{O}\left(\boldsymbol{\delta}^{\ell}(s)^{\beta-1}\right)$. □

Corollary A.45. *For each node s with degree at least n^{ε}, the number of neighbors of s is $\Theta\left(\rho_s^{\beta-1}\right)$.*

Proof. Apply the previous lemma with $\ell = 0$. □

Using the last two results, we can transform statements dealing with the number of nodes to statements dealing with weights. For this reason, we can analyze the weights, which are much simpler.

Lemma A.46. *The probability that a node v with weight ρ_v is connected to a node with weight at most ρ is $\mathcal{O}\left(\frac{n\rho_v\rho^{2-\beta}}{M}\right)$.*

Proof. First, we can assume that $\rho_v \ll \frac{M}{n\rho^{2-\beta}}$, otherwise the thesis of the lemma is trivially true.

In the CM, let us pair all the stubs of v in order. At each step, the probability that we hit a stub whose node has weight at most ρ is $\mathcal{O}\left(\frac{1}{M}\sum_{w:\rho_w<\rho}\rho_w\right)$, because we have paired at most $\rho_v \ll M$ nodes. Summing over all stubs of v, we obtain $\frac{\rho_v}{M}\sum_{w:\rho_w<\rho}\rho_w = \mathcal{O}\left(\frac{n\rho_v\rho^{2-\beta}}{M}\right)$ by Lemma A.43.

In IRG, this probability is $\sum_{w:\rho_w<\rho} f\left(\frac{\rho_v\rho_w}{M}\right) = (1+o(1))\frac{\rho_v}{M}\sum_{w:\rho_w<\rho}\rho_w = \mathcal{O}\left(\frac{n\rho_v\rho^{2-\beta}}{M}\right)$ by Lemma A.43. □

Lemma A.47. *A node v with degree at least $\log^2 n$ is w.h.p. connected to all nodes with weight at least εM.*

Proof. In IRG, this lemma follows from our assumptions on f. In the CM, let v be a node with degree at least $\log^2 n$, let w be a node with degree at least εM, and let $a_1, \dots, a_k$ be the stubs of v. Let us pair the stubs a_i in order: at each step, the probability that a_i is connected to a stub in w is at least ε. Hence, by Azuma's inequality (Lemma A.5), at least one of the stubs a_i is connected to a stub in W. □

Lemma A.48. *For each node s with degree at most $n^{1-\varepsilon}$, $\mathbb{P}\left(\boldsymbol{\tau}_s\left(n^{1-\varepsilon}\right)=2\right) = 1-\frac{1}{n^{\mathcal{O}(\varepsilon)}}$.*

Proof. Since $\deg(s) < n^{1-\varepsilon}$, $\boldsymbol{\tau}_s\left(n^{1-\varepsilon}\right) \geq 2$. For the lower bound, if s is connected to a node with weight $M^{1-\frac{\varepsilon}{2}}$, then $\boldsymbol{\tau}_s\left(n^{1-\varepsilon}\right) \leq 2$ by Corollary A.45. By Lemma A.47, this happens w.h.p. if $\deg(v) > \log^2 n$: for this reason, the only remaining case is when $\deg(v) < \log^2 n$, and v is not connected to any node with weight $n^{1-\frac{\varepsilon}{2}}$. In this case, we prove that v is likely to be isolated: indeed, let us bind the probability that v is connected to a node w with degree at most $n^{1-\frac{\varepsilon}{2}}$ (hence, with weight at most $M^{1-\varepsilon'}$). By Lemma A.46, this probability is $\mathcal{O}\left(\frac{n\rho_v M^{(2-\beta)\left(1-\varepsilon'\right)}}{M}\right) = \mathcal{O}\left(n\log n M^{1-\beta}M^{-\varepsilon'(2-\beta)}\right) = \mathcal{O}\left(n^{-\varepsilon''}\right)$. This means that, by Markov inequality, the number of nodes that are not isolated and not connected to a node with degree $n^{1-\varepsilon}$ is at most $n^{1-\varepsilon''}$, a.a.s.. We conclude that $T\left(d \to n^{1-\varepsilon}\right) \geq 2+\mathcal{O}\left(n^{-\varepsilon''}\right)$. □

Let us now estimate the deviations from this probability.

Lemma A.49. *For each node s, $\mathbb{P}\left(\boldsymbol{\tau}_s\left(n^{x}\right)=\ell\right) \leq n^{1-\frac{2-\beta}{\beta-1}(\ell-2-x)+o(1)}$.*

Proof. If $\deg(s) > \log^2 n$, $\boldsymbol{\tau}_s\left(n^x\right) \leq 2$ w.h.p.. Otherwise, since all nodes with degree at least $\log^2 n$ are connected to the node with maximum degree, $\boldsymbol{\tau}_s\left(n^x\right) = \ell$ implies that all nodes at distance at most $\ell - 3$ from s have degree at most $\log^2 n$. Hence, $\boldsymbol{\gamma}^i(v) \leq \log^{2\ell} n = n^{o(1)}$ for each $i \leq \ell - 3$. This means that, for each $i \leq \ell - 4$, there is a node in $\boldsymbol{\gamma}^i(v)$ with weight $n^{o(1)}$ connected to another node with weight $n^{o(1)}$. The probability that this happens is at most $n^{-\frac{2-\beta}{\beta-1}+o(1)}$, because there are $n^{o(1)}$ such nodes, and we may apply Lemma A.46 to each of them.

Since these events are independent, if we multiply the probabilities for each i between 0 and $\ell - 4$, the probability becomes $n^{-(\ell-3)\frac{2-\beta}{\beta-1}+o(1)}$. Finally, all nodes in $\boldsymbol{\gamma}^{\ell-3}(s)$ should be connected to nodes with degree at most n^x, and hence to nodes with weight at most $\mathcal{O}\left(n^{\frac{x}{\beta-1}}\right)$. Again by Lemma A.46, the probability that this event happens is $n^{o(1)}\frac{nn^{o(1)}n^{\frac{x(2-\beta)}{\beta-1}}}{M} = n^{1-\frac{1}{\beta-1}+\frac{x(2-\beta)}{\beta-1}+o(1)} = n^{-\frac{2-\beta}{\beta-1}(1-x)+o(1)}$. Overall, the probability that $\boldsymbol{\tau}_s\left(n^x\right) = \ell$ is at most $n^{-\frac{2-\beta}{\beta-1}(\ell-2-x)+o(1)}$.

□

Lemma A.50. *For each node s with degree 1,*

$$\mathbb{P}\left(\boldsymbol{\tau}_s\left(n^x\right) = \ell\right) \geq n^{1-\frac{2-\beta}{\beta-1}(\ell-2-x)+o(1)}.$$

Proof. Let s be a node of weight 1: we want to estimate the probability that s is connected to a node of weight 2 in the CM, with weight 1 in IRG. This probability is $\frac{2n\lambda(2)}{M} = n^{-\frac{2-\beta}{\beta-1}+o(1)}$ in the CM, $1 - \left(1 - f\left(\frac{1}{M}\right)\right)^n = 1 - \left(1 - \frac{1+o(1)}{M}\right)^n = 1 - e^{\frac{-n(1+o(1))}{M}} = \frac{n(1+o(1))}{M} = n^{-\frac{2-\beta}{\beta-1}+o(1)}$. Assuming this event holds, the probability that s is not connected to any other node is 1 in the CM, and it is $\mathcal{O}(1)$ in IRG, assuming the maximum weight is smaller than $(1-\varepsilon)M$. This means that, with probability $\mathcal{O}(1)n^{-\frac{2-\beta}{\beta-1}+o(1)} = n^{-\frac{2-\beta}{\beta-1}+o(1)}$, s is connected to a single node s_1 with weight 1 in IRG, 2 in the CM. We may re-iterate the process with s_1, finding a new node s_2, and so on, for $\ell - 3$ steps. The probability that we find a path of length $\ell - 3$ is $n^{-(\ell-3)\frac{2-\beta}{\beta-1}+o(1)}$. Then, let us estimate the probability that $s_{\ell-3}$ is connected only to a node with degree at most n^x. In the CM, the number of stubs of nodes with degree at most n^x is $\sum_{\rho_w < n^{\frac{x}{\beta-1}}} \rho_w = \mathcal{O}\left(n^{1+x\frac{2-\beta}{\beta-1}}\right)$, and hence the probability that we hit a stub of a node with degree at most n^x is $\mathcal{O}\left(\frac{n^{1+x\frac{2-\beta}{\beta-1}}}{M}\right) = n^{-(1-x)\frac{2-\beta}{\beta-1}+o(1)}$. In IRG, the probability is $1 - \prod_{\rho_w < n^{\frac{x+o(1)}{\beta-1}}} \left(1 - f\left(\frac{\rho_w}{M}\right)\right) = 1 - \prod_{\rho_w < n^{\frac{x+o(1)}{\beta-1}}} \left(1 - \frac{\rho_w(1+o(1))}{M}\right) = 1 - \prod_{\rho_w < n^{\frac{x+o(1)}{\beta-1}}} e^{-\frac{\rho_w(1+o(1))}{M}} = 1 - e^{-\frac{n^{1+x\frac{2-\beta}{\beta-1}+o(1)}}{M}} = 1 - e^{-n^{-(1-x)\frac{2-\beta}{\beta-1}+o(1)}} = n^{-(1-x)\frac{2-\beta}{\beta-1}+o(1)}$.

In both cases, we proved that the probability of having a path of length $\ell - 2$ followed by a node with degree at most n^x is at most $n^{-(\ell-2-x)\frac{2-\beta}{\beta-1}+o(1)}$. It is clear that in this case $\boldsymbol{\tau}_s\left(n^x\right) \geq \ell$.

□

Summarizing the results obtained in this section, we have proved the following theorem.

Theorem A.51. *Let $G = (V, E)$ be a random graph with degree distribution λ, which is power law with $1 < \beta < 2$. Then, if $s \in V$, $\deg(v) = d$, for each x between 0 and 1, the following hold:*

- $\boldsymbol{\tau}_s\left(n^x\right) \leq 2$ *a.a.s.;*
- $\mathbb{P}\left(\boldsymbol{\tau}_s\left(n^x\right) \geq \alpha + 2\right) \leq nc^{\alpha-x+o(1)}$*;*
- $\mathbb{P}\left(\boldsymbol{\tau}_s\left(n^x\right) \geq \alpha + 2\right) \geq nc^{\alpha+1-x+o(1)}$.

A.5 Applying the Probabilistic Bounds

Until now, we have proved bounds on the probability that $\boldsymbol{\tau}_s\left(n^x\right)$ has certain values. In this section, we turn these probabilistic bounds into bounds on the number of nodes that satisfy a given constraint, concluding the proof of the main theorems, and of the values in Table 8.2. The main tool used in the following lemma.

Lemma A.52. *For each node t, let $\boldsymbol{E}_\ell(t)$ be an event that only depends on the structure of $\boldsymbol{N}^\ell(t)$. Then, for each set $\boldsymbol{T} \subseteq V$, $0 < x < 1$, if $\boldsymbol{E}(t)$ is the event $\forall \ell < \boldsymbol{\tau}_t\left(n^x\right) - 1, \boldsymbol{E}_\ell(t)$*

$$|\{t \in \boldsymbol{T} : \boldsymbol{E}(t)\}| = \big(1 \pm o(1)\big) \sum_{t \in \boldsymbol{T}} \mathbb{P}\left(\boldsymbol{E}(t) \middle| t \in \boldsymbol{T}\right) \pm |\boldsymbol{T}| \frac{M^{2x}}{M^{1-o(1)}}.$$

If we condition on the structure of a neighbor with volume at most n^y, a very similar result holds:

$$|\{t \in T : \boldsymbol{E}(t)\}| = \big(1 \pm o(1)\big) \sum_{t \in T} \mathbb{P}\left(\boldsymbol{E}(t) \middle| t \in T\right) \pm |T| \left(\frac{M^{x+y} + M^{2x}}{M^{1-o(1)}}\right).$$

Proof. First of all, we assume without loss of generality that $|T| < n^{2\varepsilon}$, by dividing T in several sets if this is not the case. Let us sort the nodes in T, obtaining $t_1, \dots, t_k$, let $\boldsymbol{X}_i$ be 1 if $\boldsymbol{E}(t_i)$ holds, 0 otherwise, and let us assume that we know the structure of $\boldsymbol{N}^{\boldsymbol{\tau}_{t_j}\left(n^x\right)-2}(t_j)$ for each $j < i$ (in other words, let $\mathfrak{A}_i$ be the σ-field generated by all possible structures of $\boldsymbol{N}^{\boldsymbol{\tau}_{t_j}\left(n^x\right)-2}(t_j)$ for each $j < i$, and of the neighbor with volume at most n^y). Then, the probability that $\boldsymbol{N}^{\boldsymbol{\tau}_{t_i}\left(n^x\right)-2}(t_i)$ touches $\boldsymbol{N}^{\boldsymbol{\tau}_{t_j}\left(n^x\right)-2}(t_j)$ is at most

$$\begin{aligned}
&\sum_{\ell, \ell' < \mathcal{O}(\log n)} \mathbb{P}\left(\ell \leq \boldsymbol{\tau}_{t_j}\left(n^x\right) - 2 \wedge \ell' \leq \boldsymbol{\tau}_{t_i}\left(n^x\right) - 2 \wedge \boldsymbol{\Gamma}^\ell(t_i) \cap \boldsymbol{\Gamma}^{\ell'}(t_j) \neq \emptyset\right) \\
&\leq \sum_{\ell, \ell' < \mathcal{O}(\log n)} \mathbb{P}\left(\boldsymbol{\Gamma}^\ell(t_i) \cap \boldsymbol{\Gamma}^{\ell'}(t_j) \neq \emptyset \middle| \ell \leq \boldsymbol{\tau}_{t_j}\left(n^x\right) - 2 \wedge \ell' \leq \boldsymbol{\tau}_{t_i}\left(n^x\right) - 2\right) \\
&\leq \mathcal{O}(\log^2 n) \frac{M^{2x} + M^{x+y}}{M^{1-\varepsilon}},
\end{aligned}$$

because $\boldsymbol{\gamma}^{\boldsymbol{\tau}_{t_i}\left(n^x\right)-1}(t_i) < n^x$ for each i, and consequently $\boldsymbol{\delta}^{\boldsymbol{\tau}_{t_i}\left(n^x\right)-2}(t_i) < M^{x+\varepsilon}$ w.h.p., by Lemmas A.11, A.16 and A.44. As a consequence $p_i = \mathbb{P}\left(\boldsymbol{E}(t_i)\right) - \frac{M^{2x+2\varepsilon}}{M} \leq \mathbb{P}(\boldsymbol{E}(t_i) | \mathfrak{A}_i) \leq \mathbb{P}\left(\boldsymbol{E}(t_i)\right) + \frac{M^{2x+2\varepsilon}}{M} = q_i$.

We have proved that $\boldsymbol{S}_k = \sum_{i=1}^k \boldsymbol{X}_i - p_i, \boldsymbol{S'}_k = \sum_{i=1}^k q_i - \boldsymbol{X}_i$ are submartingales. If $p = \sum_{i=1}^k p_i$, by the strengthened version of Azuma's inequality (Lemma A.6), $\mathbb{P}\left(\boldsymbol{S}_k > \varepsilon k p\right) \leq e^{-\mathcal{O}\left(\frac{\varepsilon^2 k^2 p^2}{kp + \varepsilon kp}\right)} \leq e^{-\varepsilon^3 kp} \leq e^{-\varepsilon^3 n^\varepsilon}$. This proves that $|\{t \in T : \boldsymbol{E}(t)\}| \geq (1-\varepsilon) \sum_{t \in V} \mathbb{P}\left(\boldsymbol{E}_\ell(t) \middle| \ell < \boldsymbol{\tau}_t\left(n^x\right) - 1, t \in T\right) + |T| \frac{M^{2x} + M^{x+y}}{M^{1-\varepsilon}}$, w.h.p.. The other inequality follows from a very similar argument applied to $\boldsymbol{S'}_k$. □

Corollary A.53. *Let $p = \mathbb{P}(\boldsymbol{\tau}_t\left(n^x\right)) \leq \ell | \deg(t) = d)$, and let us assume that $p > M^{2x+\varepsilon-1}$. Then, $(1-\varepsilon)p|T| \leq |\{t \in T : \boldsymbol{\tau}_s\left(n^x\right)) \leq \ell\} \leq (1+\varepsilon)p|T|$.*

Proof. We apply Lemma A.52 with T as the set of nodes of degree d, $\boldsymbol{E}_\ell(t)$ as the event that $\ell \leq 2 + (1-\varepsilon)F\left(k \to n^x\right)$. We obtain that $|\{t \in T : \boldsymbol{\tau}_s\left(n^y\right) \leq (1-\varepsilon)F\left(d \to n^x\right)\}| = |\{t \in T : \forall \ell < \boldsymbol{\tau}_s\left(n^y\right) - 1, \ell \leq (1-\varepsilon)F\left(d \to n^x\right) - 2\}| = (1 \pm o(1))p|T| \pm |T| \frac{M^{2x+o(1)}}{M} = (1 \pm o(1))p|T|$. □

A.6 Proof of Theorem 8.32

Proof that Axiom 1 holds. For the first statement, if $\deg(s) = n^\alpha$ with $\alpha > \varepsilon$, in the case $\beta > 2$, we know by Theorem A.9 that $\boldsymbol{\tau}_s\left(n^y\right) \leq \boldsymbol{\tau}_s\left(n^\alpha\right) + (1+\varepsilon)F\left(n^\alpha \to n^y\right) \leq 1 + (1+$

$\varepsilon)T\left(n^\alpha \to n^y\right) \leq (1+2\varepsilon)T\left(n^\alpha \to n^y\right)$. In the case $1 < \beta < 2$, we know by A.47 that s is connected to the maximum degree node, which has degree $\Theta(n)$: hence, $\boldsymbol{\tau}_s\left(n^x\right) \leq 2 = T\left(n^\alpha \to n^x\right)$.

For the other statements, if x is small enough, this result follows by Corollary A.53 and Theorems A.40 and A.51. For bigger values of x, we can extend it with Theorem A.9. □

Proof that Axiom 2 holds, CM. Let us recall the definition of $\boldsymbol{\Delta}^\ell(s)$ as the set of stubs of nodes at distance ℓ from s, not paired with stubs at distance $\ell - 1$. We know that $\boldsymbol{\Delta}^\ell(s) \geq \boldsymbol{\Gamma}^{\ell+1}(s)^{\max\left(\frac{1}{\beta-1}\right)}$ by Lemmas A.11 and A.44. For $\ell_s = \boldsymbol{\tau}_s\left(n^x\right) - 1$, $\ell_t = \boldsymbol{\tau}_t\left(n^y\right) - 1$, $\boldsymbol{\delta}^{\ell_s}(s) \geq \boldsymbol{\gamma}^{\ell_s+1}(s)^{\max\left(1, \frac{1}{\beta-1}\right)} n^{-\varepsilon} \geq n^{x \max\left(1, \frac{1}{\beta-1}\right) - \varepsilon} \geq M^{x-\varepsilon}$, and similarly $\boldsymbol{\delta}^{\ell_t}(t) \geq M^{y-\varepsilon}$. Consequently, $\boldsymbol{\Delta}^{\ell_s}(s)\boldsymbol{\Delta}^{\ell_t}(t) \geq M^{x+y-2\varepsilon} \geq M^{1+\varepsilon'}$. We claim that, w.h.p., a stub in $\boldsymbol{\Delta}^{\ell_s}(s)$ is paired with a stub in $\boldsymbol{\Delta}^{\ell_t}(t)$, and consequently $\operatorname{dist}(s,t) \leq \ell_s + \ell_t + 1 = \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right) - 1$, proving the theorem. To prove this claim, let us first observe that if $\boldsymbol{N}^{\ell_s}(s)$ and $\boldsymbol{N}^{\ell_t}(t)$ touch each other, then $\operatorname{dist}(s,t) \leq \ell_s + \ell_t < \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right) - 1$, and the result follows. Otherwise, let us assume without loss of generality that $x < y$ (if $x > y$, we swap the roles of s and t, if $x = y$, we can decrease x by a small amount, and we change the value of ε). Let us consider the $M^{x-\varepsilon}$ unpaired stubs $a_1, \dots, a_{M^{x-\varepsilon}}$ in $\boldsymbol{\Delta}^{\ell_t}(s)$, and let us pair these stubs one by one, by defining $\boldsymbol{X}_i = 1$ if the stub is paired to a stub in $\boldsymbol{\Delta}^{\ell_t}(t)$, 0 otherwise. Note that, conditioned on all possible pairings of a_j with $j < i$, $\mathbb{E}[\boldsymbol{X}_i] \geq \frac{M^{y-\varepsilon} - M^{x-\varepsilon}}{M} \geq \frac{M^{y-2\varepsilon}}{M}$. Hence, $\boldsymbol{S}_k = \frac{kM^{y-2\varepsilon}}{M} - \sum_{i=1}^k \boldsymbol{X}_i$ is a supermartingale, and $\operatorname{Var}\left[\boldsymbol{X}_i\right] \leq \mathbb{E}\left[\boldsymbol{X}_i^2\right] \leq \mathbb{E}\left[\boldsymbol{X}_i\right] \leq \frac{M^{y-2\varepsilon}}{M}$. By a strengthened version of Azuma's inequality (Lemma A.6), $\mathbb{P}\left(\sum_{i=1}^k \boldsymbol{X}_i = 0\right) \leq \mathbb{P}\left(k\frac{M^{y-2\varepsilon}}{M} - \sum_{i=1}^k \boldsymbol{X}_i < \varepsilon i \frac{M^{y-2\varepsilon}}{M}\right) \leq e^{\frac{-\varepsilon^2 k^2 M^{2(y-2\varepsilon)}}{\Omega\left(kM^{y-2\varepsilon}M\right)}} = e^{-\Omega\left(\frac{\varepsilon^2 kM^{y-2\varepsilon}}{M}\right)}$. For $k = M^{x-\varepsilon}$, we have proved that, w.h.p., the number of stubs in $\boldsymbol{\Delta}^{\ell_s}(s)$ that are paired with stubs in $\boldsymbol{\Delta}^{\ell_t}(t)$ is at least $(1-\varepsilon)\frac{M^{x+y-3\varepsilon}}{M} \geq 1$, and consequently $\operatorname{dist}(s,t) \leq \ell_s + \ell_t < \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right) - 1$. □

Proof that Axiom 2 holds, IRG. As in Appendix A.2, let $\boldsymbol{\Delta}^\ell(s)$ be the volume of $\boldsymbol{\Gamma}^\ell(s)$, and let $\ell_s = \boldsymbol{\tau}_s\left(n^x\right) - 1$, $\ell_t = \boldsymbol{\tau}_t\left(n^y\right) - 1$. If $\beta > 2$, by Lemma A.16, $\boldsymbol{\delta}^{\ell_s}(s) > (1-\varepsilon)M^x$, and $\boldsymbol{\delta}^{\ell_t}(t) > (1-\varepsilon)M^y$. The probability that a node $v \in \boldsymbol{\Gamma}^{\ell_s}(s)$ is not connected to any node $w \in \boldsymbol{\Gamma}^{\ell_t}(t)$ is $\prod_{w \in \boldsymbol{\Gamma}^{\ell_t}(t)} 1 - f\left(\frac{\rho_v \rho_w}{M}\right)$. We have to consider different cases separately.

- If $\sum_{v \in \boldsymbol{\Gamma}^{\ell_s}(s), \rho_v < \frac{M}{M^y}} \rho_v > \frac{Mn^\varepsilon}{M^y}$, by removing some nodes we can assume that all nodes in $\boldsymbol{\Gamma}^{\ell_s}(s)$ have weight at most $\frac{M}{M^y}$. In this case, the number of nodes $v \in \boldsymbol{\Gamma}^{\ell_s}(s)$ having a connection to $\boldsymbol{\Gamma}^{\ell_t}(t)$ is $\sum_{v \in \boldsymbol{\Gamma}^{\ell_s}(s)} \boldsymbol{X}_v$, where the $\boldsymbol{X}_v$s are independent random variables with success probability $1 - \prod_{w \in \boldsymbol{\Gamma}^{\ell_t}(t)} 1 - f\left(\frac{\rho_v \rho_w}{M}\right) = 1 - \prod_{w \in \boldsymbol{\Gamma}^{\ell_t}(t)} e^{-\Omega\left(\frac{\rho_v \rho_w}{M}\right)} = 1 - e^{-\Omega\left(\frac{\rho_v M^y}{M}\right)} = \Omega\left(\frac{\rho_v M^y}{M}\right)$. We conclude by a straightforward application of the multiplicative form of Chernoff bound (Lemma A.1).

- If we do not fall into the previous case, $\sum_{v \in \boldsymbol{\Gamma}^{\ell_s}(s), \rho_v < \frac{M}{M^y}} \rho_v < \frac{Mn^\varepsilon}{M^y}$, and by slightly decreasing x we can assume without loss of generality that all nodes in $\boldsymbol{\Gamma}^{\ell_s}(s)$ have weight at least $\frac{M}{M^y}$. By changing the roles of s and t, we can also assume that all nodes in $\boldsymbol{\Gamma}^{\ell_t}(t)$ have weight at least $\frac{M}{M^x}$. Assuming this, we still have to divide the analysis in two possible cases.

  - if $\boldsymbol{\gamma}^{\ell_s}(s)\boldsymbol{\gamma}^{\ell_t}(t) > n^\varepsilon$, the number of connections between $\boldsymbol{\Gamma}^{\ell_s}(s)$ and $\boldsymbol{\Gamma}^{\ell_t}(t)$ is at least $\sum_{v \in \boldsymbol{\Gamma}^{\ell_s}(s), w \in \boldsymbol{\Gamma}^{\ell_t}(t)} \boldsymbol{X}_{v,w}$, where the $\boldsymbol{X}_{v,w}$s are independent random variables with success probability $f\left(\frac{\rho_v \rho_w}{M}\right) = \Theta(1)$. Since the sum is made by at least n^ε terms, we can conclude by a straightforward application of the multiplicative form of Chernoff bound (Lemma A.1).

- If $\boldsymbol{\gamma}^{\ell_s}(s)\boldsymbol{\gamma}^{\ell_t}(t) < n^{\varepsilon}$, there is at least a node $v \in \boldsymbol{\Gamma}^{\ell_s}(s)$ with weight $n^{x-\varepsilon}$, and a node $w \in \boldsymbol{\Gamma}^{\ell_t}(t)$ with weight $n^{y-\varepsilon}$. Then, $\mathbb{P}(E(v,w)) = f\left(\frac{\rho_v \rho_w}{M}\right) = f\left(\frac{M^{x+y-2\varepsilon}}{M}\right) \geq f\left(M^{\varepsilon}\right) \geq 1-o(M^{\varepsilon k})$ for each k (we recall that, in our assumptions, $f(x) = 1-o(x^k)$ for each k, if x tends to infinity). We conclude because this means that v is connected to w w.h.p..

□

Proof that Axiom 3 holds. Let us fix $x \geq \frac{1}{2}$, let s be any node, and let us fix an integer ℓ_s such that $\boldsymbol{\delta}^{\ell_s}(s) < M^{x+\varepsilon}$.

Let us consider a node $t \in W$, and let ℓ_t be an integer such that $\boldsymbol{\delta}^{\ell_t}(t) < M^{y+\varepsilon}$: if $\boldsymbol{E}\left(\boldsymbol{\delta}^{\ell_s}(s), \boldsymbol{\delta}^{\ell_t}(t)\right)$ is the event that there is an edge between $\boldsymbol{\Delta}^{\ell_s}(s)$ and $\boldsymbol{\Delta}^{\ell_t}(t)$, $\mathbb{P}\left(\boldsymbol{E}\left(\boldsymbol{\delta}^{\ell_s}(s), \boldsymbol{\delta}^{\ell_t}(t)\right)\middle|\boldsymbol{\delta}^{\ell_s}(s) < M^{x+\varepsilon}, \boldsymbol{\delta}^{\ell}(t) < M^{y+\varepsilon}\right) < \frac{M^{x+y+3\varepsilon}}{M}$. Hence,

$$\begin{aligned}
&P\left(\exists \ell_s, \ell_t : \boldsymbol{\delta}^{\ell_s}(s) < M^{x+\varepsilon} \wedge \boldsymbol{\delta}^{\ell_t}(t) < M^{y+\varepsilon} \wedge \boldsymbol{E}\left(\boldsymbol{\delta}^{\ell_s}(s), \boldsymbol{\delta}^{\ell_t}(t)\right)\right) \\
&\leq \sum_{\ell_s,\ell_t=0}^{\mathcal{O}(\log n)} \mathbb{P}\left(\boldsymbol{\delta}^{\ell_s}(s) < M^{x+\varepsilon} \wedge \boldsymbol{\delta}^{\ell_t}(t) < M^{y+\varepsilon} \wedge \boldsymbol{E}\left(\boldsymbol{\delta}^{\ell_s}(s), \boldsymbol{\delta}^{\ell_t}(t)\right)\right) \\
&\leq \sum_{\ell_s,\ell_t=0}^{\mathcal{O}(\log n)} \mathbb{P}\left(\boldsymbol{E}\left(\boldsymbol{\delta}^{\ell_s}(s), \boldsymbol{\delta}^{\ell_t}(t)\right)\middle|\boldsymbol{\delta}^{\ell_s}(s) < M^{x+\varepsilon}, \boldsymbol{\delta}^{\ell_t}(t) < M^{y+\varepsilon}\right) \\
&\leq \frac{M^{x+y+4\varepsilon}}{M}.
\end{aligned}$$

This means that, with probability $1 - \frac{M^{x+y+4\varepsilon}}{M}$, $\operatorname{dist}(s,t) \geq \ell_s + \ell_t + 2$, where ℓ_s (resp., ℓ_t) is the maximum integer such that $\boldsymbol{\delta}^{\ell_s}(s) < M^{x+\varepsilon}$ (resp., $\boldsymbol{\delta}^{\ell_t}(t) < M^{y+\varepsilon}$). By definition of ℓ_s, ℓ_t, $\boldsymbol{\delta}^{\ell_s+1}(s) > n^{x+\varepsilon}$, and by Lemmas A.11, A.16 and A.44, $\boldsymbol{\gamma}^{\ell_s+2}(s) > n^{x}$, meaning that $\boldsymbol{\tau}_s\left(n^x\right) \leq \ell_s+2$, w.h.p.. Since the same holds for t, $\operatorname{dist}(s,t) \geq \ell_s+\ell_t+2 \geq \boldsymbol{\tau}_s\left(n^x\right)+\boldsymbol{\tau}_t\left(n^y\right)-2$, with probability $1 - \frac{M^{x+y+4\varepsilon}}{M}$.

We have to translate this probabilistic result into a result on the number of nodes t such that $\operatorname{dist}(s,t) < \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right) - 2$. To this purpose, we apply Lemma A.52, by fixing s, conditioning on $\boldsymbol{N}^{\boldsymbol{\tau}_s(n^x)-2}(s)$ (which has volume at most n^x), and defining $\boldsymbol{E}(t)$ as $\operatorname{dist}(s,t) < \boldsymbol{\tau}_s\left(n^x\right) + \boldsymbol{\tau}_t\left(n^y\right) - 2$. Since $y < x$, and $x+y < 1$, $|\{t \in T : \operatorname{dist}(s,t) < \boldsymbol{\tau}_s\left(n^x\right)\} \leq (1+o(1))|T|\frac{M^{x+y+4\varepsilon-1}}{M} \pm |T|\frac{M^{x+y}+M^{2x}}{M^{1-o(1)}} \leq |T|M^{x+y+5\varepsilon-1}$. □

Proof that Axiom 4 holds. For values of d bigger than n^{ε}, by Lemmas A.11, A.16 and A.44, a node with weight d has degree $\Theta\left(d^{\max(1,n^{\frac{1}{\beta-1}})}\right)$. Hence, since the number of nodes with weight at least d is $\Theta\left(\frac{n}{d^{\beta-1}}\right)$, the conclusion follows.

For smaller values of d, a node with weight d has degree bigger than $\frac{1}{2}d^{\max\left(1,\frac{1}{\beta-1}\right)}$ with probability $p = \mathcal{O}(1)$: through simple concentration inequalities it is possible to prove that the degree number of nodes with degree at least $\frac{1}{2}d^{\max\left(1,\frac{1}{\beta-1}\right)}$ is $\mathcal{O}(|\{v \in V : \rho_v \geq d\}|) = \mathcal{O}(\frac{n}{d^{\beta-1}})$. By defining $d' = d^{\max\left(1,\frac{1}{\beta-1}\right)}$, we conclude.

□

A.7 Other Results

Before concluding, we need to prove some lemmas that are used in some probabilistic analyses, even if they do not follow from the main theorems.

Lemma A.54. *Assume that $\beta > 2$, and let T be the set of nodes with degree at least n^x. Then,* $\operatorname{dist}(s,T) := \min_{t \in T} \operatorname{dist}(s,t) \leq \boldsymbol{\tau}_s\left(n^{x(\beta-2)+\varepsilon}\right) + 1$ *w.h.p..*

Proof. By removing some nodes from T, we can redefine T as the set of nodes with weight at least $n^{x+\varepsilon}$ (because each node with weight at least $n^{x+\varepsilon}$ has degree at least n^x by Lemmas A.11, A.16 and A.44). After this modification, the number of nodes in T is $\Theta\left(\frac{n}{n^{(x+\varepsilon)(\beta-1)}}\right) = \Theta\left(n^{1-(x+\varepsilon)(\beta-1)}\right)$, and the volume of T is $\Omega\left(n^{1-(x+\varepsilon)(\beta-1)+x+\varepsilon}\right) = \Omega\left(n^{1-(x+\varepsilon)(\beta-2)}\right)$. We recall the definition of $\boldsymbol{\delta}^\ell(s)$: in the CM, it is the number of stubs at distance ℓ from s, not paired with stubs at distance $\ell-1$, while in IRG it is the volume of the set of nodes at distance ℓ from s. By Lemmas A.11, A.16 and A.44, if $\ell = \boldsymbol{\tau}_s\left(n^{(x+3\varepsilon)(\beta-2)}\right) - 1$, $\boldsymbol{\delta}^\ell(s) \geq n^{(x+2\varepsilon)(\beta-2)}$. In the CM, since the pairing of stubs is random, there is w.h.p. a stub in $\boldsymbol{\Delta}^\ell(s)$ which is paired with a stub of a node in T. In IRG, the probability that a node in $\boldsymbol{\Gamma}^\ell(s)$ is paired with a node in T is at least $\sum_{v \in \boldsymbol{\Gamma}^\ell(s)} \sum_{t \in T} \boldsymbol{X}_{vt}$, where the $\boldsymbol{X}_{vt}s$ are Bernoulli random variables with success probability $f\left(\frac{\rho_v \rho_t}{M}\right)$. We conclude by a straightforward application of Chernoff bound (Lemma A.1). □

Lemma A.55. *Given a node v and an integer ℓ, assume that $n^\varepsilon < \boldsymbol{\gamma}^\ell(v) < n^{1-\varepsilon}$, and let $S = \{s \in V : n^\alpha < \deg(s) < n^{\alpha+\varepsilon}\}$, for some $\alpha > 0$. Then, $|S \cap \boldsymbol{\Gamma}^\ell(v)| \leq \boldsymbol{\gamma}^\ell(v)|S|n^{-1+\alpha+\varepsilon}$ w.h.p..*

Proof for the CM. By Lemma A.11, we can assume that $n^\varepsilon \leq \boldsymbol{\delta}^{\ell-1}(v) \leq n^{1-\varepsilon}$. Let us sort the stubs in $\boldsymbol{\Delta}^{\ell-1}(v)$, and let $\boldsymbol{X}_i$ be 1 if the i-th stub is paired with a stub of a node in S, 0 otherwise. Clearly, $|S \cap \boldsymbol{\Gamma}^\ell(v)| \leq \sum_{i=1}^{\boldsymbol{\delta}^{\ell-1}(v)} \boldsymbol{X}_i$. Since $\boldsymbol{\delta}^{\ell-1}(v) < n^{1-\varepsilon}$, conditioned on the outcome of the previous variables $\boldsymbol{X}_j$, $\mathbb{P}\left(\boldsymbol{X}_i = 1\right) = \mathcal{O}\left(\frac{1}{n}\sum_{v \in S} \rho_v\right) \leq |S|n^{-1+\alpha+\varepsilon}$ (because we have already paired at most $o(n)$ stubs). Hence, $\mathbf{S}_k = k|S|n^{-1+\alpha+\varepsilon} - \sum_{i=0}^k \boldsymbol{X}_i$ is a submartingale, and if $k = \boldsymbol{\delta}^{\ell-1}(v)$, by the strenghtened version of Azuma's inequality (Lemma A.6), w.h.p., $\boldsymbol{S}_k \geq -k|S|n^{-1+\alpha+\varepsilon}$, that is, $k|S|n^{-1+\alpha+\varepsilon} - \sum_{i=0}^k \boldsymbol{X}_i \geq -k|S|n^{-1+\alpha+\varepsilon}$, and $|S \cap \boldsymbol{\Gamma}^\ell(v)| \leq \sum_{i=0}^k \boldsymbol{X}_i \leq 2k|S|n^{-1+\alpha+\varepsilon}$. The result follows. □

Proof for IRG. By Lemma A.16, we can assume that $n^\varepsilon \leq \boldsymbol{\delta}^{\ell-1}(v) \leq n^{1-\varepsilon}$. The probability that a node $s \in S$ is not linked to any node in $\boldsymbol{\Gamma}^{\ell-1}(v)$ is

$$\begin{aligned}
prod_{w \in \boldsymbol{\Gamma}^{\ell-1}(v)}\left(1 - f\left(\frac{\rho_w \rho_s}{n}\right)\right) &= \prod_{w \in \boldsymbol{\Gamma}^{\ell-1}(v)}\left(1 - \mathcal{O}\left(\frac{\rho_w \rho_s}{n}\right)\right) \\
&\leq \prod_{w \in \boldsymbol{\Gamma}^{\ell-1}(v)} e^{-\mathcal{O}\left(\frac{\rho_w \rho_s}{n}\right)} \\
&= e^{-\mathcal{O}\left(\frac{\boldsymbol{\delta}^{\ell-1}(v)\rho_s}{n}\right)} \\
&= e^{-\mathcal{O}\left(\frac{\boldsymbol{\delta}^{\ell-1}(v)n^{\alpha+\varepsilon}}{n}\right)}.
\end{aligned}$$

If $\boldsymbol{\delta}^{\ell-1}(v) > n^{1-\alpha-2\varepsilon}$, the result of the lemma is trivial, if we change the value of ε. If $\boldsymbol{\delta}^{\ell-1}(v) < n^{1-\alpha-2\varepsilon}$, the probability that a node in S is not linked to any node in $\boldsymbol{\delta}^{\ell-1}(s)$ is $e^{-\mathcal{O}\left(\frac{\boldsymbol{\delta}^{\ell-1}(s)n^{\alpha+\varepsilon}}{n}\right)} = 1 - \mathcal{O}\left(\frac{\boldsymbol{\delta}^{\ell-1}(s)n^{\alpha+\varepsilon}}{n}\right)$, and hence the probability that it is connected to a node in $\boldsymbol{\delta}^{\ell-1}(s)$ is $\mathcal{O}\left(\frac{\boldsymbol{\delta}^{\ell-1}(s)n^{\alpha+\varepsilon}}{n}\right)$. By a straightforward application of Chernoff bound, the number of nodes in S that belong to $\boldsymbol{\Gamma}^{\ell}(s)$ is $\mathcal{O}\left(\frac{|S|\boldsymbol{\delta}^{\ell-1}(s)n^{\alpha+\varepsilon}}{n}\right)$, w.h.p.. □

Lemma A.56. *Assume that $\beta > 3$, and let v a node with degree $\omega(1)$. Let S be the set of nodes with degree between n^{α} and $n^{\alpha+\varepsilon}$. Then, the number of pairs of nodes $s, t \in S$ such that* $\operatorname{dist}(s, v) + \operatorname{dist}(v, t) \leq c \log_{M_1(\mu)} n$*, and* $\operatorname{dist}(s, w) + \operatorname{dist}(w, t) > c \log_{M_1(\mu)} n$ *for each w such that* $\deg(w) > \deg(v)$ *is at most* $\deg(v)^2|S|^2 n^{-2+c+2\alpha+\varepsilon}$.

Proof. First of all, we want to assume without loss of generality that v is the node with maximum degree. To this purpose, we remove from the graph all nodes with degree bigger than $\deg(v)$: by Lemma A.38, we obtain a new random graph G', and the value of $M_1(\mu)$ changes by $o(1)$. Furthermore, v is the node with maximum degree in the new graph, and the shortest paths not passing from nodes with degree bigger than $\deg(v)$ are conserved.

Consequently, let us assume that v is the node with highest degree. By Corollaries A.14 and A.19, $\boldsymbol{\gamma}^i(v) \leq n^{\varepsilon} \deg(v)(M_1(\mu) + \varepsilon)^i$, and by Lemma A.55, $|S \cap \boldsymbol{\Gamma}^i(v)| \leq \boldsymbol{\gamma}^i(v)|S|n^{-1+\alpha+\varepsilon} \leq \deg(v)|S|(M_1(\mu) + \varepsilon)^i n^{-1+\alpha+2\varepsilon}$. We conclude that the number of pairs $(s, t) \in S^2$ such that $\operatorname{dist}(s, v) + \operatorname{dist}(v, t) \leq c \log_{M_1(\mu)} n$ is at most:

$$\begin{aligned}
&\sum_{i+j=c\log_{M_1(\mu)} n} |\{s : \operatorname{dist}(s, v) \leq i\}||\{t : \operatorname{dist}(t, v) \leq c \log_{M_1(\mu)} n - i\}| \\
&\leq \sum_{i+j=c\log_{M_1(\mu)} n} i \deg(v)|S|(M_1(\mu) + \varepsilon)^i \cdot j \deg(v)|S|(M_1(\mu) + \varepsilon)^j n^{-2+2\alpha+4\varepsilon} \\
&\leq \sum_{i+j=c\log_{M_1(\mu)} n} \deg(v)^2|S|^2(M_1(\mu) + \varepsilon)^{i+j} n^{-2+2\alpha+5\varepsilon} \\
&\leq \deg(v)^2|S|^2 n^{-2+c+2\alpha+\varepsilon'}.
\end{aligned}$$

□

Bibliography

[1] Amir Abboud, Fabrizio Grandoni, and Virginia V. Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In *Proceedings of the 26th ACM-SIAM Symposium on Discrete Algorithms*, pages 1681–1697, 2015.

[2] Amir Abboud and Virginia V. Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proceedings of the 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 434–443, 2014.

[3] Amir Abboud, Virginia V. Williams, and Joshua Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter. In *Proceedings of the 26th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 377–391, 2016.

[4] Amir Abboud, Virginia V. Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 39–51, 2014.

[5] Ittai Abraham and Amos Fiat. Highway dimension, shortest paths, and provably efficient algorithms. In *Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 782–793, 2010.

[6] Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive sampling for k-means clustering. *Proceedings of the 12th International Workshop on Approximation, Randomization, and Combinatorial Optimization Algorithms and Techniques (APPROX)*, pages 15–28, 2009.

[7] Takuya Akiba, Yoichi Iwata, and Yuki Kawata. An exact algorithm for diameters of large real directed graphs. In *Proceedings of the 14th International Symposium on Experimental Algorithms (SEA)*, pages 56–67, 2015.

[8] Takuya Akiba, Yoichi Iwata, and Yuichi Yoshida. Fast exact shortest-path distance queries on large networks by pruned landmark labeling. In *Proceedings of the 2013 ACM-SIGMOD International Conference on Management of Data*, pages 349–360, 2013.

[9] Réka Albert, Bhaskar DasGupta, and Nasim Mobasheri. Topological implications of negative curvature for biological and social networks. *Physical Review E*, 89(3):032811, 2014.

[10] Hend Alrasheed and Feodor F. Dragan. Core-periphery models for graphs based on their δ-hyperbolicity: an example using biological networks. *Proceedings of the 6th Workshop on Complex Networks (CompleNet)*, pages 65–77, 2015.

[11] Jacob M. Anthonisse. *The Rush in a Directed Graph.* Stichting Mathematisch Centrum. Mathematische Besliskunde, 1971.

[12] Krishna B. Athreya and Peter Ney. *Branching processes.* Dover Books on Mathematics Series. Springer-Verlag Berlin Heidelberg New York, 1972.

[13] David A. Bader, Shiva Kintali, Kamesh Madduri, and Milena Mihail. Approximating betweenness centrality. In *Proceedings of the 5th international conference on Algorithms and models for the web-graph (WAW)*, pages 124–137, 2007.

[14] Jorgen Bang-Jensen and Gregory Gutin. *Digraphs Theory, Algorithms and Applications.* Springer, 2008.

[15] Albert L. Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[16] Alex Bavelas. A Mathematical Model for Group Structures. *Human Organization*, 7(3):16–30, 1948.

[17] Alex Bavelas. Communication patterns in task-oriented groups. *Journal of the Acoustical Society of America*, 22:725–730, 1950.

[18] Boaz Ben-Moshe, Binay Bhattacharya, Qiaosheng Shi, and Arie Tamir. Efficient algorithms for center problems in cactus networks. *Theoretical Computer Science*, 378(3):237–252, 2007.

[19] Elisabetta Bergamini. Personal communication, 2016.

[20] Elisabetta Bergamini, Michele Borassi, Pierluigi Crescenzi, Andrea Marino, and Henning Meyerhenke. Computing top-k closeness centrality faster in unweighted graphs. In *Proceedings of the Meeting on Algorithm Engineering and Experiments (ALENEX)*, pages 68–80, 2016.

[21] Elisabetta Bergamini and Henning Meyerhenke. Fully-dinamic approximation of betweenness centrality. In *Proceedings of the 23rd European Symposium on Algorithms (ESA)*, 2015.

[22] Marián Boguná, Fragkiskos Papadopoulos, and Dmitri Krioukov. Sustaining the internet with hyperbolic mapping. *Nature Communications*, 1(62):1–8, 2010.

[23] Paolo Boldi and Sebastiano Vigna. The webgraph framework I: compression techniques. In *Proceedings of the 13th international conference on World Wide Web (WWW)*, pages 595–602, 2004.

[24] Paolo Boldi and Sebastiano Vigna. In-core computation of geometric centralities with hyperball: A hundred billion nodes and beyond. In *Proceedings of the 2013 IEEE 13th International Conference on Data Mining Workshops*, pages 621–628, 2013.

[25] Paolo Boldi and Sebastiano Vigna. Axioms for centrality. *Internet Mathematics*, 10(3-4):222–262, 2014.

[26] Béla Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European Journal of Combinatorics*, 1(4):311–316, 1980.

[27] Béla Bollobás, Svante Janson, and Oliver Riordan. The phase transition in inhomogeneous random graphs. *Random Structures and Algorithms*, 31(1):3–122, 2007.

[28] Michele Borassi. A Note on the Complexity of Computing the Number of Reachable Vertices in a Digraph. *Information Processing Letters*, 116(10):628–630, 2016.

[29] Michele Borassi, Alessandro Chessa, and Guido Caldarelli. Hyperbolicity measures democracy in real-world networks. *Physical Review E*, 92(3):032812, 2015.

[30] Michele Borassi, David Coudert, Pierluigi Crescenzi, and Andrea Marino. On computing the hyperbolicity of real-world graphs. In *Proceedings of the 23rd European Symposium on Algorithms (ESA)*, pages 215–226. Springer, 2015.

[31] Michele Borassi, Pierluigi Crescenzi, and Michel Habib. Into the square - On the complexity of some quadratic-time solvable problems. *Electronic Notes in Computer Science*, 322:51–67, 2016.

[32] Michele Borassi, Pierluigi Crescenzi, Michel Habib, Walter A. Kosters, Andrea Marino, and Frank W. Takes. Fast diameter and radius BFS-based computation in (weakly connected) real-world graphs - With an application to the Six Degrees of Separation games. *Theoretical Computer Science*, 586:59–80, 2014.

[33] Michele Borassi, Pierluigi Crescenzi, Michel Habib, Walter A. Kosters, Andrea Marino, and Frank W. Takes. On the solvability of the Six Degrees of Kevin Bacon game - A faster graph diameter and radius computation method. In *Proceedings of the 7th International Conference on Fun with Algorithms (FUN)*, pages 57–68, 2014.

[34] Michele Borassi, Pierluigi Crescenzi, and Andrea Marino. Fast and simple computation of top-k closeness centralities. *arXiv preprint 1507.01490*, 2015.

[35] Michele Borassi, Pierluigi Crescenzi, and Luca Trevisan. An axiomatic and an average-case analysis of algorithms and heuristics for metric properties of graphs. In *Proceedings of the 27th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2017. Accepted.

[36] Michele Borassi and Emanuele Natale. KADABRA is an adaptive algorithm for betweenness via random approximation. In *Proceedings of the 24th European Symposium on Algorithms*, 2016.

[37] Stephen P. Borgatti and Martin G. Everett. A graph-theoretic perspective on centrality. *Social Networks*, 28(4):466–484, 2006.

[38] Pawel Brach, Marek Cygan, Jakub Lacki, and Piotr Sankowski. Algorithmic complexity of power law networks. In *Proceedings of the 26th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1306–1325, 2016.

[39] Ulrik Brandes. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[40] Ulrik Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30(2):136–145, 2008.

[41] Ulrik Brandes and Thomas Erlebach, editors. *Network Analysis: Methodological Foundations*, volume 3418. Springer, 2005.

[42] Ulrik Brandes and Daniel Fleischer. Centrality Measures Based on Current Flow. In *Proceedings of the 22nd annual conference on Theoretical Aspects of Computer Science (STACS)*, pages 533–544, 2005.

[43] Ulrik Brandes and Christian Pich. Centrality Estimation in Large Networks. *International Journal of Bifurcation and Chaos*, 17(7):2303–2318, 2007.

[44] Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *Proceedings of the 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 661–670, 2014.

[45] Andrei Z. Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet L. Wiener. Graph Structure in the Web. *Computer Networks*, 33(1-6):309–320, 2000.

[46] Marco L. Carmosino, Gao Jiawei, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic Extensions of the Strong Exponential Time Hypothesis and Consequences for Non-reducibility. In *Proceedings of the ACM Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 261–270, 2016.

[47] Shiri Chechik, Daniel H. Larkin, Liam Roditty, Grant Schoenebeck, Robert E. Tarjan, and Virginia V. Williams. Better Approximation Algorithms for the Graph Diameter. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1041–1052, 2014.

[48] Duanbing Chen, Linyuan Lü, Ming-Sheng Shang, Yi-Cheng Zhang, and Tao Zhou. Identifying influential nodes in complex networks. *Physica A: Statistical Mechanics and its Applications*, 391(4):1777–1787, 2012.

[49] Wei Chen, Wenjie Fang, Guangda Hu, and Michael W. Mahoney. On the hyperbolicity of small-world and treelike random graphs. *Internet Mathematics*, 9(4):434–491, 2013.

[50] Victor Chepoi, Fedor F. Dragan, Bertrand Estellon, Michel Habib, and Yann Vaxès. Diameters, centers, and approximating trees of δ-hyperbolic geodesic spaces and graphs. In *Proceedings of the 24th Annual Symposium on Computational Geometry*, pages 59–68, 2008.

[51] Victor Chepoi, Feodor F. Dragan, Bertrand Estellon, Michel Habib, Yann Vaxès, and Yang Xiang. Additive spanners and distance and routing labeling schemes for hyperbolic graphs. *Algorithmica*, 62(3):713–732, 2012.

[52] Victor Chepoi and Bertrand Estellon. Packing and covering δ-hyperbolic spaces by balls. In *Proceedings of the 10th International Workshop on Approximation and the 11th International Workshop on Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX-RANDOM)*, Lecture Notes in Computer Science, pages 59–73. Springer, 2007.

[53] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, Alessandro Panconesi, and Prabhakar Raghavan. Models for the compressible web. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 331–340, 2009.

[54] Fan Chung and Linyuan Lu. *Complex graphs and networks.* American Mathematical Society, Boston, MA, USA, 2006.

[55] Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F. Werneck. Computing classic closeness centrality, at scale. In *Proceedings of the second ACM conference on Online social networks (COSN)*, pages 37–50. ACM, 2014.

[56] Nathann Cohen, David Coudert, Guillaume Ducoffe, and Aurélien Lancin. Applying clique-decomposition for computing Gromov hyperbolicity. Technical report, HAL, 2014.

[57] Nathann Cohen, David Coudert, and Aurélien Lancin. Exact and approximate algorithms for computing the hyperbolicity of large-scale graphs. Technical report, HAL, 2013.

[58] Nathann Cohen, David Coudert, and Aurélien Lancin. On computing the Gromov hyperbolicity. *Journal of Experimental Algorithms*, 20:1.6, 2015.

[59] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms (3rd edition).* MIT Press, 2009.

[60] Derek G. Corneil, Feodor F. Dragan, and Ekkehard Köhler. On the power of BFS to determine a graph's diameter. *Networks*, 42(4):209–222, 2003.

[61] Pierluigi Crescenzi, Roberto Grossi, Michel Habib, Leonardo Lanzi, and Andrea Marino. On computing the diameter of real-world undirected graphs. *Theoretical Computer Science*, 514:84–95, 2013.

[62] Pierluigi Crescenzi, Roberto Grossi, Claudio Imbrenda, Leonardo Lanzi, and Andrea Marino. Finding the diameter in real-world graphs experimentally turning a lower bound into an upper bound. In *Proceedings of the 18th European Symposium on Algorithms (ESA)*, pages 302–313, Berlin, Heidelberg, 2010. Springer-Verlag.

[63] Pierluigi Crescenzi, Roberto Grossi, Leonardo Lanzi, and Andrea Marino. On computing the diameter of real-world directed (weighted) graphs. In *Proceedings of the 11th International Symposium on Experimental Algorithms (SEA)*, pages 99–110, 2012.

[64] Gábor Csárdi and Tamás Nepusz. The igraph software package for complex network research. *InterJournal, Complex Systems*, page 1695, 2006.

[65] Dennis De Champeaux. Bidirectional heuristic search again. *Journal of the ACM (JACM)*, 30(1):22–32, 1983.

[66] Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck. Hub label compression. In *Proceedings of the 12th International Symposium on Experimental Algorithms (SEA)*, pages 18–29, 2013.

[67] Daniel Delling and Renato F. Werneck. Faster customization of road networks. In *Proceedings of the 12th International Symposium on Experimental Algorithms (SEA)*, pages 30–42, 2013.

[68] Shlomi Dolev, Yuval Elovici, and Rami Puzis. Routing betweenness centrality. *Journal of the ACM*, 57(4):1–27, 2010.

[69] Andreas Dress, Katharina T. Huber, Jacobus Koolen, Vincent Moulton, and Andreas Spillner. *Basic phylogenetic combinatorics.* Cambridge University Press, Cambridge, UK, 2012.

[70] David Easley and Jon M. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World.* Cambridge University Press, New York, 2010.

[71] Amr Elmasry. The Subset Partial Order: Computing and Combinatorics. In *Proceedings of the Meeting on Algorithm Engineering and Expermiments*, pages 27–33, 2010.

[72] David Eppstein and Joseph Wang. Fast Approximation of Centrality. *Journal of Graph Algorithms and Applications*, 8(1):39–45, 2004.

[73] Dóra Erdös, Vatche Ishakian, Azer Bestavros, and Evimaria Terzi. A Divide-and-Conquer Algorithm for Betweenness Centrality. *arXiv preprint 1406.4173*, pages 433–441, 2015.

[74] Paul Erdös. *Extremal problems in graph theory.* Theory of Graphs and Its Applications: Proceedings of the Symposium Held in Smolenice in June 1963. Pub. House of the Czechoslovak Academy of Sciences, 1964.

[75] Wenjie Fang. *On hyperbolic geometry structure of complex networks.* Internship Report, Microsoft Research Asia, 2011.

[76] Daniel Fernholz and Vijaya Ramachandran. The diameter of sparse random graphs. *Random Structures and Algorithms*, 31(4):482–516, 2007.

[77] Hervé Fournier, Anas Ismail, and Antoine Vigneron. Computing the Gromov hyperbolicity of a discrete metric space. *Information Processing Letters*, 115(6):576–579, 2015.

[78] Linton C. Freeman. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1):35–41, 1977.

[79] Alan Frieze and Colin McDiarmid. Algorithmic theory of random graphs. *Random Structures and Algorithms*, 10(1-2):5–42, 1997.

[80] Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry*, 5(3):165–185, 1995.

[81] François Le Gall. Powers of Tensors and Fast Matrix Multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 296–303, 2014.

[82] Robert Geisberger, Peter Sanders, and Dominik Schultes. Better Approximation of Betweenness Centrality. In *Proceedings of the 10th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 90–100, 2008.

[83] Andrew V. Goldberg and Chris Harrelson. Computing the shortest path: A* search meets graph theory. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 156–165, 2005.

[84] Mikhael Gromov. Hyperbolic groups. *Essays in Group Theory*, 8:75–265, 1987.

[85] Rishi Gupta, Tim Roughgarden, and Comandur Seshadhri. Decompositions of Triangle-Dense Graphs. *SIAM Journal on Computing*, 45(2):197–215, 2016.

[86] Michael Gurevich. *The Social Structure of Acquaintanceship Networks.* PhD thesis, Massachusetts Institute of Technology, 1961.

[87] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy)*, pages 11–15, 2008.

[88] Gabriel Y. Handler. Minimax Location of a Facility in an Undirected Tree Graph. *Transportation Science*, 7(3):287–293, 1973.

[89] Frank Harary. *Graph Theory.* Addison-Wesley series in mathematics. Perseus Books, 1994.

[90] Shlomo Havlin and Reuven Cohen. *Complex networks: structure, robustness and function.* Cambridge University Press, Cambridge, 2010.

[91] Kaave Hosseini. *3SUM.* Online book, available at `http://cseweb.ucsd.edu/~skhossei/3sum.pdf`, 2015.

[92] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[93] Riko Jacob, Dirk Kosch, Katharina Anna Lehmann, and Leon Peeters. Algorithms for centrality indices. In *Network Analysis - Methodological Foundations*, pages 62–82. Springer, 2005.

[94] Edmond A. Jonckheere and Poonsuk Lohsoonthorn. Geometry of network security. In *Proceedings of the American Control Conference*, volume 2, pages 976–981, Boston, MA, USA, 2004. IEEE.

[95] H. Kaindl and G. Kainz. Bidirectional Heuristic Search Reconsidered. *Journal of Artificial Intelligence Research*, 7:283–317, 1997.

[96] U Kang, Spiros Papadimitriou, Jimeng Sun, and Tong Hanghang. Centralities in large networks: Algorithms and observations. In *Proceedings of the 2011 SIAM International Conference on Data Mining (SDM)*, pages 119–130, 2011.

[97] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.

[98] W. Sean Kennedy, Onuttom Narayan, and Iraj Saniee. On the Hyperbolicity of Large-Scale Networks. *arXiv preprint 1307.0031*, pages 1–22, 2013.

[99] James King. *A survey of 3SUM-hard problems.* Available at `http://www.ccs.neu.edu/home/viola/classes/papers/King04Survey3sum.pdf`, 2004.

[100] Robert Krauthgamer and James R. Lee. Algorithms on negatively curved spaces. In *Proceedings of the 47th Symposium on Foundations of Computer Science (FOCS)*, pages 119–132, 2006.

[101] Dmitri Krioukov, Fragkiskos Papadopoulos, Marián Boguñá, and Amin Vahdat. Greedy forwarding in scale-free networks embedded in hyperbolic metric spaces. *Proceedings of the 29th conference on Information communications (INFOCOM)*, pages 2973–2981, 2010.

[102] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, and Amin Vahdat. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

[103] Jérôme Kunegis. KONECT – the Koblenz network collection. In *Proceedings of the 22nd International Conference on World Wide Web Companion*, pages 1343–1350, 2013.

[104] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1):016118, 2009.

[105] Vito Latora and Massimo Marchiori. A measure of centrality based on network efficiency. *New Journal of Physics*, 9:188, 2007.

[106] Erwan Le Merrer, Nicolas Le Scouarnec, and Gilles Trédan. Heuristical Top-k: Fast Estimation of Centralities in Complex Networks. *Information Processing Letters*, 114(8):432–436, 2014.

[107] Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. Graph Evolution: Densification and Shrinking Diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):1–41, 2007.

[108] Jure Leskovec and Rok Sosič. SNAP: A general purpose network analysis and graph mining library in C++. `http://snap.stanford.edu/snap`, 2014.

[109] Yeon-sup Lim, Daniel S. Menasché, Bruno Ribeiro, Don Towsley, and Prithwish Basu. Online estimating the k central nodes of a network. In *Proceedings of the 2011 IEEE Network Science Workshop (NSW)*, pages 118–122, 2011.

[110] Nan Lin. *Foundations of social research.* McGraw-Hill, 1976.

[111] Richard J. Lipton and Jeffrey F. Naughton. Estimating the size of generalized transitive closure. In *Proceedings of the 15th international conference on Very large data bases (VLDB)*, pages 165–171, 1989.

[112] Richard J. Lipton and Jeffrey F. Naughton. Query size estimation by adaptive sampling. *Journal of Computer and System Sciences*, 51:18–25, 1995.

[113] Maarten Löffler and Jeff M. Phillips. Shape fitting on point sets with probability distributions. In *Proceedings of the 17th European Symposium on Algorithms (ESA)*, pages 313–324, 2009.

[114] Clémence Magnien, Matthieu Latapy, and Michel Habib. Fast computation of empirically tight bounds for the diameter of massive graphs. *Journal of Experimental Algorithmics (JEA)*, 13:1.10:1–1.10:9, 2009.

[115] Zoltan A. Mann and Anikò Szajkó. Average-case complexity of backtrack search for coloring sparse random graphs. *Journal of Computer and System Sciences*, 79(8):1287–1301, 2013.

[116] Andrea Marino. *Algorithms for biological graphs - analysis and enumeration.* Atlantis Press, 2015.

[117] Stanley Milgram. The Small World Problem. *Psychology Today*, 2:60–67, 1967.

[118] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and Analysis of Online Social Networks. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (MC)*, pages 29–42, 2007.

[119] Dieter Mitsche and Pawel Pralat. On the hyperbolicity of random graphs. *The Electronic Journal of Combinatorics*, 21(2):P2.39, 2014.

[120] Onuttom Narayan and Iraj Saniee. The Large Scale Curvature of Networks. *Physical Review E*, 84:066108, 2011.

[121] Onuttom Narayan, Iraj Saniee, and Gabriel H. Tucci. Lack of hyperbolicity in asymptotic Erdös–Renyi sparse random graphs. *Internet Mathematics*, 11(3):277–288, 2015.

[122] Mark E. J. Newman. Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. *Physical Review E*, 64:016132, 2001.

[123] Mark E. J. Newman. Assortative mixing in networks. *Physical Review Letters*, 89(20):208701, 2002.

[124] Mark E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67(2):026126, 2003.

[125] Mark E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[126] Mark E. J. Newman. *Networks: an introduction.* OUP Oxford, 2010.

[127] Ilkka Norros and Hannu Reittu. On a conditionally Poissonian graph process. *Advances in Applied Probability*, 38(1):59–75, 2006.

[128] Kazuya Okamoto, Wei Chen, and Xiang-Yang Li. Ranking of closeness centrality for large-scale social networks. *Frontiers in Algorithmics*, 5059:186–195, 2008.

[129] Paul W. Olsen, Alan G. Labouseur, and Jeong-Hyon Hwang. Efficient top-k closeness centrality search. In *Proceedings of the 30th IEEE International Conference on Data Engineering (ICDE)*, pages 196–207, 2014.

[130] Mihai Patrascu and Liam Roditty. Distance Oracles Beyond the Thorup–Zwick Bound. *SIAM Journal on Computing*, 43(1):300–311, 2014.

[131] Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. *Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1065–1075, 2010.

[132] Jürgen Pfeffer and Kathleen M. Carley. K-centralities: local approximations of global measures based on shortest paths. In *Proceedings of the 21st Annual Conference on World Wide Web (WWW)*, pages 1043–1050, 2012.

[133] Andrea Pietracaprina, Matteo Riondato, Eli Upfal, and Fabio Vandin. Mining top-k frequent itemsets through progressive sampling. *Data Mining and Knowledge Discovery*, 21(2):310–326, 2010.

[134] Ira Pohl. *Bi-directional and heuristic search in path problems.* PhD thesis, Stanford University, 1969.

[135] Paul Pritchard. On computing the subset graph of a collection of sets. *Journal of Algorithms*, 33(2):187–203, 1999.

[136] Matteo Riondato and Evgenios M. Kornaropoulos. Fast approximation of betweenness centrality through sampling. *Data Mining and Knowledge Discovery*, 30(2):438–475, 2015.

[137] Matteo Riondato and Eli Upfal. ABRA: Approximating Betweenness Centrality in Static and Dynamic Graphs with Rademacher Averages. *arXiv preprint 1602.05866*, pages 1–27, 2016.

[138] Liam Roditty and Virginia V. Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the 45th ACM Symposium on Theory of Computing (STOC)*, pages 515–524, New York, New York, USA, 2013. ACM Press.

[139] Sheldon M. Ross. *Introductory Statistics.* Elsevier, Canada, third edit edition, 2010.

[140] Benjamin Rossman. *Average-case complexity of detecting cliques.* PhD thesis, Massachussets Institute of Technology, 2010.

[141] Peter Sanders and Dominik Schultes. Highway hierarchies hasten exact shortest path queries. In *Proceedings of the 13th European Symposium on Algorithms (ESA)*, pages 568–579, 2005.

[142] Ahmet E. Sarıyüce, Kamer Kaya, Erik Saule, and Umit V. Catalyurek. Incremental algorithms for closeness centrality. In *IEEE International Conference on Big Data*, pages 118–122, 2013.

[143] Ahmet Erdem Sarıyüce, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. Shattering and Compressing Networks for Centrality Analysis. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 686–694, 2012.

[144] Yilun Shang. Non-Hyperbolicity of Random Graphs with Given Expected Degrees. *Stochastic Models*, 29(4):451–462, 2013.

[145] Marvin E. Shaw. Group Structure and the Behavior of Individuals in Small Groups. *The Journal of Psychology*, 38(1):139–149, 1954.

[146] Alfonso Shimbel. Structural parameters of communication networks. *The Bulletin of Mathematical Biophysics*, 15(4):501–507, 1953.

[147] Jeremy G. Siek, Lie Quan Lee, and Andrew Lumsdaine. *The Boost Graph Library: User Guide and Reference Manual.* Addison-Wesley Longman Publishing Co., 2001.

[148] Lenie Sint and Dennis de Champeaux. An Improved Bidirectional Heuristic Search Algorithm. *Journal of the ACM*, 24(2):177–191, 1977.

[149] Christian Sommer, Elad Verbin, and Wei Yu. Distance oracles for sparse graphs. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 703–712, 2009.

[150] Mauricio A. Soto Gómez. *Quelques propriétés topologiques des graphes et applications à internet et aux réseaux*. PhD thesis, Université Paris Diderot (Paris 7), 2011.

[151] Christian L. Staudt, Aleksejs Sazonovs, and Henning Meyerhenke. Networkit: an interactive tool suite for high-performance network analysis. *arXiv preprint 1403.3005*, pages 1–25, 2014.

[152] William Stein and David Joyner. Sage: system for algebra and geometry experimentation. *SIGSAM Bulletin*, 39(2):61–64, 2005.

[153] Frank W. Takes. *Algorithms for Analyzing and Mining Real-World Graphs*. PhD thesis, Universiteit Leiden, 2014.

[154] Frank W. Takes and Walter A. Kosters. Determining the diameter of small world networks. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1191–1196, New York, New York, USA, 2011. ACM Press.

[155] Frank W. Takes and Walter A. Kosters. Computing the eccentricity distribution of large graphs. *Algorithms*, 6(1):100–118, 2013.

[156] Kristin Thompson and David Bordwell. *Film History: an Introduction*. McGraw-Hill Higher Education, 2009.

[157] Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM*, 52(1):1–24, 2005.

[158] Henri van den Esker, Remco van der Hofstad, Gerard Hooghiemstra, and Dmitri Znamenski. Distances in random graphs with infinite mean degrees. *Extremes*, 8(3):111–141, 2005.

[159] Remco van der Hofstad. *Random graphs and complex networks*, volume II. Online book, available at `http://www.win.tue.nl/~rhofstad/NotesRGCNII.pdf`, 2014.

[160] Remco van der Hofstad. *Random graphs and complex networks*, volume I. Online book, available at `http://www.win.tue.nl/~rhofstad/NotesRGCNII.pdf`, 2016.

[161] Remco van der Hofstad, Gerard Hooghiemstra, and Piet Van Mieghem. Distances in random graphs with finite variance degrees. *Random Structures and Algorithms*, 27(1):76–123, 2005.

[162] Remco van der Hofstad, Gerard Hooghiemstra, and Dmitri Znamenski. Distances in random graphs with finite mean and infinite variance degrees. *Electronic Journal of Probability*, 12:703–766, 2007.

[163] Flavio Vella, Giancarlo Carbone, and Massimo Bernaschi. Algorithms and Heuristics for Scalable Betweenness Centrality Computation on Multi-GPU Systems. *arXiv preprint 1602.00963*, pages 1–26, 2016.

[164] Sebastiano Vigna. Personal communication, 2016.

[165] Aravindan Vijayaraghavan. *Beyond worst-case analysis in approximation algorithms*. PhD thesis, Princeton, 2012.

[166] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications.* Structural Analysis in the Social Sciences. Cambridge University Press, 1994.

[167] Douglas B. West. *Introduction to Graph Theory.* Prentice Hall, 2 edition, 2000.

[168] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005.

[169] Ryan Williams and Huacheng Yu. Finding orthogonal vectors in discrete structures. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1867–1877, 2014.

[170] Virginia V. Williams. Multiplying matrices faster than Coppersmith-Winograd. In Howard J Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing (STOC)*, pages 887–898. ACM, 2012.

[171] Virginia V. Williams and Ryan Williams. Subcubic Equivalences between Path, Matrix and Triangle Problems. In *Proceedings of the 51st Annual Symposium on Foundations of Computer Science*, pages 645–654, 2010.

[172] Yaokun Wu and Chengpeng Zhang. Hyperbolicity and chordality of a graph. *The Electronic Journal of Combinatorics*, 18(1):P43, 2011.

[173] Daniel M. Yellin and Charanjit S. Jutla. Finding Extremal Sets in Less Than Quadratic Time. *Information Processing Letters*, 48(1):29–34, 1993.

[174] Uri Zwick. All Pairs Shortest Paths using Bridging Sets and Rectangular Matrix Multiplication. *Journal of the ACM*, 49(3):289–317, 2002.

Index

Italic page numbers indicate definitions, while bold page numbers indicate whole chapters or sections.