Alcom-FT Technical Report Series
ALCOMFT-TR-01-157

Layout Problems

Tesi doctoral presentada al
Departament de Llenguatges i Sistemes Informàtics
de la Universitat Politècnica de Catalunya

per optar al grau de
Doctor Enginyer en Informàtica

per
Jordi Petit i Silvestre

sota la direcció del Doctor
Josep Díaz Cort

Barcelona, 25 de maig de 2001

This thesis was presented at the Departament de Llenguatges i Sistemes Informàtics of the Universitat Politècnica de Catalunya on May 25th, 2001.

The disseration committe consisted of:

- *Jordi Cortadella Fortuny* (Chair) — Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya.
- *María José Serna Iglésias* (Secretary) — Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya.
- *Burkhard Monien* — Department of Mathematics and Computer Science, University of Paderborn.
- *Philippe Flajolet* — INRIA Rocquencourt.
- *Oriol Serra Albó* — Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya.

Dedico aquest treball a la mare i a la Mireia.

Resum

Aquesta tesi doctoral presenta aspectes algorísmics provinents de l'estudi de problemes d'arranjaments lineals de grafs. En particular, s'hi tracten els problemes següents: Minimum Linear Arrangement, Bandwidth, Cutwidth, Vertex Separation, Sum Cut, Modified Cut, Edge Bisection i Vertex Bisection. Aquests problemes representen una important classe de problemes computacionalment difícils amb diferents aplicacions en diverses disciplines.

En primer lloc, s'ofereix un recull dels coneixements disponibles sobre problemes d'arranjaments lineals. S'hi consideren aplicacions, resultats de complexitat, fites inferiors, algorismes d'aproximació i heurístiques.

Les aportacions comencen amb un estudi experimental pel problema Minimum Linear Arrangement. Es presenten diferents mètodes per obtenir fites inferiors i diferents heurístiques, que inclouen algorismes voraços, seqüenciació espectral i cerca local. Aquests mètodes s'avaluen sobre un joc de proves amb grafs aleatoris i grafs provinents d'aplicacions reals. Els resultats obtinguts s'utilitzen per dissenyar una nova heurística paral·lelitzable per obtenir millors solucions més ràpidament. Aquests resultats experimentals motiven certs aspectes que són desenvolupats posteriorment.

A continuació, es tracten diversos problemes d'arranjaments lineals sobre grafs $\mathcal{G}_{n,p}$. S'hi demostra que, amb probabilitat aclaparant, determinats problemes d'arranjaments lineals es poden aproximar amb un factor constant. De fet, els resultats mostren que el cost de qualsevol solució factible es troba a un factor constant del cost òptim.

Després, s'entra en l'estudi dels problemes d'arranjaments lineals per a grafs amb certa estructura geomètrica. Es mostra que Bandwidth, Cutwidth i Vertex Separation continuen essent **NP**-complets fins i tot quan es restringeixen a grafs graella. També s'aporten solucions per a Vertex Separation, Sum Cut, Bandwidth i Vertex Bisection sobre graelles quadrades, i es donen fites superiors per a grafs graella generals.

Posteriorment, es consideren grafs graella aleatoris i grafs geomètrics aleatoris. Es presenten teoremes de convergència que caracteritzen els costos dels problemes en el règim subcrític. També es presenten algorismes d'aproximació que, per a alguns problemes, són asimptòticament òptims en el règim supercrític. Aquests resultats es troben fortament relacionats amb el teorema de BHH i l'algorisme de dissecció de Karp per al TSP. Els resultats obtinguts són emprats per comparar experimentalment diferents heurístiques usades habitualment pel problema Edge Bisection.

Després, tot utilitzant el marc desenvolupat per tractar problemes d'arranjaments lineals i grafs geomètrics, s'analitzen certes propietats de les xarxes geomètriques aleatòries: hamiltonicitat, emulació i arranjaments lineals.

Per acabar, es presenta l'anàlisi d'alguns problemes de comunicació en arbres per a algunes classes de grafs aleatoris. S'obtenen algorimes d'aproximació.

Abstract

This doctoral dissertation deals with algorithmic issues arising in the study of graph layout problems. Specifically, the following layout problems are considered: Minimum Linear Arrangement, Bandwidth, Cutwidth, Vertex Separation, Sum Cut, Modified Cut, Edge Bisection and Vertex Bisection. These problems represent an important class of hard problems with different applications in various disciplines.

We start with a survey trying to give a complete view of the current state of the art with respect to layout problems. We consider applications, complexity results, lower bounds, approximation algorithms and heuristics.

The first contribution is an experimental study on the Minimum Linear Arrangement problem. Several lower bounding methods and heuristics are presented, implemented and analyzed on a test suite of graphs. The results of the resulting benchmarking are used in order to design a new parallel heuristic to quickly discover better solutions. Latter, we develop some topics motivated by these experimental results.

The study of layout problems on $\mathcal{G}_{n,p}$ graphs is then considered. We show that, with overwhelming probability, several well known layout problems are approximable within a constant on these graphs. In fact, our results establish that the cost of any algorithm is within a constant factor of the optimal cost.

Afterwards, we undertake an study of layout problems on graphs with a geometrical structure. We show that Bandwidth, Cutwidth and Vertex Separation remain **NP**-complete even when restricted to grid graphs. We also give solutions for Vertex Separation, Sum Cut, Bandwidth and Vertex Bisection on square grids, and prove tight upper bounds for several layout problems on general grid graphs.

We then consider random grid graphs and random geometric graphs. We present convergence theorems that characterize the behavior of layout problems on subcritical random grid graphs and subcritical random geometric graphs. We present approximation algorithms that, for some layout problems, are asymptotically optimal for supercritical random geometric graphs. These results are strongly related to the BHH theorem and Karp's dissection algorithm for the TSP. We use these theoretical results to draw an experimental analysis to compare some popular heuristics for the Edge Bisection problem.

We then apply the framework developed to treat layout problems and geometric graphs to analyze several properties of faulty random geometric networks. The considered properties are Hamiltonicity, emulation and layout costs.

Finally, we present an analysis for some communication tree problems on several classes of random graphs. Approximation algorithms are obtained.

Acknowledgements

Many people have helped me while conducting this research. I would like to thank, particularly, Josep Díaz and María Serna, with whom I have worked constantly within this project. Since the first moment, they have guided me in this research, and, continuously, they had their office's doors open to share discussions, answer my questions, and ask theirs.

During the development of this thesis, I also had the privilege to work with Luca Trevisan, Paul Spirakis, Greg Sorkin, Mathew Penrose, Carme Àlvarez and Rafel Cases. Some of them have became my coauthors. I would like to thank them for being so patient and kind.

I am grateful to many people who have drawn my attention to certain articles, have sent me their programs and have helped me executing them, or have sent me graphs instances. Facing the risk of forgetting some of them, let me cite, in no particular order, Thomas Römke, Ralf Diekmann, Robert Preis, Marcus Peinado, Bruce Hendrickson, Georg Kliewer, Georg Skorobohatyj and Sergei Bezrukov. Likewise, I would like to express my gratitude to my professors José Luis Balcázar, Joaquim Gabarró, Ricard Gavaldà and Conrado Martínez who helped me in many ways. I would also thank the help and fun I have got from my good companions Gabriel de Dietrich, Josep Llorenç Cruz, "xoku" Fatos Xhafa, Sandra Moral, Xavier Molinero, Jordi Marco and Maria Josep Blesa.

Finally, I am also indebted with Burkhard Monien, who has hosted me with his impressive work group in Paderborn, while working in the ALCOM-IT project. I also thank Philippe Flajolet, from whom I have received much encouragement. I am much obliged to them for accepting being part of my thesis committee.

The MP3 file in the accompaining CD-ROM is a fragment of *Einstein on the Beach*, an opera by Philip Glass. Reproduced with kind permission.

This research was partially supported by the ALCOM-IT Project (Algorithms and Complexity in Information Technology, European Union ESPRIT LTR Project 20244) and by the ALCOM-FT Project (Algorithms and Complexity in Future Technologies, European Union Project IST-1999-14186).

Contents

Introduction

Understanding is one of those things that science is about.

D. S. Johnson

Several well-known problems in Computer Science can be formulated as graph layout problems. Loosely speaking, layout problems ask to order the vertices of an input graph, in such a way that a particular cost function is optimized. Layout problems are an important class of hard problems with many different applications in various disciplines. Since their introduction in the sixties, layout problems have been a great source of inspiration for researchers. This doctoral dissertation deals with algorithmic issues arising in the study of layout problems.

The results exposed in this thesis were triggered by a question from my advisor: "Why does this heuristic work?". This is a difficult question, for which this thesis does only provide a very partial answer. Rather, this dissertation deepens in the study of layout problems on several restricted types of instances, most of them given by a probabilistic distribution. But this thesis also includes complexity results, average case results, experimental results and a few exact algorithms, all oriented towards that original question.

This thesis is organized into seven chapters, all of them written as self-contained as possible. In particular, each chapter finishes with its concluding remarks, which include a discussion on the contributions of the chapter, some open questions related to the material presented in the chapter, and the publications of the work presented in the chapter. I have preferred this presentation over the alternative of a final chapter of conclusions. On the other hand, outside of this Introduction, I have tried to write the thesis with an impersonal style; the use of the "we" pronoun should be understood as "the reader and me."

With the exception of Chapter 1, titled *"A survey on layout problems,"* the development of this dissertation roughly follows the chronological order in which I performed this research. The choices of the topics considered in each chapter were basically marked by the evolution of this research. In the following, I would like to present this evolution and how it is reflected in the organization of the thesis.

I started the research presented in this dissertation with an experimental study on the Minimum Linear Arrangement problem. Chapter 2, *"Experiments on the* MinLA *Problem,"* presents and discusses this empirical work, which involves different types of sequential and parallel heuristics as well as different families of sparse graphs. The theoretical study done in the subsequent three chapters is a consequence of some of the outcomes of these experimental observations.

As a way to understand the empirical results obtained for the Minimum Linear Arrangement problem on $\mathcal{G}_{n,p}$ graphs, in Chapter 3, *"Layout Problems and Binomial Random Graphs,"* I present a theoretical study for several layout problems on this type of graphs. Briefly stated, the obtained results show that the ratio between the best and worst feasible solutions is bounded by a small constant. The conclusion being that $\mathcal{G}_{n,p}$ graphs do not offer an informative framework in which to analyze heuristics for layout problems, I looked for alternative distributions of random graphs.

The selected alternative was already present in Chapter 2, where I had observed major differences in the behavior of binomial random graphs and random geometric graphs. So I entered upon the study of random geometric graphs, which also carried the study of random grid graphs. Chapter 5, *"Layout Problems and Random Unit Disk Graphs,"* presents the results obtained within this study. These random graphs have a different behavior with respect to layout problems according to two regimes well distinguished by percolation theory. For the so-called subcritical regime, I present several convergence theorems that precisely characterize the behavior of layout problems on random grid graphs and random geometric graphs. For the so-called supercritical regime of random geometric graphs, I present approximation algorithms that, for some layout problems, are asymptotically optimal. These theoretical results are strongly related with classical results on probabilistic analysis of Euclidean optimization problems, but to the best of my knowledge, they have never been observed on graphs. In order to show that these new analytical results on supercritical random geometric graphs may give more insight to analyze and compare heuristics, Chapter 5 also reports on new smaller but more focussed experiments.

To obtain these results, in Chapter 4, *"Layout Problems and Unit Disk Graphs,"* I present solutions for some open layout problems on square grid graphs, and tight upper bounds to layout problems for grid graphs. I also consider the hardness of some layout problems on geometric instances.

While the development of the results in Chapters 4 and 5 ran in parallel, I think it is more clear to expose first the deterministic results, and afterwards the probabilistic ones.

The topic of Chapter 6, *"Faulty Random Geometric Networks,"* was motivated by the observation that the geometric graphs introduced in Chapter 5 could be used as a model for a wireless network. A natural question to investigate where the the properties of these networks in the presence of random faults. In that chapter, I present algorithms to find Hamiltonian paths in faulty random geometric graphs, algorithms to emulate random geometric graphs on faulty random geometric graphs with a small slowdown, and an analysis of some layout problems on faulty random geometric graphs.

Finally, some recent interesting communication tree problems that generalize layout problems were drawn to my attention. Chapter 7, *"Communication Tree Problems,"* presents the obtained results. That chapter analyses several communication tree parameters for certain proposed algorithms on the different classes of random graphs already introduced in the previous chapters.

The dissertation finishes with an Appendix. Its goal is to introduce some tools from probability theory that are used all through the theoretical developments in this thesis, and to summarize my notation.

I feel that this dissertation exposes thoroughly the properties of several layout problems on some families of graphs, thus contributing to their understanding. Due to my original engineering training, I am particularly glad to have undertaken this study using both analytic and experimental techniques, trying to gain insight from one and another. As stated above, several questions arising in Chapter 2 have not been tackled as, for instance, the theoretical insight of some heuristics on random geometric graphs. I believe this is an exciting open field for future work.

This thesis is distributed with a CD-ROM that contains programs, inputs, and an hypertext electronic version of this dissertation as a Portable Document File (PDF). Since a few pictures are in color, the PDF file will enable the reader to print them on a color printer, or display them on a screen.

1

A Survey on Layout Problems

1.1 Introduction

Graph layout problems are a particular class of combinatorial optimization problems whose goal is to find a linear layout of an input graph in such way that a certain objective function is optimized. A linear layout is a labelling of the vertices of a graph with distinct integers. A large amount of relevant problems in different domains can be formulated as graph layout problems. These include optimization of networks for parallel computer architectures, VLSI circuit design, information retrieval, numerical analysis, computational biology, graph theory, scheduling and archaeology. Moreover, the minimal values of some layout costs are also related to interesting graph theoretic invariants of graphs. Most interesting graph layout problems are **NP**-hard and their decisional versions **NP**-complete, but, for most of their applications, feasible solutions with an almost optimal cost are sufficient. As a consequence, approximation algorithms and effective heuristics are welcome in practice.

Because of their importance, there exist a lot of results related with layout problems. In this chapter we try to give a complete view of the current state of the art with respect to graph layout problems. Our focus is biased to algorithmic issues. There exist other surveys that deal with several aspects of graph layout problems, and this chapter generously intersects with them; see references [24, 46, 48, 60, 164, 183, 186].

The road map for this chapter is as follows: First of all, in Section 1.2, we formally define the layout problems we are interested in. In Section 1.3, we state some basic but useful results for these layout problems. Motivations and applications for the study of layout problems are surveyed in Section 1.4. In Section 1.5, we present complexity results on layout problems, including fixed parameterized results and results for particular classes of graphs. In Section 1.6 we present several proposed techniques to obtain lower bounds. Finally, in Sections 1.7 and 1.8, we survey approximation algorithms and heuristics.

1.2 Definitions

In this section we give the definition of several graph layout problems and associated concepts. They will be used all through the thesis.

We use graph theoretic definitions and notations that, for the most part, conform to the standard ones in computer science. Unless mentioned otherwise, graphs are finite, undirected, and without loops. Given a graph G, its vertex set is denoted by $V(G)$ and its edge set by $E(G)$. The notation uv stands for the undirected edge $\{u, v\}$. The degree of a vertex u in a graph G is denoted as $\deg(u) = \deg_G(u)$ and the maximal degree of G as $\Delta(G)$. The notation $\Gamma(u) = \Gamma_G(u)$ stands for the neighborhood of a vertex u in G, that is, the set $\{v \in V(G) \ : \ uv \in E(G)\}$.

A *linear layout*, or simply a *layout*, of an undirected graph $G = (V, E)$ with $n = |V|$ vertices is a bijective function $\varphi\colon V \to [n]$. The set of all layouts of G is denoted $\Phi(G)$. A layout is also called a *linear arrangement*, a *labeling* or a *numbering* of the vertices of a graph, because each of its n vertices receives a different label in $1, 2, \ldots, n$.

Given a layout φ of a graph $G = (V, E)$ and an integer i, we define the sets $L(i, \varphi, G) = \{u \in V \ : \ \varphi(u) \leqslant i\}$ and $R(i, \varphi, G) = \{u \in V \ : \ \varphi(u) > i\}$. The *edge cut* at position i of φ is defined as

$$\theta(i, \varphi, G) = |\{uv \in E \ : \ u \in L(i, \varphi, G) \wedge v \in R(i, \varphi, G)\}|$$

and the *modified edge cut* at position i of φ as

$$\zeta(i, \varphi, G) = |\{uv \in E \ : \ u \in L(i, \varphi, G) \wedge v \in R(i, \varphi, G) \wedge \varphi(u) \neq i\}|.$$

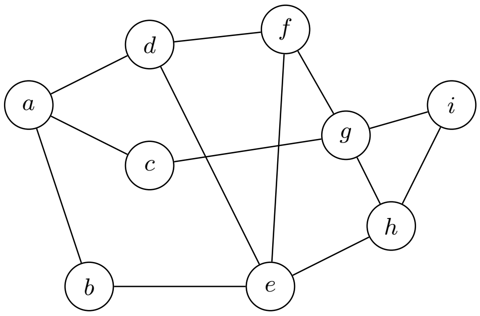
The *vertex cut* or *separation* at position i of φ is defined as

$$\delta(i, \varphi, G) = |\{u \in L(i, \varphi, G) \ : \ \exists v \in R(i, \varphi, G) : uv \in E\}|.$$

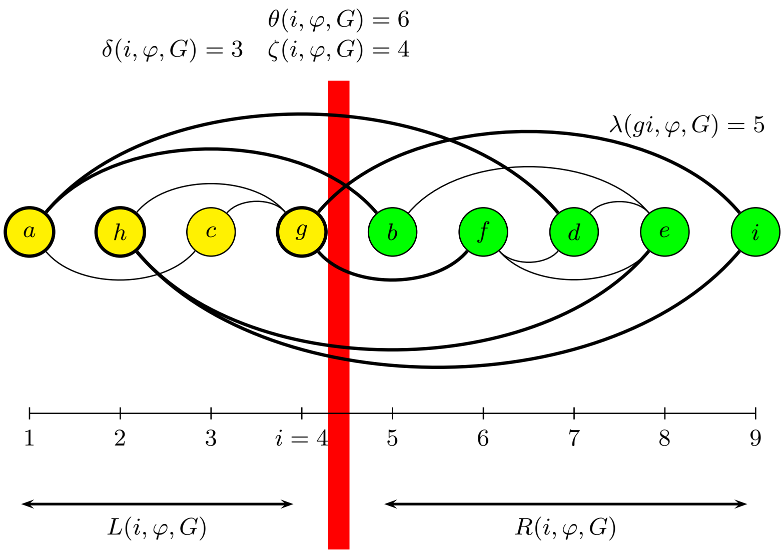
Given a layout φ of G and an edge $uv \in E$, the *length* of uv on φ is

$$\lambda(uv, \varphi, G) = |\varphi(u) - \varphi(v)|.$$

To ease future reference, these measures are summarized in Table 1.1 on page 10.



(a) $G = (V, E)$.



(b) Graphical representation of φ. Dividing the layout at $i = 4$, left vertices are shown in light yellow, right vertices in dark green, cut edges and separator vertices are bold.

Figure 1.1: A graph G and a graphical representation of the layout $\varphi = \{(a, 1), (b, 5), (c, 3), (d, 7), (e, 8), (f, 6), (g, 4), (i, 9), (h, 2)\}$ together with some layout measures.

A common way to represent a layout φ of a graph G is to align its vertices on a line, mapping each vertex u to position $\varphi(u)$, as shown in Figure 1.1. This graphical representation gives an easy understanding of the previously defined measures: By drawing a vertical line just after position i and before position $i+1$, the vertices at the left of the line belong to $L(i,\varphi,G)$ and the vertices at the right of the line belong to $R(i,\varphi,G)$. It is easy to compute the cut $\theta(i,\varphi,G)$ by counting the number of edges that cross the vertical line. The modified cut $\zeta(i,\varphi,G)$ counts all the edges in $\theta(i,\varphi,G)$ except those that have vertex $\varphi^{-1}(i)$ as endpoint. It is also easy to compute the separation $\delta(i,\varphi,G)$ by counting the number of vertices at the left of the vertical line that are joined with some vertex at the right of the vertical line. Finally, the length $\lambda(uv,\varphi,G)$ of an edge uv corresponds to the natural distance between its endpoints.

Given a layout φ of a graph $G=(V,E)$, its *reversed layout* is denoted φ^R and is given by $\varphi^R(u) = |V| - \varphi(u) + 1$ for all $u \in V$.

A *layout cost* is a function F that associates to each layout φ of a graph G an integer $F(\varphi,G)$. Let F be a layout cost; the optimization layout problem associated with F consists in determining some layout $\varphi^* \in \Phi(G)$ of an input graph G such that

$$F(\varphi^*,G) = \min_{\varphi\in\Phi(G)} F(\varphi,G).$$

In the following we will use the following notation for any F and G:

$$\begin{aligned}
\textsc{min}F(G) &= \min_{\varphi\in\Phi(G)} F(\varphi,G),\\
\textsc{max}F(G) &= \max_{\varphi\in\Phi(G)} F(\varphi,G),\\
\textsc{avg}F(G) &= \frac{1}{|V(G)|!}\sum_{\varphi\in\Phi(G)} F(\varphi,G).
\end{aligned}$$

The particular costs we are interested in are listed below, together with the layout problems they give raise to:

- *Bandwidth* (Bandwidth): Given a graph $G=(V,E)$, find a layout $\varphi^* \in \Phi(G)$ such that $\textsc{bw}(\varphi^*,G) = \textsc{minbw}(G)$ where

$$\textsc{bw}(\varphi,G) = \max_{uv\in E} \lambda(uv,\varphi,G).$$

- *Minimum Linear Arrangement* (MinLA): Given a graph $G=(V,E)$, find a layout $\varphi^* \in \Phi(G)$ such that $\textsc{la}(\varphi^*,G) = \textsc{minla}(G)$ where

$$\textsc{la}(\varphi,G) = \sum_{uv\in E} \lambda(uv,\varphi,G).$$

- *Cutwidth* (Cutwidth): Given a graph $G = (V, E)$, find a layout $\varphi^* \in \Phi(G)$ such that $\textsc{cw}(\varphi^*, G) = \textsc{mincw}(G)$ where

$$\textsc{cw}(\varphi, G) = \max_{i \in [|V|]} \theta(i, \varphi, G).$$

- *Modified Cut* (ModCut): Given a graph $G = (V, E)$, find a layout $\varphi^* \in \Phi(G)$ such that $\textsc{mc}(\varphi^*, G) = \textsc{minmc}(G)$ where

$$\textsc{mc}(\varphi, G) = \sum_{i \in [|V|]} \zeta(i, \varphi, G).$$

- *Vertex Separation* (VertSep): Given a graph $G = (V, E)$, find a layout $\varphi^* \in \Phi(G)$ such that $\textsc{vs}(\varphi^*, G) = \textsc{minvs}(G)$ where

$$\textsc{vs}(\varphi, G) = \max_{i \in [|V|]} \delta(i, \varphi, G).$$

- *Sum Cut* (SumCut): Given a graph $G = (V, E)$, find a layout $\varphi^* \in \Phi(G)$ such that $\textsc{sc}(\varphi^*, G) = \textsc{minsc}(G)$ where

$$\textsc{sc}(\varphi, G) = \sum_{i \in [|V|]} \delta(i, \varphi, G).$$

- *Profile* (Profile): Given a graph $G = (V, E)$, find a layout $\varphi^* \in \Phi(G)$ such that $\textsc{pr}(\varphi^*, G) = \textsc{minpr}(G)$ where

$$\textsc{pr}(\varphi, G) = \sum_{u \in V} \left(\varphi(u) - \min_{v \in \Gamma^*(u)} \varphi(v) \right)$$

  and $\Gamma^*(u) = \{u\} \cup \{v \in V \ : \ uv \in E\}$.

- *Edge Bisection* (EdgeBis): Given a graph $G = (V, E)$, find a layout $\varphi^* \in \Phi(G)$ such that $\textsc{eb}(\varphi^*, G) = \textsc{mineb}(G)$ where

$$\textsc{eb}(\varphi, G) = \theta(\left\lfloor \tfrac{1}{2}|V| \right\rfloor, \varphi, G).$$

- *Vertex Bisection* (VertBis): Given a graph $G = (V, E)$, find a layout $\varphi^* \in \Phi(G)$ such that $\textsc{vb}(\varphi^*, G) = \textsc{minvb}(G)$ where

$$\textsc{vb}(\varphi, G) = \theta(\left\lfloor \tfrac{1}{2}|V| \right\rfloor, \varphi, G).$$

The definitions of these layout problems are summarized in Table 1.2. Strictly speaking, the Edge Bisection and Vertex Bisection problems are not layout problems. However, they fit in our framework for layout problems and we shall consider them in the following.

$L(i,\varphi,G)$	$=$	$\{u \in V \ : \ \varphi(u) \leqslant i\}$
$R(i,\varphi,G)$	$=$	$\{u \in V \ : \ \varphi(u) > i\}$
$\theta(i,\varphi,G)$	$=$	$\lvert\{uv \in E \ : \ u \in L(i,\varphi,G) \wedge v \in R(i,\varphi,G)\}\rvert$
$\zeta(i,\varphi,G)$	$=$	$\lvert\{uv \in E \ : \ u \in L(i,\varphi,G) \wedge v \in R(i,\varphi,G) \wedge \varphi(u) \neq i\}\rvert$
$\delta(i,\varphi,G)$	$=$	$\lvert\{u \in L(i,\varphi,G) \ : \ \exists v \in R(i,\varphi,G) : uv \in E\}\rvert$
$\lambda(uv,\varphi,G)$	$=$	$\lvert\varphi(u) - \varphi(v)\rvert, \qquad uv \in E$

Table 1.1: Layout measures for a layout φ of a graph $G = (V,E)$.

Problem	Name	Cost
Bandwidth	BANDWIDTH	$\text{BW}(\varphi, G) = \max_{uv \in E} \lambda(uv, \varphi, G)$
Min. Lin. Arrangement	MINLA	$\text{LA}(\varphi, G) = \begin{cases} \sum_{uv \in E} \lambda(uv, \varphi, G) \\ \sum_{i=1}^{n} \theta(i, \varphi, G) \end{cases}$
Cutwidth	CUTWIDTH	$\text{CW}(\varphi, G) = \max_{i=1}^{n} \theta(i, \varphi, G)$
Modified Cut	MODCUT	$\text{MC}(\varphi, G) = \sum_{i=1}^{n} \zeta(i, \varphi, G)$
Vertex Separation	VERTSEP	$\text{VS}(\varphi, G) = \max_{i=1}^{n} \delta(i, \varphi, G)$
Sum Cut	SUMCUT	$\text{SC}(\varphi, G) = \sum_{i=1}^{n} \delta(i, \varphi, G)$
Profile	PROFILE	$\text{PR}(\varphi, G) = \begin{cases} \varphi(u) - \min_{v \in \Gamma^*(u)} \varphi(v) \\ \text{SC}(\varphi^R, G) \end{cases}$
Edge Bisection	EDGEBIS	$\text{EB}(\varphi, G) = \theta(\lfloor n/2 \rfloor, \varphi, G)$
Vertex Bisection	VERTBIS	$\text{VB}(\varphi, G) = \delta(\lfloor n/2 \rfloor, \varphi, G)$

Table 1.2: Layout problems and costs for a graph $G = (V, E)$ with $|V| = n$.

1.3 Basic observations

At this point it is relevant to point out some basic but important facts:

Observation 1.1. For any graph $G = (V, E)$ and any layout φ of G, the total edge length equals the sum of all edge cuts in the layout:

$$\sum_{uv \in E} \lambda(uv, \varphi, G) = \sum_{i \in [|V|]} \theta(i, \varphi, G).$$

This fact was first noticed by Harper [120]. It follows from the observation that any edge $uv \in E$ with $\varphi(u) < \varphi(v)$ contributes $\varphi(v) - \varphi(u)$ to the left hand side and 1 to each one of the terms $\theta(\varphi(u), \varphi, G), \theta(\varphi(u) + 1, \varphi, G), \ldots, \theta(\varphi(v), \varphi, G)$ in the right hand side.

Observation 1.2. For any graph $G = (V, E)$ and any layout φ of G,

$$\textsc{pr}(\varphi, G) = \textsc{sc}(\varphi^R, G).$$

This is due to the fact that each vertex $u \in V$ contributes one unit $\varphi(u) - \min_{v \in \Gamma^*(u)} \varphi(v)$ times to the sum cut in the reversed layout. As a consequence, Profile and SumCut are equivalent problems!

Observation 1.3. It is important to stress that the graph layout problems we have formulated explicitly ask for an optimal layout rather than the cost of an optimal layout:

> *"Given a graph G, find a layout $\varphi^* \in \Phi(G)$ such that $F(\varphi^*, G) = \textsc{min}F(G)$."*

All the problems can however be restated as decisional problems, where the task is to decide whether or not a graph admits a layout with cost not greater than an integer given as part of the input:

> *"Given a graph G and an integer K, is there some layout $\varphi \in \Phi(G)$ such that $F(\varphi, G) \leqslant K$?"*

In the following, we will not insist too much if we are considering the optimization or the decisional version of some problem, because they will be clearly differentiated by the context.

The following lemma gives simple but useful relations between some layout costs. It is a basic consequence of the previous definitions.

Lemma 1.1. Let G be any graph with n vertices and m edges, and let φ be any layout of G. Then,

$$\begin{aligned}
\textsc{la}(\varphi,G) &\leqslant n\cdot\textsc{cw}(\varphi,G), & \textsc{minla}(G) &\leqslant n\cdot\textsc{mincw}(G),\\
\textsc{la}(\varphi,G) &\leqslant m\cdot\textsc{bw}(\varphi,G), & \textsc{minla}(G) &\leqslant m\cdot\textsc{minbw}(G),\\
\textsc{mc}(\varphi,G) &\leqslant \textsc{la}(\varphi,G), & \textsc{minmc}(G) &\leqslant \textsc{minla}(G),\\
\textsc{sc}(\varphi,G) &\leqslant n\cdot\textsc{vs}(\varphi,G), & \textsc{minsc}(G) &\leqslant n\cdot\textsc{minvs}(G),\\
\textsc{vs}(\varphi,G) &\leqslant \textsc{bw}(\varphi,G), & \textsc{minvs}(G) &\leqslant \textsc{minbw}(G),\\
\textsc{eb}(\varphi,G) &\leqslant \textsc{cw}(\varphi,G), & \textsc{mineb}(G) &\leqslant \textsc{mincw}(G),\\
\textsc{vb}(\varphi,G) &\leqslant \textsc{vs}(\varphi,G), & \textsc{minvb}(G) &\leqslant \textsc{minvs}(G),\\
\textsc{cw}(\varphi,G) &\leqslant \Delta(G)\cdot\textsc{bw}(\varphi,G), & \textsc{mincw}(G) &\leqslant \Delta(G)\cdot\textsc{minbw}(G).
\end{aligned}$$

The next lemma relates some layout costs of a graph in terms of its connected components.

Lemma 1.2. Let G be a graph and $G_1,\dots,G_k$ its connected components. Then,

$$\begin{aligned}
\textsc{minbw}(G) &= \max_{i\in[k]}\textsc{minbw}(G_i), & \textsc{minmc}(G) &= \sum_{i\in[k]}\textsc{minmc}(G_i),\\
\textsc{mincw}(G) &= \max_{i\in[k]}\textsc{mincw}(G_i), & \textsc{minla}(G) &= \sum_{i\in[k]}\textsc{minla}(G_i),\\
\textsc{minvs}(G) &= \max_{i\in[k]}\textsc{minvs}(G_i), & \textsc{minsc}(G) &= \sum_{i\in[k]}\textsc{minsc}(G_i).
\end{aligned}$$

A useful consequence of the previous lemma is that, for any layout cost $F\in\{\textsc{bw},\textsc{cw},\textsc{vs},\textsc{la},\textsc{mc},\textsc{sc}\}$, it is possible to obtain an optimal layout for F of a graph just by computing the optimal layouts of its connected components. Observe, however, that the bisection costs ($\textsc{eb}$ and $\textsc{vb}$) do not share this property.

The following lemma shows that the cost of the layout problems can only decrease after the deletion of an edge or a vertex from a graph. We shall refer to this property as *monotonicity*.

Lemma 1.3. Let H be a subgraph of a graph G. Then, for any layout cost $F\in\{\textsc{bw},\textsc{cw},\textsc{vs},\textsc{la},\textsc{mc},\textsc{sc}\}$, it holds that $\textsc{min}F(H)\leqslant\textsc{min}F(G)$. Moreover, if $V(G)=V(H)$, then $\textsc{mineb}(H)\leqslant\textsc{mineb}(G)$ and $\textsc{minvb}(H)\leqslant\textsc{minvb}(G)$.

1.4 Motivations and applications

In this section we present some information that motivates research on layout problems, as well as some of their applications. We start with a historical overview.

Historical perspective. The Minimum Linear Arrangement problem (MinLA) was stated in 1964 by Harper [119]. Harper's aim was to design error-correcting codes with minimal average absolute errors on certain classes of graphs [119, 120]. More recently, this problem was also considered by Mitchison and Durbin as an over-simplified model of some nervous activity in the cortex [182]. MinLA also has applications in single machine job scheduling [2, 214] and in graph drawing [223]. The Minimum Linear Arrangement problem has received some alternative names, as the *Optimal Linear Ordering*, the *Edge Sum problem* or the *Minimum-*1*-sum*.

The Bandwidth had received much attention during the fifties in order to speed up several computations on sparse matrices , but the introduction of the Bandwidth problem for graphs (Bandwidth) was first stated in 1967 by Harary [117].

The Cutwidth problem (Cutwidth) was first used in the seventies as a theoretical model for the number of channels in an optimal layout of a circuit [3, 177]. In general, the cutwidth of a graph times the order of the graph gives a measure of the area needed to represent the graph in a VLSI layout when vertices are laid out in a row [172]. More recent applications of this problem include network reliability [144], automatic graph drawing [190] and information retrieval [39].

The Sum Cut and Profile problems (SumCut and Profile) were independently defined by Díaz *et al.* [62] and Kuo and Chang [163]. The Sum Cut problem was originally proposed as a simplified version of the δ-operator problem [59]. The Profile problem was proposed as a way to reduce the amount of storage of sparse matrices. Both problems turn to be equivalent to the Interval Graph Completion problem [214], which has applications in archaeology [151] and clone fingerprinting [147].

The Vertex Separation problem (VertSep) was originally motivated by the general problem of finding good separators for graphs [173], and has applications in algorithms for VLSI design [169]. As we shall see, this problem is also equivalent to some other well known problems.

The Edge Bisection problem (EdgeBis) has a wide range of applications, notably in the area of parallel computing and VLSI [26, 73, 129, 167, 226]. The Vertex Bisection problem (VertBis) only seems to have been considered in [157], where it is related with the complexity of sending messages to processors in interconnection networks via vertex-disjoint paths.

In the remaining of this section, we present several further applications related to layout problems.

Layout problems in numerical analysis. In the area of numerical analysis, it is desirable for many engineering applications to reorder the rows and columns of very large sparse symmetric matrices in such a way that their non-zero entries lie as much close as possible to the diagonal. Recall that a matrix is sparse when it has very few non-zero entries. Specifically, the *bandwidth* of a symmetric matrix M is the largest integer b for which there is a non-zero entry at $M[i, i+b]$ and the *profile* of M is $\sum_{i \in [n]}(i - p_i)$ where p_i is the index of the first non-zero entry of row i. Reducing the bandwidth and/or the profile of a matrix leads to a reduction of the amount of space needed for some storage schemes and to an improvement of the performance of several common operations such as Choleski factorization of non-singular systems of equations [219]. The problem of reducing the bandwidth or the profile of a matrix M consists in finding a permutation matrix P (an identity matrix with the same size of M whose columns have been permuted) such that the bandwidth or the profile of $M' = P \cdot M \cdot P^T$ is minimal.

Observe that if we identify the non-zero entries of a symmetric matrix with the edges of a graph and the permutations of rows and columns with flips of the vertex labels, then the bandwidth of the graph equals the bandwidth of the matrix, and the profile of the graph equals the profile of the matrix.

The problem of reducing the bandwidth or the profile of an sparse symmetric matrix has a long history since it originated in the fifties (see e.g. references in [101]). Nowadays, there exist general sparse methods that are more efficient than these "envelope schemes." However, many commercial packages still offer functions to reduce the bandwidth or the profile of sparse matrices as a preprocessing step. Thus, improvements in these methods can be ported to this software without a complete reorganization of their architecture [19]. Efficient algorithms to perform several operations on matrices with small bandwidth can be found, for instance, in [219]. Information retrieval to browse hypertext is a recent area where bandwidth and profile reduction techniques are also used [39, 227].

Layout problems in VLSI. Many layout problems are originally motivated as simplified mathematical models of VLSI layout. Given a set of modules, the VLSI layout problem consists in placing the modules on a board in a non-overlapping manner and wiring together the terminals on the different modules according to a given wiring specification and in such a way that the wires do not interfere among them. There are two stages in VLSI layout: placement and routing. The *placement problem* consists in placing the modules on a board; the *routing problem* consists in wiring together the terminals on different modules that should be connected. A VLSI circuit can be modeled by the means of a graph, where the edges represent the wires and the vertices represent modules. Of course, this graph is an over-simplified model of the circuit, but understanding and solving problems in this simple model can help to obtain better solutions for the real-world model. For a nice survey about the algorithms and techniques used for VLSI layout in practice, see [220]. For a more theoretic point of view, see e.g. [26, 226].

A possible approach to solve the placement problem consists in finding recursively minimal cuts with minimal capacity among all cuts that separates the graph into two components of equal size. The Edge Bisection problem we have presented aims at this approach [227]. Another approach to solve the placement phase is to use the Minimum Linear Arrangement problem in order to minimize the total wire length [3, 121].

It must be noticed that current integrated circuit technology has changed substantially.

Layout problems in graph drawing. Perhaps the most important goal in graph drawing is to produce aestethic representations of graphs. Reducing the number of crossing edges is a way to improve the readability and comprehension of a graph. A bipartite drawing (or 2-layer drawing) is a graph representation where the vertices of a bipartite graph are placed in two parallel lines and the edges are drawn with straight lines between them. The bipartite crossing number of a bipartite graph is the minimal number of edge crossings over all bipartite drawings. Shahrokhi *et al.* [196, 223] prove that for a large class of bipartite graphs, reducing the bipartite crossing number is equivalent to reducing the total edge length, that is, to the Minimum Linear Arrangement problem. Moreover, an approximate solution of MinLA can be used to generate an approximate solution to the Bipartite Crossing Number problem.

Layout problems as embedding problems. Linear arrangements are a particular case of embedding graphs in d-dimensional grids or other graphs. In its most general form, the embedding of a graph G into a host graph H consists in defining an injective function mapping the vertices of G to the vertices of H and associating a path in H for each edge of G. Three parameters are fundamental to assess the quality of an embedding: the dilation, the congestion and the load. The *dilation* of an embedding is the length of the largest associated path. The *congestion* of an embedding is the maximal number of paths that share an edge of H. The *load* of an embedding is the maximum number of vertices of G that are mapped to a vertex of H. Making use of good embeddings is essential in certain contexts, as in parallel computing where embeddings can be used to simulate an algorithm designed for one type of network on a parallel machine with a different type of network; see [186] for a nice survey.

The case in which a graph with n vertices must be embedded into a path graph P_n of n vertices with load 1 is perhaps the simplest nontrivial embedding problem and has intensively been studied in the literature [2, 3, 39, 122, 147, 151, 167, 177, 214, 216, 219]. In this particular case, some layout problems and embedding problems are closely related. Specifically, the bandwidth of a graph corresponds to the minimal dilation and the cutwidth to the minimal congestion.

Layout problems in parallel processing. Many parallel computers in use today are made up of a set of processors with their own private memory that exchange messages by the way of a communication network. In order to get good speedups when using such a system, it is important to distribute the total amount of work between the processors as evenly as possible to minimize idle times. It is also important to reduce the amount of communication between the processors, because communicating through the network is generally much slower than the speed of the processors. Several mapping and load balancing techniques have been developed to address these situations. For certain cases, these techniques lead to graph partitioning problems. See [73, 129, 167].

The Graph Partitioning problem consists in partitioning the vertices of a given graph in k sets of nearly same size in such a way that the number of cutting edges between the k sets is as minimal as possible. The Edge Bisection problem is a particular case of Graph Partitioning where $k = 2$. Recursive bisection is a popular technique to obtain partitions when k is a power of 2. See [227] for an analysis of recursive bisection.

Edge Bisection can be of use when solving partial differential equations and finite elements methods in parallel systems. Simplifying, in these problems a particular iterative computational task has to be carried out in every vertex of a grid (or general graph) and its computation involves data from this vertex and in its neighbors. A way to distribute the total amount of computation between two processors is to assign to each one half of the vertices in the grid. But as border vertices need to communicate in order to get their operands, it is necessary to reduce the cut of the bisection.

Some equivalent problems of Vertex Separation. The Vertex Separation problem is strongly connected with several other important **NP**-complete problems: Gate Matrix Layout, Pathwidth and vertex Search Number. The Gate Matrix Layout is a well studied problem with application in CMOS circuit design [58]. The Pathwidth problem has received an enormous interest in recent years because of its relation with the Robertson–Seymour theory [216], and the vertex Search Number problem is related to strategy and search in graphs. Let us recall the definitions of these problems.

An instance of the *Gate Matrix Layout problem* consists of a collection of nets (rows) $\{N_1, \dots, N_n\}$ and their respective connection to a set of gates (columns) $\{G_1, \dots, G_m\}$. The goal of the problem is to seek a permutation of the columns that minimizes the number of tracks required to lay out the chip, which is equivalent to minimize its area. Figure 1.2 shows an example of gate matrix layout. We denote by $\textsc{mingml}(G)$ the minimal number of tracks needed by a graph G.

A *path-decomposition* of a graph $G = (V, E)$ is a sequence of subsets of vertices $(X_1, \ldots, X_r)$ such that

- $\bigcup_{i=1}^{r} X_i = V$;
- for all edges $e \in E$, some X_i contains both endpoints of e; and
- for all $i \leqslant j \leqslant k$, it is the case that $X_i \cap X_k \subseteq X_j$.

The pathwidth of a path decomposition $(X_1, \ldots, X_n)$ is

$$\textsc{minpw}(G, (X_1, \ldots, X_n)) = \max_{i \in [r]} |X_i| - 1.$$

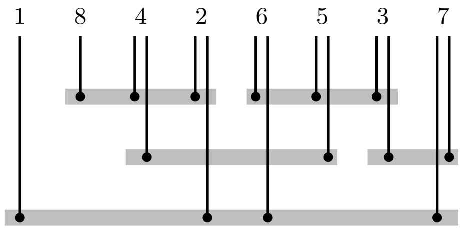
The pathwidth of G, denoted $\textsc{minpw}(G)$, is the minimal pathwidth over all possible path decompositions of G. The Pathwidth problem (Pathwidth) consists in determining a path decomposition with minimal pathwidth. Figure 1.3 shows a graph and one of its path decompositions. See [30, 78] for more information on the Pathwidth and Treewidth problems.

Kirousis and Papadimitriou introduced the Vertex Search Number problem [156]. Briefly stated, this problem asks how many searchers are needed to capture an unlimited number of intruders moving around the edges of a given graph. We denote by $\textsc{sn}(G)$ the vertex search number of graph G.

The equivalence between the Gate Matrix Layout, Search Number Pathwidth, Vertex Separation problems is a consequence of the results by Fellows and Langston [92], Kirousis and Papadimitriou [156] and Kinnersley [154]:

Theorem 1.1. For any graph G, $\textsc{minvs}(G) = \textsc{minpw}(G) = \textsc{minsn}(G) - 1 = \textsc{mingml}(G) + 1$.

Layout problems in computational biology. Physical mapping is a central issue in molecular biology and the human genome project. It consists in determining the relative position of several fragments of DNA, which can be used by biologists to characterize individual genes. As the DNA fragments are obtained "out of order," it is necessary to re-assemble them in order to get a map of the original sequence, based on the information of their pairwise overlaps [148]. The research in [47] has shown that the physical mapping problem is isomorphic to the MinLA problem.



	G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_8
N_1	1	1	0	0	0	1	1	0
N_2	0	0	0	1	1	0	0	0
N_3	0	0	1	0	0	0	1	0
N_4	0	1	0	1	0	0	0	1
N_5	0	0	1	0	1	1	0	0

Figure 1.2: Example of a gate matrix layout with three tracks (figure after [30]).

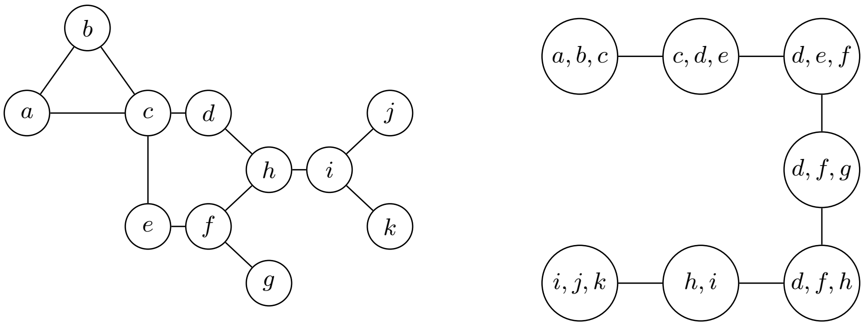


Figure 1.3: Example of a graph and one of its path decompositions with pathwidth 2 (figure after [30]).

1.5 Complexity results

In this section, we survey complexity results related to layout problems. We first consider **NP**-completeness results, then fixed parameter complexity results, and, finally, positive results for particular classes of graphs.

1.5.1 NP-completeness results

It is widely believed that showing that a problem in **NP**-complete is equivalent to prove its computational intractability [98]. The following theorem indicates the difficulty of the considered layout problems on arbitrary graphs.

Theorem 1.2. The decisional versions of the following layout problems are **NP**-complete: BANDWIDTH, MINLA, CUTWIDTH, MODCUT, VERTSEP, SUMCUT, EDGEBIS.

The reductions are due to Papadimitriou for BANDWIDTH [197], to Garey, Johnson and Stockmeyer for MINLA and EDGEBIS [99], to Gavril for CUTWIDTH [100], to Monien for MODCUT [187], to Lengauer for VERTSEP [171], and to Díaz *et al.* for SUMCUT [62].

Many layout problems remain **NP**-complete even for certain restricted classes of graphs. Garey *et al.* showed that BANDWIDTH is **NP**-complete even when restricting its inputs to trees with maximum degree three [97]. Monien improved this result proving that BANDWIDTH remains **NP**-complete for caterpillars with hairs of length at most three, and for caterpillars with at most one unbounded hair attached to the backbone [185]. Recall that a caterpillar is a particular class of tree made of a set of paths, called the hairs, attached by one of their leaves to the vertices of another path, called the backbone. On the other hand, Makedon, Papadimitriou and Sudborough proved that CUTWIDTH is **NP**-complete even for graphs with maximum degree three [176]. Latter, Monien and Sudborough strengthened this result proving that CUTWIDTH and VERTSEP and MODCUT are **NP**-complete even for planar graphs with maximum degree three [187]. Bui *et al.* proved that EDGEBIS is **NP**-complete even when it is restricted to d-regular graphs [43]. Finally, it is also known that MINLA is **NP**-complete when restricted for bipartite graphs [84], and SUMCUT is **NP**-complete when restricted to cobipartite graphs [242]. Notice that it is unknown if MINLA or SUMCUT are **NP**-complete for sparse graphs.

Table 1.3 draws a synthetic overview on these results. Remark that the complexity of VERTBIS is unknown.

Problem	**NP**-complete	Ref.
Bandwidth	in general	[197]
	for trees with maximum degree 3	[97]
	for caterpillars with hair-length $\leqslant 3$	[185]
	for caterpillars with $\leqslant 1$ hair per backbone vertex	[185]
MinLA	in general	[99]
	for bipartite graphs	[84]
Cutwidth	in general	[100]
	for graphs with maximum degree 3	[176]
	for planar graphs with maximum degree 3	[187]
ModCut	for planar graphs with maximum degree 3	[187]
VertSep	in general	[171]
	for planar graphs with maximum degree 3	[187]
	for chordal graphs	[113]
	for bipartite graphs	[105]
SumCut	in general	[62, 163]
	for cobipartite graphs	[242]
EdgeBis	in general	[99]
	for d-regular graphs	[43]

Table 1.3: Review of **NP**-completeness results for decisional graph layout problems.

1.5.2 Fixed parameter results

In many applications, practical problems are often characterized by fixed-parameter instances. The parameter may represent, for example, the number of processing elements to be employed, the maximum allowed wire length in a placement problem, or the number of crossings in a planar drawing of a graph. Parameterized complexity has been primarily motivated by concrete computational problems and can be used to get more insight on layout problems; see e.g. Chapter 1 of [78].

Specifically, a parameterized problem is *fixed-parameter* tractable if there is a constant α and an algorithm that decides whether an instance x with parameter k has a solution in time $f(k)|x|^\alpha$, where f is an arbitrary function and $|x|$ is the size of x. The class of fixed parameter tractable problems is denoted **FPT**. Downey and Fellows introduced a framework from complexity theory and derived the so-called W hierarchy of complexity classes $\mathbf{W}[t]$, whose definition is based on logical circuit families and fixed-parameter reducibility; see [78]. The class $\mathbf{W}[1]$ is the parameterized analog of **NP**: $\mathbf{W}[1]$ hardness is the basic evidence that a parameterized problem is not likely to be fixed-parameter tractable. In the following, we denote by $\Pi(k)$ the fixed parameterized version of a problem Π, where k is the parameter.

Several parameterized complexity results are known for some layout problems. The earlier result of Garey *et al.* proved that it is possible to decide Bandwidth(2) and Cutwidth(2) in linear time [97].

In the case of Cutwidth, Gurari and Sudborough presented a $O(n^k)$ algorithm to decide Cutwidth(k) for any input graph with n vertices and any constant k [112]. This result was improved by Makedon and Sudborough, with a $O(n^{k-1})$ algorithm [177]. Latter, Fellows and Langston obtained a $O(n^2)$ algorithm [90]. The result for Cutwidth(k) has recently been improved by Thilikos, Serna and Bodlaender, who present a linear time algorithm [235]. These authors also present an algorithm to compute the cutwidth of bounded degree graphs with small treewidth in polynomial time [236]. Recall that the treewidth notion is similar to the pathwidth notion, but for a tree decomposition rather than a path decomposition.

In the case of Bandwidth, Saxe presented a $O(n^{k+1})$ algorithm to decide Bandwidth(k) for any constant k [221]. This result was improved by Gurari and Sudborough, who presented a $O(n^k)$ algorithm [112]. This is essentially the best that can be done, as Bodlaender proved that Bandwidth (k) is hard for the class $\mathbf{W}[k]$ for any k [29].

On the other hand, Fellows and Langston have proved that VertSep and ModCut are fixed-parameter tractable [90]. In particular, Bodlaender has proved that VertSep(k) can be decided in linear time [31].

Table 1.4 summarizes these results. When considering their utility, it

Problem	Complexity	Ref.
BANDWIDTH(2)	$O(n)$	[97]
BANDWIDTH(k)	$O(n^{k+1})$	[221]
BANDWIDTH(k)	$O(n^k)$	[112]
BANDWIDTH(k)	$\mathbf{W}[k]$	[29]
CUTWIDTH(2)	$O(n)$	[97]
CUTWIDTH(k)	$O(n^k)$	[112]
CUTWIDTH(k)	$O(n^{k-1})$	[177]
CUTWIDTH(k)	$O(n^2)$	[91]
CUTWIDTH(k)	$O(n)$	[235]
MODCUT(k)	$O(n^2)$	[91]
VERTSEP(k)	$O(n^2)$	[90]
VERTSEP(k)	$O(n)$	[31]

Table 1.4: Fixed parameterized complexity results for layout problems (n denotes the size of the graph and k the parameter).

must be noticed that the multiplicative factor in the big-oh notation is often exponential in k. The fixed parameter complexity of the layout problems not included in the table remains open.

1.5.3 Positive results

Recall that **NP**-completeness results do not rule out the existence of efficient algorithms to get optimal solutions on particular classes of graphs. We review now this type of results for layout problems.

In the case of the Minimum Linear Arrangement problem, Harper computed the optimal value of MINLA for the de Bruijn graph of order four [121] and for hypercubes [119]. The motivation for the former case was to minimize the total edge length of the wires needed to connect a Viterbi decoder and the motivation for the latter case was to design error-correcting codes with minimal errors. In the case of a d-dimensional hypercube Q_d,

$$\text{MINLA}(Q_d) = 2^{d-1}(2^d - 1).$$

According to Chung [48], Goldberg and Klipker where the first to give an $O(n^3)$ algorithm to solve the Minimum Linear Arrangement problem for trees [104]. Adolphson and Hu gave an $O(n \log n)$ algorithm for computing the MINLA of a rooted tree with n vertices [3]. Shiloach improved the result by presenting an algorithm to solve the MINLA on unrooted trees of n vertices in $O(n^{2.2})$ time [225]. This was further improved by Chung, who gave a

$O(n^{\log 3/\log 2})$ algorithm (see [48]). The optimal value for the MinLA problem on a complete binary tree with k levels $T_{2,k}$ has an explicit expression discovered by Chung [48]:

$$\text{MINLA}(T_{2,k}) = 2^k(\tfrac{1}{3}k + \tfrac{5}{18}) + \tfrac{2}{9}(-1)^k - 2, \qquad \forall k \geqslant 2.$$

A recursive expression was also presented by Chung for the case of complete ternary trees. With respect to parallel algorithms, Díaz *et al.* proved that MinLA for trees is in **NC**, as it can be solved in $O(\log^2 n)$ time using a CREW PRAM with $O(n^{3.6})$ processors [61]; see [71, 198] for concepts on parallel complexity.

The MinLA problem on square or rectangular grids has a peculiar history: The problem was first solved in a Russian paper by Muradyan and Piliposjan for the general case of rectangular grids in 1980 [189]. Latter, in 1986, Mitchison and Durbin published the solution only for square grids [182]. In 1981, Niepel *et al.* incorrectly conjecture that the lexicographic layout is optimal for $\text{MINLA}(L_m)$ [192]. In a paper published in 1994, Nakano [191] references again this conjecture. The author of this thesis was personally aware of all that history back in 1997, after observing that the conjecture was false thanks to computational experiments. In 2000, Fishburn, Tetali and Winkler [93] have published another paper on the solution of $\text{MINLA}(L_{m,m'})$, without being aware of the previous results. We delay to Section 4.3 the presentation of the optimal solution of MinLA on square grids.

A d-dimensional c-ary clique is a graph with vertices labeled by integers from 0 to $c^d - 1$ and edges connecting vertices whose c-ary representation differ in one and only one digit. The MinLA problem on this kind of graphs was analyzed by Nakano [191].

More exact MinLA results for several other particular classes of graphs have been identified; see [24] and references therein.

The Cutwidth problem has a very similar trajectory. Harper seems to be the one who first solved it for the case of hypercubes [120]. Chung *et al.* presented a $O(n \log^{d-2} n)$ time algorithm for the cutwidth of trees with n vertices and with maximum degree d [50]. Yannakakis improved that result by giving an algorithm to determine the cutwidth of a tree of n vertices in $O(n \log n)$ time [241]. In the case of a k-level t-ary tree $T_{t,k}$, it holds that

$$\text{MINCW}(T_{t,k}) = \left\lceil \tfrac{1}{2}(k-1)(t-1) \right\rceil, \qquad \forall k \geqslant 3.$$

With respect to parallel algorithms, Díaz *et al.* proved that an optimal layout for the cutwidth of a tree with n vertices and degree Δ can be computed in $O(\Delta \log^2 n)$ time using a CREW PRAM with $O(n^{3.6})$ processors [61]. It is an open problem whether Cutwidth is in **NC** for trees with unbounded degree.

In the case of the Sum Cut or Profile problems, optimal layouts of unrooted trees can be computed in linear time with an algorithm of Díaz *et al.* [62]. These authors also gave a parallel algorithm for computing the optimal Sum Cut of an unrooted tree with n vertices in $O(\log n)$ time using a CREW PRAM with $O(n^2 \log n)$ processors. It is important to remark that these results were published prior to the ones of Kuo and Chang [163], which, moreover, only apply in the sequential case.

In the case of the Vertex Separation problem, Ellis *et al.* gave a linear algorithm to compute the optimal vertex separation of a tree, and a $O(n \log n)$ algorithm to find the optimal layout [80]. Recently, Skodonis has presented a linear time algorithm to find the optimal layout [229]. Bodlaender *et al.* have presented polynomial time algorithms to compute the vertex separation of permutation graphs [32] and cographs [33].

In the case of the Edge Bisection problem, Leighton showed how to minimize the bisection width of Cartesian products of paths of the same length, provided the length is even [167]. Nakano closed the problem for odd lengths [191]. The case of the hypercube bisection seems to have been solved by many people concurrently (see [191]).

As opposed to the rest of the layout problems, the Bandwidth problem is **NP**-complete when restricted to trees. However, in the case of a k-level t-ary tree $T_{t,k}$, it holds that

$$\textsc{minbw}(T_{t,k}) = \left\lceil \frac{t(t^{k-1}-1)}{2(k-1)(t-1)} \right\rceil .$$

The embedding of complete binary trees with optimal bandwidth was presented in [123]. There also exists a $O(n \log n)$ algorithm to determine the bandwidth of caterpillars with hairs of length at most two [13]. Other classes of graphs whose bandwidth can be computed efficiently are interval graphs [106] and chain graphs [160]. Recall that interval graphs are intersection graphs of a set of intervals over the real line, and that chain graphs are bipartite graphs $G = (X, Y, E)$ where there is an ordering $x_1, x_2, \ldots, x_{|X|}$ of X such that $\Gamma(x_1) \subseteq \Gamma(x_2) \subseteq \cdots \subseteq \Gamma(x_{|X|})$.

There appears to be few exact results for the EdgeBis problem, albeit its importance. Bui and Peck have shown that the EdgeBis can be solved in polynomial time for planar graphs such that $\textsc{mineb}(G_n) = O(\log n)$ [44].

Table 1.5 summarizes which classes of graphs are known to be solvable to optimality in polynomial time for graph layout problems.

Problem	Class of graph	Complexity	Ref.
BANDWIDTH	Caterpillars with hair-length $\leqslant 2$	$O(n \log n)$	[13]
	Interval graphs	$O(n+m)$	[106]
	Chain graphs	$O(n \log n)$	[160]
	Complete k-level t-ary tree	$O(n)$	[48]
MINLA	Trees	$O(n^3)$	[104]
	Rooted trees	$O(n \log n)$	[3]
	Trees	$O(n^{2.2})$	[225]
	Trees	$O(n^{\log 3/\log 2})$	[48]
	Rectangular meshes	$O(n)$	[189]
	Square meshes	$O(n)$	[182]
	Hypercubes	$O(n)$	[119]
	de Bruijn graph of order 4	$O(n)$	[121]
	d-dimensional c-ary cliques	$O(n)$	[191]
CUTWIDTH	Trees	$O(n \log^{\Delta-2} n)$	[50]
	Trees	$O(n \log n)$	[241]
	Hypercubes	$O(n)$	[119]
	d-dimensional c-ary cliques	$O(n)$	[191]
VERTSEP	Trees	$O(n \log n)$	[80]
	Trees	$O(n)$	[229]
	Cographs	$O(n)$	[33]
	Permutation graphs	$O(n^2)$	[32]
SUMCUT	Trees	$O(n^{1.722})$	[163]
	Trees	$O(n)$	[62]
EDGEBIS	Hypercubes	$O(n)$	[191]
	d-dimensional c-ary arrays	$O(n)$	[191]
	d-dimensional c-ary cliques	$O(n)$	[191]

Table 1.5: Review of classes of graphs optimally solvable in polynomial time (n denotes the number of vertices in the graph, m its number of edges and Δ its maximal degree).

1.6 Lower bounds

As all the layout problems presented so far are hard to solve for general graphs, good lower bounds become important. Formally, given a layout cost F we say that an algorithm L computes a lower bound of the cost of a graph G for F if $F(\varphi, G) \geqslant L(G)$ for all $\varphi \in \Phi(G)$. This section presents several approaches to get lower bounds for some layout problems under consideration. Other lower bounds for specifics graphs can be found in other surveys, and we shall present new lower bounds for MinLA in Section 2.2.1.

1.6.1 The Path method

The Path method was introduced by Juvan and Mohar in [142] to compute a lower bound for the MinLA and Bandwidth problems. Let $P_n^k = (V_n, E_n^k)$ denote the k-th power graph of the path P_n, where $V_n = [n]$ and $E_n^k = \{ij : 0 < |i - j| \leqslant k\}$. It can be seen that

$$\textsc{minla}(P_n^k) = \tfrac{1}{6}k(k+1)(3n-2k-1).$$

Let $c(n,m)$ be the largest k for which $|E(P_n^k)| \leqslant m$. Then, we have,

$$c(n,m) = n - \tfrac{1}{2}\sqrt{(2n-1)^2 - 8m} - \tfrac{1}{2}.$$

The use of these expressions to get a lower bound to minla and minbw is given by the following theorem:

Theorem 1.3 ([142]). Let G a graph with n nodes and m edges and let $k = \lfloor c(n,m) \rfloor$. Then, $\textsc{minla}(G) \geqslant \textsc{minla}(P_n^k)$ and $\textsc{minbw}(G) \geqslant k$.

1.6.2 Bounds based on spectral properties

Let $G = ([n], E)$ be a graph and let L_G be its Laplacian matrix, defined by

$$L_G[u,v] = \begin{cases} -1 & \text{if } uv \in E, \\ 0 & \text{if } uv \notin E, \\ \deg(u) & \text{if } u = v. \end{cases}$$

By construction, L_G is positive semidefinite. Therefore it has n nonnegative real eigenvalues $0 \leqslant \lambda_1 \leqslant \lambda_2 \leqslant \ldots \leqslant \lambda_n$. The sequence $\lambda_1, \lambda_2, \ldots, \lambda_n$ is known as the *spectrum* of the graph G. It is know that the multiplicity of the value 0 as an eigenvalue of L_G is equal to the number of connected components of G; in particular, if G is connected $0 = \lambda_1 < \lambda_2$ [183].

The following theorem states lower bounds involving the second smallest eigenvalue of the Laplacian of a graph:

Theorem 1.4 ([142, 183]). Let G be a connected graph with n vertices and let λ_2 be the second smallest eigenvalue of the Laplacian matrix of G. Then,

$$\begin{aligned}
\textsc{minla}(G) &\geqslant \lambda_2(n^2-1)/6,\\
\textsc{mincw}(G) &\geqslant \lambda_2 \left\lfloor \tfrac{1}{2}n \right\rfloor \left\lceil \tfrac{1}{2}n \right\rceil /n,\\
\textsc{mineb}(G) &\geqslant \begin{cases} \lambda_2 n/4 & \text{if } n \text{ is even,}\\ \lambda_2(n^2-1)/4n & \text{if } n \text{ is odd.}\end{cases}
\end{aligned}$$

Newer bounds for the EdgeBis related to the level structure of a graph can be found in [25].

The following result, due to Helmberg *et al.*, bounds the bandwidth of a graph using the ratio between the two extremal eigenvalues of its spectrum:

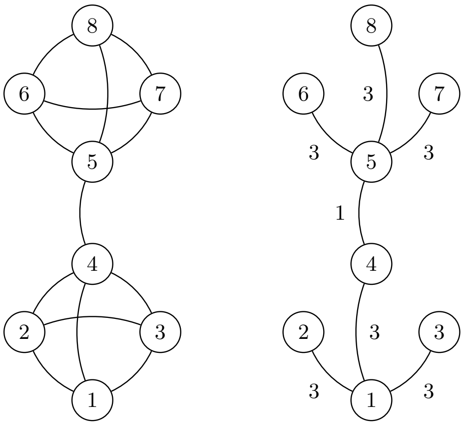
Theorem 1.5 ([124]). Let G be a graph with n vertices and at least one edge. Let λ_2 and λ_n denote the second smallest and the largest eigenvalue of the Laplacian of G, respectively. Let α be the largest integer smaller than $n\lambda_2/\lambda_n$. Then,

$$\textsc{minbw}(G) \geqslant \begin{cases} n-1 & \text{if } \alpha \geqslant n-2,\\ \alpha+1 & \text{if } \alpha \leqslant n-2 \text{ and } n \text{ is even,}\\ \alpha & \text{otherwise.}\end{cases}$$

Another lower bound for MinLA also appears in [124].

1.6.3 Bounds based on fundamental cuts

Let $G = (V, E)$ be a graph with n vertices and let s and t be two distinguished vertices of G, which we call the source and the sink, respectively. The well-known max-flow min-cut theorem states that the maximal flow value from s to t is equal to the minimal edge cut separating s and t [94]. As there exists an efficient algorithm to compute such a minimal cut, and there are $\frac{1}{2}n(n-1)$ possible choices for s and t, it is possible to build a symmetric $n \times n$ matrix f where $f[i, j]$ stores the maximal flow value between two distinct vertices i and j. Gomory and Hu showed that matrix f can simply be represented by a weighted spanning tree of G where each edge represents a *fundamental cut* of G and has weight equal to the corresponding minimal cut [107]. The maximum flow $f[i, j]$ between any pair of vertices i and j can be obtained by finding the unique path between i and j in the weighted spanning tree and finding the minimal weight over all the edges in this path. Figure 1.4 shows a graph, its corresponding Gomory–Hu tree and its corresponding matrix f. The algorithm to construct the Gomory–Hu tree can be found in [107].



f	1	2	3	4	5	6	7	8
1		3	3	3	1	1	1	1
2	3		3	3	1	1	1	1
3	3	3		3	1	1	1	1
4	3	3	3		1	1	1	1
5	1	1	1	1		3	3	3
6	1	1	1	1	3		3	3
7	1	1	1	1	3	3		3
8	1	1	1	1	3	3	3	

Figure 1.4: A graph G, its Gomory–Hu tree and its matrix f of max-flows min-cuts. Applying Theorem 1.6, $\textsc{minla}(G) \geqslant 19$, $\textsc{mincw}(G) \geqslant 3$ and $\textsc{minbw}(G) \geqslant 3$. In fact, $\textsc{minla}(G) = 21$, $\textsc{mincw}(G) = 3$ and $\textsc{minbw}(G) = 3$.

By construction, it is easy to see that the maximal fundamental cut of a graph is a lower bound of the Cutwidth problem. Moreover, Adolphson and Hu proved that the total cut capacity of the $n-1$ fundamental cuts is a lower bound on the cost of the Minimum Linear Arrangement problem [3]. Using Lemma 1.1 we have that $\textsc{minbw}(G) \geqslant \lceil \textsc{minla}(G)/n \rceil$. Therefore, we have the following lower bounds:

Theorem 1.6. Let $G = (V, E)$ be a graph and $T = (V, E', w)$ its weighted Gomory–Hu tree. Then,

$$\begin{aligned}
\textsc{mincw}(G) &\geqslant \max_{e \in E'} w(e),\\
\textsc{minla}(G) &\geqslant \sum_{e \in E'} w(e),\\
\textsc{minbw}(G) &\geqslant \left\lceil \frac{1}{|V|} \sum_{e \in E'} w(e) \right\rceil .
\end{aligned}$$

In [3] it is also proved that if the Gomory–Hu tree is a line graph, then this is the solution to the MinLA problem.

1.7 Approximation algorithms

One of the approaches to deal with intractable problems is to design an approximation algorithm that in polynomial time will give a feasible solution "close" to the optimal one [15, 98]. In this section we precise this idea and present approximability results for layout problems.

Recall that a problem $\Pi = (I, S, f)$ is an optimization problem when:

- I represents the *set of instances* of Π. If $x \in I$ we say that x is an *instance* (or an *input*) of Π. I must be recognizable in polynomial time with respect to $|x|$.
- Given an instance $x \in I$, $S(x)$ denotes the *set of feasible solutions* of x. These solutions must have polynomial length with respect to $|x|$ and $S(x)$ must be recognizable in polynomial time.
- For any instance $x \in I$ and any feasible solution $\sigma \in S(x)$, $f(\sigma, x)$ represents the cost of σ with respect to Π. The function f must be computable in polynomial time and is called the *objective function*.
- The goal of Π is to find a feasible solution that minimizes[1] f: given an input $x \in I$, determine an *optimal solution* $\sigma^* \in S(x)$ such that $f^*(x) = f(\sigma^*, x) \leqslant f(\sigma, x)$ for all $\sigma \in S(x)$.

Given a minimization problem Π, an $r(n)$-approximation algorithm is an algorithm that, for any input x of size n, finds a solution to Π whose cost is at most $r(n)$ times the cost of an optimal solution to the problem's instance. When a problem Π has some $r(n)$-approximation algorithm, it is said to be $r(n)$-*approximable*. When it has an algorithm such that, for all $\epsilon < 1$, A_ϵ returns a feasible solution σ such that the ration between the obtained value and the optimal value is less than $1 + \epsilon$ and runs in polynomial time with respect to $|x|$, A_ϵ is said to be a *polynomial time approximation scheme*. Moreover, when A_ϵ runs in polynomial time with respect to $|x|$ and $1/\epsilon$, A_ϵ is said to be a *fully polynomial time approximation scheme*. A combinatorial optimization problem belongs to the class **APX** if it is ϵ-approximable for some constant $\epsilon > 1$, to the class **PTAS** if it admits an approximation scheme and to the class **FPTAS** if it admits a fully approximation scheme. It is known that **FPTAS** $\subseteq$ **PTAS** $\subseteq$ **APX**, where the inclusions are strict if and only if **P** $\neq$ **NP**. There are also other parallel analogues to sequential approximation classes; see [71].

In the case of the considered graph layout problems, the set of instances I corresponds to the set of all undirected graphs, an instance $x \in I$ corresponds to a particular undirected graph G, the set of feasible solutions $S(G)$ corresponds

[1] Maximization problems are analogous.

to $\Phi(G)$, and the objective function is a layout cost $f \in \{\text{LA}, \text{BW}, \text{SC}, \text{VS}, \text{CW}, \text{MC}, \text{EB}, \text{VB}\}$.

With regard to the BANDWIDTH problem, some particular kinds of graphs have approximations algorithms. For γ-dense graphs, there exists a polynomial time 3-approximation, due to Karpinski *et al.* [149]. Recall that a graph with n vertices is γ-dense if its minimum degree is at least γn. There are also polynomial time $O(\log n)$-approximation algorithms for caterpillars [116], and for a more large class of trees, denoted as GHB-trees, which are characterized as trees such that for any node v, the depth difference of any two non-empty subtrees rooted at v is bounded by a constant [115]. In the case that the input graph is chain-free, there exists a polynomial time $O(\log n)$-approximation algorithm [160]. For general graphs, there exist polylogarithmic approximation algorithms running in polynomial time due to Blum *et al.* [28] and to Feige [85]. On the negative side, Blache *et al.* have shown that it is **NP**-complete to find a $\frac{3}{2}$-approximation in general, and a $\frac{4}{3}$-approximation for trees [27]. As a consequence BANDWIDTH does not belong to **PTAS**. In fact, Unger has proved that it is **NP**-complete to find any k-approximation even for cartepillars, where k is any constant [238]. Therefore, BANDWIDTH does not belong to **APX**. The approximability of BANDWIDTH between a constant and a polylogarithmic factor remains open.

For the EDGEBIS, MINLA and CUTWIDTH problems there exist fully polynomial time approximation schemes for dense graphs [11, 12, 95].

In the case of MINLA, CUTWIDTH and SUMCUT problems, several approximation algorithms have been proposed. The first nontrivial approximation algorithm for MINLA and CUTWIDTH on general graphs had a $O(\log^2 n)$ approximation ratio and was due to Leighton and Rao [168]. Hansen also proposed a $O(\log^2 n)$-approximation algorithm for MINLA [114]. This result was improved for MINLA and SUMCUT by Even *et al.*, who proposed $O(\log n \log\log n)$-approximation algorithms based on spreading metrics [83]. A *spreading metric* on a graph is an assignment of lengths to its edges or its vertices, so that nontrivial subgraphs are spread apart in the associated metric space. The volume of a spreading metric, defined as the sum of the lengths if all edges or vertices, provides a lower bound of solving the problem that guides a divide and conquer strategy. Up to date, the best polynomial time approximation algorithms for MINLA and SUMCUT are $O(\log n)$-approximations for general graphs and $O(\log\log n)$-approximations for planar graphs. Both results are due to Rao and Richa and also use the spreading metric technique [213]. Their technique works for the general case of graphs with weighted edges. The drawback of the algorithms based on spreading metrics is that they require solving a linear program with an exponential number of constraints using the Ellipsoid method.

In the case of the VERTSEP problem, Bodlaender *et al.* have presented

Problem	Approximability	Ref.
BANDWIDTH	3-approximable for dense graphs	[149]
	$O(\log n)$-approximable for caterpillars	[116]
	$O(\log n)$-approximable for GHB-trees	[115]
	$O(\log n)$-approximable for chain free graphs	[160]
	$O(\log^{11/2} n)$-approximable	[85]
	no **PTAS**	[27]
	no **APX**	[238]
VERTSEP	$O(\log^2 n)$-approximable	[159]
	$O(\log n)$-approximable for planar graph	[159]
MINLA	**PTAS** for dense graphs	[11]
	$O(\log^2 n)$-approximable	[114, 168]
	$O(\log n \log\log n)$-approximable	[83]
	$O(\log n)$-approximable	[213]
	$O(\log\log n)$-approximable for planar graphs	[213]
CUTWIDTH	**PTAS** for dense graphs	[11]
	$O(\log^2 n)$-approximable	[168]
SUMCUT	$O(\log n \log\log n)$-approximable	[83]
	$O(\log n)$-approximable	[213]
	$O(\log\log n)$-approximable for planar graphs	[213]
EDGEBIS	**PTAS** for dense graphs	[95]
	$O(\sqrt{n} \log n)$-approximable	[87]

Table 1.6: Review of approximability results for layout problems; n is the size of the input graph.

a polynomial time $O(\log^2 n)$-approximation algorithm for general graphs, and show how to use results from Robertson and Seymour to get a $O(\log n)$-approximation algorithm for planar graphs [159].

The only known approximation algorithm for the EdgeBis problem on general graphs is due to Feige, Krathgamer and Nissim, who obtain a $O(\sqrt{n}\log n)$ approximation ratio [87].

Table 1.6 summarizes these approximability results.

1.8 Heuristics

Resorting to heuristics is an alternative method to obtain solutions for optimization problems. In general, an *heuristic* is a rule of thumb, simplification or guess that reduces or limits the search for solutions in domains that are difficult and poorly understood. In the context of layout problems, an heuristic is a procedure that, given an input graph G, returns a feasible layout G. Unlike approximation algorithms, heuristics do not provide a theoretical guarantee on their cost of the returned layout nor on their running time. In spite of that, heuristics are often used in practice, but the assessment of their effectiveness and efficiency is inherently empirical: “It works well with my inputs.” In this section, we review several heuristics for layout problems and works that analyze them.

Due to their importance in engineering applications, many heuristics have been developed to reduce the bandwidth and/or profile of sparse matrices. Chinn [46] references a paper citing 49 different heuristics! The most well-known heuristics for bandwidth/profile reduction are King’s algorithm, the CutHill–McKee algorithm [55] and the Gibbs–Poole–Stockmeyer algorithms [101]. Most of them belong to a family of heuristics called *level algorithms*. Level algorithm are based on a level structure of the graph, which partitions its vertex set into levels $L_0, \ldots, L_s$ such that the endpoints of every edge in the graph are either in the same level L_i or in two consecutive levels L_i and L_{i+1}. The assessment of the goodness of level algorithms for Bandwidth has been considered by Turner, who has analyzed their behavior on a particular distribution of random graphs with bandwidth no larger than an integer B [237]. Specifically, he considers graphs $\mathcal{G}_{n,p,B}$, resulting from the following experiment: The vertex set is $[n]$, and the edge set is made by connecting, with probability $p \in (0,1)$, any pair of vertices $u, v \in [n]$ such that $|u - v| \leqslant B$. Turner first proved that for all $\epsilon > 0$, almost all $G \in \mathcal{G}_{n,p,B}$ satisfy $1 \leqslant B/\textsc{minbw}(G) \leqslant 1+\epsilon$, and then proved that for any level algorithm A, it holds that $A(G)/\textsc{minbw}(G) \leqslant (1+\epsilon)(3-p)$ for almost all $G \in \mathcal{G}_{n,p,k}$, provided that $B = \omega(\log n)$ and p is fixed. Feige and Krathgamer improve on Turner’s results, allowing smaller edges probabilities p [86].

On the other hand, the areas of VLSI and of parallel computing have given rise to many heuristics for the Edge Bisection problem; see [81] for a nice survey. One of the first proposed heuristics to bisect a graph was the Kernighan–Lin heuristic [152] (not to be confused with the Lin–Kernighan heuristic for the TSP). This heuristic, originally developed to minimize the number of connections in electronic circuits, belongs to a more general family of local search heuristics [1]. Helpful Sets is a more recent local search heuristic developed by Diekmann, Monien and Preis [76]. The basic idea of this heuristic appears in the proof of a result we have already mentioned to get upper bounds on the bisection of 4-regular graphs [129]; see also [72]. Moreover, the Helpful Sets heuristic can be combined with simulated annealing to obtain better results [74]. Spectral bisection is another popular heuristic for EdgeBis based on spectral properties (see Section 1.6.2). Its basic principle is to compute the Fiedler vector of the Laplacian matrix of an input graph $G = (V, E)$, compute the median M of its eigenvalues and return a bisection $(A, V \setminus A)$ where A is made of vertices $v \in V$ satisfying $x_v^{(2)} \leqslant M$. A related algorithm based on eigenvalues is presented and probabilistically analyzed by Boppana [38]. Also, Spielman and Teng show that spectral bisection methods work well on bounded-degree planar graphs and finite element meshes [231]. Other heuristics for EdgeBis include simulated annealing, multilevel algorithms [20], tabu search or genetic algorithms [143].

There exist several papers that compare the effectiveness of several heuristics for EdgeBis from an experimental point of view; see [21, 76, 137, 170]. Unfortunately, these studies do not seem to give an indication on the reason why these heuristics work.

Simpler local search heuristics for the Edge Bisection problem have been theoretically analyzed on particular random graphs: Jerrum and Sorkin analyzed the Metropolis algorithm for the Edge Bisection problem on graphs with a *planted bisection* [134]. Their results involve an algorithm that, starting from a random bisection, iteratively takes a pair of vertices in different sides of the current bisection and interchanges them with probability $1/(1 + \exp(\delta/t))$, where δ is the increase of the cut size and t is a parameter called temperature. Their analysis considers $\mathcal{G}_{4n,p,r}$ random graphs, generated as follows: A $\mathcal{G}_{4n,p,r}$ graph has $4n$ vertices, half of them black and half of them white; edges between vertices of the same color are included independently with probability p, whereas edges between vertices of different colors are included independently with probability $r < p$. They first prove that the planted bisection, defined by the original coloring, is, with high probability, the unique optimal bisection on $\mathcal{G}_{4n,p,r}$ random graphs. Then, they prove that, with overwhelming probability, for a certain choice of t, the Metropolis algorithm can find the planted bisection of $\mathcal{G}_{4n,p,r}$ random graphs in time $O(n^2)$, provided $p - r = \Omega(n^{-1/6})$. In his thesis [141], Juels improves these results to cope with the simpler Hillclimbing algorithm:

interchanges are accepted only if they decrement the size of the bisection. His results state that Hillclimbing can find the planted bisection of $\mathcal{G}_{4n,p,r}$ random graphs in expected time $O(n^2)$ with probability $c > 0$, provided $p - r$ is constant. Repeated executions can boost the probability of success. Juels also reports experimental results that show that Hillclimbing may be much more effective than what his theoretical results indicate on the random graphs he considers. Condon and Karp also present similar results for a generalized version of $\mathcal{G}_{4n,p,r}$ graphs, but with a successive augmentation heuristic rather than local search [52]. Their heuristic is based on a greedy algorithm that repeatedly selects a new pair of vertices and adds one to each side the bisection. They show that their linear time heuristic hits the planted bisection with overwhelming probability, provided $p - r \geqslant n^{-1/2}$.

In Chapter 2 we will present several proposed heuristics for the MinLA problem.

There exist several software libraries that implement many of the above mentioned heuristics. The Party library [212] and the Chaco library [126] are packages that include a variety of different methods to partition or bisect graphs. They will be used in subsequent chapters. Also, Metis is a library of programs for partitioning graphs and computing profile reducing orderings of sparse matrices [150].

1.9 Conclusion

In this chapter we have presented a current view on the main known results about graph layout problems. Surely the reader has observed that plenty of problems remain open: In general, everything that in this chapter is not referred as done might turn into an interesting research problem. In particular, we remark that for most problems, there exist PTAS for dense instances, but not much results exist to approximate sparse graphs. Also, we observe that developing practical approximation algorithms or efective heuristics is a challenging issue. In the latter case, ways to assert their behavior are strongly needed.

2

Experiments on the MinLA Problem

2.1 Introduction

In the field of combinatorial optimization, there exists a big gap between the observations obtained empirically and the theoretical knowledge available. The practical need to obtain good approximated solutions in reasonable time for hard problems has lead to the development of many approximation heuristics. A substantial part of these methods are routinely used, and their users obtain quite satisfactory results. However, the theoretical knowledge that we have on the benefits of these heuristic techniques is still vague.

This chapter presents, from an experimental perspective, a case study to evaluate several techniques to deal with layout problems. We undertake this study with the particular case of the Minimum Linear Arragement problem (MinLA). The goal is to obtain a first evaluation on the behavior of different families of heuristics and of lower bounding techniques that could serve as a guide for subsequent work.

In the light of the preceding survey, starting to tackle a layout problem with an experimental approach could seem a waste of energy. After all, the approximation algorithm of Rao and Richa for the MinLA problem delivers solution whose cost is guaranteed to be not more than a $O(\log n)$ factor far from the optimal [213]. However, to my knowledge this algorithm has never been implemented, maybe because it needs to solve a linear program with an exponential number of constraints using the Ellipsoid method. Moreover, the $O(\log n)$ approximation factor is asymptotic and without a clue in the constant hidden in the big-oh notation nor on its running time. Therefore, it seems reasonable to design heuristic methods that return good solutions in a moderated amount of time, and to investigate their behavior for distinct classes of graphs.

In this context, it is important to remark that experimental study in the algorithmics field is gaining increasing importance. For instance, several workshops and journals have recently appeared to address issues related to experimental algorithmics. However, there is some concern on the significance of the obtained results: For instance, Hooker claims that experimental studies of heuristics are of competitive nature rather than of scientific nature [127, 128]. Trying to address this issue, some authors as Johnson, McGeoch or Moret attempt to establish experimental algorithms as a scientific discipline [136, 178, 188].

This chapter is organized in two well separated parts. The goal of the first part is to present a fair evaluation of different upper and lower bounding techniques for the MinLA problem. This is done running computational experiments on a test suite of graphs. The result is a benchmarking of the evaluated techniques. Section 2.2 presents the lower and upper bounding methods, Section 2.3 presents the test suite, and Section 2.4 presents the experimental evaluation. The second goal of this chapter is to take profit of the results obtained in the first part in order to design a new heuristic for the MinLA problem. The challenge is to obtain faster better solutions. The result is a new parallelizable hybrid heuristic, which we also evaluate on our test suite. The chapter is concluded with a precise summary of our observations. Some of these observations lead to the development of Chapters 3, 4 and 5.

2.2 Lower and upper bounding methods

In this section, we first present new methods to find lower bounds for the MinLA problem that complement some techniques already presented in Chapter 1. Afterwards, we present several methods to get feasible solutions that provide upper bounds for the MinLA problem. These methods are heuristics based on general techniques that include Successive Augmentation, Local Search, and Spectral Sequencing. Finally, we describe the implementation of these upper and lower bounding techniques in the so-called `llsh` toolkit.

2.2.1 Lower bounds

Previously known methods have already been described in Section 1.6, as part of the review on layout problems. These include the Path method, the Gomory–Hu tree method and the Juvan–Mohar method. In the following, we present some new methods to get lower bounds.

The Edges method. The Edges method is a way to compute lower bounds for the MinLA problem that improves the Path method described in Section 1.6. Due to the floor operation taken in Theorem 1.3, the Path method ignores the length of some edges in the graph. A way to take these edges into account is to make use of the following observation: Consider any layout φ. Notice that no more than $n-1$ edges can have cost 1 in φ. Moreover, no more than $n-2$ edges can have cost 2. In general, no more than $n-c$ edges can have cost c in any layout φ. This observation gives us a simple algorithm to compute a lower bound for the MinLA problem: while uncounted edges remain, count their minimal contribution:

```
function EdgesMethod(G) : integer is
    n := |V(G)|; m := |E(G)|; i := 1; f := 0; lb := 0
    while f + n − i ⩽ m do
        f := f + n − i
        lb := lb + i(n − i)
        i := i + 1
    end while
    return lb + i(m − f)
end
```

The Degree method. The Degree method is another way to compute lower bounds for MinLA. Rather than considering edges as in the Edges method, in the Degree method we consider the degree of the vertices: Let φ be any layout of a graph $G=(V,E)$ and define the contribution of a vertex u as $\sum_{uv\in E}|\varphi(u)-\varphi(v)|$. In the best case, a vertex must have two incident edges that contribute 1, two incident edges that contribute 2, two incident edges that contribute 3, etc. (see Figure 2.1). Therefore, a vertex u with degree d cannot have a contribution greater than $\sum_{i=1}^{d/2} 2i = d^2/4 + d/2$ if d is even or $(d+1)/2 + \sum_{i=1}^{(d-1)/2} 2i = (d^2+2d+1)/4$ if d is odd. Therefore, to compute a lower bound to $\text{MINLA}(G)$ we must add the minimal contributions of each vertex and divide the sum by two (as edges have been counted twice). Here is the algorithm:

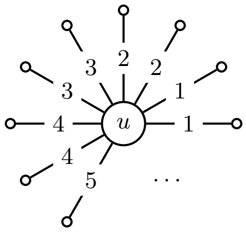


Figure 2.1: Illustration for the Degree method.

```
function DegreeMethod(G) : integer is
    lb := 0
    for all u ∈ V(G) do
        d := deg(u)
        if d mod 2 = 0 then
            lb := lb + d^2/4 + d/2
        else
            lb := lb + (d^2 + 2d + 1)/4
        end if
    end for
    return lb/2
end
```

The Mesh method. Let L_m be a $m \times m$ square grid. In Section 4.3, we will see that

$$\text{MINLA}(L_m) = \tfrac{1}{3}(4 - \sqrt{2})m^3 + O(m^2).$$

By Lemma 1.2, the MINLA of a graph is greater than the MINLA of its parts. Therefore, an alternative way to obtain a lower bound for MinLA is to decompose an input graph G in k disjoint square grids $M_1, \dots, M_k$, compute a lower bound to $\text{MINLA}(M_i)$ for all $i \in [k]$, so that $\text{MINLA}(G) \geqslant \sum_{i=1}^{k} \text{MINLA}(M_i)$.

We propose a greedy algorithm to compute a lower bound for MinLA. The algorithm works by iteratively finding maximal square grids in a given graph:

```
function MeshMethod(G) : integer is
    lb := 0;
    Let M be the largest square grid contained in G; let s be its side
    while s ⩾ 2 do
        lb := lb + (4 − √2)s^3
        G := G \ M
        Let M be the largest square grid contained in G; let s be its side
    end while
    return lb + EdgesMethod(G)
end
```

In order to find the largest square grid contained in G, it is necessary to resort to backtracking. This causes this method to be quite inefficient, although a careful implementation can prune much of the search space.

Discussion. In the case of sparse graphs where $|E| = O(|V|)$, the lower bounds obtained by the Path method, the Edges method and the Degree method are linear in $|V|$. This fact shows that these methods might not be very accurate. In contrast, the lower bounds obtained by the Juvan–Mohar method can grow faster than linearly, depending on the expansion properties of the input graphs. On the other hand, it can be expected that the bounds obtained by the Mesh method will only be effective in graphs containing many grids. This may be the case in the kind of graphs arising from applications in finite element methods.

2.2.2 Approximation heuristics

We now present several heuristics to get approximate solutions for the MinLA problem. All the heuristics described in the following are algorithms that, given a graph G, stop and return a layout φ of G. Most of them are randomized algorithms. These heuristics are expected to provide layouts whose costs are close to the optimal, but no performance guarantee is given. In what follows, we assume that the vertices of $G = (V, E)$ are $V = [n]$.

Random and Normal layouts. A simple way to generate approximated solutions consists in returning a random feasible solution. We denote such a solution a *random layout*. A similar idea consists in not permuting at all the input. This is what we call the *normal layout*:

$$\varphi[i] = i, \qquad \forall i \in [n].$$

In general, these methods will yield bad results, but at least their running time will be negligible.

Successive Augmentation heuristics. We present now a family of *Successive Augmentation heuristics*. These types of heuristics have been applied to a great variety of optimization problems, such as the Graph Coloring problem [138] or the Traveling Salesman problem (TSP) [135, 138]. Under this approach, a partial layout is extended, vertex by vertex, until all vertices have been enumerated, at which point the layout is output without any further attempt to improve it. At each step, the best possible free label is assigned to the current vertex.

Our generic heuristic is given by the following algorithm:

```
function Increment(G, φ, i, v_i, x) : integer is
    φ[v_i] := x;  c := 0
    for j := 1 to i do if v_i v_j ∈ E then c := c + |φ[v_i] − φ[v_j]|
    return c
end

function SuccessiveAugmentation(G) : φ is
    n := |V(G)|
    Select an initial ordering of vertices v_1, v_2, ..., v_n
    φ[v_1] := 0;  l := −1;  r := 1
    for i := 2 to n do
        if Increment(G, φ, i, v_i, l) < Increment(G, φ, i, v_i, r) then
            φ[v_i] := l;  l := l − 1      (∗ Put at left ∗)
        else
            φ[v_i] := r;  r := r + 1      (∗ Put at right ∗)
        end if
    end for
    (∗ Re-map φ to [n] ∗)
    for i := 1 to n do φ[i] := φ[i] − l
    return φ
end
```

To start, label 0 is assigned to a first vertex. Then, at each iteration, a new vertex is added to the layout, to its left or to its right, in the way that minimizes the partial cost. The layout will be from $l + 1$ to $r - 1$ and not from 1 to i as usual, but this has no importance, because MinLA works with differences between labels and not with the labels them-selves. In order to decide in which extreme of the current layout the new vertex must be placed, the function *Increment*(G, φ, i, v_i, x) returns the increment of the cost of the partial layout φ restricted to the vertices $v_1, \ldots, v_{i-1}$ when label x is assigned to vertex v_i. Finally, a second loop maps φ to $[n]$.

It remains to select an initial ordering of the vertices. We propose four different strategies:

Normal ordering: The vertices are ordered in the same way as they are labelled in the graph.

Random ordering: The vertices are randomly ordered. This scheme has the disadvantage of ignoring the connectivity and density of the graph.

Random breadth search: Choose an initial vertex v_1 and set $S := \{v_1\}$ and $i := 2$. While $S \neq V$, choose randomly and edge $uv_i \in E$ with $u \in S$ and $v_i \notin S$; add v_i to S setting $S := S \cup \{v_i\}$ and increment i. This initial ordering has the advantage of making use of the connectivity of the graph but lacks locality.

Breadth-first search: To create an initial ordering, perform a breadth-first search from a random vertex of the graph. In this way, the greedy heuristic would take profit of the possible locality and connectivity of the graph.

Depth-first search: An alternative to the previous strategies would be to perform a depth-first search from a random vertex of the graph.

Observe that, with the exception of the normal ordering, all the above variations result in randomized algorithms.

Spectral Sequencing. The spectral sequencing heuristic to find layouts for MinLA is due to Juvan and Mohar [142]. Given a graph G, the algorithm first computes the *Fiedler vector* of G; that is, the eigenvector $x^{(2)}$ corresponding to the second smallest eigenvalue λ_2 of the Laplacian matrix L_G of G. Then, each position of $x^{(2)}$ is ranked. Thus, the spectral sequencing heuristic returns a layout φ satisfying

$$\varphi(u) \leqslant \varphi(v) \quad \text{whenever} \quad x_u^{(2)} \leqslant x_v^{(2)}.$$

The rationale behind this heuristic is that the ordering of the vertices produced by their values in the Fiedler vector has some nice properties. In particular, vertices connected by an edge will tend to be assigned numbers that are close to each other. This property has been used already in other problems such as graph partitioning [14], chromosome mapping and matrix reordering [125].

Local Search heuristics. Local Search has been described by Papadimitriou and Steiglitz as "an area where intuition and empirical tests play a crucial role and where the design of effective Local Search is much an art" [199]. In spite of this, because of its performance and simplicity, Local Search is one of the most used techniques to approximate many optimization problems (see e.g. [1] and references therein). The basic principle of this heuristic is to iteratively improve an initial solution by performing local changes on its combinatorial structure. Usually, the initial solution is generated at random, and changes that improve the solution are accepted, whereas that changes that worse the solution are rejected.

In order to apply Local Search to an optimization problem, the following items should be recognized: the set of feasible solutions ($S = \{\sigma_i\}_i$), a cost function that assigns a numerical value to any feasible solution ($f : S \to \mathbb{R}^+$) and a neighborhood, which is a relation between feasible solutions that are "close" in some sense. The generic algorithm is as follows:

```
function LocalSearch is
    σ := Select initial random feasible solution
    while ¬Termination() do
        σ′ := Select a neighbor of σ
        δ := f(σ) − f(σ′)
        if Acceptable(δ) then σ := σ′
    end while
    return ⟨σ, f(φ)⟩
end
```

For the MinLA problem, the set of all feasible solutions is $\Phi(G)$, and the objective function is $\mathrm{LA}(G, \varphi)$. However many different neighborhoods can be taken into account. In this research we have considered the following neighborhoods:

Flip2: Two layouts are neighbors if one can go from one to the other by flipping the labels of any pair of vertices in the graph.

Flip3: Two layouts are neighbors if one can go from one to the other by rotating the labels of any trio of vertices in the graph.

FlipE: Two layouts are neighbors if one can go from one to the other by flipping the labels of two adjacent vertices in the graph.

Besides the appealing simplicity of these neighborhoods, the reasons to choose them among all the other potential candidates are the easiness to perform movements and the low effort necessary to compute incrementally the cost of the new layout.

Figure 2.2 shows a graphical representation of the landscape defined by the Flip2 neighborhood. For a random graph with 20 vertices, this figure shows the magnitude of the gain of swapping any pair of vertices. It can be remarked that the landscape is very irregular.

Below, we present some concrete methods derived from the general Local Search heuristic. These variations depend on the way the neighborhood is explored to search favorable moves or which is the criterion to accept moves that do not directly improve the solution. These algorithms are usually called *black-box heuristics* [141] because they work only with the objective function and the neighborhood structure, but they do not use problem-dependent strategies.

Hillclimbing. The Hillclimbing heuristic on the Flip2 neighborhood is implemented as follows: A first initial layout is generated at random. Then, proposed moves are generated at random and are accepted when their gain (δ) is positive or zero. Accepting moves with null gain allows the search to go across plateaus. The heuristic terminates after *max* consecutive proposed moves have

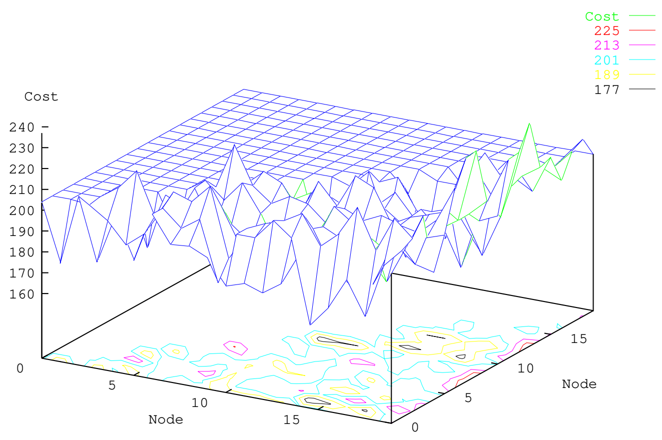


Figure 2.2: Landscape of Flip2 on a random graph.

not strictly decremented the cost of the layout. The algorithm for Hillclimbing is given below.

function *Hillclimbing2*(G, max) **is**
  $\varphi :=$ Generate an initial random layout
  $z := 0$
  while $z < max$ **do**
    $z := z + 1$
    $u :=$ Generate a random integer in $[n]$
    $v :=$ Generate a random integer in $[n]$
    $\delta :=$ *GainWhenFlip2*(G, φ, u, v)
    if $\delta \geqslant 0$ **then**
      Flip2(φ, u, v)
      if $\delta > 0$ **then** $z := 0$
    end if
  end while
  return $\langle \varphi, C_{G,\varphi} \rangle$
end

The function *GainWhenFlip*(G, φ, u, v) returns the gain (possibly negative) in the cost function when flipping the labels of u and v in φ. The procedure *Flip2*(φ, u, v) performs this flipping.

The Hillclimbing heuristic with the Flip3 neighborhood proceeds in the same way, except that three vertices are randomly chosen. For the FlipE neighborhood, a vertex u is first randomly chosen in V; then, another vertex is randomly chosen among all vertices adjacent to u. On the Flip2 and Flip3 neighborhoods, we take $max = n \log n$; on FlipE, $max = \log^2 n$.

Full search. In the full search heuristic, at each step the gain of each possible transition is computed in order to choose the move with maximum gain in the current neighborhood. The algorithm on the Flip2 neighborhood is conceptually as follows:

```
function FullSearch(G, max) is
    φ := Generate an initial random layout
    repeat
        ⟨u, v, δ⟩ := SelectBestMove(G, φ)
        if δ ⩾ 0 then
            Flip2(φ, u, v)
        end if
    until δ <⩽ 0
    return ⟨φ, C_{G,φ}⟩
end
```

According to this implementation, it seems necessary to compute at each steep the gain of the $n(n-1)/2$ possible moves. However, large savings can be done if the graph is sparse, because it is not necessary to compute again all the moves of the vertices that are not neighbors of the previously interchanged vertices. In this way, the number of iteration to perform in *SelectBestMove* is reduced from $O(n^2)$ to $O(\Delta n)$, where Δ is the maximum degree of the input graph.

Metropolis. The problem with the Local Search heuristics seen so far is that once a local optimum is found the heuristic stops, but this local optimum can be far away from the global optimum. In order to enable the heuristic to accept downhill moves, the Metropolis heuristic [181] is parametrized by a "temperature" t and proceeds as follows in the Flip2 neighborhood:

```
function Metropolis(G, r, t) is
    φ := Generate an initial random layout
    for i := 1 to r do
        u := Generate a random integer in [n]
        v := Generate a random integer in [n]
        δ := GainWhenFlip2(G, φ, u, v)
        with probability min(1, e^{-δ/t}) do Flip2(φ, u, v)
    end for
    return ⟨φ, C_{G,φ}⟩
end
```

Notice that improving movements will be automatically accepted, whereas worsening movements are accepted randomly in function of the "height" δ of the movement and the temperature t. With a high temperature the probability of accepting is high; with a small temperature, it is low. In the limit, as $t \to \infty$, Metropolis performs a random walk on the neighborhood structure, and as $t \to 0$ Metropolis proceeds as the Hillclimbing heuristic.

Simulated Annealing. The Simulated Annealing heuristic [155] is closely related to the Metropolis process. Briefly, Simulated Annealing consists in a sequence of runs of Metropolis with a progressive decrement of the temperature. For MinLA, the basic Simulated Annealing heuristic follows the following algorithm:

function *SimulatedAnnealing*(G) **is**
  $\varphi :=$ Generate an initial random layout
  $t :=$ *InitialTemperature()*
  while $\neg$*Frozen()* **do**
    while $\neg$*Equilibrium()* **do**
      $u :=$ Generate a random integer in $[n]$
      $v :=$ Generate a random integer in $[n]$
      $\delta :=$ *GainWhenFlip2*(G, φ, u, v)
      with probability $\min\left(1, e^{-\delta/t}\right)$ **do** *Flip2*(φ, u, v)
    end while
    $t := \alpha \cdot t$ $(*\ 0 < \alpha < 1\ *)$
  end while
  return $\langle \varphi, C_{G,\varphi} \rangle$
end

The main point in Simulated Annealing algorithms is the selection of their parameters: initial temperature, frozing and equilibrium detection, cooling ratio α, ... These depend not only on the optimization problem, but also on the instance of the problem. An excellent treatment of different Simulated Annealing schemes can be found in [135, 137, 138]. Rather than investigating different selections for these parameters, we have used an adaptative technique due to Aarts (see [240]).

2.2.3 The `llsh` toolkit for the MinLA problem

To enable the evaluation of the techniques described in the two previous subsections, we have implemented them. The result is the "`llsh` toolkit," a library of methods for the Minimum Linear Arrangement problem, which we briefly describe. The name `llsh` comes after "Linear Layout Shell."

The architecture of this toolkit is made of two layers: a core layer and an interface layer. The implementation of the different upper and lower bounding methods reside in the core layer, together with utilities to manipulate layouts and graphs. This core is implemented in C++. The toolkit is completed with an interface layer which enables using the core from within a Tcl interpreter, which is a scripting language for controlling and extending applications [195]. This architecture proves to be very convenient: on one side, the fact that the core is implemented in C++ enables both a high level and efficient implementation of the different methods; on the other side, offering an interpreter to the users of the toolkit enables an easy coding of the driver programs that perform the experiments and process the results.

The development of the core of the toolkit has devoted much attention to provide efficient implementations. To achieve this goal without much cost in programming effort, several preexisting libraries have been used:

- In order to quickly compute the Fiedler vector of a large sparse matrix with precision and without taking too many resources, `llsh` uses an implementation of the Lanczos algorithm [202] offered in the Chaco library developed by Hendrickson and Leland [126].

- In order to compute the Gomory–Hu tree of a graph, a public domain implementation by G. Skorobohatyj has been ported to C++ [230].

- The Simulated Annealing heuristic has been implemented using the parSA library developed at the University of Paderborn [158]. This library offers the Aarts adaptative scheduler.

- Several parts of the implementation use classes from the LEDA library developed at MPI [179]. However, arrays and graphs had to be recoded in order to obtain a better efficiency than the one obtained with LEDA. There are two reasons for this: First, LEDA classes are often too general for our purposes, and this generality affects their performance; this is the case of the `graph` class.[1] Second, LEDA methods can not be inlined by the compiler, because their implementation is not available until linking time; this is the case of the `array` and `array2`.

In order to illustrate how the interpreter of the toolkit works, Figure 2.3 shows a sample session.

2.3 Test suite

We consider now the creation of a test suite of graphs to benchmark the previous upper and lower bounding techniques for the MinLA problem.

There exist several test suites for various combinatorial optimization problems. For instance, TSPLIB is a library of sample instances for the TSP from

[1] This problem should be soon overcomed in LEDA by the introduction of a new `sgraph` class, for static graphs.

`> llsh`	Starts the `llsh` toolkit
`% Graph G "randomA1.gra"`	Creates a graph `G` loading `randomA1`
`% puts "[G vertices] [G edges]"`	Writes the number of vertices and edges of `G`
`1000 4974`	result computed by `llsh`
`% Layout L [@ G]`	Creates a layout `L` for the graph `G`
`% UB_spectral [@ G] [@ L]`	Applies the spectral method to `G` and sets the result in `L`
`% puts [L la]`	Writes the cost of the layout `L`
`1202165`	result computed by `llsh`
`% puts [LB_juvan_mohar [@ G]`	Writes the Juvan–Mohar lower bound for `G`
`140634`	result computed by `llsh`
`% exit`	Leaves `llsh`

Figure 2.3: Sample session with `llsh`.

various sources and of various types [215]. Also, QAPLIB provides a unified test bed for the Quadratic Assignment problem [45]. Both libraries are regularly used by practitioners to test new algorithms, and owe their utility to be accessible to the scientific community through the World Wide Web. In contrast, no test suite has already been proposed for MinLA.

Creating a test suite of graphs for the MinLA problem is a difficult task. Ideally, this test suite should contain graphs arising from real life applications, graphs with known optima, graphs that can support general conclusions, and generators of instances. Unfortunately, these kinds of graphs do not abound. Therefore, in order to select the graphs in the test suite, some pragmatical decisions must be taken:

- The first decision we take is to limit the scope of the test suite to sparse graphs. It has already been said that for the case of dense graphs, fully approximation schemes exist, so this family of graphs has not much interest for us.

- The second decision we take is to only include large graphs in the test suite. Here, "large" should stand for graphs that cannot be optimally solved by a brute force algorithm as Branch and Bound in a "reasonable" time. This is a vague definition, but has the advantage of being useful and dynamical. It can also encourage the study of brute force algorithms to show that, after all, these graphs were not so large. Currently, according to this definition, a 5×5 square grid graph is large. However, during this research we have felt the need to go further, and so we have decided that the graphs to be included in the test suite should have more than one thousand vertices.

- The third decision we take is that the test suite must contain several graphs for which their optimal solution is known. At the current time, the only families of graphs that satisfy this condition are the ones shown in Table 1.5. From these graphs, we select a binary tree, a hypercube and a square grid. These are natural families of graphs and represent quite different topologies. In order to have graphs with around one thousand vertices, we take a 10-hypercube (`hc10`), a 33×33 grid graph (`mesh33x33`) and a complete binary tree with 10 levels (`bintree10`).

- The fourth decision we take is that random graphs should be included in the test suite. The reason is that such graphs may be amenable to a probabilistic analysis, and thus general conclusions can be drawn from their study.

  Binomial random graphs $\mathcal{G}_{n,p}$ are obvious candidates to be included in the test suite.[2] Recall that a $\mathcal{G}_{n,p}$ graph is a graph with n vertices where each of the $n(n-1)/2$ edges appears with probability p. We include two $\mathcal{G}_{n,p}$ binomial random graphs in the test suite, taking $n = 1000$ and $p = 0.01$ and $p = 0.05$. The random graphs generated are `randomA1` and `randomA2`.

  Besides $\mathcal{G}_{n,p}$ graphs, it would be interesting to include some other class of random graphs. Random regular graphs do not seem an option, as they share many properties with binomial random graphs, which already have been included. Random graphs proposed to model the Internet would be a nice option, but no model has yet gained enough popularity.

  On the other hand, random geometric graphs seem a good option.[3] Random geometric graphs are graphs whose vertices correspond to points uniformly distributed at random in the unit square, and whose edges join vertices that are closer than some specified bound. Random geometric graphs have also often been used as instances to benchmark heuristics, see for example [23, 137, 165]. So, the test suite includes a random geometric graph, `randomG4`, with 1000 vertices and radius 0.075. Finally, the inclusion of a random geometric graph calls for the inclusion of a binomial random graph with a similar number of vertices and edges. The graph `randomA4` was therefore generated in order to discover differences with `randomG4`.

- The fifth and last decision we take is to include "real life graphs" in the test suite. As we could not obtain real life instances specific for MinLA, we have taken three families of graphs that arise in the same kinds of applications that layout problems: VLSI design (VLSI family), graphs

[2] Binomial random graphs will be treated in more detail in Chapter 3.
[3] Random geometric graphs will be treated in more detail in Chapter 5.

from finite element discretizations (FE family) and graphs from graph-drawing competitions (GD family).

The VLSI graphs `c1y`, `c2y`, `c3y`, `c4y` and `c5y` are graphs derived from circuit layouts. These graphs were provided by M. Peinado, who received them from M. Jünger.

The FE graphs where provided by R. Diekmann, who used them as part of his test suite for partitioning and load balancing algorithms. They arise from applications in computational fluid dynamics (`airfoil1` and `3elt`), earthquake wave propagation (`whitaker3`) and structural mechanics (`crack`).

Finally, the GD graphs have been obtained from several graph-drawing competitions posted in the World Wide Web. In this class, `gd95c` is a mysterious graph, `gd96a` represents a finite automaton used in a natural-language processing system, `gd96b` represents the calls made between a set of telephone numbers, `gd96c` is an artificial nice graph, and `gd96d` represents the structure of a fragment of a web site.

The main characteristics of the graphs in our test suite are shown in Table 2.1. Some of them are shown in Figure 2.4.

2.4 Experimental evaluation

In this section we present, compare and analyze some experimental results aiming to evaluate the performance of the considered methods on the test suite proposed in the previous section. We start presenting the experimental environment in which the measurements took place and the representation of the results that we report later on.

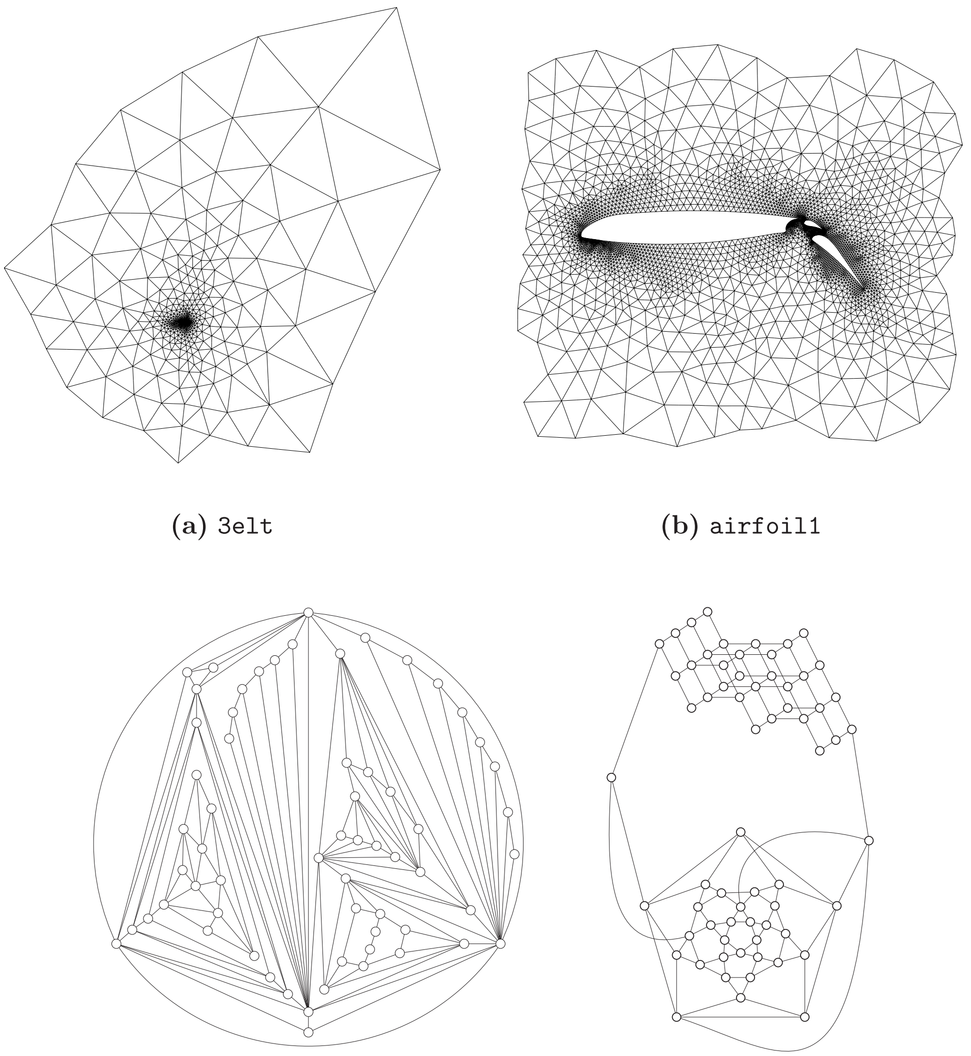
2.4.1 Experimental environment and representation of results

The experiments have been run on a cluster of nine identical personal computers with AMD K6 processors at 450 MHz and 256 Mb of memory with a Linux operating system delivering 897.84 BogoMips each one. These computers have enough main memory to run our programs without delays due to paging. Moreover, all the experiments have been executed in dedicated mode (expect for system daemons) and measure the total elapsed (wall-clock) time. Pre- and post-processing times are included in our measures, with the exception of the time needed to read the input graph.

As described in Section 2.2.3, the core of the programs has been written in the C++ programming language, and the drivers that perform the experiments

Name	Nodes	Edges	Degree	Diam	Family
`randomA1`	1000	4974	1/9.95/21	6	$\mathcal{G}_{n=1000,p=0.01}$
`randomA2`	1000	24738	28/49.47/72	3	$\mathcal{G}_{n=1000,p=0.05}$
`randomA4`	1000	8177	4/16.35/29	4	$\mathcal{G}_{n=1000,p=0.0164}$
`randomG4`	1000	8173	5/16.34/31	23	$\mathcal{G}_{n=1000}(r=0.075)$
`hc10`	1024	5120	10/10/10	10	10-hypercube
`mesh33x33`	1089	2112	2/3.88/4	64	33×33-mesh
`bintree10`	1023	1022	1/1.99/3	18	10-bintree
`3elt`	4720	13722	3/5.81/9	65	
`airfoil1`	4253	12289	3/5.78/10	65	FE
`crack`	10240	30380	3/5.93/9.00	121	
`whitaker3`	9800	28989	3/5.91/8	161	
`c1y`	828	1749	2/4.22/304	10	
`c2y`	980	2102	1/4.29/327	11	
`c3y`	1327	2844	1/4.29/364	13	VLSI
`c4y`	1366	2915	1/4.26/309	14	
`c5y`	1202	2557	1/4.25/323	13	
`gd95c`	62	144	2/4.65/15	11	
`gd96a`	1076	1676	1/3.06/111	20	
`gd96b`	111	193	2/3.47/47	18	GD
`gd96c`	65	125	2/3.84/6	10	
`gd96d`	180	228	1/2.53/27	8	

Table 2.1: Test suite. For each graph, its name, number of vertices, number of edges, degree information (minimum/average/maximum), diameter and family.



(a) `3elt` **(b)** `airfoil1`

(c) `gd95c` **(d)** `gd96c`

Figure 2.4: Some graphs from the test suite.

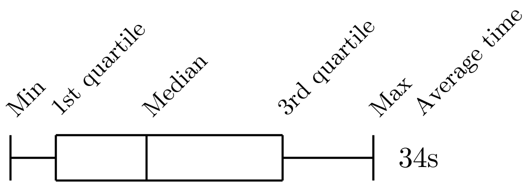


Figure 2.5: Interpretation of the boxplots.

have been written in Tcl. The programs have been compiled using the GCC 2.95.2 compiler with the `-O3` option.

The code has been tested using two different random number generators: the standard `rand()` function in `<stdlib.h>` and the random number generator available in LEDA. We did not notice any anomaly due to the use of either.

All the elemental experiments have been executed independently 200 times, except for Simulated Annealing, for which from 5 to 25 independent runs have been performed, depending on the graph size. Most results will be summarized using boxplots. A *boxplot* is a graphical way of summarizing the distribution of the values of a group of samples. By portraying the values for more than one group next to each other, one can see the major trends in the dataset. The box in a boxplot shows the median value as a line and the first (25th percentile) and third quartile (75th percentile) of the value distribution as the lower and upper parts of the box. The bars shown at the sides of the boxes represent the largest and smallest observed values. The average time elapsed to compute an individual of the sample is displayed at the right of the corresponding boxplot, using the International System of Units (SI). Figure 2.5 illustrates the meaning of the boxplots. Table 2.4 on page 77 resumes the abbreviations shown in other figures and tables.

2.4.2 Comparison of the lower bounding methods

In order to compare the different lower bounding methods presented in Section 2.2.1, we have applied each method to get a lower bound for each graph included in our test suite. Table 2.2 shows the obtained results. In the last column of this table we also include the best known upper bounds, which were found in this research or in the one developed in [17].

The first fact to be observed in Table 2.2 is that the highest obtained lower bounds are far away from the best known upper bounds; the only exception is the case of the `mesh33x33` graph, for which the optimal lower bound is obviously reached by the Mesh method.

The comparison between the different lower bounds makes clear that all

Graph	Edges	Degree	Path	J–M	G–H	Mesh	best
randomA1	14890	16176	9970	140634	9926	†	884261
randomA2	321113	323568	319475	4429294	49404	†	6528780
randomA4	37713	39531	35796	601130	16325	†	1721670
randomG4	37677	39972	35796	14667	16315	†	146996
bintree10	1022	1277	1022	173	1022	1	*3696
hc10	15395	15360	15305	349525	10230	32768	*523776
mesh33x33	3136	3135	1088	1789	4220	*31680	*31680
3elt	27010	27135	14155	8476	27435	44785	363204
airfoil1	24112	24220	12754	5571	24569	40221	277412
crack	60424	64938	30715	25826	60751	95347	1491126
whitaker3	57571	57824	29395	11611	57970	144854	1151064
c1y	2767	14101	2479	13437	3192	2819	62233
c2y	3370	16473	2935	17842	3877	3762	79429
c3y	4555	20874	3976	23417	5320	5548	123548
c4y	4651	16404	4093	21140	5518	5778	115222
c5y	4069	16935	3601	19217	4790	4626	96965
gd95c	250	292	181	36	255	174	506
gd96a	2257	4552	1095	5155	3233	377	96342
gd96b	276	702	110	43	305	113	1422
gd96c	186	191	64	38	241	130	519
gd96d	277	595	179	415	331	113	2409

† The Mesh method could not be applied
* Optimal

Table 2.2: Comparison of the lower bounding methods.

the methods have different behaviors depending on the class of graph they are applied to: 1) for the binomial random graphs, the Juvan–Mohar bound gives much better results than any other method; 2) in the case of FE graphs, the Mesh method always delivers the best lower bounds; 3) on the VLSI graphs, the Juvan-Mohar method and the Degree method clearly outperform the other methods; 4) on the random geometric graph, the Degree method obtains the best lower bound; 5) on the GD graphs, no method clearly outperforms the rest.

As expected, the Edges method always dominates the Path method. Furthermore, we can observe that the Gomory–Hu tree method and the Edges method provide bounds of similar quality, which are always dominated by the Degree method. An interpretation of the poor performance of the Gomory–Hu tree method is that, for all the graphs in our test suite, their Gomory–Hu tree is a star whose central vertex is a vertex with maximal degree.

Regarding the running times, the Edges, Degree and Path methods have a negligible running time; computing the Juvan–Mohar bound takes less than five seconds and computing the Gomory–Hu tree takes about 10 minutes on our bigger graphs. Case apart is the Mesh method, for which we had to limit its exploration to grids up to 23×23 vertices in order to get the results on the FE graphs. In spite of this limit, the Mesh method can last up to 14 hours on the `whitaker3` graph and did not finish on the random graphs, even allowing one week of computation time.

From these results, our advice to obtain the best lower bounds for the MinLA problem on large sparse graphs would be to use the Juvan–Mohar bound and the Degree method, which usually provide the best bounds. Moreover, they are quite fast, and always dominate the Path, Edges and Gomory–Hu methods. On the other hand, applying the Mesh method is only worth in the case of graphs with large sub-meshes, provided one can afford the long running time needed to compute it. Globally, the experiments evidence that none of the presented lower bounding techniques might be very tight in general.

2.4.3 Graphs with known minima

In the case of graphs where the optimal minla value is known, we can use those optimal values to measure the quality of the results obtained by the heuristics we have described. The results of applying each heuristic to these graphs are reflected in Figure 2.14 on Page 78, which shows the performance of the heuristics both in approximation quality and running time. For the sake of clarity, absolute values have been normalized by the optimal values, given in Table 2.2. The absolute non normalized values are given in Table 2.6 on Page 85.

All the graphs share in common that the best average results are found by Simulated Annealing. In the case of the hypercube (`hc10`), Simulated Annealing hits the optimum in a 25% of the tries; for the mesh (`mesh33x33`), Simulated Annealing finds solutions not exceeding a 3% of the optimum; for the binary tree (`bintree10`) the solutions obtained by Simulated Annealing almost double the optimal value.

Depending on the graph, some methods work better than others. In any case, Simulated Annealing always dominates them, albeit it has a longer running time. It must be noticed that on the hypercube, the Normal and Successive Augmentation heuristics with normal ordering reach the optimal values simply because the minla of a hypercube is reached on its normal numbering, and `hc10` follows this numbering.

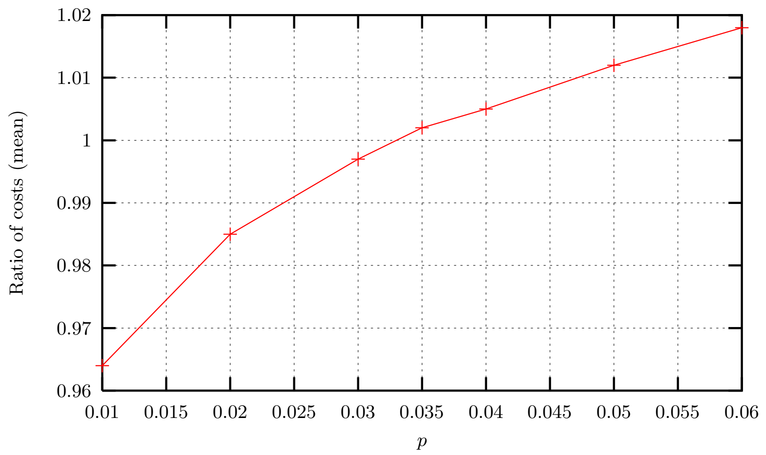


Figure 2.6: Comparison between the Flip2 and Flip3 neighborhoods with the Hillclimbing algorithm on $\mathcal{G}_{n=1000,p}$ graphs. The curve shows the ratio between the average results obtained using Flip2 and Flip3 on 100 independent runs.

2.4.4 Comparing the Flip2 and Flip3 neighborhoods

Notice that when using full search, the Flip3 neighborhood will get stuck at less local minima than the Flip2 neighborhood. However, in the case of the Hillclimbing algorithm things are not so evident. Moreover, in Flip3 it is more difficult to find good moves than in Flip2. For very sparse graphs, preliminary tests seemed to indicate that the Flip2 neighborhood was working better than the Flip3 neighborhood. To validate this impression, we analyzed the performance of Hillclimbing using the two neighborhoods on binomial random graphs $\mathcal{G}_{n,p}$ with $n = 1000$ vertices and $0.01 \leqslant p \leqslant 0.06$. The average result of performing this experiment 100 times is shown in Figure 2.6, which plots the costs of the solutions found in the Flip2 neighborhood divided by the costs of the solutions found in the Flip3 neighborhood.

As the Figure 2.6 reveals, there is a relation between the ratio of the results of the two algorithms and the edge probability p: in the case that $p \leqslant 0.03$, the Flip2 neighborhood yields slightly better results because the ratio is below 1; in the case that $p \geqslant 0.035$, the Flip3 neighborhood turns out to be better because the ratio is above 1.

2.4.5 Binomial random graphs *versus* geometric random graphs

Binomial random graphs $\mathcal{G}_{n,p}$ and geometric random graphs $\mathcal{G}_n(r)$ have very different characteristics. For instance, the graphs `randomA4` and `randomG4` have the same numbers of vertices and almost the same number of edges, but a very different diameter (see Table 2.1). How do these inherent properties of different classes of random graphs affect the behavior of the heuristics we have presented? In order to answer this question, each analyzed heuristic has been applied to the `randomA4` and `randomG4` graphs.

The results are shown in Figure 2.15 on page 79 and Table 2.5 on page 84. Notice that in this case, the values obtained cannot be normalized dividing them by the optimum, because it is unknown. In these cases, we normalize them by the solution obtained by Spectral Sequencing. This decision will be justified latter on.

In the binomial random graph, the best solutions are obtained with the Local Search heuristics. Simulated Annealing obtains the best costs, but the costs obtained with Hillclimbing are very close. The solutions obtained with Spectral Sequencing and the Successive Augmentation heuristics are worse. On the other hand, in the random geometric graph, the best solutions are, in the average, obtained with Spectral Sequencing. However, independent runs of Simulated Annealing, hit a better solution. In both graphs, the running time of Simulated Annealing is much greater than the running time of Spectral Sequencing.

Comparing the differences between the percentage above the best layout ever seen, and assuming that the best layout ever seen is close to the optimum, it can be remarked that approximating geometric random graphs is harder than approximating binomial random graphs. In this sense, it looks like that almost any layout will not be very far from the optimum in $\mathcal{G}_{n,p}$ graphs.

2.4.6 Other graphs

Summarized results to evaluate the performance of the different heuristic on the rest of graphs are given in Figures 2.16, 2.17, 2.18 and 2.19, at the end of the chapter. There are some observations that are worth to discuss.

Surprisingly, for some “real life” graphs the normal heuristic provides quite good results! This can be due to the fact that data locality is often implicit in the vertex numbering or that the layout of these graphs has previously been optimized to reduce their profile in order to improve the efficiency of some storage schemes.

Our observations also establish different trends existing between the approximation of the different families of graphs. From our measures, the Hillclimbing heuristic works well on binomial random graphs (even better than Spectral Sequencing), whereas it performs worse in graphs with an implicit geometric structure, such as random geometric graphs or finite elements computations graphs. In our set of FE graphs, Hillclimbing performs worst than successive augmentation, whereas the contrary happens for our set of VLSI instances.

It can also be observed that Spectral Sequencing is a method that, in general, produces good results in short time. Due to this fact, and because this is not a randomized algorithm, it seems a natural "standard" algorithm to compare against. This is the reason we have decided to normalize our results with Spectral Sequencing when the optimal cost is unknown.

2.4.7 Viewing layouts

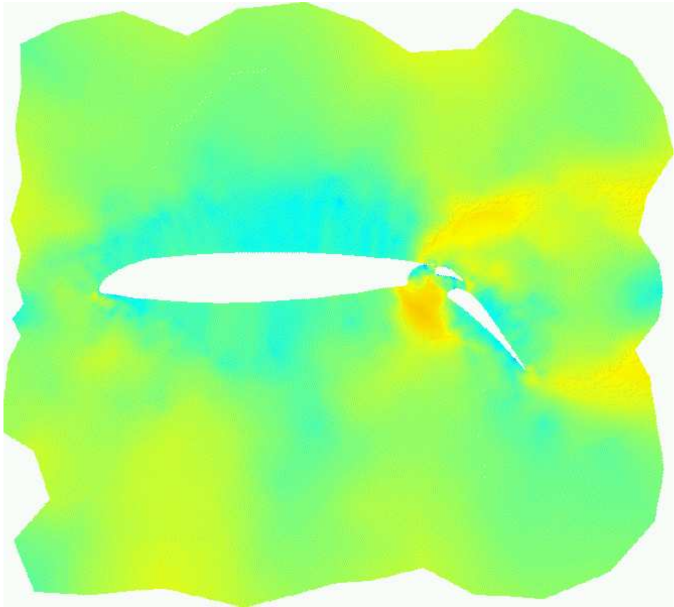
In order to get a feeling of the characteristics of the layouts computed by each heuristic, we have devised a way to "view" them on the FE graphs, for which we have graphical information. For each heuristic, we have conveyed to the edges of this graph a color according to their cost $|\varphi(u) - \varphi(v)|$. Here, "cold colors" represent low costs and "hot colors" represent high costs. For the `airfoil1` graph, Figure 2.7(a) reproduces the Spectral solution, and Figure 2.7(b) reproduces a Simulated Annealing solution. A shadowing process has also been applied to this figure.

It is interesting to see in this way the quality and the geometry of the different heuristics considered. For instance, for the `airfoil1` graph, in Figure 2.7 one can appreciate that the solution obtained by Spectral Sequencing is quite uniform over all the edges, whereas the solution obtained by Simulated Annealing is made of different zones of lower cost separated by grooves with a higher cost.

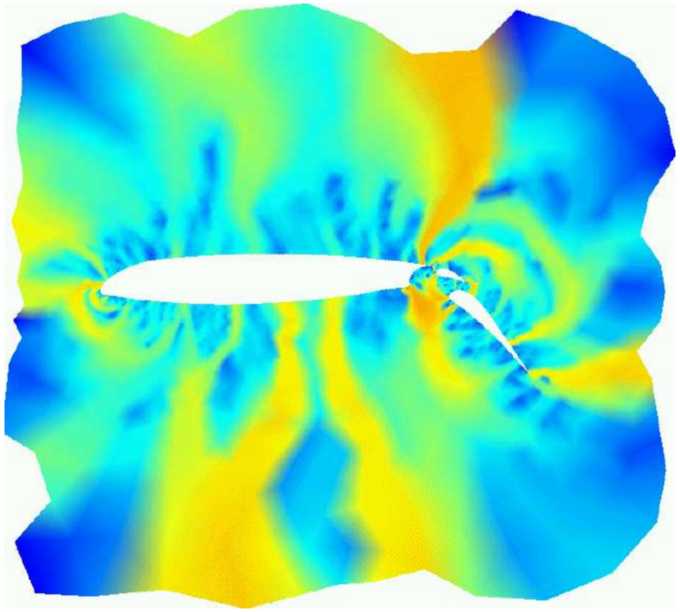
2.5 The SS+SA heuristic

In the previous sections, we have presented a number of heuristics to approximate the MinLA problem, and we have measured their behavior on a test suite of graphs. For the bigger graphs, our results are twofold: On one hand, the best results are obtained with Simulated Annealing, which consumes an inordinate amount of time. On the other hand, results obtained with Spectral Sequencing are not much larger than the ones obtained by Simulated Annealing and are computed much more faster. For instance, in the considered machines, for the `airfoil1` graph, Simulated Annealing lasts almost two hours to generate a layout of cost 338,120, whereas Spectral Sequencing lasts less than two seconds to generate a layout of cost 353,399.

In this section we address the design and evaluation of a new heuristic for the MinLA problem. Our goal is to improve better solutions in a faster time. In this case, we are targeting our larger graphs. To do so, we will present an hybridization between the Simulated Annealing and the Spectral Sequencing heuristics. The resulting heuristic will be called SS+SA, for obvious reasons. Afterwards, we will present two strategies to parallelize the SS+SA heuristic on



(a) Spectral Sequencing



(b) Simulated Annealing

(c) Scale

Figure 2.7: Graphical view of two layouts on the `airfoil1` graph.

a cluster of personal computers. Finally, we will evaluate the performance and efficiency of the sequential and parallel SS+SA heuristics.

2.5.1 The sequential SS+SA heuristic

Notice that Spectral Sequencing is a constructive heuristic that builds a layout from scratch, while Simulated Annealing is a Local Search heuristic that improves a given layout. Therefore, it makes sense to try to hybridize both methods in order to obtain a new heuristic to find better approximations in a faster time. For others problems, it has previously been pointed that spectral methods offer solutions that have a good global quality but combined with a local weakness, and that the global strength of Spectral Sequencing combined with Local Search can lead to heuristics significantly better than either alone [125]. Figure 2.7 suggests that this can also be the case for the MinLA problem.

So, the basic idea of the SS+SA heuristic consists in building first a "globally good layout" by the means of Spectral Sequencing, and then improving it locally through the use of Simulated Annealing. This has several consequences.

First of all, notice that when using Simulated Annealing with random solutions, cooling schedules start with high temperatures that accept, say, one half of the generated movements. Of course, this behavior is not suitable for our SS+SA heuristic, as this would completely destroy the solution generated by Spectral Sequencing. As a consequence, it will be necessary to start Simulated Annealing at a low temperature.

Since the Simulated Annealing process will be started with a quite good solution at a low temperature, it could be expected to have a high number of rejected moves in the Flip2 neighborhood. As a consequence, finding acceptable moves will be a difficult and long task that we wish to speed up. In order to reduce the time of this search, we can make use of the following idea: On a good solution, changes that are worth to be tried must be close in the current layout. It does not make sense to try to flip vertices that are far away! Figure 2.8 supports this heuristic affirmation: for each one of the $n(n-1)/2$ possible moves in the Flip2 neighborhood, we have computed their gain δ and have accepted or rejected the move according to the Simulated Annealing criteria for different temperatures. The figure shows how many moves have been accepted in function of the distance between the vertices in the layout obtained by Spectral Sequencing. Here, the distance between two vertices u and v in a layout φ is defined as $|\varphi(u) - \varphi(v)|$. The distributions clearly follow a half Gauss bell, and show that moves involving close vertices in the layout will have more probability of being accepted.

This gives rise to the use of a new neighborhood relation, or more specifically, a new distribution for the Flip2 neighborhood. Just after obtaining the solution found by Spectral Sequencing, we perform a scanning of the Flip2

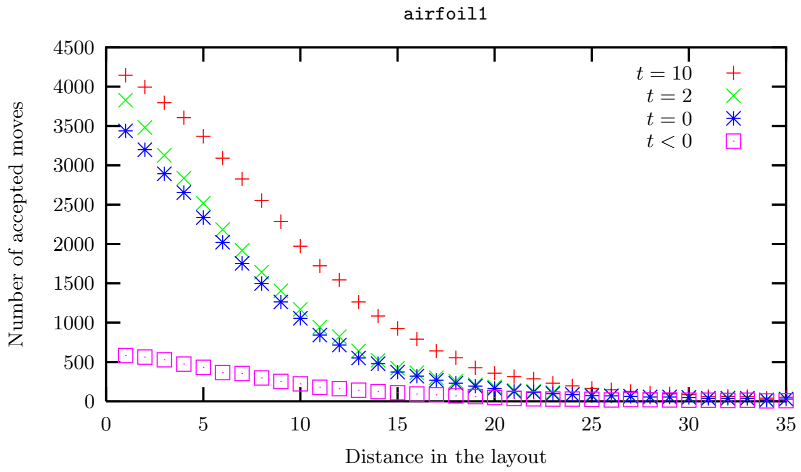


Figure 2.8: Number of accepted moves in function of their distance and the temperature (t) on the solution found by Spectral Sequencing for the `airfoil1` graph. ($t < 0$ means that only strictly descendent moves are accepted.)

neighborhood and compute the mean μ and standard deviation σ of the distances between acceptable moves. Then, during the Simulated Annealing process, moves will be generated producing pairs of vertices whose distance follow a normal distribution $\mathcal{N}(\mu, \sigma)$. We call this new neighborhood FlipN.

Generating moves in the FlipN neighborhood can be done very fast with adequate data structures: storing and updating the inverse layout φ^{-1} of φ suffices. Moreover, computing the gain of one move can also be done very efficiently on sparse graphs without having to recompute the resulting cost from scratch.

Selecting a cooling schedule and tuning it has always been a key problem when using Simulated Annealing [137, 138]. The design of our cooling schedule is very pragmatical, close to the classical ones and in a big extend influenced by our experience and the requirements on the running time and quality solution of SS+SA. Specifically, we use a geometric cooling schedule that starts at an initial temperature t_0 and, at each round, decrements it by a factor of α. For each temperature, a Metropolis round of r moves will be generated. The Simulated Annealing process ends when the temperature drops below t_f.

The pseudo-code of the sequential SS+SA heuristic is as follows:

function *SS+SA*(G) **is**
  Generate an initial layout φ using Spectral Sequencing
  Scan the neighborhood at t_0 to obtain μ and σ
  $t := t_0;\ n := |V(G)|;\ r := \beta n^{3/2}$
  while $t > t_f$ **do**
    repeat r **times**
      Select u and v with $|\varphi(u) - \varphi(v)|$ drawn from $\mathcal{N}(\mu, \sigma)$
      $\delta :=$ *GainWhenFlip2*(G, φ, u, v)
      with probability $\min\left(1, e^{-\delta/t}\right)$ **do**
        Flip2(φ, u, v)
      end with
    end repeat
    $t := \alpha \cdot t$
  end while
  return φ
end

The concrete values of the parameters we have used are $t_0 = 10$ to set up the initial temperature, $t_f = 0.2$ to set the final temperature and $\alpha = 0.95$ to decrement the temperature after each round. Moreover, the number of moves to apply in each Metropolis round has been set to $r = \beta n\sqrt{n}$ with $\beta = 20$.

2.5.2 The parallel SS+SA heuristics

We now present two strategies to parallelize the Metropolis loop of the SS+SA heuristic on a cluster made of P machines. All the terms relative to parallel computing can be found in any standard reference, such as [162].

A problem independent parallel Simulated Annealing is implemented in the Paderborn Simulated Annealing Library [158] that we have used in Section 2.2. That implementation is based on a general framework to parallelize Simulated Annealing on distributed memory machines presented by Diekmann, Lüling and Simon in [75]. The basic idea consists in generating and evaluating moves independently in different processors. Rather than pursuing that idea, in this chapter we tailor parallelization techniques for Simulated Annealing used in shared memory computers (see e.g. [56]). The difference here is that we will have several processors updating concurrently a unique solution.

Before presenting the parallel algorithm, we need some definitions. Given an integer P, a layout φ on a graph $G = (V, E)$ with n vertices, and an increasing sequence of indices $j_0, j_1, \ldots, j_P$ with $j_0 = 0$ and $j_P = n$, let us define a P-partition $(V_1, V_2, \ldots, V_P)$ of V by

$$V_i = \{u \in V \mid j_{i-1} < \varphi(u) \leqslant j_i\}, \qquad \forall i \in [n].$$

Moreover, let $\mathcal{V}_0$ be the P-partition induced by

$$j_i = i \cdot n/P, \qquad \forall i \in [n-1],$$

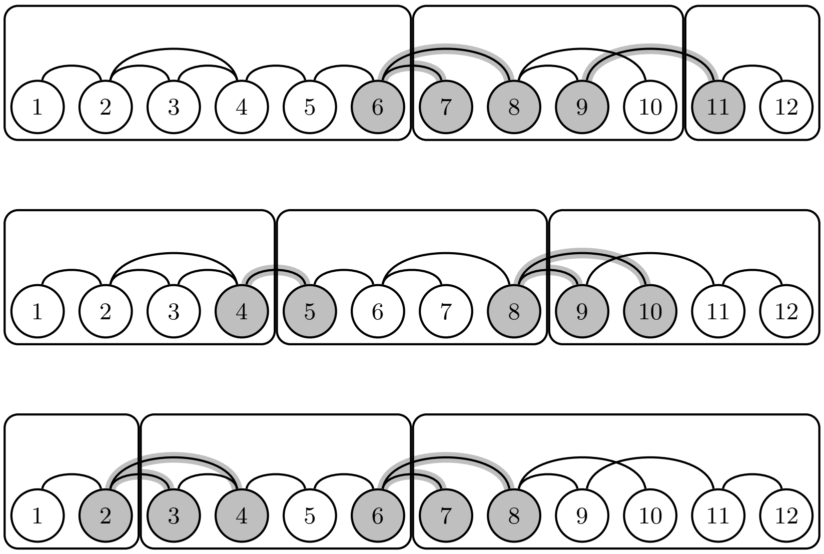


Figure 2.9: Examples of partitions for three processors: $\mathcal{V}_{+1}$ at the top, $\mathcal{V}_0$ at the middle and $\mathcal{V}_{-1}$ at the bottom.

let $\mathcal{V}_{+1}$ be the P-partition induced by

$$j_i = i \cdot n/P + n/2P, \qquad \forall i \in [n-1],$$

and let $\mathcal{V}_{-1}$ be the P-partition induced by

$$j_i = i \cdot n/2 - n/2P, \qquad \forall i \in [n-1].$$

Given a P-partition, edges whose vertices are in different partitions are said to be in the *cut*. Vertices that have an adjacent edge in the cut are said to be in the *frontier*; the remaining vertices are said to be *free*. As an example, Figure 2.9 shows a layout of a graph and its three P-partitions defined as before with the cut edges and the frontier vertices marked. Notice that these three partitions group vertices that are consecutive in the layout φ.

The *exact strategy* for the parallel SS+SA heuristic on P processors starts computing the initial solution by Spectral Sequencing. This step is done sequentially, because it can be done very fast (by Amdahl's Law its parallelization would not be worth). After this step, a first layout φ is obtained. Then, the Flip2 neighborhood is scanned in parallel by the P processors, to obtain the values of μ and σ. At this moment, the Simulated Annealing algorithm begins. The cooling schedule is the same as in the sequential algorithm, but now the Metropolis process is different. The main part of the parallel SS+SA for P processors is as follows:

function *ExactParallel SS+SA*(G) **is**
  Sequentially, generate an initial layout φ using Spectral Sequencing
  In parallel, sample the neighborhood at t_0 to obtain μ and σ
  $t := t_0;\ n := |V(G)|;\ r := \beta n^{3/2}/4P$
  while $t > t_l$ **do**
    repeat$r$ **times**
      Metropolis(0)
      Metropolis$(+1)$
      Metropolis(0)
      Metropolis(-1)
      $t := \alpha t$
    end repeat
  end while
  return φ
end

The *Metropolis* function is the one that contains parallelism. Given a graph $G = (V, E)$, the basic idea here is to compute P-partitions of V using the current layout φ, assigning one partition to each processor. Then, the processors concurrently generate moves in the FlipN neighborhood on their own partitions. If these moves are not forbidden (i.e., none of the vertices that it flips are in the frontier), they are accepted according to the Metropolis criterium; otherwise they are directly rejected. This is necessary in order to ensure that, during all this phase, the information owned by each processor is maintained coherent with respect to the information stored in the other processors. Notice that allowed moves do not need to be communicated to the remaining of processors, since they will never use this information. As a consequence, forbidding moves removes the need of an expensive communication while not affecting the correctness of the algorithm. After performing r iterations, a synchronization phase between all the processors is performed. During this synchronization, a global layout is easily rebuilt through the combination of the layouts in each processor and the P-partition.

The high level algorithm follows this pseudo-code:

function *Metropolis*(x) **is**
  for each processor $i \in [P]$ **do in parallel**
    Get a private copy of layout φ
    From $\mathcal{V}_x$ compute V_i and compute the frontier vertices
    (∗ Perform Metropolis in $\mathcal{V}_x$ ∗)
    repeat r **times**
      Select u, v in V_i with $|\varphi(u) - \varphi(v)|$ drawn from $\mathcal{N}(\mu, \sigma)$
      if u, v are free w.r.t. V_i **then**
        $\delta :=$ *GainWhenFlip2*(G, φ, u, v)
        with probability $\min\left(1, e^{-\delta/t}\right)$ **do**
          Flip2(φ, u, v)

```
                end with
            end if
        end repeat
    end for
    (* Gather data *)
    Rebuild a global layout φ from the ones distributed
        among processsors, according to V_x
end
```

On sparse graphs, partitions induced by a good layout are them-selves expected to have a few cutting edges. As a consequence, only a few moves inside a partition will be forbidden. So, in each call to the Metropolis process there will be many opportunities to optimize the individual partitions. The trick of using three different partitions intercalated in four phases is used in order to not always forbid the same moves. Otherwise we would obtain layouts which would be well arranged inside each partition, but locally bad arranged near the frontiers. With respect to the efficiency of the previous algorithm, it is clear that if r is large enough, the time spent in the syncronization, broadcasting and gathering phases will be low in comparison with the time spent in the computation phases. Thus, large speedups and high processor use can be expected.

It is clear that forbidding moves enables the processors to allways have an up-to-date information. Unfortuntely, these forbidden moves restrict the possibilities of optimizing, as they are directly rejected. If we admit that processors freely move frontierer vertices, we get the *chaotic strategy* for the SS+SA heuristic. In this case, as moves are applied into a concrete partition but cannot cross the partitions, the global state of the system will still be coherent, since it represents a valid layout. However, after moving a fronterier vertex, other processors would compute the gains of their moves with an out-of-date information, thus commiting an error. Intuitively, this error should not be very large, and it would descrease as the temperature is lowered. A way to reduce the error would be to perform more frequent syncronizations in which all the processors perform an all-to-all communication in order to recover the more recent up-to-date layout. In this case, a trade-off between this additional communication time and error precision should then be done.

2.5.3 Experimental evaluation

We now present, compare and analyze some empirical results aiming to evaluate the performance of our new heuristic. In particular, we are interested in the relative performance and efficiency of SS+SA compared to Spectral Sequencing and Simulated Annealing, and in the scalability in time and in quality between the different parallel SS+SA strategies. To do so, we use the larger graphs included in the test suite designed in Section 2.3. In the following, some experimental results are only shown for the particular case of the `airfoil1` graph but, if not otherwise stated, similar behaviours were observed on the other graphs.

Experimental conditions. The sequential SS+SA heuristic has been coded in the C++ programming language and forms part of to the `llsh` toolkit described in Section 2.2.3. The parallel heuristics use C++ augmented by the MPI message passing library [111]. All the programs have been run on a Linux cluster made of 9 personal computers described in Section 2.4.1. It is important to remark that the Linux cluster is interconnected with an inexpensive Fast Ethernet network shared with other machines. As we did in Section 2.4, we always measure the total ellapsed (wallclock) time, starting at the moment that the input graph is read. The programs have been compiled using maximum optimization compiler options and linked against LAM, an implementation of MPI.

Comparing FlipN with Flip2. The following experiments were designed to evaluate the effect of using the FlipN neighorhood instead of the more traditional Flip2 neighorhood, i.e. to favor moves among close vertices in the current layout with respect to try to move arbitray pairs of vertices.

For the particular case of the `airfoil1` graph, Figure 2.10(a) shows a trace of the cost as function of the time when using Flip2 and FlipN. From the curves, the benefits of stretching the neighborhood in order to reduce the runtime and to increase the quality of the solution are evident.

Evaluation of the sequential SS+SA heuristic. In order to compare SS+SA against Spectral Sequencing and Simulated Annealing alone, we have tried to apply SS+SA to the graphs in our test suite with more than 1000 vertices.

The first event that we observed is that the heuristic argument that lead us to select the FlipN neighborhood is not valid for the binomial random graphs and the hypercube in our test-suite. In these graphs, the average distance between accepted moves (μ) was greater than one third the number of vertices. For the remaining graphs, μ was much smaller (arround 20). This shows that the strengths of SS+SA will only apply to clustered and structured graphs, as the ones in the FE and VLSI families.

Table 2.3 compares the solution quality and running time of SS+SA with the ones of Spectral Sequencing and the better ones of Simulated Annealing. For the FE graphs, it can be observed that SS+SA improves the Spectral Sequencing and Simulated Annealing solutions by more than 20%, while reducing the running time to a 25% of Simulated Annealing. For the remaining graphs, SS+SA allways improves the Spectral Sequencing solutions and only for the `c5y`

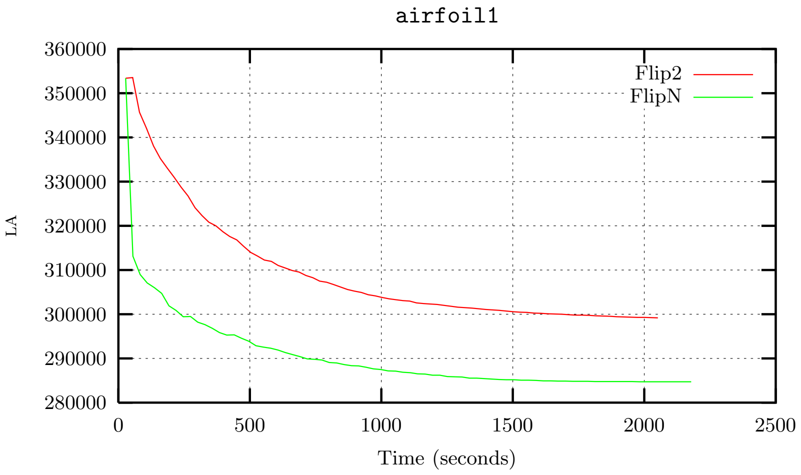


Figure 2.10: SS+SA on `airfoil1` using the Flip2 or FlipN neighborhoods.

	Cost			Time		
Graph	SS	SA	SS+SA	SS	SA	SS+SA
`3elt`	429164	428907	363686	2	9200	2564
`airfoil1`	353399	338120	285597	1	7427	2065
`crack`	1641119	1576500	1496671	4	46178	9310
`whitaker3`	1251709	1214090	1169642	3	41144	8127
`c1y`	104316	66894	63675	0	108	133
`c2y`	97218	84337	79429	0	183	174
`c3y`	181124	131022	123548	0	436	289
`c4y`	137701	123266	116140	1	514	293
`c5y`	144625	101498	103958	0	325	247
`randomG4`	197298	159370	151074	1	266	319
`mesh33x33`	36468	32605	31929	0	536	198
`gd96a`	170903	96366	108804	1	250	201

Table 2.3: Comparison between cost and time (in seconds) for Spectral Sequencing (SS), Simulated Annealing and sequential SS+SA.

and `gd96a` graphs is it unable to improve the Simulated Annealing solution. The running times are usually lower for SS+SA than for Simulated Annealing.

Overall, these results show that the SS+SA heuristic is a valuable improvement over the Spectral Sequencing and Simulated Annealing alone, with some exceptions. In the case of the larger graph (the FE family) the improvements are substantial, both in quality and time.

Evaluation of the parallel SS+SA heuristics. In order to compare the behavior of the exact and chaotic strategies for parallel SS+SA, we have measured the running time and solution quality of these heuristics on our graphs. The assesment of their goodness is done by comparison with the sequential heuristic: Let T the time needed to execute the sequential SS+SA heuristic and C the cost obtained, let T_P be the time of the parallel SS+SA heuristic ran on P processors and let C_P the cost obtained. Recall that the *speedup* on P processors is T/T_P, and that the *efficiency factor* is $T/T_P P$. In order to compare the quality of the heuristics, we define the *quality factor* as C/C_P. Good parallelitzations ought to have the efficiency and quality factors close to one. In principle, the efficiency factor of any algorithm can never be larger than one, but due to the random nature of our heuristics, this is possible in our case.

The measured efficiency factor and quality factor for exact SS+SA are shown in Table 2.10 on page 89. The results for chaotic SS+SA are shown in Table 2.11 in page 90. Both tables are shown graphically in Figures 2.11 and 2.12. Traces of the parallel annealings curves for the `airfoil1` graph are shown in Figure 2.13.

From these measures, the following facts can be observed for the exact SS+SA strategy:

- For the VLSI graphs, exact parallel SS+SA does not work at all. The reason is that the number of forbidden moves is to high. A closer inspection (see the maximal degree information in Table 2.1) reveals that in all these graphs, there is a vertex connected with more than 300 neighbors that forbids too many moves as soon as $P \geqslant 3$.

- The quality factor of exact parallel SS+SA decreases as the number of processors increases. The trend shows that the quality factor depends on the size of the graph: For a fixed number of processors, larger graphs have a better quality factor.

- For the medium sized graphs, the efficiency factor of exact parallel SS+SA decreases as the number of processors increases. This does not seem to be the case for the larger graphs, for which the efficiency factor is close to 1.

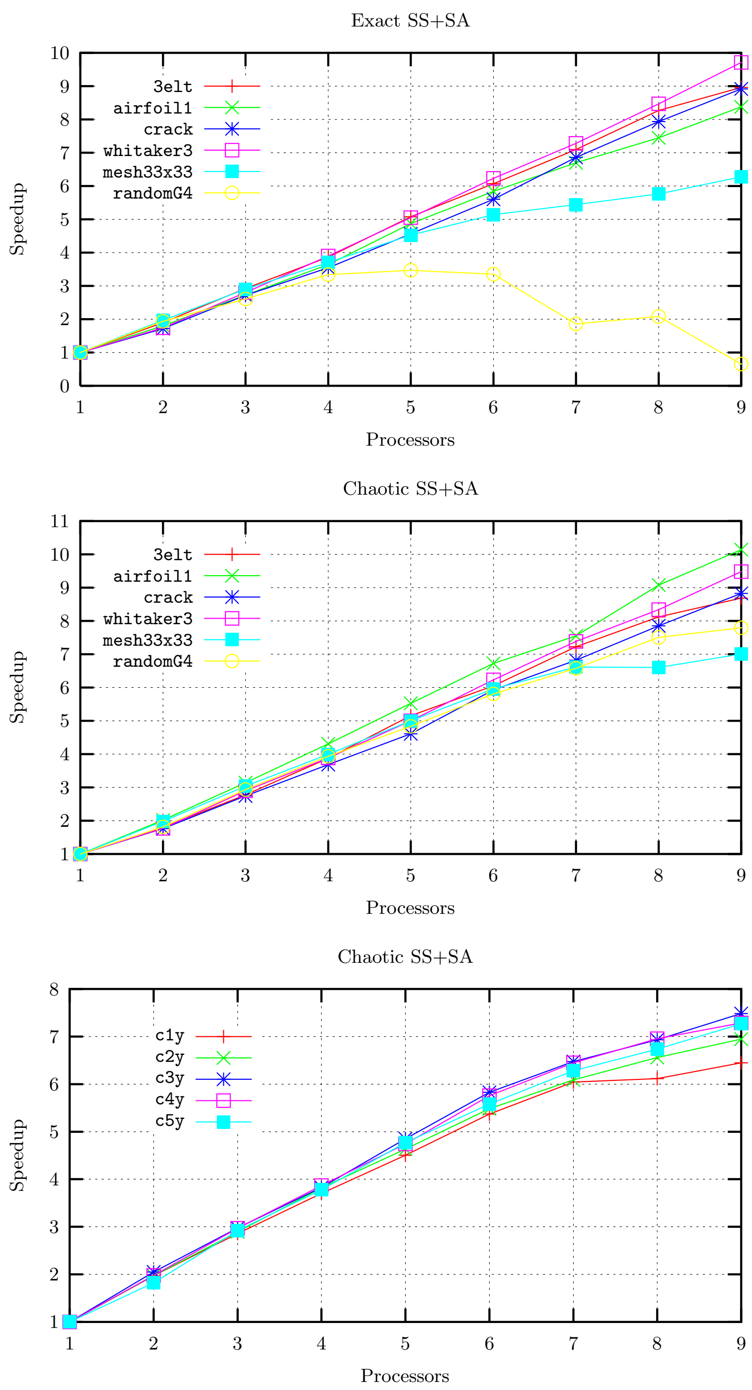


Figure 2.11: Speedup in function of the number of processors for the exact and chaotic SS+SA heuristics.

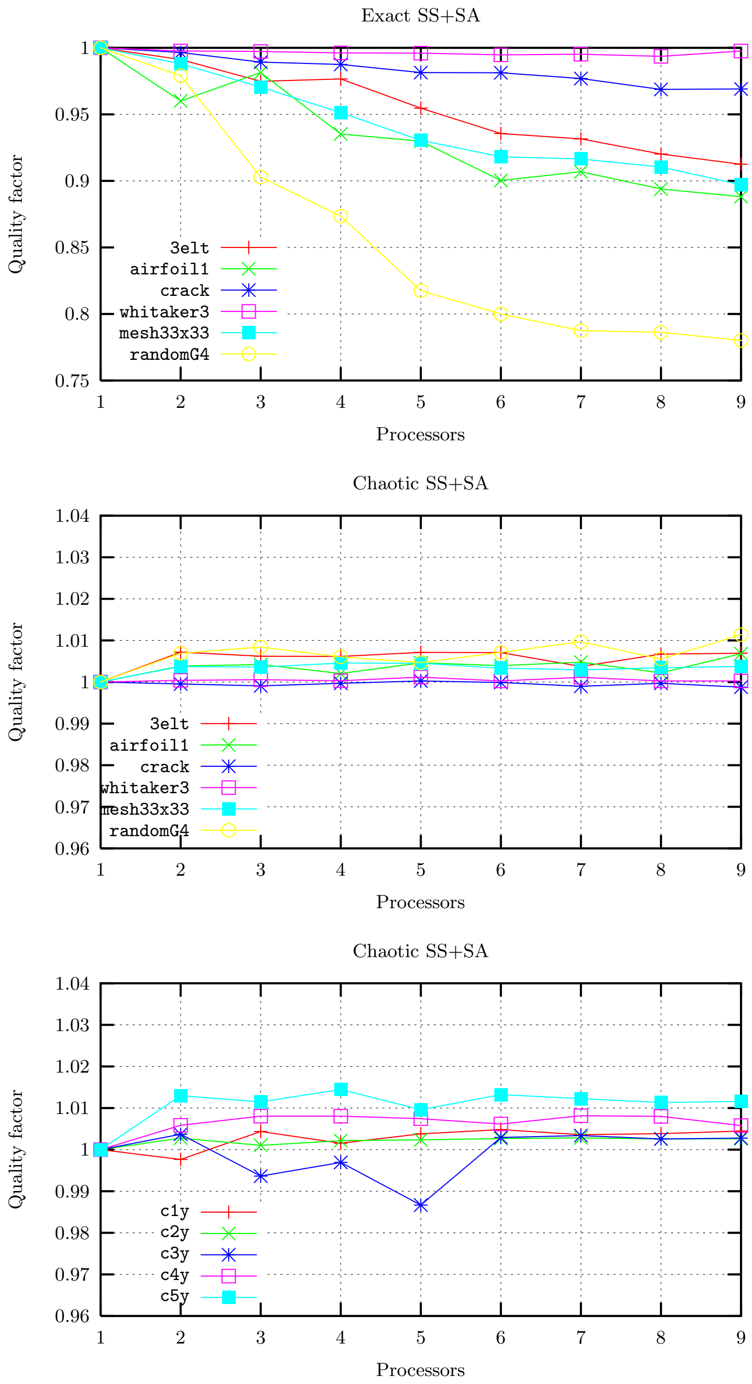
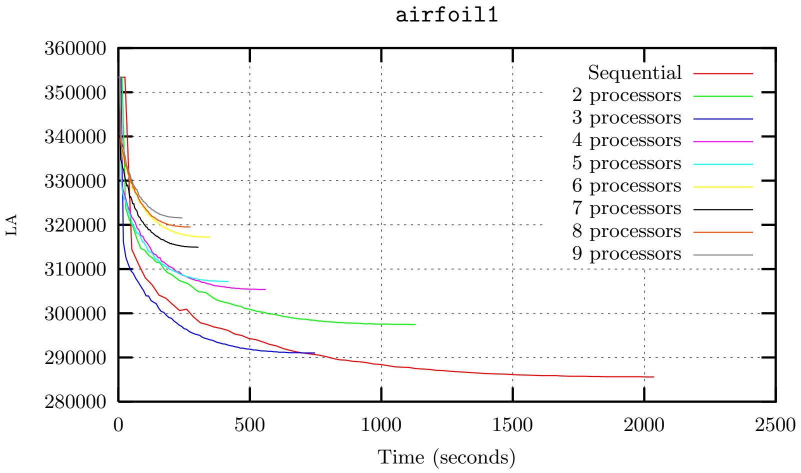
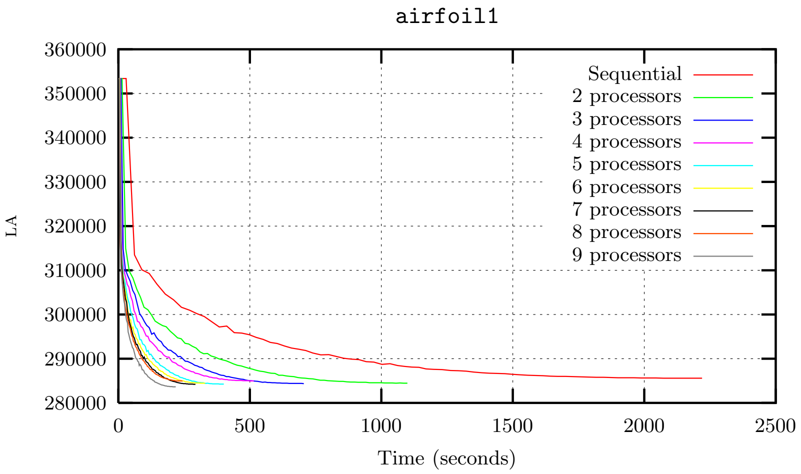


Figure 2.12: Quality factor in function of the number of processors for the exact and chaotic SS+SA heuristics.



(a) Exact parallel SS+SA



(b) Chaotic parallel SS+SA

Figure 2.13: Exact and chaotic parallel SS+SA on `airfoil1`: Cost in function of time depending on the number of processors.

Glabally, these observation show that the exact strategy for SS+SA is not very scalable and heavily depends on the number of forbidden moves.

In the case of the chaotic strategy for parallel SS+SA, the following observations can be made:

- Chaotic SS+SA can process all the inputs, because no movements are forbidden.
- The quality factor of chaotic parallel SS+SA is always close to 1 for the FE graphs. Specifically, its range is between 0.99 and 1.01. As a consequence, the cost of the solutions obtained by chaotic parallel SS+SA and the cost of the solutions obtained by sequential SS+SA differ at most by 2%. This variability is the same than the found for independent runs of sequential SS+SA.
- The efficiency factor of exact parallel SS+SA for medium sized graphs slightly decreases as the number of processors increases. The worst speedup is achieved with the `mesh33x33` graph with 9 processors: its efficiency factor is 0.779, which on a network of personal computer looks good. For the larger graphs, the efficiency factor is usually grater than 95%.

From the previous observations the following facts can be infered: With regards to solution quality, chaotic parallel SS+SA returns solutions of the same quality than sequential SS+SA. This is not true for exact SS+SA, for which the solution quality degrades as the number of processors increase. This shows, maybe surprisingly, that it is usefull to let the processors act without a complete knowledge of the state of the whole layout. No additional syncronization phases seem necessary to gossip the current layout. With regards to the time efficiency, the results of chaotic SS+SA on 9 processors present an excelent speedup for the larger graphs and a quite reasonable speedup for the medium sized graphs. In definitive, our mesurements show that the chaotic parallel SS+SA heuristic maintains solution quality, decreases the running time and offers an excellent speedup when ran in parallel, on a commodity network of personal computers.

2.6 Conclusions

This chapter has considered the Minimum Linear Arrabgement problem from an experimental point of view. In Sections 2.2 to 2.4 we have presented different heuristics to approximate the MinLA problem and to obtain lower bounds. We have also built a test suite made of regular graphs, random graphs and graphs arising in real life applications. These upper and lower bounding methods have been applied to this test suite, empirically obtaining the first comparative results on the MinLA problem.

With respect to the methods to find lower bounds, we have presented several new techniques, and we have observed that for certain classes of graphs, these deliver better bounds than previous existing ones.

With respect to the methods to find upper bounds, we have presented heuristics based on well known general techniques: Successive Augmentation, Spectral Sequencing and Local Search. Adapting these general techniques to the particular problem of MinLA is easy, but many decisions that have a great effect on their behavior have to be taken; for instance, the initial ordering of vertices in the Successive Augmentation heuristics, the neighborhood structure in Local Search, and the parameter tuning for Simulated Annealing. Due to the lack of theoretical results in the literature, the only way to characterize these decisions seems to be empirical testing. In contrast, Spectral Sequencing has the benefit to be an algorithm without parameters.

The extensive experimentation and the benchmarking presented suggest that, when measuring the quality of the solutions, Simulated Annealing is the best heuristic to approximate the MinLA problem on sparse graphs. However, this heuristic is extremely slow whereas Spectral Sequencing gives results not too far from those obtained by Simulated Annealing, but in much less time. In the case that a good approximation suffices, Spectral Sequencing would clearly be the method of choice.

Facing this dichotomy, in Section 2.5 we have focused on the design and the evaluation of the SS+SA heuristic. This new heuristic hybridizes Spectral Sequencing with Simulated Annealing. The aim was to accelerate the Simulated Annealing heuristic and to obtain better solutions. Doing so, we have addressed three main points in the improvement of Simulated Annealing techniques: the use of better-than-random initial solutions, the use of an adequate neighborhood specially adapted to improve good solutions, and the use of parallelism.

The use of initial solutions computed by Spectral Sequencing has proved to be very valuable, as it provides a simple way to enable Simulated Annealing to reach good solutions in shorter time, specially in large graphs. Moreover, we have seen that Simulated Annealing is capable to further improve these solutions by factors around 20%.

The use of better-than-random solutions to start Simulated Annealing directly implies the use of a lower starting temperature; but more subtely, it also gives rise to the use of more refined neighborhood relations that have a great influence on the runtime and the quality of the obtained solution. In the particular case of the MinLA problem, we have shown how to dynamically choose a neighborhood distribution depending on the input. This has been possible due to the implicit geometry in some of our graph instances and the good behaviour of the Spectral Sequencing heuristic.

Regarding the parallelization of Simulated Annealing, we have proposed a tailored parallelization that enables different processors to perform Metropolis concurrently on different partitions of the graph. In the exact strategy, the global state is always coherent, in the sense that it represents a feasible solution, and the processors have the entire knowledge of it. This is ensured by forbiding the moves that would render the knowledge incomplete or uncertain. Some syncronization phases and different partitioning schemes are used so as not allways forbid the same moves. In the chaotic strategy, the global state is still maintained coherent, but processors are allowed to perform moves having only an approximation of the global state. Therefore, in this case, computing the gain of a move is subject to an error. Infrequent syncronization phases where the global state is broadcasted to all the processors are used in order to not allow large errors.

The experiments and the benchmarkings we have presented on large sparse graphs show the viability and practibility of our approaches based on SS+SA. On one hand, using sequential SS+SA, we have found better approximations in radically less time. On the other hand, we have seen that the chaotic strategy for the parallel SS+SA has an excellent behaviour in terms of running time and solution quality compared to the sequential heuristic when running on a network of personal computers. The drawback of this heuristic is that there is no improvement for the graphs with small diameter.[4]

Globally, our experiments evidence that there exists a large gap between the best known upper bounds the best known lower bounds. The conclusion we draw is that, from the empirical point of view, it is not only hard to get good upper bounds the MinLA problem, but also it is difficult to get good estimates for the lower bounds. Therefore, one question remains: How good are our solutions?

There exist a number of other heuristics that can probably be adapted to the Minimum Linear Arrangement but have not been considered in this study; Genetic Algorithms [57], Tabu Search [102], and linear relaxations [213] are only a few examples of these. Investigating them and applying them to the proposed test suite is left as work to be done.

In order to help to reproduce our experiments, the code for `llsh` and the test suite are available through the World Wide Web and are attached in the accompaining CD-ROM.

A preliminary version of this chapter, without the SS+SA heuristic and the Mesh method, was presented at the workshop *Algorithms and Experiments*—ALEX '98 *(Trento,* 1998*)* [208]. A journal version of this paper is currently under revision [210]. The Mesh method and considerations on the difficulty to get good lower bounds for the MinLA problem were presented at the *Exper-*

[4] In the following chapter we will see that this is not really a problem, as any layout of these graphs is quite good.

imental Algorithms seminar (*Dagstuhl*, 2000) [209]. The study on the SS+SA heuristic has been also submited for publication [207].

Very recently, Bar-Yehuda, Even, Feldman and Naor have used the test suite of graphs presented in Section 2.3, a seminal version of the `llsh` toolkit described in Section 2.2.3, and the experimental results on heuristics given in Section 2.4 in order to experiment heuristics based on the computation of optimal orientations of balanced decomposition trees for MinLA and Cutwidth [17]. These authors report experimental results on their own heuristics and compare them to the ones presented in this chapter. Their results are of roughly the same quality than the ones obtained by Simulated Annealing but at a fraction of its running time. Our results with SS+SA offer a better quality and are obtained faster, particularly in the FE graphs; see Table 2.12.

On the other hand, our comparative study also establishes the big differences existing between the approximation of randomly generated graphs and the approximation graphs arising from real life applications. These will be analysed in Chapters 3 and 5.

Legend	Meaning
spectral	Spectral Sequencing
sa	Simulated Annealing
random	Random method
hillc_E	Hillclimbing using the FlipE neighborhood
hillc_3	Hillclimbing using the Flip3 neighborhood
hillc_2	Hillclimbing using the Flip2 neighborhood
greedy_rnd	Successive Augmentation method with random initial ordering
greedy_rbs	Successive Augmentation method with random breadth search
greedy_nor	Successive Augmentation method with normal ordering
greedy_dfs	Successive Augmentation method with depth-first search
greedy_bfs	Successive Augmentation method with breadth-first search
normal	Normal method
best	Best known upper bound
Edges	Edges method
Degree	Degree method
Path	Path method
J–M	Juvan–Mohar method
G–H	Gomory–Hu Tree method
Mesh	Mesh method
s	Second
min	Minute
h	Hour

Table 2.4: Abbreviations used in figures and tables.

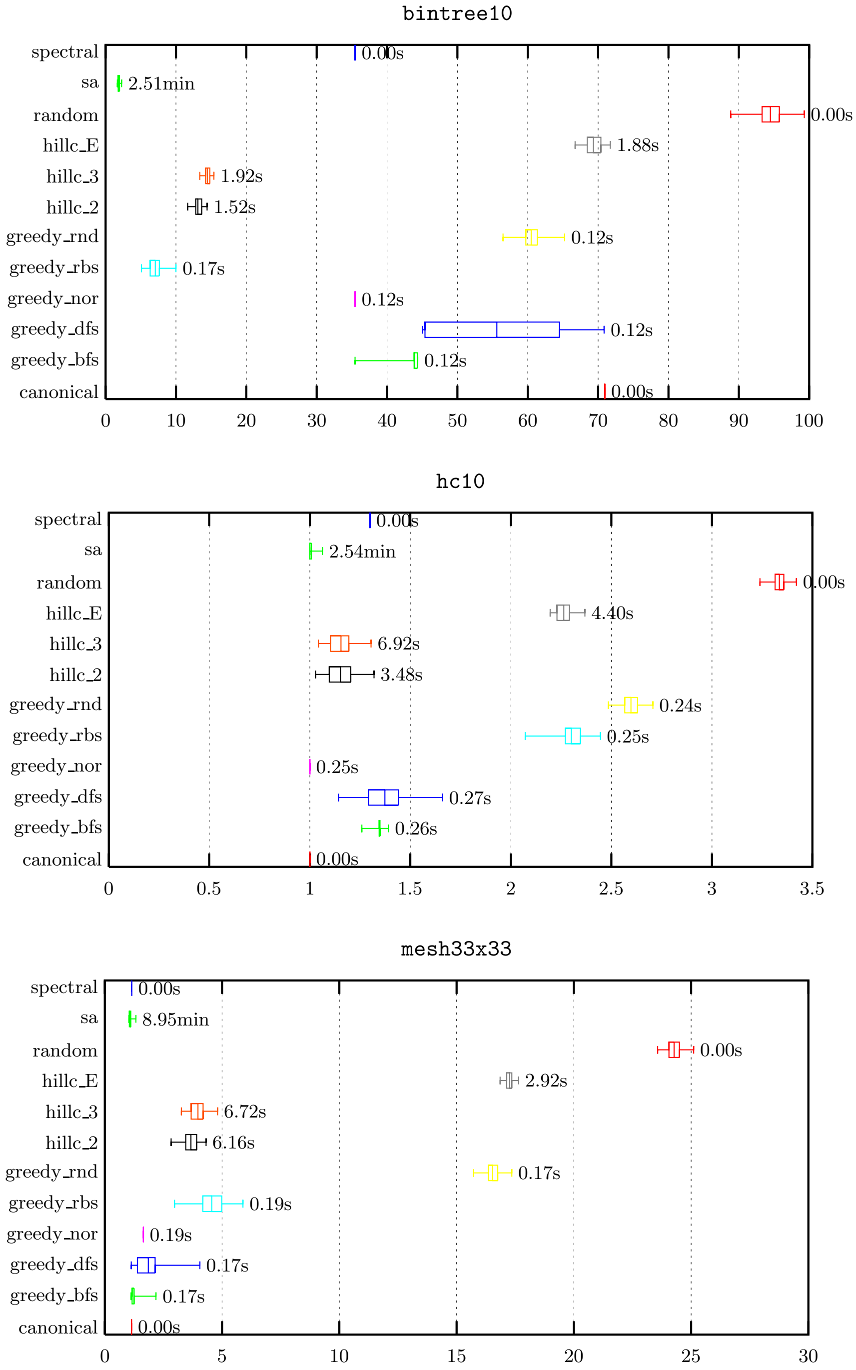


Figure 2.14: Comparison of heuristics, boxplots (Part 1). The absolute costs have been normalized dividing them by the respective optimal costs.

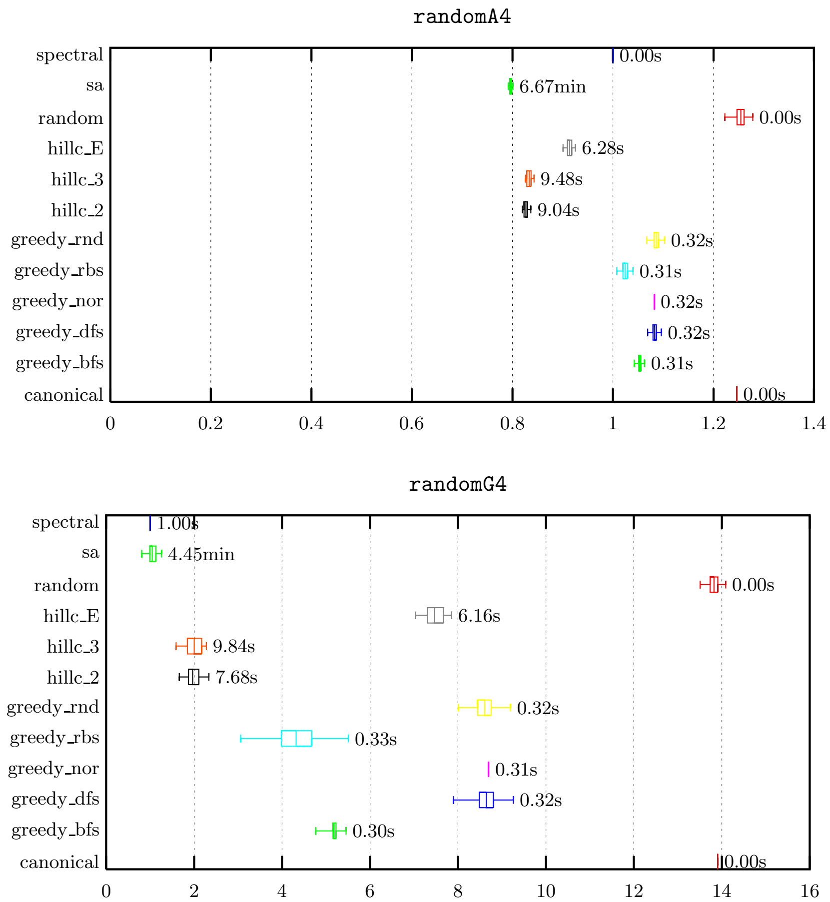


Figure 2.15: Comparison of heuristics, boxplots (Part 2). The absolute costs have been normalized dividing them by the the cost found by Spectral Sequencing.

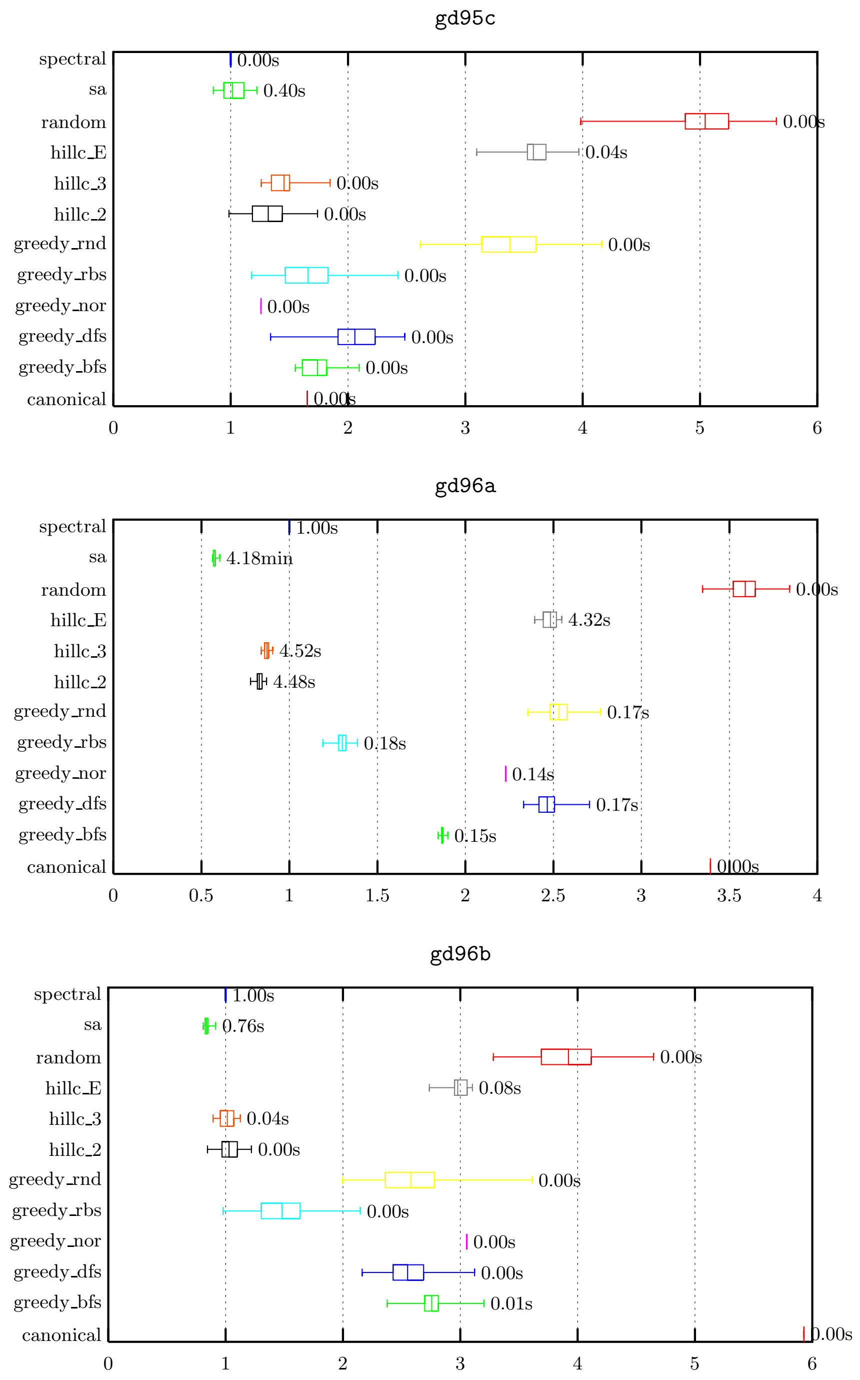


Figure 2.16: Comparison of heuristics, boxplots (Part 3).

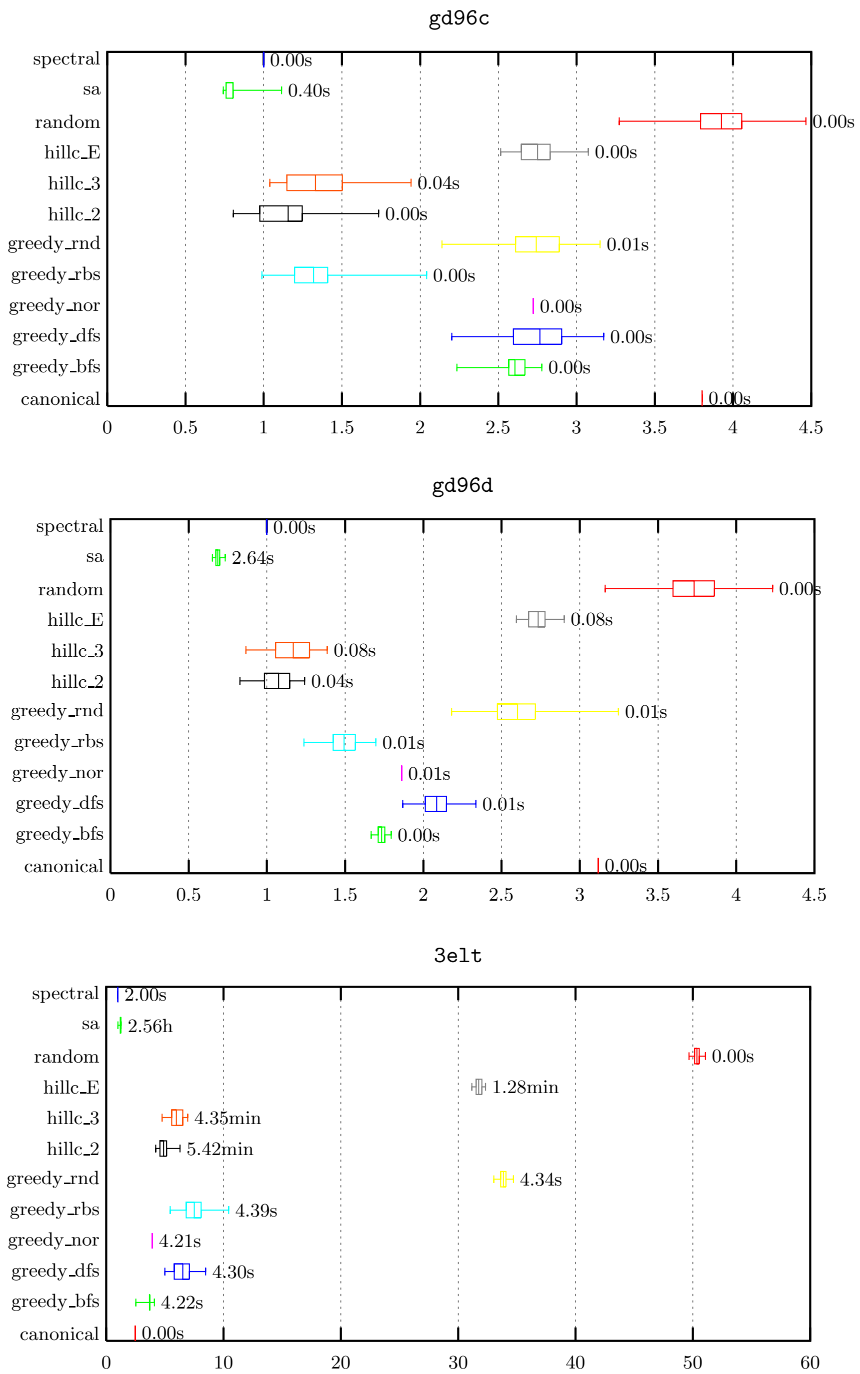


Figure 2.17: Comparison of heuristics, boxplots (Part 4).

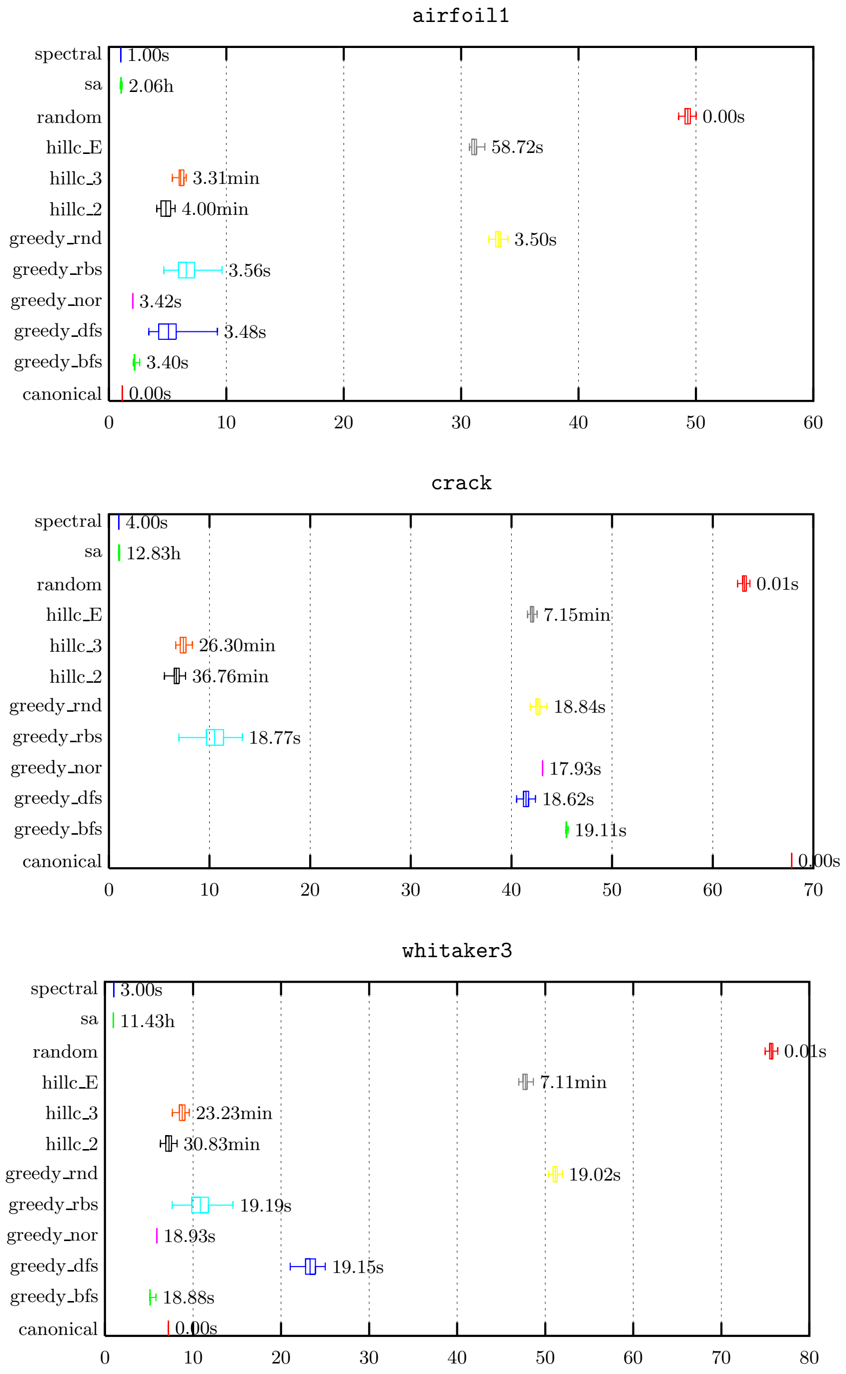
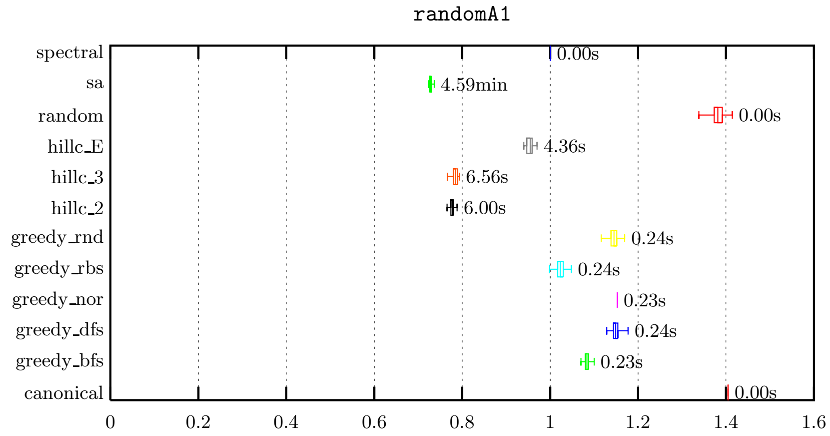


Figure 2.18: Comparison of heuristics, boxplots (Part 5).



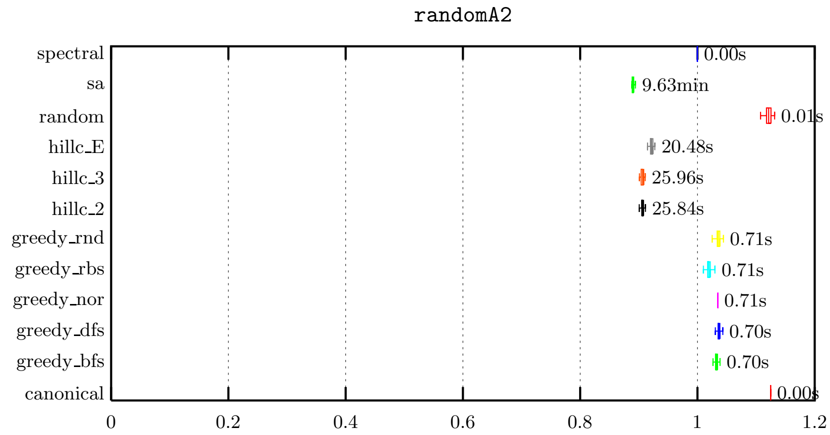


Figure 2.19: Comparison of heuristics, boxplots (Part 6).

Graph	Method	Min	1st quart.	Average	3rd quart.	Max	Time	Reps
randomA1	normal	1688528	1688528	1688528	1688528	1688528	0	1
	random	1608960	1650709	1660809	1672807	1700162	0	200
	spectral	1202165	1202165	1202165	1202165	1202165	0	1
	greedy_nor	1385226	1385226	1385226	1385226	1385226	0	200
	greedy_rnd	1341696	1368795	1376738	1384369	1406151	0	200
	greedy_rbs	1200746	1222818	1229829	1237855	1260444	0	200
	greedy_bfs	1286602	1298921	1302894	1307361	1322309	0	200
	greedy_dfs	1356687	1375501	1381587	1387065	1415392	0	200
	hillc_E	1130087	1138174	1147399	1152772	1166051	4	25
	hillc_2	920542	930760	934802	937625	947777	6	25
	hillc_3	920781	937596	942849	949223	953914	6	25
	sa	869648	872872	875343	878222	885592	275	25
randomA2	normal	8278100	8278100	8278100	8278100	8278100	0	1
	random	8153381	8228183	8254256	8284345	8331340	0	200
	spectral	7361167	7361167	7361167	7361167	7361167	0	1
	greedy_nor	7619276	7619276	7619276	7619276	7619276	0	200
	greedy_rnd	7546703	7609219	7623879	7642058	7687473	0	200
	greedy_rbs	7436358	7488739	7505766	7522002	7580046	0	200
	greedy_bfs	7556635	7588784	7600226	7612381	7645291	0	200
	greedy_dfs	7586741	7620507	7631157	7642426	7680278	0	200
	hillc_E	6735877	6771871	6786815	6802899	6825760	20	25
	hillc_2	6630749	6661508	6673037	6684140	6708047	25	25
	hillc_3	6632039	6653199	6672108	6686529	6706585	25	25
	sa	6536540	6542800	6553901	6559770	6580910	577	25
randomA4	normal	2710738	2710738	2710738	2710738	2710738	0	1
	random	2658957	2711939	2727987	2742260	2779787	0	200
	spectral	2174723	2174723	2174723	2174723	2174723	0	1
	greedy_nor	2353193	2353193	2353193	2353193	2353193	0	200
	greedy_rnd	2320797	2352277	2361570	2370935	2397905	0	200
	greedy_rbs	2190885	2217589	2227031	2237612	2261270	0	200
	greedy_bfs	2267018	2285808	2290380	2295783	2311333	0	200
	greedy_dfs	2325356	2348372	2355859	2363149	2383846	0	200
	hillc_E	1958524	1978164	1986684	1996556	2011961	6	25
	hillc_2	1782901	1789549	1798613	1805831	1818999	9	25
	hillc_3	1796599	1802109	1812306	1820063	1832943	9	25
	sa	1721490	1728000	1731627	1735620	1741510	400	25
randomG4	normal	2743658	2743658	2743658	2743658	2743658	0	1
	random	2665512	2709910	2727396	2744030	2780200	0	200
	spectral	197298	197298	197298	197298	197298	1	1
	greedy_nor	1715980	1715980	1715980	1715980	1715980	0	200
	greedy_rnd	1580249	1666365	1698157	1727682	1813529	0	200
	greedy_rbs	603158	785879	852453	921799	1086533	0	200
	greedy_bfs	939972	1017738	1022454	1031605	1076151	0	200
	greedy_dfs	1558053	1674085	1705817	1737451	1827220	0	200
	hillc_E	1387889	1441892	1473687	1513016	1550085	6	25
	hillc_2	328416	369537	388533	416378	461041	7	25
	hillc_3	313305	364673	395211	428128	449448	9	25
	sa	159370	195836	206479	222573	248380	266	25

Table 2.5: Comparison of heuristics, absolute values (Part 1).

Graph	Method	Min	1st quart.	Average	3rd quart.	Max	Time	Reps
bintree10	normal	262143	262143	262143	262143	262143	0	1
	random	328361	344826	349243	353976	366980	0	200
	spectral	131072	131072	131072	131072	131072	0	1
	greedy_nor	131072	131072	131072	131072	131072	0	200
	greedy_rnd	208796	220803	223554	226827	241135	0	200
	greedy_rbs	18867	23407	26149	28272	36989	0	200
	greedy_bfs	131072	162117	162315	163728	163964	0	200
	greedy_dfs	166386	167725	205525	238459	261856	0	200
	hillc_E	246574	252969	256202	260171	265146	1	25
	hillc_2	43168	47331	48526	50491	53256	1	25
	hillc_3	49514	52528	53408	54787	56948	1	25
	sa	6205	6614	6984	7282	8429	150	25
hc10	normal	523776	523776	523776	523776	523776	0	1
	random	1696668	1735922	1747370	1758914	1791510	0	200
	spectral	680388	680388	680388	680388	680388	0	1
	greedy_nor	523776	523776	523776	523776	523776	0	200
	greedy_rnd	1301454	1344530	1360348	1377480	1417932	0	200
	greedy_rbs	1084962	1189402	1205017	1228546	1280982	0	200
	greedy_bfs	659612	704674	705005	706636	728822	0	200
	greedy_dfs	598470	676690	719642	754326	869378	0	200
	hillc_E	1149752	1167642	1185332	1200668	1240598	4	25
	hillc_2	538816	574462	603577	630544	691156	3	25
	hillc_3	546120	577720	605090	625782	683324	6	25
	sa	523776	523776	527715	523808	556544	152	25
mesh33x33	normal	35904	35904	35904	35904	35904	0	1
	random	746880	762134	768999	776288	795451	0	200
	spectral	36468	36468	36468	36468	36468	0	1
	greedy_nor	52272	52272	52272	52272	52272	0	200
	greedy_rnd	497988	518055	523990	530845	549791	0	200
	greedy_rbs	94176	132453	144575	157984	186545	0	200
	greedy_bfs	35559	36945	39506	39381	69145	0	200
	greedy_dfs	35842	44164	58887	67939	128458	0	200
	hillc_E	534072	542915	546800	549711	559098	2	25
	hillc_2	89582	109518	116233	124143	136937	6	25
	hillc_3	103298	116533	125731	132911	152482	6	25
	sa	32605	33171	34666	35212	42136	536	25

Table 2.6: Comparison of heuristics, absolute values (Part 2).

Graph	Method	Min	1st quart.	Average	3rd quart.	Max	Time	Reps
3elt	normal	1060932	1060932	1060932	1060932	1060932	0	1
	random	21320524	21519449	21607450	21691862	21915401	0	200
	spectral	429164	429164	429164	429164	429164	2	1
	greedy_nor	1684916	1684916	1684916	1684916	1684916	4	200
	greedy_rnd	14181869	14430016	14530357	14626358	14893864	4	200
	greedy_rbs	2338237	2927279	3222942	3472921	4476144	4	200
	greedy_bfs	1083604	1583242	1588782	1591995	1764712	4	200
	greedy_dfs	2145119	2489899	2798318	3039555	3639601	4	200
	hillc_E	13376292	13531060	13623110	13741548	13875429	76	25
	hillc_2	1813778	1964188	2100562	2205995	2700645	325	25
	hillc_3	2044785	2399181	2569562	2798290	2981052	261	25
	sa	428907	527159	512629	530245	546590	9200	5
airfoil1	normal	407921	407921	407921	407921	407921	0	1
	random	17149384	17341726	17423203	17504076	17676431	0	200
	spectral	353399	353399	353399	353399	353399	1	1
	greedy_nor	718566	718566	718566	718566	718566	3	200
	greedy_rnd	11439872	11642293	11719155	11799037	12008340	3	200
	greedy_rbs	1650302	2092549	2329099	2576867	3406870	3	200
	greedy_bfs	734283	767776	772806	775012	922526	3	200
	greedy_dfs	1204510	1496655	1791999	2027001	3261700	3	200
	hillc_E	10849400	10920463	10999010	11064460	11314710	58	25
	hillc_2	1439406	1565493	1717401	1856146	1989228	240	25
	hillc_3	1911201	2111029	2159552	2253527	2326138	198	25
	sa	338120	359002	368420	374381	399534	7427	5
crack	normal	111329381	111329381	111329381	111329381	111329381	0	1
	random	102505474	103357469	103651106	103956543	104534672	0	200
	spectral	1641119	1641119	1641119	1641119	1641119	4	1
	greedy_nor	70753206	70753206	70753206	70753206	70753206	17	200
	greedy_rnd	68730362	69663447	69971935	70270782	71443429	18	200
	greedy_rbs	11406337	15921816	17243012	18710735	21764543	18	200
	greedy_bfs	74483018	74582007	74604066	74624097	74893737	19	200
	greedy_dfs	66488754	67610879	68028543	68455398	69562912	18	200
	hillc_E	68252742	68746410	69048432	69258494	69825470	428	25
	hillc_2	9059517	10661548	10993289	11462027	12490825	2205	25
	hillc_3	10891691	11638994	12138627	12584082	13615953	1577	25
	sa	1576500	1610520	1647800	1756380	1756380	46178	3
whitaker3	normal	9029276	9029276	9029276	9029276	9029276	0	1
	random	93842080	94489158	94720883	94957871	95634550	0	200
	spectral	1251709	1251709	1251709	1251709	1251709	3	1
	greedy_nor	7415719	7415719	7415719	7415719	7415719	18	200
	greedy_rnd	63092055	63720780	64013600	64307195	65039074	19	200
	greedy_rbs	9553062	12342349	13568382	14717919	18210032	19	200
	greedy_bfs	6399474	6408126	6424074	6420849	7246221	18	200
	greedy_dfs	26342064	28526623	29164281	29925884	31301560	19	200
	hillc_E	58847006	59446135	59780810	60027241	60898668	426	25
	hillc_2	7901602	8655390	9112885	9472800	10268095	1849	25
	hillc_3	9598131	10600303	10980574	11373250	11977102	1393	25
	sa	1214090	1214090	1214090	1214090	1214090	41144	1

Table 2.7: Comparison of heuristics, absolute values (Part 3).

Graph	Method	Min	1st quart.	Average	3rd quart.	Max	Time	Reps
c1y	normal	369905	369905	369905	369905	369905	0	1
	random	445563	467333	482476	497232	536365	0	200
	spectral	104316	104316	104316	104316	104316	0	1
	greedy_nor	278524	278524	278524	278524	278524	0	200
	greedy_rnd	312882	329352	341383	349023	396368	0	200
	greedy_rbs	173796	195016	200477	206899	219548	0	200
	greedy_bfs	170620	173901	174576	174760	179760	0	200
	greedy_dfs	272827	316520	321447	332508	349924	0	200
	hillc_E	326967	339991	343249	348524	351931	3	25
	hillc_2	99807	109656	114891	121314	127929	2	25
	hillc_3	99809	112143	118124	125487	132521	2	25
	sa	63145	66894	68580	70845	74556	108	25
c2y	normal	517737	517737	517737	517737	517737	0	1
	random	642478	667499	687478	705238	752918	0	200
	spectral	97218	97218	97218	97218	97218	0	1
	greedy_nor	376023	376023	376023	376023	376023	0	200
	greedy_rnd	448882	466052	482289	494042	553589	0	200
	greedy_rbs	247507	269417	276950	284847	305920	0	200
	greedy_bfs	243932	251583	252117	252744	258705	0	200
	greedy_dfs	359982	427100	441885	460761	481673	0	200
	hillc_E	479575	484418	488861	492908	501914	4	25
	hillc_2	135081	144423	150906	155503	170584	3	25
	hillc_3	146080	153561	160706	167957	170004	4	25
	sa	79673	84337	88336	91560	95438	183	25
c3y	normal	787049	787049	787049	787049	787049	0	1
	random	1170820	1227185	1259207	1286481	1352826	0	200
	spectral	181124	181124	181124	181124	181124	0	1
	greedy_nor	610049	610049	610049	610049	610049	0	200
	greedy_rnd	823692	853338	880323	902213	982089	0	200
	greedy_rbs	418346	460157	474888	489417	528791	0	200
	greedy_bfs	376559	383416	385251	385120	407359	0	200
	greedy_dfs	686079	756057	779166	806142	847669	0	200
	hillc_E	867123	881192	894030	904758	932313	9	25
	hillc_2	234019	238949	251907	258872	305307	9	25
	hillc_3	234911	257910	269887	281528	299966	9	25
	sa	126470	131022	136826	143322	149318	436	25
c4y	normal	919089	919089	919089	919089	919089	0	1
	random	1267909	1302717	1330982	1355140	1422005	0	200
	spectral	137701	137701	137701	137701	137701	1	1
	greedy_nor	612069	612069	612069	612069	612069	0	200
	greedy_rnd	867197	895358	921694	942188	1020830	0	200
	greedy_rbs	411138	462853	477248	493945	534373	0	200
	greedy_bfs	355683	360597	361786	362020	382696	0	200
	greedy_dfs	693293	790328	801906	823722	862950	0	200
	hillc_E	914375	924058	937773	945934	965645	7	25
	hillc_2	211434	239098	252345	261431	288450	9	25
	hillc_3	247831	262053	272256	282782	293202	9	25
	sa	116724	123266	128926	134566	141220	514	25
c5y	normal	743485	743485	743485	743485	743485	0	1
	random	959876	1002534	1028436	1056645	1115226	0	200
	spectral	144625	144625	144625	144625	144625	0	1
	greedy_nor	517030	517030	517030	517030	517030	0	200
	greedy_rnd	662412	693253	716073	735884	796897	0	200
	greedy_rbs	342150	374915	386193	397432	437088	0	200
	greedy_bfs	273303	298867	299549	300045	338821	0	200
	greedy_dfs	525419	616015	626278	639429	680050	0	200
	hillc_E	703472	717221	725449	732311	745970	6	25
	hillc_2	176621	196409	207727	218067	231266	7	25
	hillc_3	203184	210635	221375	229091	252912	7	25
	sa	97791	101498	105586	109362	116188	325	25

Table 2.8: Comparison of heuristics, absolute values (Part 4).

Graph	Method	Min	1st quart.	Average	3rd quart.	Max	Time	Reps
gd95c	normal	990	990	990	990	990	0	1
	random	2386	2920	3021	3141	3385	0	200
	spectral	599	599	599	599	599	0	1
	greedy_nor	754	754	754	754	754	0	200
	greedy_rnd	1567	1881	2024	2157	2493	0	200
	greedy_rbs	706	878	993	1096	1453	0	200
	greedy_bfs	929	965	1041	1089	1255	0	200
	greedy_dfs	803	1147	1232	1336	1488	0	200
	hillc_E	1855	2114	2144	2209	2376	0	25
	hillc_2	590	710	790	863	1042	0	25
	hillc_3	755	807	871	900	1107	0	25
	sa	511	564	607	667	733	0	25
gd96a	normal	579874	579874	579874	579874	579874	0	1
	random	572245	602068	613546	623242	656677	0	200
	spectral	170903	170903	170903	170903	170903	1	1
	greedy_nor	380975	380975	380975	380975	380975	0	200
	greedy_rnd	402485	424062	432576	440655	473195	0	200
	greedy_rbs	203431	218678	222189	225786	236996	0	200
	greedy_bfs	315205	318912	319299	320069	324777	0	200
	greedy_dfs	398422	413305	421313	428463	462458	0	200
	hillc_E	409109	417324	424430	430329	435360	4	25
	hillc_2	133158	139907	141600	144318	148777	4	25
	hillc_3	143558	146829	149072	150807	154680	4	25
	sa	96366	97077	98658	99083	103321	250	25
gd96b	normal	10887	10887	10887	10887	10887	0	1
	random	6024	6776	7200	7558	8533	0	200
	spectral	1836	1836	1836	1836	1836	1	1
	greedy_nor	5608	5608	5608	5608	5608	0	200
	greedy_rnd	3668	4333	4735	5105	6633	0	200
	greedy_rbs	1797	2392	2717	3002	3944	0	200
	greedy_bfs	4362	4950	5060	5163	5878	0	200
	greedy_dfs	3971	4455	4683	4935	5732	0	200
	hillc_E	5024	5417	5467	5615	5697	0	25
	hillc_2	1551	1779	1889	2018	2240	0	25
	hillc_3	1638	1754	1863	1964	2067	0	25
	sa	1484	1514	1537	1559	1678	0	25
gd96c	normal	2665	2665	2665	2665	2665	0	1
	random	2294	2658	2751	2843	3131	0	200
	spectral	701	701	701	701	701	0	1
	greedy_nor	1909	1909	1909	1909	1909	0	200
	greedy_rnd	1500	1830	1922	2026	2208	0	200
	greedy_rbs	693	839	924	987	1431	0	200
	greedy_bfs	1567	1799	1826	1872	1947	0	200
	greedy_dfs	1544	1820	1938	2036	2224	0	200
	hillc_E	1763	1855	1928	1984	2156	0	25
	hillc_2	565	683	810	874	1216	0	25
	hillc_3	728	805	933	1053	1361	0	25
	sa	520	532	563	564	781	0	25
gd96d	normal	11537	11537	11537	11537	11537	0	1
	random	11704	13310	13807	14286	15664	0	200
	spectral	3701	3701	3701	3701	3701	0	1
	greedy_nor	6891	6891	6891	6891	6891	0	200
	greedy_rnd	8069	9155	9631	10055	12018	0	200
	greedy_rbs	4575	5270	5525	5796	6278	0	200
	greedy_bfs	6169	6334	6416	6492	6643	0	200
	greedy_dfs	6915	7450	7715	7947	8647	0	200
	hillc_E	9603	9890	10120	10280	10736	0	25
	hillc_2	3061	3645	3978	4240	4591	0	25
	hillc_3	3206	3904	4326	4707	5128	0	25
	sa	2414	2493	2537	2583	2714	2	25

Table 2.9: Comparison of heuristics, absolute values (Part 5).

Processors	1	2	3	4	5	6	7	8	9
`3elt`	1.000	0.991	0.975	0.977	0.955	0.936	0.932	0.920	0.913
	1.000	0.950	0.974	0.967	1.016	1.011	1.013	1.034	0.996
`airfoil1`	1.000	0.960	0.981	0.935	0.930	0.900	0.907	0.894	0.888
	1.000	0.901	0.908	0.910	0.974	0.973	0.957	0.931	0.931
`crack`	1.000	0.996	0.989	0.987	0.981	0.981	0.977	0.969	0.969
	1.000	0.859	0.907	0.887	0.913	0.934	0.980	0.992	0.991
`whitaker3`	1.000	0.998	0.997	0.996	0.996	0.995	0.995	0.994	0.998
	1.000	0.872	0.934	0.975	1.010	1.038	1.040	1.058	1.079
`mesh33x33`	1.000	0.988	0.971	0.951	0.930	0.918	0.916	0.910	0.897
	1.000	0.984	0.968	0.926	0.906	0.857	0.778	0.720	0.698
`randomG4`	1.000	0.979	0.903	0.874	0.817	0.800	0.788	0.786	0.780
	1.000	0.968	0.870	0.835	0.694	0.559	0.266	0.261	0.073

Table 2.10: Quality and efficiency factors for exact parallel SS+SA relative to sequential SS+SA. For each graph, the top number is the solution quality factor and the bottom number is the time efficency factor.

Processors	1	2	3	4	5	6	7	8	9
3elt	1.000	1.007	1.006	1.006	1.007	1.007	1.004	1.007	1.007
	1.000	0.900	0.926	0.969	1.031	1.009	1.032	1.015	0.965
airfoil1	1.000	1.004	1.004	1.002	1.005	1.004	1.005	1.002	1.007
	1.000	1.008	1.047	1.077	1.105	1.120	1.079	1.134	1.127
crack	1.000	1.000	0.999	1.000	1.000	1.000	0.999	1.000	0.999
	1.000	0.880	0.914	0.922	0.921	0.988	0.974	0.982	0.981
whitaker3	1.000	1.000	1.001	1.000	1.001	1.000	1.001	1.000	1.000
	1.000	0.885	0.967	0.970	1.000	1.038	1.055	1.042	1.054
mesh33x33	1.000	1.004	1.004	1.005	1.005	1.003	1.003	1.003	1.004
	1.000	0.991	1.013	0.998	1.000	0.994	0.945	0.825	0.779
randomG4	1.000	1.007	1.008	1.006	1.005	1.007	1.010	1.006	1.011
	1.000	0.906	0.978	0.984	0.967	0.968	0.940	0.938	0.866
c1y	1.000	0.998	1.004	1.002	1.004	1.005	1.004	1.004	1.004
	1.000	0.984	0.952	0.925	0.901	0.895	0.863	0.765	0.716
c2y	1.000	1.003	1.001	1.002	1.002	1.003	1.003	1.003	1.003
	1.000	0.984	0.964	0.955	0.927	0.914	0.870	0.820	0.772
c3y	1.000	1.004	0.994	0.997	0.987	1.003	1.003	1.003	1.003
	1.000	1.026	0.991	0.957	0.970	0.972	0.925	0.866	0.832
c4y	1.000	1.006	1.008	1.008	1.007	1.006	1.008	1.008	1.006
	1.000	0.988	0.989	0.967	0.949	0.959	0.921	0.869	0.810
c5y	1.000	1.013	1.012	1.014	1.010	1.013	1.012	1.011	1.012
	1.000	0.911	0.973	0.945	0.953	0.930	0.898	0.842	0.808

Table 2.11: Quality and efficiency factors for chaotic parallel SS+SA relative to sequential SS+SA. For each graph, the top number is the solution quality factor and the bottom number is the time efficency factor.

Graph	Cost	Time
randomA1	884261	2328
randomA2	6576912	191645
randomA4	14289214	10869
randomG4	146996	8376
hc10	523776	2545
mesh33x33	33531	278
bintree10	3762	60
3elt	363204	5757
airfoil1	289217	4552
crack	—	—
whitaker3	1200374	29937
c1y	62333	205
c2y	79571	299
c3y	127065	553
c4y	115222	580
c5y	96956	446
gd95c	506	2
gd96a	99944	268
gd96b	1422	3
gd96c	519	1
gd96d	2409	4

Table 2.12: Results of [17]. Times are given in seconds for an Intel P-III 600 MHz machine (about 1200 BogoMips).

3

Layout Problems and Binomial Random Graphs

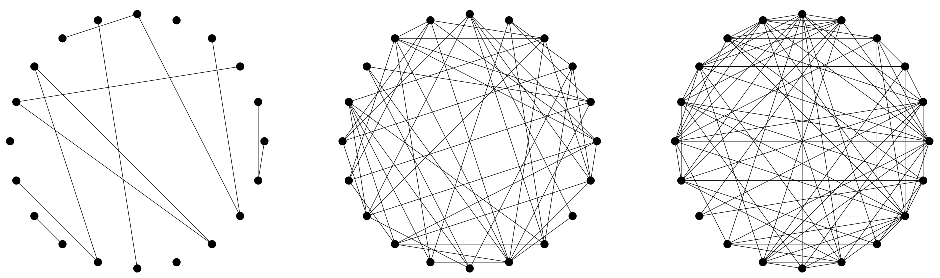
3.1 Introduction

Random graphs are an active area of research that begun in 1960 in a paper by Erdős and Rényi [82]. In that paper, random graphs were defined by the uniform model, denoted $\mathcal{G}_{n,m}$, that consists in taking uniformly one graph among all possible graphs with n vertices and m undirected edges. In this chapter, we deal with a closely related model, the binomial model, denoted $\mathcal{G}_{n,p}$, which consists of graphs with n vertices where each possible undirected edge is present with probability p. The terms "uniform random graph" and "binomial random graph" are taken from [133]. More formally:

Definition 3.1 (Binomial random graphs). Let n be a natural and p a probability. The class of *binomial random graphs*, denoted by $\mathcal{G}_{n,p}$, is a probability space over the set of undirected graphs $G = (V, E)$ on the vertex set $V = [n]$ determined by

$$\mathbf{Pr}\left[uv \in E\right] = p$$

with these events mutually independent.



(a) n=20, p=.075 **(b)** n=20, p=.15 **(c)** n=20, p=.45

Figure 3.1: Binomial random graphs $\mathcal{G}_{n,p}$.

Binomial random graphs can be generated by the outcome of the following experiment (which is a Bernoulli process): Take $V = [n]$ and then, for each $u \in V$ and each $v \in V \setminus \{u\}$, include the edge uv in E with probability p. Different binomial random graphs generated in this way are shown in Figure 3.1.

In order to establish asymptotic results, it is often necessary to consider random graphs whose edge probability p_n or number of edges m_n are functions of n. It is well know that the $\mathcal{G}_{n,p_n}$ and $\mathcal{G}_{n,m_n}$ models of random graphs have similar properties when $p_n \sim m_n/\binom{n}{2}$ [34, 133]. Also, it is known that binomial random graphs exhibit a *phase transition* at $p_n = 1/n$: when $p_n = c/n$ with $c < 1$, binomial random graphs consist of many small connected components whose largest size is $\Theta(\log n)$; when $p_n = c/n$ with $c > 1$, binomial random graphs turn to have a *giant component*, that is, a connected component with $\Theta(n)$ vertices [5, 34, 82, 133].

There exist a number of interesting results on the approximability of several optimization problems on binomial random graphs; see [96] for a nice survey on random graphs and algorithmics. In the case of layout problems, Turner [237] has shown that for all $\epsilon > 0$, with high probability, it is the case that $n/\text{MINBW}(\mathcal{G}_{n,p_n}) \leqslant 1 + \epsilon$ when $p_n \geqslant c \log n/n$ with $c > 0$. On the other hand, Bollobás [35] proved that for almost all random regular graphs, the optimal edge bisection grows linearly with the number of vertices. The same happens for binomial random graphs, as remarked in [38] and [43]. These results show that the order of growth of the minimal bandwidth and the minimal edge bisection match the trivial upper bounds. So the previous authors conclude that binomial random graphs may not serve well to distinguish really good heuristics from bad ones; for instance, the ones that try to maximize instead of minimizing. From the previous results, a natural question arises: Is this the case for all layout problems defined in Section 1.2?

The goal of this chapter is to answer this question in the positive. Our main result (Theorem 3.1) states that, under certain conditions for p_n, with overwhelming probability, all the layout problems defined in Section 1.2 are approximable within a constant on binomial random graphs by *any* approximation algorithm.

The organization of this chapter is as follows. In Section 3.2 we first introduce the concept of gap between the maximal and the minimal costs of a layout problem for a graph. This is relevant, because any bound on this gap implies a bound to the approximation factor of any approximation algorithm. Then, we define the deterministic class of mixing graphs, for which we are able to give approximation results. This class of mixing graphs is useful since we can prove that, with overwhelming probability, binomial random graphs are mixing graphs. The chapter ends in Section 3.3 with a discussion on the present results and their implication in experimental testing.

3.2 Approximation results

A natural question is to ask whether there is any relation between the approximability of the maximization and minimization versions of our layouts problems. A second question is the possibility of inferring some consequences for the minimization versions from our understanding of the maximization versions. In order to answer these questions, it makes sense to introduce and estimate the gap between the maximal and the minimal costs of a problem for some instance:

Definition 3.2 (Gap). For a layout cost $F \in \{\text{BW}, \text{LA}, \text{CW}, \text{MC}, \text{SC}, \text{VS}, \text{EB}, \text{VB}\}$ and a graph G, we define the *gap* of F on G as the ratio between its minimal and maximal values, that is,

$$\text{GAP}F(G) = \frac{\text{MAX}F(G)}{\text{MIN}F(G)} = 1 + \frac{\text{MAX}F(G) - \text{MIN}F(G)}{\text{MIN}F(G)}.$$

Observe that any bound on the gap of a function F gives a bound on the approximation ratio for any approximation algorithm that computes a layout for a graph G.

We introduce now a class of graphs that captures the properties we need to bound the gaps for our layout problems on binomial random graphs.

Definition 3.3 (Mixing graphs). Let $\epsilon \in (0, \frac{1}{2})$, $\gamma \in (0, 1)$ and set $C_{\epsilon,\gamma} = 3(1 + \ln 3)(\epsilon\gamma)^{-2}$. Consider a sequence $(c_n)_{n\in\mathbb{N}}$ such that $C_{\epsilon,\gamma} \leqslant c_n \leqslant n$ for all $n \geqslant n_0$ for some natural n_0. A graph $G = (V, E)$ with $|V| = n$ and $|E| = m$ is said to be (ϵ, γ, c_n)-*mixing* if

$$m \leqslant (1 + \gamma)\tfrac{1}{2} n c_n, \tag{3.1}$$

and if, for any two disjoint subsets $A, B \subset V$ such that $|A| \geqslant \epsilon n$ and $|B| \geqslant \epsilon n$, it is the case that

$$\left|\theta(A,B) - \frac{c_n}{n}|A||B|\right| \leqslant \gamma\frac{c_n}{n}|A||B|, \tag{3.2}$$

where $\theta(A, B)$ denotes the number of edges in E having one endpoint in A and another in B.

Observe that condition (3.2) is equivalent to

$$1 - \gamma \leqslant \left.\frac{\theta(A,B)}{|A||B|}\right/\frac{c_n}{n} \leqslant 1 + \gamma,$$

which establishes the maximal variation between the density of the cutting edges and the sizes of the subsets of vertices.

Mixing graphs are strongly related to expander graphs. From the many variations of the definition of expander graphs, let us reproduce the one given in [5]: A graph $G = (V, E)$ is an (n, d, c)-*expander* if it has n vertices, its degree is d, and for every set $W \subset V$ of cardinality $|W| \leqslant \frac{1}{2}|V|$, the inequality $|\theta(W, V \setminus W)| \geqslant c|W|$ holds. The main differences in the two definitions are that expanders are required to be bounded-degree graphs and the expansion condition has to be satisfied for all not too large subsets W of vertices, while the mixing condition only involves sets of vertices with size $\Theta(n)$.

Expander graphs have the property that the largest and second largest eigenvalue of their adjacency matrix are well-separated. This is a sufficient condition for a graph to have good mixing properties, as the following lemma asserts:

Lemma 3.1. Let $G = (V, E)$ be a regular graph of degree d, let λ be the second largest eigenvalue of the adjacency matrix of G. Then for every ϵ, γ such that $\epsilon\gamma > \lambda/d$, G is (ϵ, γ, d)-mixing.

Proof. The Expander Mixing Lemma (see Chapter 1 of [5]) states that if G is a d-regular graph and λ is the second largest eigenvalue of the adjacency matrix of G, then for every two subsets of vertices A and B,

$$\left|\theta(A,B) - \frac{d}{n}|A||B|\right| \leqslant \lambda\sqrt{|A||B|}.$$

If we have $|A| \geqslant \epsilon n$, $|B| \geqslant \epsilon n$, and $\epsilon\gamma > \lambda/d$, then we get

$$\left|\theta(A,B) - \frac{d}{n}|A||B|\right| \leqslant \lambda\frac{|A||B|}{\epsilon n} \leqslant \gamma\frac{d}{n}|A||B|.$$

Moreover, as G is d-regular, $|E| = \frac{1}{2}dn$. It follows that, under the hypothesis of the lemma, G is (ϵ, γ, d)-mixing. ☐

The best known explicit construction of expander graphs with strong separation between degree and second largest eigenvalue is due to Lubotzky, Phillips and Sarnak [174]. Their construction yields regular graphs of degree d where the second largest eigenvalue is less than $2\sqrt{d}$. So, taking $d = O(\epsilon^{-2}\gamma^{-2})$, the construction of [174] gives (ϵ, γ, d)-mixing graphs for every ϵ and γ. See also [54] for additional discussions on mixing graphs, explicit constructions and other applications.

In this chapter, our interest in mixing graphs is motivated by the fact that, for an appropriate choice of the sequence $(c_n)_{n\in\mathbb{N}}$, with overwhelming probability, a binomial random graph in the $\mathcal{G}_{n,c_n/n}$ model is mixing, as will be proved below in Lemma 3.2, and the fact that the layout problems defined in Chapter 1 are easy to approximate on mixing graphs, as will be proved in Lemma 3.4. Together, these two results imply that layout problems are easy to approximate on binomial random graphs.

Lemma 3.2. Let $\epsilon \in (0, \frac{1}{2})$, $\gamma \in (0,1)$ and set $C_{\epsilon,\gamma} = 3(1+\ln 3)(\epsilon\gamma)^{-2}$. Consider a sequence $(c_n)_{n\in\mathbb{N}}$ such that $C_{\epsilon,\gamma} \leqslant c_n \leqslant n$ for all $n \geqslant n_0$ for some natural n_0. Then, for all $n \geqslant n_0$, binomial random graphs drawn from $\mathcal{G}_{n,p_n}$ with $p_n = c_n/n$ are (ϵ, γ, c_n)-mixing with probability at least $1 - 2^{-\Omega(n)}$.

Proof. Let $n \geqslant n_0$ and consider a binomial random graph $G = (V, E)$ drawn from $\mathcal{G}_{n,p_n}$. Let M be a random variable counting its number of edges. We estimate first the probability that condition (3.1) fails. The expected number of edges of G is $\mathbf{E}[M] = \frac{1}{2}(n-1)c_n$. Applying Chernoff's bounds (Theorem A.9) we get,

$$\begin{aligned}\mathbf{Pr}\left[M > (1+\gamma)\tfrac{1}{2}nc_n\right] &\leqslant \mathbf{Pr}\left[M > (1+\gamma)\mathbf{E}[M]\right]\\ &\leqslant \exp\left(-\tfrac{1}{3}\gamma^2\mathbf{E}[M]\right)\\ &\leqslant \exp\left(-\tfrac{1}{6}\gamma^2(n-1)c_n\right) \leqslant 2^{-\Omega(n)}, \qquad (3.3)\end{aligned}$$

by the hypothesis $c_n \geqslant C_{\epsilon,\gamma} > 1$.

We estimate now the probability that condition (3.2) does not hold. Consider any two disjoint sets $A, B \subset V$ such that $|A|, |B| \geqslant \epsilon n$. There are $k = |A||B|$ possible edges having an endpoint in A and an endpoint in B. Order these k possible edges. Let $X_1, \ldots, X_k$ be the Bernoulli random variables such that $X_i = 1$ if the i-th of such edges is in the graph and $X_i = 0$ otherwise. The probability of existence of the i-th edge is c_n/n. Let μ_n be the expectation of $\sum_{i=1}^k X_i$. By linearity of the expectation,

$$\mu_n = \mathbf{E}\left[\sum_{i=1}^k X_i\right] = \sum_{i=1}^k \mathbf{E}[X_i] = |A||B|c_n/n.$$

Applying again Chernoff's bounds we have that,

$$\mathbf{Pr}\left[(1-\gamma)\mu_n < \sum_{i=1}^{k} X_i < (1+\gamma)\mu_n\right] \geqslant 1 - 2\exp\left(-\tfrac{1}{3}\gamma^2\mu_n\right).$$

Any vertex can either be in A, in B, or not in A and not in B. Therefore, there are at most 3^n choices for the sets A and B. By Boole's inequality (Theorem A.1) it follows that the probability that condition (3.2) is not satisfied is at most

$$3^n \cdot 2\exp\left(-\tfrac{1}{3}\gamma^2\mu_n\right) \leqslant \exp\left(\ln 2 + n\ln 3 - \tfrac{1}{3}\gamma^2\epsilon^2 n c_n\right) \leqslant 2^{-\Omega(n)} \tag{3.4}$$

by the hypothesis $c_n \geqslant C_{\epsilon,\gamma} = 3(1+\ln 3)(\epsilon\gamma)^{-2}$.

Overall, by Boole's inequality, the failure probability of the mixing conditions can be estimated by the sum of their individual failure probabilities, given by Equations (3.3) and (3.4), which is $2^{-\Omega(n)}$. □

To prove our next result, we need the following technical lemma.

Lemma 3.3. Let G be an arbitrary graph with n vertices and m edges. Then,

$$\textsc{avgla}(G) = \tfrac{1}{3}m(n^2-1)/n \quad \text{and} \quad \textsc{avgmc}(G) = \tfrac{1}{3}m(n^2-3n+2)/n.$$

Proof. By definition,

$$\begin{aligned}\textsc{avgla}(G) &= 2\sum_{i=1}^{n-1}\sum_{uv\in E}\mathbf{Pr}\left[\varphi(u)\leqslant i\right]\cdot\mathbf{Pr}\left[\varphi(v)>i\right]\\ &= 2m\sum_{i=1}^{n-1}\left(\frac{i}{n}\cdot\frac{n-i}{n}\right) = \tfrac{1}{3}m(n^2-1)/n\end{aligned}$$

and

$$\begin{aligned}\textsc{avgmc}(G) &= 2\sum_{i=1}^{n-1}\sum_{uv\in E}\mathbf{Pr}\left[\varphi(u)< i\right]\cdot\mathbf{Pr}\left[\varphi(v)>i\right]\\ &= 2m\sum_{i=1}^{n-1}\left(\frac{i-1}{n}\cdot\frac{n-i}{n}\right) = \tfrac{1}{3}m(n^2-3n+2)/n,\end{aligned}$$

which prove the lemma. □

The following result states that, for the considered layout problems, the gap is bounded on mixing graphs.

Lemma 3.4. Let $\epsilon \in (0, \frac{1}{6})$, $\gamma \in (0,1)$ and set $C_{\epsilon,\gamma} = 3(1+\ln 3)(\epsilon\gamma)^{-2}$. Consider a sequence $(c_n)_{n\in\mathbb{N}}$ such that $C_{\epsilon,\gamma} \leqslant c_n \leqslant n$ for all $n \geqslant n_0$ for some natural n_0. Let $n \geqslant n_0$ and let $G = (V,E)$ be any (ϵ, γ, c_n)-mixing graph with $|V| = n$. Then,

$$
\begin{aligned}
\textsc{gapcw}(G) &\leqslant 2(\gamma+1)/(1-\gamma) + O\left(1/n\right), && (3.5)\\
\textsc{gapeb}(G) &\leqslant 2(\gamma+1)/(1-\gamma) + O\left(1/n\right), && (3.6)\\
\textsc{gapla}(G) &\leqslant (\gamma+1)/(1-6\epsilon)(1-\gamma), && (3.7)\\
\textsc{gapmc}(G) &\leqslant (\gamma+1)/(1-12\epsilon)(1-\gamma), && (3.8)\\
\textsc{gapvs}(G) &\leqslant 1/(\tfrac{1}{2}-\epsilon) + O\left(1/n\right), && (3.9)\\
\textsc{gapvb}(G) &\leqslant 1/(\tfrac{1}{2}-\epsilon) + O\left(1/n\right), && (3.10)\\
\textsc{gapsc}(G) &\leqslant 1/(1-4\epsilon), && (3.11)\\
\textsc{gapbw}(G) &\leqslant 1/(1-2\epsilon) + O\left(1/n\right). && (3.12)
\end{aligned}
$$

Proof. Let $m = |E|$. To prove (3.5), consider any layout φ of G. Take $A = \left\{u \in V \ :\ \varphi(u) \leqslant \left\lfloor \frac{1}{2}n \right\rfloor\right\}$ and $B = V \setminus A$. Then $|A|, |B| \geqslant \frac{1}{2}n - 1 \geqslant \epsilon n$. So,

$$
\begin{aligned}
\textsc{cw}(\varphi, G) &\geqslant \theta\left(\left\lfloor \tfrac{1}{2}n \right\rfloor \varphi, G\right) \geqslant \theta(A,B)\\
&\geqslant (1-\gamma)\frac{c_n}{n}|A||B| \geqslant (1-\gamma)\frac{c_n}{n}\left(\tfrac{1}{2}n - 1\right)^2.
\end{aligned}
$$

Therefore, $\textsc{mincw}(G) \geqslant (1-\gamma)c_n(\frac{1}{2}n-1)^2/n$. The number of edges of G is an upper bound on $\textsc{maxcw}(G)$. As G is a (ϵ,γ,c_n)-mixing graph, $\textsc{maxcw}(G) \leqslant |E| \leqslant (1+\gamma)\frac{1}{2}(n-1)c_n$. So, the computation of the gap yields $\textsc{gapcw}(\varphi, G) \leqslant 2\frac{\gamma+1}{1-\gamma} + O\left(n^{-1}\right)$. The proof of (3.6) is identical.

To prove (3.7), let φ be any layout of G. We have

$$
\textsc{la}(\varphi, G) = \sum_{i=1}^{n} \theta(i,\varphi,G) \geqslant \sum_{i=\epsilon n}^{(1-\epsilon)n} \theta(i,\varphi,G) \geqslant (1-\gamma)\frac{c_n}{n}\sum_{i=\epsilon n}^{(1-\epsilon)n} i(n-i).
$$

On the other hand, the sum of the cuts at the first ϵn and of the last $(1-\epsilon)n$ positions is, for each one, not larger than $\epsilon n m$. So,

$$
\begin{aligned}
\textsc{la}(\varphi, G) &= \sum_{i=1}^{n} \theta(i,\varphi,G) \leqslant 2\epsilon n m + \sum_{i=\epsilon n}^{(1-\epsilon)n} \theta(i,\varphi,G)\\
&\leqslant 2\epsilon n m + (1+\gamma)\frac{c_n}{n}\sum_{i=\epsilon n}^{(1-\epsilon)n} i(n-i).
\end{aligned}
$$

Therefore, making $S = c_n n^{-1}\sum_{i=\epsilon n}^{(1-\epsilon)n} i(n-i)$, we have

$$
\textsc{minla}(G) \geqslant (1-\gamma)S,
$$

$$\text{MAXLA}(G) \leqslant 2\epsilon nm + (1+\gamma)S.$$

By Lemma 3.3 we have $\text{AVGLA}(G) \geqslant m(n+1)/3$, and therefore, there is some layout φ satisfying $\text{LA}(\varphi, G) \geqslant m(n+1)/3$. So, $2\epsilon nm + (1+\gamma)S \geqslant m(n+1)/3 > mn/3$ and thus $2\epsilon nm \leqslant \frac{6\epsilon}{(1-6\epsilon)}(1+\gamma)S$. Hence,

$$\text{GAPLA}(G) \leqslant 1 + \frac{\frac{6\epsilon}{(1-6\epsilon)}(1+\gamma)S + 2\gamma S}{(1-\gamma)S} \leqslant \frac{\gamma+1}{(6\epsilon-1)(\gamma-1)}.$$

The proof of (3.8) is similar; let φ be any layout of G. We have

$$\begin{aligned}\text{MC}(\varphi, G) = \sum_{i=1}^{n-1} \zeta(i, \varphi, G) &\geqslant \sum_{i=\epsilon n+1}^{(1-\epsilon)n-1} \zeta(i, \varphi, G) \\ &\geqslant (1-\gamma)\frac{c_n}{n} \sum_{i=\epsilon n+1}^{(1-\epsilon)n-1} (i-1)(n-i).\end{aligned}$$

For the lower bound, we have

$$\begin{aligned}\text{MC}(\varphi, G) \leqslant \sum_{i=1}^{n-1} \zeta(i, \varphi, G) &\leqslant 2(\epsilon n+1)m + \sum_{i=\epsilon n+1}^{(1-\epsilon)n-1} \zeta(i, \varphi, G) \\ &\leqslant 2\epsilon nm + (1+\gamma)\frac{c_n}{n} \sum_{i=\epsilon n+1}^{(1-\epsilon)n-1} (i-1)(n-i),\end{aligned}$$

where the last inequality holds because $\zeta(1, \varphi, G) = \zeta(n, \varphi, G) = 0$ for any layout φ. Therefore, making $T = c_n n^{-1} \sum_{i=\epsilon n+1}^{(1-\epsilon)n-1} (i-1)(n-i)$, we have

$$\begin{aligned}\text{MINMC}(G) &\geqslant (1-\gamma)T, \\ \text{MAXMC}(G) &\leqslant 2\epsilon nm + (1+\gamma)T.\end{aligned}$$

As $\text{AVGMC}(G) = m(n-5)/3$, there is a layout that gives at least this value. So, $2\epsilon nm + (1+\gamma)T \geqslant m(n-5)/3 > mn/6$ and thus $2\epsilon nm \leqslant \frac{12\epsilon}{(1-12\epsilon)}(1+\gamma)T$. As a consequence,

$$\text{GAPMC}(G) \leqslant 1 + \frac{\frac{12\epsilon}{(1-12\epsilon)}(1+\gamma)T + (1+\gamma)T)}{(1-\gamma)T} \leqslant \frac{\gamma+1}{(12\epsilon-1)(\gamma-1)}.$$

To prove (3.9), consider any layout φ of G. Notice that in a (ϵ, γ, c_n)-mixing graph there cannot be ϵn vertices on the left of i and ϵn vertices on the right of i without any connection. Thus, it holds that $\delta(i,\varphi,G) \geqslant i - \epsilon n$ for every $\epsilon n < i < (1-\epsilon)n$. As a consequence,

$$\textsc{vs}(\varphi,G) \geqslant \delta(\lfloor \tfrac{1}{2}n \rfloor, \varphi, G) \geqslant \lfloor \tfrac{1}{2}n \rfloor - \epsilon n \geqslant \left(\tfrac{1}{2} - \epsilon\right) n - 1$$

and therefore $\textsc{minvs}(G) \geqslant \left(\frac{1}{2} - \epsilon\right) n - 1$. Obviously $\textsc{maxvs}(G) \leqslant n$, so $\textsc{gapvs} \leqslant \left(\frac{1}{2} - \epsilon\right)^{-1} + O\left(n^{-1}\right)$. The proof of (3.10) is similar.

To prove (3.11), consider again any layout φ of G. We have,

$$\begin{aligned}\textsc{sc}(\varphi,G) &= \sum_{i=1}^{n-1} \delta(i,\varphi,G) \geqslant \sum_{i=\epsilon n}^{(1-\epsilon)n} \delta(i,\varphi,G) \geqslant \sum_{i=\epsilon n}^{(1-\epsilon)n} (i - \epsilon n) \\ &\geqslant \tfrac{1}{2}n^2 - 2\epsilon n^2 + \tfrac{1}{2}n(1-\epsilon) + 3\epsilon^2 n^2 \geqslant \tfrac{1}{2}n^2 - 2\epsilon n^2.\end{aligned}$$

Therefore, $\textsc{minsc}(G) \geqslant \frac{1}{2}n^2 - 2\epsilon n^2$. Moreover,

$$\textsc{sc}(\varphi,G) = \sum_{i=1}^{n-1} \delta(i,\varphi,G) \leqslant \sum_{i=1}^{n-1} i \leqslant (n-1)\tfrac{1}{2}n \leqslant \tfrac{1}{2}n^2.$$

Therefore, $\textsc{maxsc}(G) \leqslant \frac{1}{2}n^2$. Thus, $\textsc{gapsc} \leqslant (1-4\epsilon)^{-1}$.

Finally, let us prove (3.12). Notice that $\textsc{maxbw}(G) < n$. Consider any layout φ of G. As G is a (ϵ,γ,c_n)-mixing graph, there must be some edge between the $\lceil \epsilon n \rceil$ first vertices of φ and the $\lfloor (1-\epsilon)n \rfloor$ last vertices of φ. The length of this edge must be at least $(1-2\epsilon)n-4$, and so $\textsc{minbw}(G) \geqslant (1-2\epsilon)n-4$. Thus $\textsc{gapbw}(G) \leqslant (1-2\epsilon)^{-1} + O\left(n^{-1}\right)$. □

The combination of Lemmas 3.2 and 3.4 implies our main result:

Theorem 3.1. Let $\epsilon \in (0, \frac{1}{6})$, $\gamma \in (0,1)$ and set $C_{\epsilon,\gamma} = 3(1+\ln 3)(\epsilon\gamma)^{-2}$. Consider a sequence $(c_n)_{n\in\mathbb{N}}$ such that $C_{\epsilon,\gamma} \leqslant c_n \leqslant n$ for all $n \geqslant n_0$ for some natural n_0. Then, with overwhelming probability, the problems Bandwidth, MinLA, Cutwidth, ModCut, SumCut, VertSep, EdgeBis and VertBis can be approximated within a constant on binomial random graphs $\mathcal{G}_{n,p_n}$, where $p_n = c_n/n$.

3.3 Conclusion

This chapter has considered the approximation of several layout problems on binomial random graphs. Theorem 3.1 states that, with overwhelming probability, all the layout problems defined in Section 1.2 are easily approximable within a constant on binomial random graphs $\mathcal{G}_{n,p_n}$ such that $p_n = c_n/n$ and $c_n \geqslant C_{\epsilon,\gamma} > 1$. The reason is that the ratio between the maximal value and the minimal value of a layout cost on such a graph is bounded by a constant. As said, similar results already existed for the BANDWIDTH and EDGEBIS problems [38, 43, 237]. An important contribution of this chapter is to have extended this kind of result to several other layout problems using a *unique* framework involving "mixing graphs."

The approximation ratios obtained are very tight: they can be arbitrarily close to 1 in the case of the BANDWIDTH, MINLA, SUMCUT and MODCUT problems, and arbitrarily close to 2 in the case of the remaining problems. It is also remarkable that our method is valid both for sparse and dense binomial random graphs, though in the case of sparse graphs, we must ensure the existence of a giant component. However, as mentioned in Section 1.7, there exist better approximation results for dense graphs.

One of our motivations to study binomial random graphs for layout problems was to enable us to analyze heuristics. Unfortunately, our results show that *any* algorithm computing a layout, no matter how good or bad, will perform rather well on binomial random graphs, pointing out that such an evaluation may be unworthy for layout problems.

An overview of the results in this chapter in the weaker case $p_n = c/n$ was presented at the *Fifth Czech-Slovak International Symposium on Combinatorics, Graph Theory, Algorithms and Applications (Prague,* 1998*)*; all the results on this chapter will appear in *Discrete Mathematics* [70].

A recent result due to Luczak and McDiarmid [175] improves our result for EDGEBIS: They show that if $c > \ln 4$, then $\text{MINEB}(\mathcal{G}_{n,c/n}) = \Theta(n)$ with high probability; while if $c < \ln 4$, then $\text{MINEB}(\mathcal{G}_{n,c/n}) = 0$ with high probability. Their techniques could possibly be applied to strengthen the results given in this chapter.

4

Layout Problems and Unit Disk Graphs

4.1 Introduction

An important characteristic of layout applications is the specificity of their instances. As seen in Chapter 1, most layout problems are used in specific domains. Therefore, it makes sense to try to take advantage on the kind of inputs appearing in these domains. In particular, in routing and circuit design applications, input graphs tend to have some *geometrical structure* and are likely to be very *sparse*. In this chapter, we are concerned with unit disk graphs and grid graphs, which have been proposed to model such situations. For these classes of graphs, not much is known, particularly on layout problems.

Let us introduce unit disk graphs and some of their basic properties. Unit disk graphs are defined as follows:

Definition 4.1 (Unit disk graphs). Given a norm $\|\cdot\|$, a *unit disk* with center $\mathbf{x}$ is the set of points $\mathbf{y}$ such that $\|\mathbf{x}-\mathbf{y}\| \leqslant 1$. A graph is said to be a *unit disk graph* if each vertex can be mapped to a unit disk in the plane such that two distinct vertices are adjacent in the graph if and only if their corresponding disks intersect.

Unit disk graphs arise as a simplified model for several applications. These include VLSI circuit design, cluster analysis, random test case (benchmarking), molecular graphics, decoding of noisy data, radio frequency assignment, broadcast networks, mobile phones and ship to ship communications [41, 42, 51, 131]. A good source of information on unit disk graphs is Chapter 3 of Breu's PhD thesis [41].

According to the terminology of Clark, Colbourn and Johnson [51], Definition 4.1 corresponds to the *intersection model* for unit disk graphs. Another model for unit disk graphs is the following: for n equal-sized circles in the plane, form a graph with n vertices corresponding to the n circles and add an edge between two vertices if one of the circles contains the other's center. This is the *containment model* for unit disk graphs. Yet another model is formed by n points in the plane, which correspond to n vertices, and there is an edge between two different vertices if and only if the distance between their corresponding points is at most some specified bound d. This is the *proximity model* of unit disk graphs. All these models are equivalent [51]. Figure 4.1 illustrates the equivalence and Figure 4.2 summarizes how to transform a model to another.

A *realization* of a unit disk graph is a mapping between its vertices and the structures of any of the previous three models. The recognition problem for unit disk graphs (i.e., given a graph, determine if it has a realization) is **NP**-hard [42]. As a consequence, it is customary in practice to assume that a realization of the graph is given [131, 137, 165]. The following reasons lead to justify this assumption:

- As said, the recognition problem is **NP**-hard.
- For most applications, the graph is implicitly specified using one of the three models.
- In the design and analysis of VLSI circuits, unit disk graphs are specified hierarchically: Huge circuits often share hierarchical regular structures that make possible to succinctly describe their designs [131].

Some **NP**-complete problems for general graphs remain **NP**-complete even when restricted to unit disk graphs. These include CHROMATIC NUMBER, INDEPENDENT SET, VERTEX COVER, CONNECTED DOMINATING SET, HAMILTONIAN CIRCUIT and STEINER TREE [51]. However, for the problems where approximation algorithms exist, the approximation ratios on unit disk graphs are better than on general graphs [41, 131]. On the other hand, the MAX CLIQUE problem on unit disk graphs can be solved in polynomial time, provided that a realization is given as input [51]. A remarkable fact is that the complexities of all these problems on unit disk graphs and planar graphs with maximum degree 4 seem to agree. In fact, Breu conjectures in his thesis:

Conjecture 4.1 ([41]). If a problem Π is **NP**-complete for planar graphs with maximum degree 4, then Π is **NP**-complete for unit disk graphs, too.

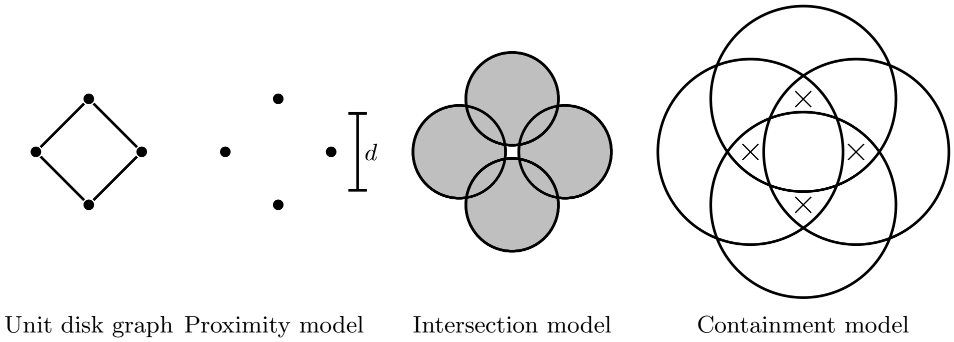


Figure 4.1: Unit disk graphs: Models in l_2.

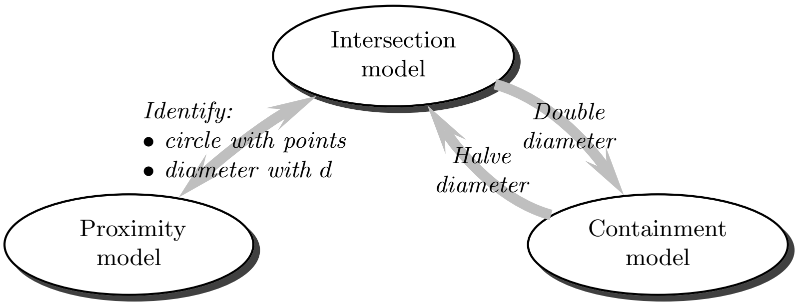


Figure 4.2: Unit disk graphs: Converting from one model to another.

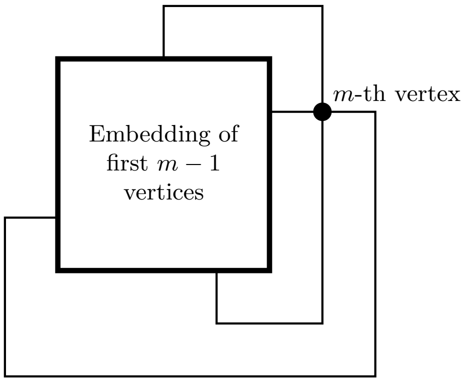


Figure 4.3: Embedding the m-th vertex in Valiant's algorithm (figure reproduced from [239]).

This conjecture is based on the existence of a "plan of attack" for reducing a problem restricted to planar graphs with maximum degree 4 to the same problem restricted to unit disk graphs, based on embedding the planar graph on a grid and simulating its edges by "clever" strings of disks. In order to embed any planar graph with maximum degree 4 on a grid without crossovers of edges, the plan of attack uses *Valiant's algorithm* [239]:

1. Start with any embedding of the graph in the plane.

2. Strip off one vertex at a time from the perimeter of the embedding.

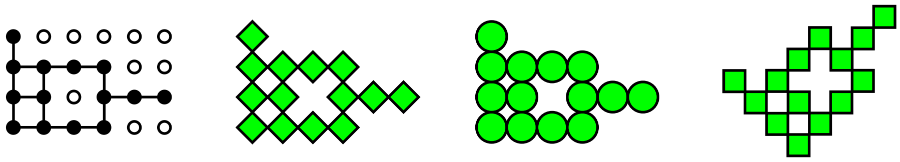
3. Embed these vertices in reverse order into the grid. The m-th vertex is typically embedded as shown in Figure 4.3.

Let us now present grid graphs. Grid graphs are defined as follows:

Definition 4.2 (Grid graphs). A graph is said to be a *grid graph* if it is a finite vertex-induced subgraph of the infinite lattice, that is, its vertex set is a subset of $\mathbb{Z}^2$ and two vertices are connected whenever their Euclidean distance is 1.

Notice that any grid graph is a unit disk graph in any l_p norm ($p \in \mathbb{N} \cup \{\infty\}$); Figure 4.4 illustrates it.

A result relating grid graphs and layout problems is an exact polynomial time algorithm for EdgeBis on grid graphs *without* holes by Papadimitriou and Sideri [201]. In the case of rectangular or square grid graphs, more results exist.



(a) Grid graph **(b)** l_1 **(c)** l_2 **(d)** l_∞

Figure 4.4: Any grid graph is a unit disk graph in any l_p norm, $p \in \mathbb{N} \cup \{\infty\}$.

Definition 4.3 (Rectangular and square grids). Given two naturals m and m', the $m \times m'$ *rectangular grid* is the grid graph with vertex set $\{0, \ldots, m-1\} \times \{0, \ldots, m'-1\}$. Given a natural m, the $m \times m$ *square grid* is the grid graph with vertex set $\{0, \ldots, m-1\}^2$. Throughout this thesis, $L_{m,m'}$ denotes an $m \times m'$ rectangular grid and $L_m = L_{m,m}$ denotes an $m \times m$ square grid.

In this chapter we consider layout problems on unit disk graphs, grid graphs and rectangular or square grid graphs. At the light of Conjecture 4.1, it seems natural to investigate whether Breu's plan of attack can be of use to tackle the complexity of some layout problems restricted to unit disk graphs. This issue will be explored in Section 4.2. On the other hand, square grid graphs are a so regular structure, that it makes sense to think out which is their optimal solution for the different layout problems presented in Section 1.2. The optimal values for the EdgeBis and Cutwidth problems on square grids are well known, but the result for MinLA is not so widely known and deserves some attention. This is why in Section 4.3 we will study the optimal solutions for several layout problems on square grids, including new results for the VertSep, SumCut, Bandwidth and VertBis problems. Afterwards, in Section 4.4, we will present tight upper bounds to several layout problems on general grid graphs. Beyond their intrinsic interest, these bounds will be useful in Chapter 5. Finally, this chapter will be closed with an overview of the results, conclusions and open problems.

4.2 Complexity results

In this section, we show that some layout problems remain hard to solve, even when restricted to geometric instances. To do so, we use the unit disk graph model, which also includes grid graphs as a particular case. We shall primarily use Breu's "plan of attack" described in the previous section.

To prove complexity results of layout problems on spatial graphs, all through this section we shall consider their decisional counterparts (see Observation 1.3). In the standard way, we shall use K as the constant parameter in the definition of the decisional versions [98]. All the proofs in this section are valid for any l_p norm ($p \in \mathbb{N} \cup \{\infty\}$), but for readability purposes, the explanation and the figures use the Euclidean norm.

Definition 4.4 (Subdivision and homeomorphism). A graph H is a *subdivision* of a graph G if H can be constructed from G by subdividing some of its edges, that is, replacing an edge by a path of vertices with degree 2. Two graphs are *homeomorphic* if they are subdivisions of the same graph.

Lemma 4.1. Let H be a subdivision of a graph G. Then,

$$\textsc{mincw}(G) = \textsc{mincw}(H).$$

Proof. We prove the statement of the lemma for the insertion of a new vertex w between an edge uv of G. Let φ be an optimal layout of G, without lost of generality assume that $\varphi(u) < \varphi(v)$. Let ϕ be a layout of H that corresponds to insert w just after u. Then, $\textsc{cw}(G, \varphi) = \textsc{cw}(H, \phi)$. To show that ϕ is optimal for H, suppose the contrary, then there exists a layout ϕ' of H such that $\textsc{cw}(H, \phi') < \textsc{cw}(H, \phi)$. Let φ' be the layout obtained by removing w from ϕ'. Then $\textsc{cw}(G, \varphi') \leqslant \textsc{cw}(H, \phi') < \textsc{cw}(H, \phi) \leqslant \textsc{cw}(G, \varphi)$, which is a contradiction to the optimality of φ. $\square$

The following known complexity results are the base for our reductions:

Theorem 4.1 ([185]). Bandwidth remains **NP**-complete even when restricted to caterpillars with at most one hair attached to each vertex of the body.

Theorem 4.2 ([187]). Cutwidth and VertSep remain **NP**-complete even when restricted to planar graphs with maximum vertex degree 3.

Using the above results, we can prove now the following result:

Theorem 4.3. Bandwidth, Cutwidth and VertSep remain **NP**-complete even when restricted to grid graphs (and therefore, even when restricted to unit disk graphs).

Proof. Bandwidth remains **NP**-complete even when restricted to grid graphs because caterpillars with at most one hair attached to each vertex of the body are grid graphs. See Figure 4.5.

We present a reduction from the Cutwidth problem restricted to planar graphs with maximum vertex degree 3 to the Cutwidth problem restricted to

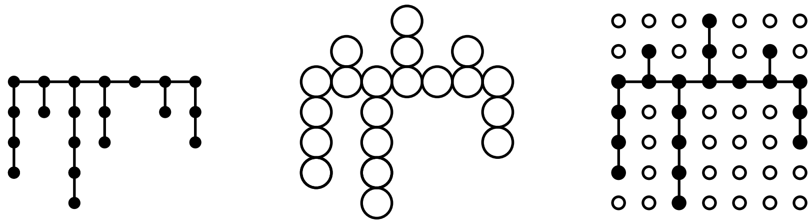


Figure 4.5: Any caterpillar with at most one hair attached to each vertex of the body is a unit disk graph and a grid graph.

grid graphs. Let $\langle G, K\rangle$ be an instance of CUTWIDTH restricted to planar graphs with maximum vertex degree 3. Using Valiant's algorithm (see Section 4.1), we can draw G in such a way that its vertices are located at positions $(6x, 6y)$ for some $x, y \in \mathbb{N}$ and the edges only follow horizontal and vertical paths without crossing. This embedding uses an area polynomial in the size of G. Then, replace each edge by a string of unit disks to produce a grid graph H (see Figure 4.6). By construction, H is a subdivision of G. Therefore we have that $\text{MINCW}(G) \leqslant K$ if and only if $\text{MINCW}(H) \leqslant K$, which proves the result.

Observe that the previous reduction creates graphs with maximum degree 3. Recall that for graphs with maximum degree 3, the SEARCHNB problem is identical to the CUTWIDTH problem [177]. Therefore, as a corollary, we get that SEARCHNB remains **NP**-complete even when restricted to grid graphs. In what follows, we denote by SN the cost for SEARCHNB.

For any graph G, the vertex separation of a homeomorphic image of G is identical to the search number of G [80]. Let us reduce SEARCHNB restricted to planar graphs with maximum vertex degree 3 to VERTSEP restricted to grid graphs using the same transformation that we used for CUTWIDTH. The resulting graph H is a grid graph homeomorphic to the input graph G, so we get $\text{MINVS}(H) \leqslant K$ if and only if $\text{MINSN}(G) \leqslant K$. □

We could not obtain similar results for SUMCUT, MODCUT, MINLA, EDGEBIS or VERTBIS. However, for the bisection problems, we are able to produce a weaker result, which gives an indication of their hardness, based on Papadimitriou and Sideri's conjecture on the **NP**-completeness of the edge bisection problem on planar graphs [201].

Theorem 4.4. If EDGEBIS (resp. VERTBIS) is **NP**-complete even when restricted to planar graphs with maximum vertex degree 4, then EDGEBIS (resp. VERTBIS) is **NP**-complete even when restricted to unit disk graphs.

Proof. Assume that the input graph G is of even order (otherwise, add a disconnected vertex to G). We only give the proof for EDGEBIS, as the proof for

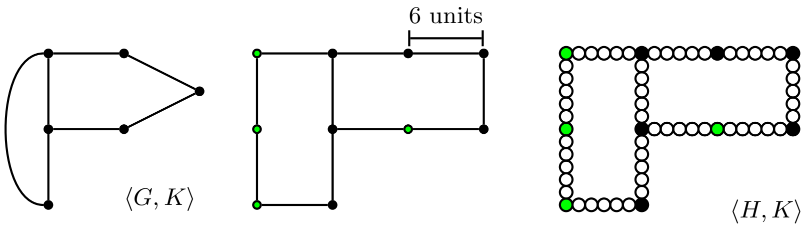


Figure 4.6: Reduction from CUTWIDTH restricted to planar graphs with maximum vertex degree 3 to CUTWIDTH restricted to grid graphs. At left, the input graph; at center, the input graph embedded with Valiant's algorithm; at right, substitution of the edges with paths of disks.

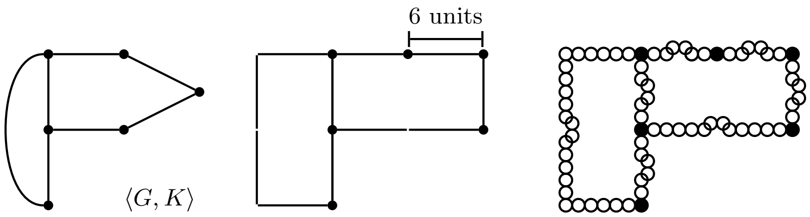


Figure 4.7: Reduction from EDGEBIS restricted to planar graphs with maximum vertex degree 4 to EDGEBIS restricted to unit disk graphs (1). At left, the input graph with $n = 6$ vertices; at center, the input graph embedded with Valiant's algorithm; at right, substitution of the edges with paths of disks with even length.

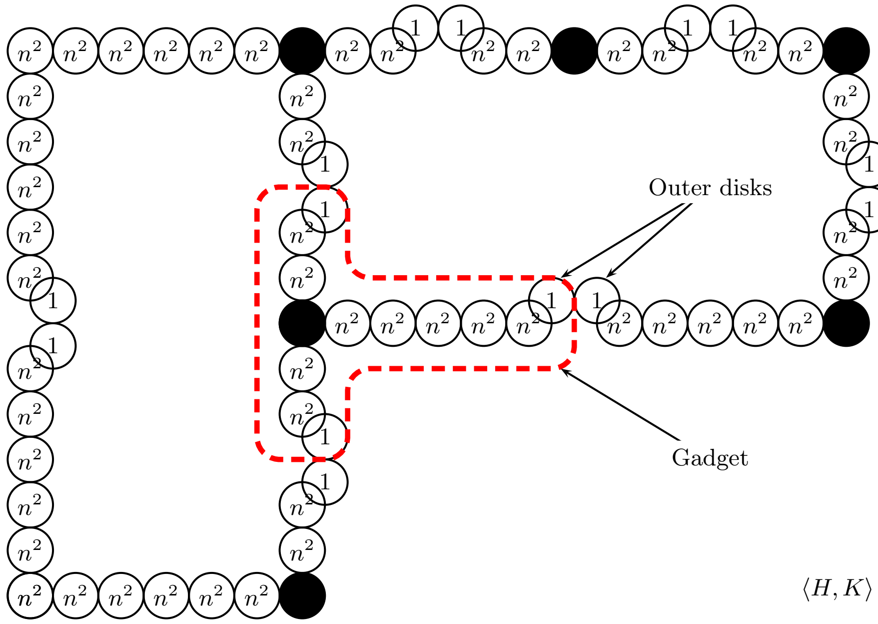


Figure 4.8: Reduction from EdgeBis restricted to planar graphs with maximum vertex degree 4 to EdgeBis restricted to unit disk graphs (2). Non outer disks receive multiplicity n, outer disks get multiplicity 1 and original vertices receive the required multiplicity in order to ensure that all the gadgets contain the same number of disks.

VertBis is identical.

Let $\langle G, K\rangle$ be an instance of EdgeBis where G is a planar graph with n vertices and maximum vertex degree 4. We will reduce it to an instance $\langle H, K\rangle$ of EdgeBis where H is a unit disk graph such that $\text{MINEB}(G) = \text{MINEB}(H)$.

As in the proof of Theorem 4.3, using Valiant's algorithm, we embed G on the plane in such a way that its vertices are located at positions $(6x, 6y)$ for some $x, y \in \mathbb{N}$ and that the edges follow horizontal and vertical paths without crossing. We identify each "original" vertex of the embedding with a unit disk and replace each half edge of length l with a string of disks of length $\left\lfloor \frac{1}{2} l \right\rfloor$. As edges had an odd length, we must join the strings using two additional "outer" disks as shown in Figure 4.7. Therefore, each edge has been replaced by an even number of disks. For each original vertex u in $V(G)$, define its "gadget" as the set of disks that represent its adjacent half edges. Notice that a gadget includes the outer disks, where it ends. We give to each non-outer disk multiplicity n^2, while outer disks retain multiplicity 1. Also, we add multiplicity to the original vertices in such a way that every gadget receives the same amount of disks. The second part of this reduction is depicted in Figure 4.8. Let H be the resulting graph of this reduction, where disks with multiplicity m consist of m different disks on the same position and form a clique. The reduction can be computed in polynomial time.

We need to prove that $\langle G, K \rangle$ is a positive instance of EdgeBis if and only if $\langle H, K \rangle$ is also a positive instance of EdgeBis. We do so by showing that gadgets in H behave as the original vertices in G.

If $\langle G, K \rangle$ is a positive instance of EdgeBis then there exists a bisection B of G such that $\text{EB}(G, B) \leqslant K$. Coloring each gadget of H according to B, the bisection of H coincides with the bisection B and is a legal bisection, in the sense that each gadget has the same number of vertices). Therefore $\langle H, K \rangle$ is a positive instance of EdgeBis.

On the other hand, if $\langle H, K \rangle$ is a positive instance of EdgeBis, we have two cases: When $K > 2n$, as G has maximum degree 4, the bisection width of G cannot exceed $2n$, thus $\langle G, K \rangle$ surely is a positive instance of EdgeBis. For the case $K \leqslant 2n$, let us consider any gadget. Each of the vertices of this gadget must be on the same side of the bisection (otherwise, the bisection width would be larger than $2n$ because of the cliques of size n^2 introduced in H). Taking a bisection of G that coincides with the one given to the gadgets of H, we get that $\langle G, K \rangle$ is a positive instance of EdgeBis. □

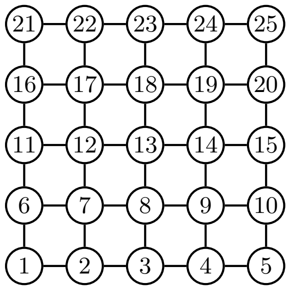
4.3 Optimal layouts for square grids

In this section we consider the optimal solutions of several layout problems on square grids. We start reviewing the results for the EdgeBis, Cutwidth and MinLA problems; afterwards we present new results for the VertSep, SumCut, Bandwidth and VertBis problems.

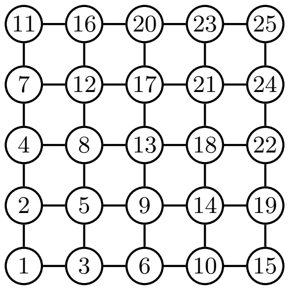
Recall that L_m denotes the $m \times m$ square grid. In the following, we will encounter two different natural ways to order the vertices of square grids: Let u and u' be two distinct vertices of a square grid and let (x, y) be the coordinates of u and (x', y') the coordinates of u'. In the *lexicographic layout*, denoted by φ_L, u precedes u' whenever $y < y'$ or $y = y'$ and $x < x'$. In the *diagonal layout*, denoted by φ_D, u precedes u' whenever $x + y < x' + y'$ or whenever $x + y = x' + y'$ and $x < x'$. The lexicographic and diagonal layouts are illustrated in Figure 4.9(a) and Figure 4.9(b) respectively.

EdgeBis and Cutwidth on square grids. The optimal values for the Edge Bisection and Cutwidth problems on L_m are well known:

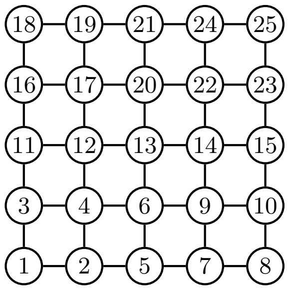
$$\text{MINEB}(L_m) = \text{MINCW}(L_m) = m + (\text{odd } m).$$



(a) Lexicographic layout



(b) Diagonal layout



(c) Muradyan–Piliposjan layout

Figure 4.9: Optimal layouts for 5×5 square grids. The lexicographic layout is optimal for VertSep, Bandwidth, EdgeBis and Cutwidth; the Muradyan–Piliposjan layout is optimal for MinLA; the diagonal layout is optimal for VertSep, VertBis, SumCut and Bandwidth.

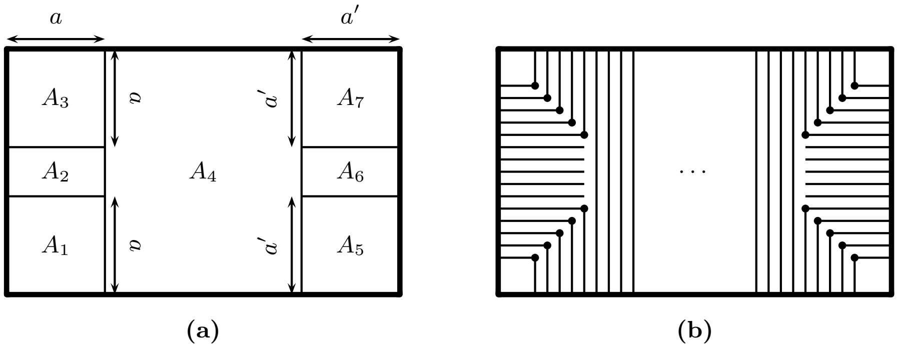


Figure 4.10: Schematic representation of the optimal layout of MINLA$(L_{m,m'})$ for rectangular $m \times m'$ grids (figure reproduced from [189]).

The optimal values can be achieved by the lexicographic layout φ_L. These values play an important role in the analysis of the efficiency of parallel algorithms on meshes of processors [123, 167].

MinLA **on rectangular grids.** Recall from Section 1.5.3 that there exists some disconcert with respect to the solution of MinLA for square grids. Indeed, the optimal layout for MINLA$(L_{m,m'})$ on $m \times m'$ rectangular grids has an interesting solution, well described in [24]. The optimal numbering is schematically shown in Figure 4.10. The numbering starts with the left lower corner of the grid and then fills the areas $A_1, A_2, \ldots, A_7$, where A_1, A_3 are $a \times a$ squares and A_5, A_7 are $a' \times a'$ squares (see Figure 4.10(a)). The values of a and a' must satisfy

$$a, a' \in \left\{ \left\lceil m - \tfrac{1}{2} - \sqrt{\tfrac{1}{2}m^2 - \tfrac{1}{2}m + \tfrac{1}{4}} \right\rceil, \left\lfloor m + \tfrac{1}{2} - \sqrt{\tfrac{1}{2}m^2 - \tfrac{1}{2}m + \tfrac{1}{4}} \right\rfloor \right\}.$$

The way to number the areas is shown in Figure 4.10(b). Each square must be numbered sequentially. A_1 must be filled row after column. A_2 must be filled by consecutive rows, from bottom to top and from left to right. A_3 must be enumerated with the reverse order with respect to A_1. A_4 must be numbered by columns from bottom to top and from left to right. Finally A_5, A_6 and A_7 are filled in the same way. Using this solution,

$$\text{MINLA}(L_{m,m'}) = -\tfrac{2}{3}a^3 + 2ma^2 - \left(m^2 + m - \tfrac{2}{3}\right) a + m'(m^2 + m - 1) - m$$

and

$$\text{MINLA}(L_m) = \tfrac{4-\sqrt{2}}{3}m^3 + O(m^2).$$

An example for the 5×5 square grid is shown in Figure 4.9(b).

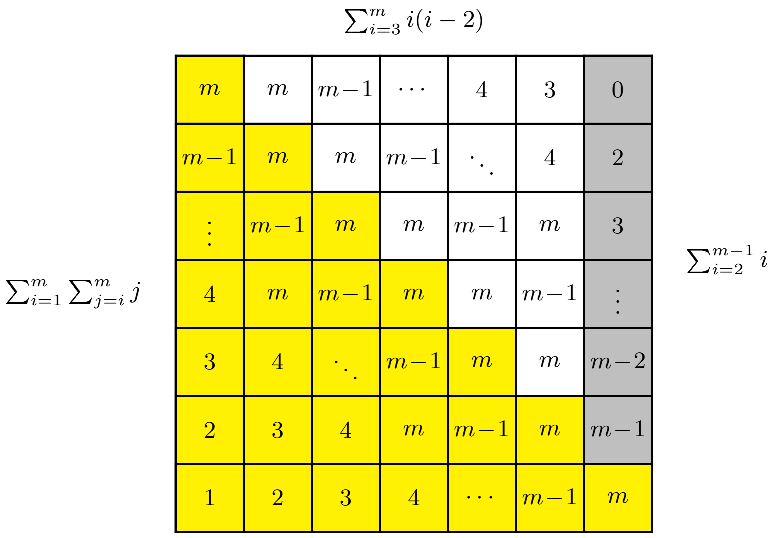


Figure 4.11: Vertex cuts of the diagonal layout φ_D on L_m.

Vertex problems on square grids. The next results concern the optimality of the *diagonal ordering* on L_m for the VERTSEP, SUMCUT, BANDWIDTH and VERTBIS problems. Notice that, for all k, $\delta(k, \varphi_D, L_m) \leqslant m$ (see Figure 4.11).

We begin by presenting a vertex isoperimetric inequality that will help us to characterize the optimal layouts for some problems on square grid graphs L_m. The next Lemma is a special case of Corollary 9 in [37], who in fact prove the d-dimensional version for arbitrary d. Nevertheless, the proof for this particular case will be of interest by itself, as we shall consider its continuous analog latter on in Section 5.3.

Lemma 4.2 (Vertex isoperimetric inequality). Let $m \in \mathbb{N}$. For any layout φ on L_m and any $k \in [m^2]$, it is the case that $\delta(k, \varphi, L_m) \geqslant \delta(k, \varphi_D, L_m)$.

Proof. Given φ and k, let A be the set $L(k, \varphi, L_m)$, and let $\partial_{\text{in}} A$ be the number of boundary elements of A, i.e. elements of A having neighbors in $L_m \backslash A$. Thus $\partial_{\text{in}} A = \delta(k, \varphi, L_m)$.

Let A' be the set in L_m obtained by "pushing each vertical section of A down as far as possible towards the x-axis"; more precisely, setting $S_i(A) = \{j \ : \ (i,j) \in A\}$ for $i \in \{0, \ldots, m-1\}$, let

$$A' = \bigcup_{\substack{i \in \{0,\ldots,m-1\} \\ S_i(A) \neq \emptyset}} \{i\} \times \{0, \ldots, |S_i(A)| - 1\}.$$

Notice that $|A'| = |A|$. Moreover, we claim that $\partial_{\text{in}} A' \leqslant \partial_{\text{in}} A$: Consider first the vertical contribution of the vertices of a column $i \in \{0, \ldots, m-1\}$. If all vertices in column i are in A, then this contribution is zero both in A and A'. Otherwise, at least one vertex participates in the contribution in A, and exactly one vertex participates in the contribution in A'. Consider now the contribution to the right at a column $i \in \{0, \ldots, m-2\}$. Assume that there are r vertices of A in column i and that there are s vertices of A in column $i+1$. If $r \leqslant s$, there will not be any vertex at column i that contributes to the right in A'. Otherwise, there are at least $r-s$ vertices at column i that contribute to the right, but there will be exactly $r-s$ vertices at column i that will contribute to the right in A'. A symmetrical argument holds for the contributions to the left of a column $i \in \{1, \ldots, m-1\}$. So, joining the three contributions together, we have $\partial_{\text{in}} A' \leqslant \partial_{\text{in}} A$.

Let A'' be the set in L_m, obtained by "pushing each horizontal section of A' sideways as far as possible towards the y-axis": setting $S_j(A) = \{i \;:\; (i,j) \in A'\}$ for $j \in \{0, \ldots, m-1\}$, let

$$A'' = \bigcup_{\substack{j \in \{0,\ldots,m-1\} \\ S_j(A') \neq \emptyset}} \{j\} \times \{0, \ldots, |S_j(A')| - 1\}.$$

Then $|A''| = |A'| = |A|$. Moreover, using the same argument as in the previous paragraph, $\partial_{\text{in}} A'' \leqslant \partial_{\text{in}} A' \leqslant \partial_{\text{in}} A$. Furthermore, A'' is a *down-set*, that is, it has the property that for any $\mathbf{x} \in A''$, all vertices of L_m lying directly below or directly to the left of $\mathbf{x}$ are in A''. From now on we assume that A is a down-set. If this was not the case, one could convert it to a down-set A'' with smaller or equal boundary. We consider three cases:

Case 1: $(0, m-1) \notin A$ and $(m-1, 0) \notin A$. Choose the positive integer r so that $r(r-1)/2 < k \leqslant r(r+1)/2$. Then there must be a point $\mathbf{x} = (x, y) \in A$ with $x + y \geqslant r + 1$. Choose such a point $\mathbf{x}$, having neighbors in $L_m \setminus A$ both to its right and above it. Then there is a path of y or more boundary points of A from the bottom of the square to $\mathbf{x}$, and another path of x or more boundary points of A from the left of the square to $\mathbf{x}$, and these paths do not intersect each other except at $\mathbf{x}$. Therefore,

$$\delta(k, \varphi, L_m) = \partial_{\text{in}} A = y + x - 1 \geqslant r.$$

If $r \leqslant m$ then we have $\delta(k, \varphi_D, L_m) = r \leqslant \delta(k, \varphi, L_m)$, while if $r > m$ then we have $\delta(k, \varphi_D, L_m) \leqslant m \leqslant \delta(k, \varphi, L_m)$. So, $\delta(k, \varphi, L_m) \geqslant \delta(k, \varphi_D, L_m)$.

Case 2: $(0, m-1) \in A$ and $(m-1, 0) \in A$. Choose the positive integer r so that $r(r-1)/2 < m^2 - k \leqslant r(r+1)/2$. Then there must be a point $\mathbf{x} = (x, y) \in L_m \backslash A$ with $(m+1-x)+(m+1-y) \geqslant r+1$, that is, $x+y \leqslant 2m+1-r$.

Choose such a point $\mathbf{x}$, having neighbors in A both to its left and above it. Then there is a path of at least $m - y + 1$ boundary points of A from the top of the square to the point just to the left of $\mathbf{x}$, and another path of at least $n - x + 1$ boundary points of A from the right of the square to just below $\mathbf{x}$, and these paths do not intersect. Therefore,

$$\delta(k, \varphi, L_m) = \partial_{\text{in}} A \geqslant 2m - x - y + 2 \geqslant r + 1.$$

If $r < m$ then $\delta(k, \varphi_D, L_m) = r + 1 \leqslant \delta(k, \varphi, L_m)$, while if $r \geqslant m$ then $\delta(k, \varphi_D, L_m) \leqslant m \leqslant \delta(k, \varphi, L_m)$. So, $\delta(k, \varphi, L_m) \geqslant \delta(k, \varphi_D, L_m)$.

Case 3: Only one of the the corners $(m-1, 0)$ or $(0, m-1)$ is in A. In this case we have $\partial_{\text{in}} A = m \geqslant \delta(k, \varphi_D, L_m)$. Again, $\delta(k, \varphi, L_m) \geqslant \delta(k, \varphi_D, L_m)$. □

Using the above isoperimetric inequality, we can assert the optimality of the diagonal ordering for vertex layout problems:

Theorem 4.5. For any natural m, the diagonal ordering is optimal for the VertSep, VertBis, Bandwidth and SumCut problems on a $m \times m$ square grid L_m. Moreover,

$$\begin{aligned} \textsc{minvs}(L_m) &= m, \\ \textsc{minvb}(L_m) &= m, \\ \textsc{minbw}(L_m) &= m, \\ \textsc{minsc}(L_m) &= \tfrac{2}{3}m^3 + \tfrac{1}{2}m^2 - \tfrac{7}{6}m. \end{aligned}$$

Proof. From Lemma 4.2, it follows that φ_D is optimal for the minvs, minvb and minsc costs on L_m. Moreover, $\textsc{minvs}(L_m) = \textsc{minvb}(L_m) = m$.

On the other hand, using Lemma 1.1, we have $m = \textsc{bw}(\varphi_D, L_m) \geqslant \textsc{minbw}(L_m) \geqslant \textsc{minvs}(L_m) = m$.

Finally, decomposing $\textsc{sc}(\varphi_D, L_m)$ as shown in Figure 4.11, we have

$$\textsc{sc}(\varphi_D, L_m) = \sum_{i=1}^{m}\sum_{j=1}^{m} j + \sum_{i=3}^{m} i(i-2) + \sum_{i=2}^{m-1} i = \tfrac{2}{3}m^3 + \tfrac{1}{2}m^2 - \tfrac{7}{6}m,$$

which concludes the proof. □

As $\textsc{minbw}(L_m) = \textsc{minvb}(L_m) = \textsc{minvs}(L_m) = m$, the lexicographic layout is also an optimal layout for the Bandwidth, Vertex Bisection and Vertex Separation problems on square grids:

Corollary 4.1. The lexicographic layout is optimal for the Bandwidth, VertBis and VertSep problems on square grids.

4.4 Upper bounds for grid graphs

In this section we present upper bounds for layout problems on general grid graphs with n vertices. The following lemma is the base to achieve algorithmically these upper bounds.

Lemma 4.3. Let $n \in \mathbb{N}$. For any grid graph L with n vertices, and any $k \in [n]$, there is a layout φ on L such that $\theta(k, \varphi, L) \leqslant 2\sqrt{2n} + 1$.

Proof. We are looking for a subset S of L consisting of k vertices, such that there are at most $2\sqrt{2n}+1$ edges between S and $L \backslash S$. Figure 4.12 sketches this proof.

Let $\alpha > 0$ be a constant, to be chosen later. For $x \in \mathbb{Z}$ let $S_x = \{y \in \mathbb{Z} \,:\, (x, y) \in L\}$ and let $V = \{x \in \mathbb{Z} \,:\, |S_x| \geqslant \alpha\sqrt{n}\}$. For $i \in \mathbb{Z}$, let H_i denote the half-space $(-\infty, i] \times \mathbb{Z}$. Set

$$i_0 = \min\{i \in \mathbb{Z} \,:\, |L \cap H_i| \geqslant k\}.$$

Consider the case $i_0 \notin V$. Then define S to be a set of the form

$$S = L \cap (H_{i_0-1} \cup (\{i_0\} \times (-\infty, j]))$$

with j chosen so that S has precisely k elements.

With this definition of S for $i_0 \notin V$, the number of horizontal edges between S and $L \backslash S$ is at most $|S_{i_0}|$, and hence is at most $\alpha\sqrt{n}$. On the other hand, there is at most one vertical edge between S and $L \backslash S$, so the total number of edges from S to $L \backslash S$ is at most $1 + \alpha\sqrt{n}$ when $i_0 \notin V$.

Now consider the other case $i_0 \in V$. Let $I = [i_1, i_2]$ be the largest integer interval which includes i_0 and is contained in V. Then $i_1 - 1 \notin V$, and $i_2 + 1 \notin V$. Also, as $|V| \leqslant \sqrt{n}/\alpha$, we have $i_2 - i_1 + 1 \leqslant \sqrt{n}/\alpha$. Thus,

$$|L \cap H_{i_1-1}| < k \leqslant |L \cap H_{i_2}|.$$

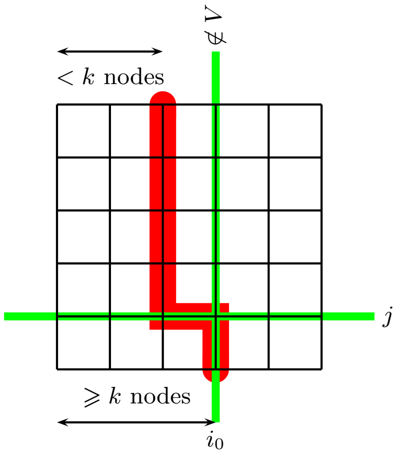
For $j \in \mathbb{Z}$ let $T_j = [i_1, i_2] \times (-\infty, j]$. Choose j_0 so that

$$|L \cap (H_{i_1-1} \cup T_{j_0-1})| < k \leqslant |L \cap (H_{i_1-1} \cup T_{j_0})|,$$

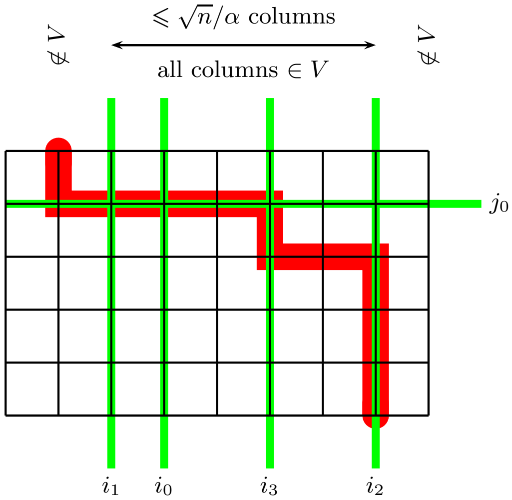
and let S be $L \cap (H_{i_1-1} \cup T_{j_0-1} \cup ([i_1, i_3] \times \{j_0\}))$, with $i_3 \in [i_1, i_2]$ chosen so that S has precisely k elements.

We estimate the number of edges between S and $L \backslash S$ for the case $i_0 \in V$. Since $i_1 - 1 \notin V$ and $i_2 + 1 \notin V$, the number of horizontal edges between S and $L \setminus S$ is at most $2\alpha\sqrt{n} + 1$ (the 1 comes from the cut in i_3). Moreover, since $i_2 - i_1 + 1 \leqslant \sqrt{n}/\alpha$, the number of vertical edges between S and $L \backslash S$ is at most $\sqrt{n}/\alpha$. Combining these estimates we find that there are at most $(2\alpha + 1/\alpha)\sqrt{n} + 1$ edges between S and $L \backslash S$, whether or not $i_0 \in V$.

The minimum value of $2\alpha + 1/\alpha$ is $2\sqrt{2}$ and is achieved at $\alpha = 1/\sqrt{2}$. So, setting $\alpha = 1/\sqrt{2}$, we have the required partition. □



(a) Case where $i_0 \notin V$



(b) Case where $i_0 \in V$

Figure 4.12: Sketch of the proof of Theorem 4.3. The thick line marks the vertices in the cut.

The previous lemma gives the key to prove the following upper bounds:

Theorem 4.6. Let $n \in \mathbb{N}$. For any grid graph L with n vertices,

$$\begin{aligned}
\text{MINEB}(L) &\leqslant 2\sqrt{2n}+1,\\
\text{MINCW}(L) &\leqslant 14\sqrt{n},\\
\text{MINVB}(L) &\leqslant 2\sqrt{2n}+1,\\
\text{MINVS}(L) &\leqslant 14\sqrt{n},\\
\text{MINLA}(L) &\leqslant 14n\sqrt{n},\\
\text{MINMC}(L) &\leqslant 14n\sqrt{n},\\
\text{MINSC}(L) &\leqslant 14n\sqrt{n}.
\end{aligned}$$

Proof. Taking $k = \lfloor \frac{1}{2}n \rfloor$ in Lemma 4.3 we get the upper bound on the cost of the minimal edge bisection.

In order to prove the CW result, suppose first that $n = 2^k$ for a natural k. The proof is based on recursive bisection, with the cut size guaranteed by Lemma 4.3. The following recurrence gives an upper bound to the cost of MINCW(L), where $f(k)$ is the MINCW cost of a grid with 2^k vertices,

$$f(k) \leqslant \begin{cases} 0 & \text{if } k = 0,\\ 2^{3/2}2^{k/2}+1+f(k-1) & \text{otherwise.}\end{cases}$$

Its resolution yields

$$f(k) \leqslant \sum_{j=1}^{k}(2^{3/2}2^{j/2}+1) \leqslant 4(2^{1/2}+1)(2^{k/2}-1)+k.$$

We can drop the assumption that $n = 2^m$, by taking k so that $n \leqslant 2^k < 2n$, and adding extra points until one has a set of size 2^k. By monotonicity (see Lemma 1.3), this process does not reduce the MINCW cost, so

$$\begin{aligned}
\text{MINCW}(L) &\leqslant 2^{5/2}(\sqrt{2}+1)\sqrt{n}+(\log n+1)-4(\sqrt{2}+1)\\
&\leqslant 13.657\sqrt{n}+\log n-8.
\end{aligned}$$

Since for any $x > 0$ we have $(\log x - 8)/\sqrt{x} < 0.067$, the above bound for MINCW(L) is at most $14\sqrt{n}$ for all n.

The remaining bounds are obtained using Lemma 1.1. □

It is important to observe that in the case of L_m (where $n = m^2$), the above bounds are within a constant of their optimal costs (see Theorem 4.5 and [167, 182, 189]). This shows that these bounds are best possible, disregarding a constant factor.

4.5 Conclusion

In this chapter we have considered layout problems on unit disk graphs and grid graphs. First of all we have shown that the decisional versions of the BANDWIDTH, CUTWIDTH and VERTSEP problems remain **NP**-complete even when restricting their inputs to unit disk graphs. Our results reaffirm the close relation between disk graphs and planar graphs as it was already noticed by Clark, Colbourn and Johnson [51]. We have also presented a weaker result, which asserts that EDGEBIS and VERTBIS are **NP**-complete on unit disk graphs if they are **NP**-complete on planar graphs with maximum degree 4. Papadimitriou and Sideri conjecture that bisection of planar graphs is **NP**-complete [201]. We suppose that SUMCUT, MINLA and MODCUT have the same behavior, but in this case, the link with planar graphs with restricted degree is not clear. These complexity results will appear in *Journal of Algorithms* [66].

Afterwards, we have considered layout problems on square grid graphs. The optimal solutions for EDGEBIS, CUTWIDTH and MINLA were already known. Our new results prove the optimality of the diagonal layout for the BANDWIDTH, VERTSEP, SUMCUT, and VERTBIS problems on square grids. An overview of optimal values for layout problems on square grids is shown in Table 4.1. These results were presented in *Computing and Combinatorics — 5th Annual International Conference* COCOON99 *(Tokyo* 1999*)* [64]. The optimal layout for the MODCUT problem on square grids remains an open problem: Is the Muradyan–Piliposjan layout optimal for MODCUT?

Finally, we have presented several worst case upper bounds for arbitrary grid graphs. Our bounds only involve the number of vertices in the graph and can be achieved algorithmically. They are best possible disregarding a constant factor. An overview of these results is shown in Table 4.2. Part of these upper bounds were presented in [64]. A non-trivial upper bound on the Bandwidth of an arbitrary grid graph remains to be found; we suspect that $\textsc{minbw}(L) = O(\sqrt{n})$ for any grid graph L with n vertices.

This chapter has dealt with layout problems and unit disk graphs and grid graphs *in the worst case*. These results will be of use in the following chapter, which treats random models of unit disk graphs and grid graphs.

Optimal costs			
MINBW(L_m)	$=$	m	
MINCW(L_m)	$=$	$m + (\text{odd } m)$	[167]
MINEB(L_m)	$=$	$m + (\text{odd } m)$	[167]
MINLA(L_m)	$=$	$\frac{1}{3}(4-\sqrt{2})m^3 + O(m^2)$	[182, 189]
MINSC(L_m)	$=$	$\frac{2}{3}m^3 + \frac{1}{2}m^2 - \frac{7}{6}m$	
MINVB(L_m)	$=$	m	
MINVS(L_m)	$=$	m	

Table 4.1: Overview of the optimal costs of several layout problems on a $m \times m$ square grid graph L_m.

Upper bounds		
MINCW(L)	$\leqslant$	$14\sqrt{n}$
MINEB(L)	$\leqslant$	$2\sqrt{2n}+1$
MINLA(L)	$\leqslant$	$14n\sqrt{n}$
MINMC(L)	$\leqslant$	$14n\sqrt{n}$
MINSC(L)	$\leqslant$	$14n\sqrt{n}$
MINVB(L)	$\leqslant$	$2\sqrt{2n}+1$
MINVS(L)	$\leqslant$	$14\sqrt{n}$

Table 4.2: Overview of upper bounds for the cost of several layout problems on a grid graph L with n vertices.

5

Layout Problems and Random Unit Disk Graphs

5.1 Introduction

As we have seen in Chapter 3, with overwhelming probability, the ratios between the maximal and minimal costs of many layout problems on binomial random graphs are bounded by a constant. Therefore, binomial random graphs do not provide an informative framework to analyze heuristics for layout problems. Besides, it is doubtful that binomial random graphs are representative of real life inputs. This situation calls for the study of alternative models of random graphs. In this chapter we consider geometrical alternatives.

In the previous chapter we have seen that layout problems on unit disk graphs can have a certain relevance and are not simpler. Thus, it seems appropriate to study the approximability properties of layout problems on unit disk graphs generated at random.

We shall consider two random models of unit disk graphs: random grid graphs and random geometric graphs. Informally, a *random grid graph* with parameters $m \in \mathbb{N}$ and $p \in (0, 1)$ is obtained through the random selection of each vertex from a $m \times m$ square grid chosen independently with probability p, together with the addition of all edges between vertices at distance 1. Figure 5.1 shows a random grid graph. On the other hand, a *random geometric graph* with parameters $n \in \mathbb{N}$ and $r \in \mathbb{R}$ is a graph with n vertices that correspond to n points scattered at random in the unit square, together with all the edges between any pair of distinct vertices whose distance is smaller than r. Figure 5.2 shows a random geometric graph. Notice that both random grid graphs and random geometric graphs are unit disk graphs.

Very little work has been done on random grid graphs. In the best of our knowledge, this is the first systematic study of the behavior of some type of algorithm on this type of random graphs.

On the other hand, random geometric graphs have been proposed as a plausible and relevant model to take into account the structural characteristics of instances that appear in most of the practical applications for partitioning problems, such as finite element graphs and VLSI circuits. As a consequence, many empirical studies have used random geometric graphs for partitioning problems [23, 137, 165]. We have also done so, in Chapter 2, in our experimental study on the Minimum Linear Arrangement problem. However, up to now, the previous theoretical study of random geometric graphs has been mainly focused on parameters as their clique or chromatic number and their connectivity properties (more details will be given latter on), but these results have never been related to algorithmic issues. Therefore, the analysis of layout problems in random geometric graphs certainly seems worthwhile to study.

The first goal of this chapter is to present a probabilistic analysis on the behavior of several layout problems on random grid graphs and random geometric graphs. We shall see that the results we obtain are strongly related to classical results on probabilistic analysis of Euclidean optimization problems, which we will soon introduce. The second goal of this chapter is to make use of the obtained results on random geometric graphs to give empirical evidence of the goodness of several well-known heuristics for layout and partitioning problems. These heuristics include several global methods (Spectral, Multilevel and Greedy methods), some Local Search methods (Simulated Annealing, Kernighan–Lin and Helpful Sets), and two algorithms presented in this chapter.

In the remaining of this section we give the background for the concepts we study. This includes more formal definitions of random grid graphs and random geometric graphs, as well as their basic properties. We also introduce the two frameworks on which this chapter is sustained: percolation theory and the probabilistic theory of Euclidean optimization problems. Afterwards, in Section 5.2 we will present our results on layout problems and subcritical random grid graphs. Then, in Section 5.3 we will study a particular family of random geometric graphs. Finally, in Section 5.4, we will deal with subcritical random geometric graphs. This chapter will be concluded in Section 5.5 with a summary of results, conclusions and open problems.

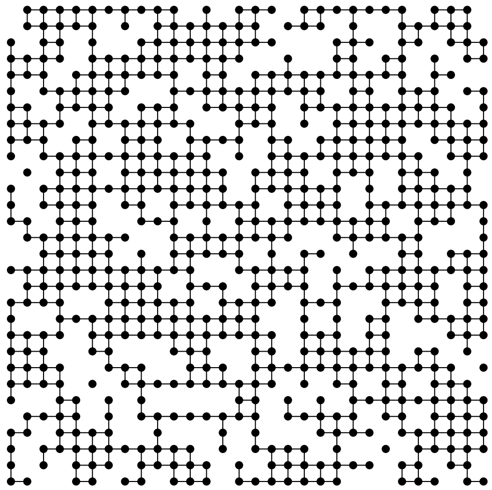


Figure 5.1: A random grid graph $\mathcal{L}_{m,p}$ with $m = 30$ and $p = \frac{3}{4}$.

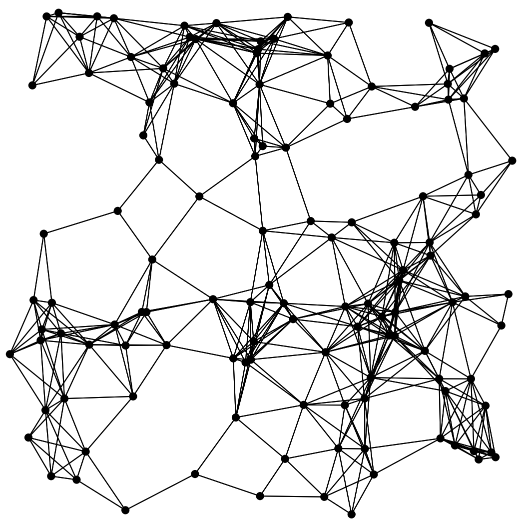


Figure 5.2: A random geometric graph with $n = 125$ vertices and radius $r = 0.155$.

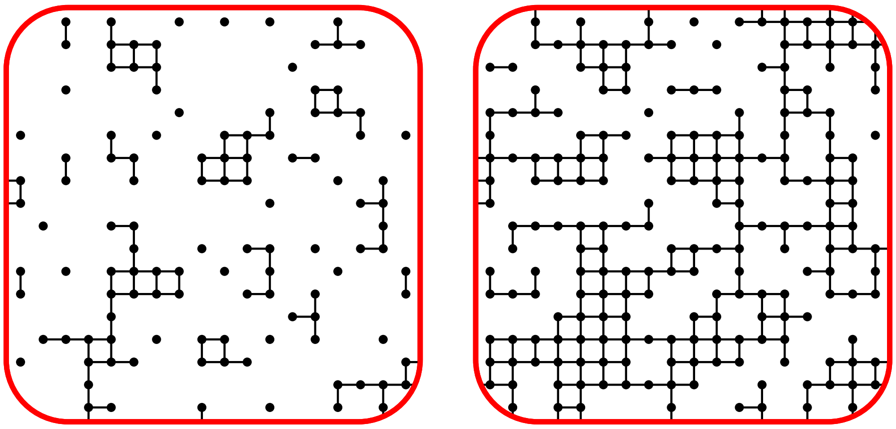


Figure 5.3: Partial view of two realizations of site percolation on a square grid for $p = 0.35$ (left) and $p = 0.65$ (right).

5.1.1 Random grid graphs and site percolation

Percolation is a discipline with applications to many diverse Natural Sciences, such as physics, mechanics or chemistry [180]. In this chapter, percolation theory provides a framework to study random grid graphs and random geometric graphs. We start describing some fundamental results for one of the simplest forms of percolation theory: site percolation in $\mathbb{Z}^2$. These will lead us to random grid graphs. More details on percolation theory can be found in Grimmett's book [109].

Definition 5.1 (Site percolation). Let $p \in (0,1)$. *Site percolation* is a probability space over the set of infinite grid graphs $G = (V, E)$ on the vertex set $V \subseteq \mathbb{Z}^2$ determined by

$$\mathbf{Pr}\left[u \in V\right] = p,$$

with these events mutually independent and $E = \{uv \,:\, u, v \in V \wedge \|u - v\|_2 = 1\}$.

Figure 5.3 shows a partial view of two infinite graphs produced by site percolation.

A related model to site percolation is *bond percolation*, where edges are selected at random rather than vertices. Most properties hold both on site and bond percolation; see [109] for the differences. We shall concentrate in site percolation, because its study will help us latter on, when considering random geometric graphs.

A *cluster* of a site percolation process is a connected component in the resulting infinite graph. Let C_x denote the cluster that includes the point $x \in \mathbb{Z}^2$

and C the cluster that includes the origin $(0,0)$. In the following, $|C_x|$ denotes the number of vertices in C_x. A basic question in percolation theory is whether or not a node of this graph belongs to an infinite cluster. Because of translation invariance, it is customary to study $|C|$. Let $\vartheta(p)$ denote the probability that the size of this cluster is unbounded:

$$\vartheta(p) = \mathbf{Pr}\left[|C| = \infty\right].$$

A fundamental result is that there exists a *critical value* of p defined by

$$p_c = \inf\{p \ : \ \vartheta(p) > 0\},$$

such that

$$\vartheta(p) \begin{cases} = 0 & \text{if } p < p_c, \\ > 0 & \text{if } p > p_c. \end{cases}$$

The case $p \in (0, p_c)$ is called the *subcritical phase* and the case $p \in (p_c, 1)$ is called the *supercritical phase*. The case $p = p_c$ is called the *critical point*. In the subcritical phase all components are finite almost surely: when $p \in (0, p_c)$, the probability that there exists an infinite cluster is 0. On the other hand, in the supercritical phase there exists infinite clusters almost surely: when $p \in (p_c, 1)$, the probability that there exists an infinite cluster is 1. For bond percolation, it is known that $p_c = \frac{1}{2}$. However, for site percolation, it is only known that $p_c \in (\frac{1}{2}, 1)$, although computational experiments suggest a value close to 0.59 (see Section 1.6 in [109]).

The next results show that the size and the radius of a component decay exponentially fast in the subcritical phase. In [109], these theorems are proved for bond percolation, but they also hold for site percolation.

Theorem 5.1 ([109]). Let $p \in (0, p_c)$. Then, there exists a constant $\alpha = \alpha(p) > 0$ such that $\mathbf{Pr}\left[|C| = k\right] \leqslant \exp\left(-\alpha k\right)$ for all $k \in \mathbb{N}$. Moreover, there exists a constant $\rho = \rho(p) > 0$ such that $\mathbf{Pr}\left[|C| > k\right] \leqslant \exp\left(-\rho k\right)$ for all $k \in \mathbb{N}$.

Theorem 5.2 ([109]). Let $p \in (0, p_c)$ and let $R = \min\{j : C \subset [-j, j]^2\}$. Then, there exists a constant $\gamma = \gamma(p) > 0$ such that $\mathbf{Pr}\left[R = k\right] \leqslant \exp\left(-\gamma k\right)$ for all $k \in \mathbb{N}$.

In the following, a random grid graph with parameters m and p will be obtained by restricting a site percolation process with parameter p to the vertices $\{0, \ldots, m-1\}^2$ of L_m:

Definition 5.2 (Random grid graph). Consider a site percolation process on $\mathbb{Z}^2$ with parameter p. For all $m \in \mathbb{N}$, we write $V_m = \{0, \ldots, m-1\}^2$ and call $\mathcal{L}_{m,p}$ the *random grid graph* obtained by taking the open vertices in V_m and the edges joining them.

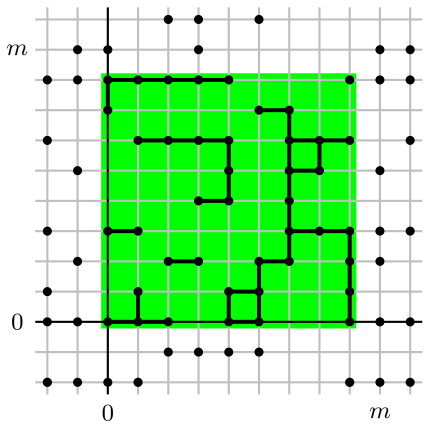


Figure 5.4: A random grid graph $\mathcal{L}_{m,p}$ in the subcritical phase with $m = 9$ and $p = 0.45$. There are 7 connected components.

Figure 5.4 illustrates this definition.

It must be observed that random grid graphs are defined by an *incremental model*: A sequence of random grid graphs $(\mathcal{L}_{m,p})_{m\in\mathbb{N}}$ is obtained on a unique site percolation process generated on the infinite grid. We shall be interested in knowing the properties of the random variables $F(\mathcal{L}_{m,p})$ as m goes to infinity. In most cases, F will be the optimal cost of a layout problem. On the other hand, it must be stressed that p is fixed; therefore, values depending only on p will be considered constants. This includes expressions as $\mathbf{E}[\cdot]$, which is a constant depending on p, and $\text{MINLA}(C)$, which is a random variable also depending on p.

Let us introduce some more notation: In the following, C_x^m denotes the cluster in $\mathcal{L}_{m,p}$ that includes the point $x \in V_m$, and C^m denotes the cluster in $\mathcal{L}_{m,p}$ that includes $(0,0)$. Also, $|C_x^m|$ denotes the number of vertices in C_x^m. Notice that $C_x^m = C_x \cap V_m$ and thus $C_x^m \subseteq C_x$ and $|C_x^m| \leqslant |C_x|$.

While studying together subcritical random grid graphs, Penrose analyzed the behavior of several layout problems on supercritical random grid graphs. The following theorem describes the order of magnitude of their respective costs:

Theorem 5.3 ([205]). Let $p \in (p_c, 1)$. Then, there exist a constant $c > 0$ such that, with overwhelming probability,

$$\begin{aligned}
cm^2 \leqslant\ & \text{MINSC}(\mathcal{L}_{m,p}) && \leqslant m^2,\\
cm^2 \leqslant\ & \text{MINLA}(\mathcal{L}_{m,p}) && \leqslant 4m^2,\\
cm \leqslant\ & \text{MINBW}(\mathcal{L}_{m,p}) && \leqslant m,\\
cm \leqslant\ & \text{MINVS}(\mathcal{L}_{m,p}) && \leqslant m,
\end{aligned}$$

$$cm \leqslant \quad \text{MINCW}(\mathcal{L}_{m,p}) \quad \leqslant 4m.$$

Moreover, for all p such that $\vartheta(p) > \frac{1}{2}$, with overwhelming probability,

$$cm \leqslant \text{MINEB}(\mathcal{L}_{m,p}) \leqslant 4m.$$

5.1.2 Random geometric graphs

After having presented random grid graphs, we now introduce random geometric graphs. Let us first define a geometric graph:

Definition 5.3 (Geometric graphs). Consider any norm $\|\cdot\|$. Let V be a set of points in $[0,1]^2$ and let r be a positive real. The *geometric graph* $\mathcal{G}(V;r)$ is the graph (V,E) where $E = \{uv \ : \ u, v \in V \wedge 0 < \|u - v\| \leqslant r\}$.

Notice that by the proximity model introduced in Section 4.1, geometric graphs are unit disk graphs. We can now define the class of random geometric graphs:

Definition 5.4 (Random geometric graphs). Consider any norm $\|\cdot\|$. Let $(r_n)_{n\in\mathbb{N}}$ be a sequence of positive numbers and let $X = (X_n)_{n\in\mathbb{N}}$ be a sequence of independently and uniformly distributed random points in $[0,1]^2$. For any $n \in \mathbb{N}$, we write $\mathcal{X}_n = \{X_1, \ldots, X_n\}$ and call $\mathcal{G}(\mathcal{X}_n; r_n)$ the *random geometric graph* of n vertices on X with radius r_n.

As with random grid graphs, random geometric graphs are defined by an *incremental model*: A sequence of random geometric graphs $(\mathcal{G}(\mathcal{X}_n; r_n))_{n\in\mathbb{N}}$ is obtained by a unique infinite sequence of independent and uniformly distributed random points in $[0,1]^2$ and a unique infinite sequence of radius. We shall be interested in knowing the properties of the random variables $F(\mathcal{G}(\mathcal{X}_n; r_n))$ as n goes to infinity.

Several properties of random geometric graphs have been studied by Apple and Russo. These include characterizations of their minimum and maximum vertex degree, clique number, chromatic number or independence number [8, 9, 10]. Connectivity is also an important issue on random geometric graphs: Depending on the specific values of n and r_n, random geometric graphs can be dense or sparse, connected or disconnected, etc. The following lemma, given in [10] without proof, states a basic result for random geometric graphs on l_∞:

Lemma 5.1 ([10]). Let G be a random geometric graph with n vertices and radius $r \in [0,1]$ measured on the l_∞ norm. Then,

$$\mathbf{Pr}\left[uv \in E(G)\right] = (2r - r^2)^2.$$

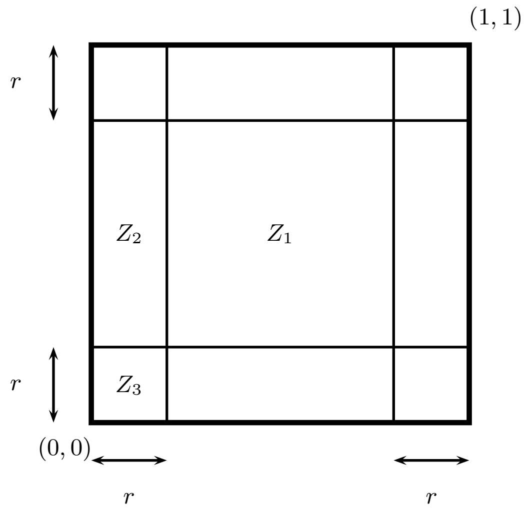


Figure 5.5: Illustration for the proof of Lemma 5.1.

Proof. Decompose the unit square in the nine zones shown in Figure 5.5. Because of symmetry, we have

$$\begin{aligned}
\mathbf{Pr}\left[uv \in E(G)\right] &= \mathbf{Pr}\left[u \in Z_1\right] \cdot \mathbf{Pr}\left[\|u-v\|_\infty \leqslant r \ : \ u \in Z_1\right] + \\
&\quad + 4 \cdot \mathbf{Pr}\left[u \in Z_2\right] \cdot \mathbf{Pr}\left[\|u-v\|_\infty \leqslant r \ : \ u \in Z_2\right] + \\
&\quad + 4 \cdot \mathbf{Pr}\left[u \in Z_3\right] \cdot \mathbf{Pr}\left[\|u-v\|_\infty \leqslant r \ : \ u \in Z_3\right] \\
&= \int_{x_u=r}^{1-r} \int_{y_u=r}^{1-r} \int_{x_v=x_u-r}^{x_u+r} \int_{y_v=y_u-r}^{y_v+r} \mathrm{d}y_v \, \mathrm{d}x_v \, \mathrm{d}y_u \, \mathrm{d}x_u + \\
&\quad + 4 \int_{x_u=0}^{r} \int_{y_u=r}^{1-r} \int_{x_v=0}^{x_u+r} \int_{y_v=y_u-r}^{y_v+r} \mathrm{d}y_v \, \mathrm{d}x_v \, \mathrm{d}y_u \, \mathrm{d}x_u + \\
&\quad + 4 \int_{x_u=0}^{r} \int_{y_u=0}^{r} \int_{x_v=0}^{x_u+r} \int_{y_v=0}^{y_v+r} \mathrm{d}y_v \, \mathrm{d}x_v \, \mathrm{d}y_u \, \mathrm{d}x_u \\
&= (2r-r^2)^2,
\end{aligned}$$

which proves the lemma. □

When interpreting this lemma, it must be remarked that the edges in random geometric graphs are not independent.

According to the above result, the expectation of the random variable M_n that counts the number of edges in a random geometric graph G_n with n vertices and radius r is $\mathbf{E}\left[M\right] = \binom{n}{2}(2r-r^2)^2$. In fact, M_n converges almost surely to $\binom{n}{2}(2r-r^2)^2$. This means that in the case that r is independent of

n, then $M_n = \Theta(n^2)$ and thus a sequence of dense graphs is obtained. As our focus is on sparse graphs, we must select sequences of radius satisfying $r_n \to 0$. The following theorem establishes another connectivity property of random geometric graphs.

Theorem 5.4 ([8]). Let $X = (X_n)_{n\in\mathbb{N}}$ be a sequence of independent and uniformly distributed random points in $[0,1]^2$ and let $\mathcal{X}_n = \{X_i\}_{i=1}^n$. Define the connectivity distance ρ_n of a random geometric graph by

$$\rho_n = \inf\{r \ : \ \mathcal{G}(\mathcal{X}_n; r) \text{ is connected}\}.$$

Then,

$$\frac{\rho_n}{\sqrt{\log n/n}} \xrightarrow{\text{as}} \tfrac{1}{2}.$$

Penrose [206] generalized the previous result to any metric l_p with $p \in \mathbb{N}\cup\{\infty\}$, proving that, with high probability, if one starts with single vertices and adds the corresponding edges as r increases, the resulting graph becomes $k+1$ connected at the moment it achieves a minimum degree of $k+1$. Penrose has also shown that, similarly to site percolation, random geometric graphs exhibit a phase transition [203]: Suppose $nr_n^2 \to \lambda$; then there exists a critical parameter λ_c such that when $\lambda < \lambda_c$, graphs $\mathcal{G}(\mathcal{X}_n; r_n)$ are likely to have at most $O(\log n)$ vertices in each connected component, while when $\lambda > \lambda_c$, there is likely to be a component with $\Theta(n)$ vertices. When $\lambda < \lambda_c$, we will speak about *subcritical random geometric graphs*; when $\lambda > \lambda_c$, we will speak about *supercritical random geometric graphs*. We shall give more details latter on.

In [205], Penrose also analyzes the behavior of several layout problems on supercritical random geometric graphs. He obtained the order of magnitude of their respective minimal costs:

Theorem 5.5 ([205]). Suppose $nr_n^2 \to \lambda \in (\lambda_c, \infty)$. Then, with high probability,

$$\begin{aligned}
\textsc{minvs}(\mathcal{G}(\mathcal{X}_n; r_n)) &= \Theta(nr_n),\\
\textsc{minbw}(\mathcal{G}(\mathcal{X}_n; r_n)) &= \Theta(nr_n),\\
\textsc{minsc}(\mathcal{G}(\mathcal{X}_n; r_n)) &= \Theta(n^2 r_n),\\
\textsc{mineb}(\mathcal{G}(\mathcal{X}_n; r_n)) &= \Theta(n^2 r_n^3),\\
\textsc{mincw}(\mathcal{G}(\mathcal{X}_n; r_n)) &= \Theta(n^2 r_n^3),\\
\textsc{minla}(\mathcal{G}(\mathcal{X}_n; r_n)) &= \Theta(n^3 r_n^3).
\end{aligned}$$

In this chapter we will obtain a better characterization of the previous results, at the expenses of increasing r_n to the point where $\mathcal{G}(\mathcal{X}_n; r_n)$ graphs become connected almost surely. We will also find algorithms that reach these bounds.

5.1.3 The Euclidean model

As shall be clear in the development of this chapter, our study on layout problems for random unit disk graphs fits in the framework of the probabilistic theory of Euclidean optimization problems in the plane. The monographs of Yukich [243] and Steele [234] are the basic references for this field. In the following, we introduce the Euclidean model and its most important properties related to optimization problems.

The *Euclidean* model considers weighted complete graphs whose edge weights correspond to the Euclidean distance between a set of independent and identically distributed points in the d-dimensional unit square, $[0,1]^d$.

Let us consider the traveling salesman problem (TSP). The Euclidean TSP is an **NP**-hard problem, but it can be approximated within a $\frac{3}{2}$ factor [98]. In one of the seminal papers in the area of probabilistic analysis of combinatorial optimization problems on the Euclidean model, Beardwood, Halton and Hammersley proved the following result, often referred as he BHH theorem.

Theorem 5.6 (BHH [22]). Let $X = \{X_n\}_{n\in\mathbb{N}}$ be a sequence of independent and uniformly distributed points in $[0,1]^d$. Let $\textsc{mintsp}(n)$ denote the length of the optimal solution of the TSP among the n first points of X. Then, there exists a constant $\beta_{\text{TSP}}(d) \in (0,\infty)$ such that

$$\frac{\textsc{mintsp}(n)}{n^{(d-1)/d}} \xrightarrow{\text{as}} \beta_{\text{TSP}}(d).$$

The result of the BHH theorem for the TSP has been extended to other combinatorial problems for the Euclidean model on $[0,1]^d$: the Minimum Matching [200], the Steiner Minimal Tree [232], and the Minimum Spanning Tree [233, 16, 204]. In fact, Steele [234] generalized the BHH theorem for a class of combinatorial problems that can be formulated as a *subadditive Euclidean functional* from finite subsets of $[0,1]^d$ to the nonnegative real numbers that measure the cost that the problems optimize as, for instance, the length of a tour or the weight of a spanning tree. Specifically, F is said to be a *subadditive functional* if it has the following properties: for all $x_i \in [0,1]^d$:

1. $F(\emptyset) = 0$.
2. $F(cx_1,\dots,cx_n) = cF(x_1,\dots,x_n), \qquad \forall c > 0$.
3. $F(x_1+y_1,\dots,x_n+y_n) = F(x_1,\dots,x_n), \qquad \forall y_i \in [0,1]^d$.
4. $F(x_1,\dots,x_n) \leqslant F(x_1,\dots,x_n,x_{n+1})$.
5. If Q_i is the i-th square of size $1/m \times 1/m$, then the *geometric subadditivity condition* holds: $\exists c > 0 : \ \forall m,n \geqslant 1 :$

$$F(x_1,\dots,x_n) \leqslant \sum_{i=1}^{m^2} F(\{x_1,\dots,x_n\} \cap Q_i) + cm.$$

Steele proved that for each problem in this class, there exists a constant $\beta_F(d)$ such that

$$\frac{\text{MIN}F(n)}{n^{(d-1)/d}} \xrightarrow{\text{Pr}} \beta_F(d)$$

where $\text{MIN}F(n)$ denotes the cost of an optimal solution to F among the first n points in a set of independent and identically distributed points in $[0,1]^d$.

It is an open problem to determine the values of $\beta_F(d)$ for most functionals, even for the case $d = 2$. Up to date, the best estimation for the TSP was empirically obtained by Johnson *et al.* who claim that $\beta_{\text{TSP}}(2) = 0.7124 \pm 0.0002$ [140].

To finish our presentation of the Euclidean model, we present an important algorithmic result: *Karp's dissection technique* [146]. Given a uniform independent and identically distributed set of points X in the unit square, Karp's algorithm consists in dissecting $[0,1]^2$ into n/m_n squares $\{Q_i\}$, each one of size $\sqrt{(m_n/n)} \times \sqrt{(m_n/n)}$. With high probability, every square will contain not much more than m_n points. Then, use dynamic programming to construct an optimal tour in each square in polynomial time; this is possible provided that m_n is of logarithmic size (see e.g. Section 5.6 in [40]). In order to join the individual tours, consider each tour in each square Q_i as a single point p_i, and for any $i, j \in [n/m_n]$, define the distance between p_i and p_j as the shortest Euclidean distance between any point in Q_i and any point in Q_j. Construct in $O(n \log n)$ steps the minimal-length spanning tree joining all the $\{p_i\}_i$ points [224]. The algorithm returns the closed walk that traverses each subtour once and each tree edge twice.

Karp proves that, for any $\epsilon > 0$, with high probability, his algorithm runs in $O(n^2 \log n)$ steps and produces a tour of length within $1 + \epsilon$ of the optimal one.

5.2 Subcritical random grid graphs

In this section we consider layout problems on random grid graphs $\mathcal{L}_{m,p}$ generated by subcritical limiting phases; that is, with $p \in (0, p_c)$. First, we give tight bounds for the Cutwidth and Vertex Separation problems on $\mathcal{L}_{m,p}$. Afterwards, we will present convergence results for the Minimum Linear Arragement, Modified Cut and Sum Cut problems. These convergence results can be regarded as discrete analogues of the BHH theorem for layout problems. In the case of Minimum Linear Arragement, we report on a computational experiment to determine the limiting constant in the BHH-like theorem.

5.2.1 Order of growth of MINVS and MINCW

The following theorem shows that, with probability 1, the minimal costs of the VERTSEP and CUTWIDTH problems on subcritical random grid graphs $\mathcal{L}_{m,p}$ are $\Theta(\sqrt{\log m})$.

Theorem 5.7. Let $p \in (0, p_c)$. Then, there exists two constants $0 < c_1 < c_2$ such that, with probability 1, for all m large enough,

$$c_1 \leqslant \frac{\text{MINVS}(\mathcal{L}_{m,p})}{\sqrt{\ln m}} \leqslant \frac{\text{MINCW}(\mathcal{L}_{m,p})}{\sqrt{\ln m}} \leqslant c_2.$$

Proof. First, we will obtain an upper bound for $\text{MINCW}(\mathcal{L}_{m,p})/\sqrt{\ln m}$, holding with high probability. Recall that, by Lemma 1.2, the MINCW cost of a disconnected graph is the maximum of the MINCW costs of its connected components. Hence, for any positive constant c_2,

$$\begin{aligned}\mathbf{Pr}\left[\text{MINCW}(\mathcal{L}_{m,p}) \geqslant c_2\sqrt{\ln m}\right] &= \\ &= \mathbf{Pr}\left[\vee_{x \in V_m} \text{MINCW}(C_x^m) \geqslant c_2\sqrt{\ln m}\right].\end{aligned}$$

By the exponential decay property presented in Theorem 5.1, there exists a constant $\alpha > 0$ such that $\mathbf{Pr}\left[|C| \geqslant k\right] \leqslant e^{-\alpha k}$. So, using Theorem 4.6, we obtain

$$\begin{aligned}\mathbf{Pr}\left[\text{MINCW}(\mathcal{L}_{m,p}) \geqslant c_2\sqrt{\ln m}\right] &\leqslant \mathbf{Pr}\left[\vee_{x \in V_m} |C_x^m| \geqslant \left(\tfrac{1}{14}c_2\right)^2 \ln m\right] \\ &\leqslant m^2 \exp(-\alpha \left(\tfrac{1}{14}c_2\right)^2 \ln m).\end{aligned}$$

Choosing $c_2 = 18\sqrt{2/\alpha}$, we get

$$\mathbf{Pr}\left[\text{MINCW}(\mathcal{L}_{m,p})/\sqrt{\ln m} \leqslant c_2\right] \geqslant 1 - m^{-64/49}, \tag{5.1}$$

which implies that $\text{MINCW}(\mathcal{L}_{m,p})/\sqrt{\ln m} \leqslant c_2$ occurs with high probability.

Next, we will get a lower bound for $\text{MINVS}(\mathcal{L}_{m,p})/\sqrt{\ln m}$ holding with overwhelming probability. Let $\delta > 0$ and let $T_1, \ldots, T_{j(m)}$ be disjoint grid subsquares of L_m, each of side $\lfloor\sqrt{\delta \ln m}\rfloor$, where $j(m) = \lfloor m/\lfloor\sqrt{\delta \ln m}\rfloor\rfloor^2$. Set $\gamma = \ln(1/p)$ so that $p = e^{-\gamma}$.

We claim that, at least, one of these subsquares is completely filled: For all $i \in [j(m)]$, let A_i be the event that all sites in T_i are open. Then,

$$\mathbf{Pr}\left[A_i\right] = p^{\lfloor\sqrt{\delta \ln m}\rfloor^2} = \exp(-\gamma\lfloor\sqrt{\delta \ln m}\rfloor^2) \geqslant m^{-\gamma\delta}.$$

As a consequence, assuming m sufficiently large and noticing that $j(m) \geqslant m^2/m^{\delta\gamma}$,

$$\mathbf{Pr}\left[\wedge_{i=1}^{j(m)} A_i^c\right] \leqslant \left(1 - m^{-\gamma\delta}\right)^{j(m)} \leqslant e^{-m^{-\gamma\delta} j(m)} \leqslant e^{-m^{2-\gamma\delta}/\delta m^{\delta\gamma}} \leqslant e^{-m}.$$

So, by de Morgan's law,

$$\mathbf{Pr}\left[\vee_{i=1}^{j(m)} A_i\right] \geqslant 1 - e^{-m},$$

which proves the claim.

Assume that all sites in a subsquare T_i are open. By Theorem 4.5 and and Theorem 1.3,

$$\textsc{minvs}(\mathcal{L}_{m,p}) \geqslant \textsc{minvs}(L_{\lfloor\sqrt{\delta \ln m}\rfloor}) = \left\lfloor \sqrt{\delta \ln m} \right\rfloor \geqslant \tfrac{1}{2}\sqrt{\delta \ln m}.$$

So, by the previous claim, choosing $c_1 = \frac{1}{2}\sqrt{\delta}$, we get

$$\mathbf{Pr}\left[c_1 \leqslant \textsc{minvs}(\mathcal{L}_{m,p})/\sqrt{\ln m}\right] \geqslant 1 - e^{-m}, \tag{5.2}$$

which implies that $\textsc{minvs}(\mathcal{L}_{m,p})/\sqrt{\ln m} \geqslant c_1$ occurs with overwhelming probability.

Using Boole's inequality on equations (5.1) and (5.2) we get

$$\begin{aligned}\mathbf{Pr}\left[c_1 < \frac{\textsc{minvs}(\mathcal{L}_{m,p})}{\sqrt{\ln m}} \bigvee \frac{\textsc{mincw}(\mathcal{L}_{m,p})}{\sqrt{\ln m}} > c_2\right] &\leqslant e^{-m} + m^{-64/49}\\ &\leqslant 2m^{-64/49}.\end{aligned}$$

As $\sum_{m\in\mathbb{N}} 2m^{-64/49}$ is bounded, using the Borel–Cantelli Lemma (see Theorem A.5), we get

$$\mathbf{Pr}\left[\left(c_1 \leqslant \frac{\textsc{minvs}(\mathcal{L}_{m,p})}{\sqrt{\ln m}} \bigwedge \frac{\textsc{mincw}(\mathcal{L}_{m,p})}{\sqrt{\ln m}} \leqslant c_2\right) \ \mathbf{a.a.}\right] = 1,$$

where **a.a.** stands for *almost always* (see Appendix A.2.1). By Lemma 1.1, we have $\textsc{minvs}(G) \leqslant \textsc{mincw}(G)$, which implies the theorem. □

5.2.2 Convergence results for MINLA, MINMC and MINSC

We now seek a stronger result for the MinLA, ModCut and SumCut problems, namely that $\textsc{minla}(\mathcal{L}_{m,p})/m^2$, $\textsc{minmc}(\mathcal{L}_{m,p})/m^2$ and $\textsc{minsc}(\mathcal{L}_{m,p})/m^2$ converge in probability to a constant when $p \in (0, p_c)$.

In the next lemma we prove that for subcritical site percolation with parameter p, the expected ratio between $\textsc{minla}(C)$ and $|C|$ is finite. The same property holds for the ModCut and the SumCut problems. Throughout the remainder of the section, we use the convention $0/0 = 0$ to cover the case $C = \emptyset$.

Lemma 5.2. Let $p \in (0, p_c)$ and $M \in \{\textsc{la}, \textsc{mc}, \textsc{sc}\}$. Then,

$$\mathbf{E}\left[\frac{\textsc{min}M(C)}{|C|}\right] \in (0, \infty).$$

Proof. Let $R = \min\{k : C \subset [-k,k]^2\}$. By considering the lexicographic layout, $\textsc{minla}(L_m) \leqslant m^3$, which together with monotonicity (see Lemma 1.3) gives us that $\textsc{minla}(C) \leqslant (2R+1)^3$. By Theorem 5.2, R decays exponentially in k. So,

$$\begin{aligned} 0 < \mathbf{E}\left[\frac{\textsc{minla}(C)}{|C|}\right] &\leqslant \sum_{k\geqslant 0} \mathbf{Pr}\left[C \subset [-k,k]^2\right] \frac{(2k+1)^3}{(2k+1)^2} \\ &\leqslant \sum_{k\geqslant 0} e^{-\alpha k}(2k+1) < \infty. \end{aligned}$$

The proofs for ModCut and SumCut are similar. □

We use the previous lemma to prove a convergence result on random grid graphs. It is important to remark that the next theorem can be viewed as an analogue of the BHH Theorem for the MinLA, ModCut and SumCut problems on subcritical random grid graphs.

Theorem 5.8. Let $p \in (0, p_c)$; then, there exist three constants $\beta_{\text{LA}}(p) > 0$, $\beta_{\text{SC}}(p) > 0$ and $\beta_{\text{SC}}(p) > 0$ such that, as $m \to \infty$,

$$\begin{aligned} \textsc{minla}(\mathcal{L}_{m,p})/m^2 &\xrightarrow{\text{Pr}} \beta_{\text{LA}}(p), \\ \textsc{minmc}(\mathcal{L}_{m,p})/m^2 &\xrightarrow{\text{Pr}} \beta_{\text{MC}}(p), \\ \textsc{minsc}(\mathcal{L}_{m,p})/m^2 &\xrightarrow{\text{Pr}} \beta_{\text{SC}}(p). \end{aligned}$$

Proof. Let $M \in \{\textsc{minla}, \textsc{minmc}, \textsc{minsc}\}$. Then,

$$\begin{aligned} \frac{M(\mathcal{L}_{m,p})}{m^2} &= \frac{1}{m^2} \sum_{x\in V_m} \frac{M(C_x^m)}{|C_x^m|} \\ &= \frac{1}{m^2} \sum_{x\in V_m} \left(\frac{M(C_x^m)}{|C_x^m|} - \frac{M(C_x)}{|C_x|}\right) + \frac{1}{m^2} \sum_{x\in V_m} \frac{M(C_x)}{|C_x|}. \end{aligned} \tag{5.3}$$

Call $t_1(m)$ and $t_2(m)$ the first and second terms respectively of the previous equation.

Let us work with $t_1(m)$. By monotonicity, we have $M(C_x^m) \leqslant M(C_x)$. So,

$$\begin{aligned} t_1(m) &\leqslant \frac{1}{m^2} \sum_{x\in V_m} \left(\frac{M(C_x)}{|C_x^m|} - \frac{M(C_x)}{|C_x|}\right) \\ &\leqslant \frac{1}{m^2} \sum_{x\in V_m} \left|\frac{M(C_x)}{|C_x^m|} - \frac{M(C_x)}{|C_x|}\right| \\ &= \frac{1}{m^2} \sum_{\substack{x\in V_m \\ C_x \neq C_x^m}} \left|\frac{M(C_x)}{|C_x^m|} - \frac{M(C_x)}{|C_x|}\right|. \end{aligned}$$

But $C_x \neq C_x^m$ implies $|C_x| > |C_x^m| \geqslant 0$, and $a \geqslant b \geqslant 0$ implies $|a-b| \leqslant 2a$, so we have

$$t_1(m) \leqslant \frac{1}{m^2} \sum_{\substack{x \in V_m \\ C_x \neq C_x^m}} \frac{2M(C_x)}{|C_x^m|} \leqslant \frac{2}{m^2} \sum_{y \in \partial V_m} M(C_y),$$

where ∂V_m stands for the set of vertices $x \in V_m$ with grid neighbors in $\mathbb{Z}^2 \setminus V_m$.

Let $\epsilon > 0$. Call $\pi(m)$ the probability that $t_1(m) > \epsilon$. We have:

$$\pi(m) \leqslant \mathbf{Pr}\left[\sum_{y \in \partial V_m} M(C_y) > \tfrac{1}{2}\epsilon m^2\right] \leqslant \mathbf{Pr}\left[\bigvee_{y \in \partial V_m} M(C_y) > \tfrac{1}{2}\epsilon m^2\right],$$

which by Boole's inequality gives

$$\pi(m) \leqslant \sum_{y \in \partial V_m} \mathbf{Pr}\left[M(C_y) > \tfrac{1}{2}\epsilon m^2\right].$$

Because of translation invariance, $\mathbf{Pr}\left[M(C_y) > \frac{1}{2}\epsilon m^2\right]$ is independent on y. Therefore, we have

$$\pi(m) \leqslant |\partial V_m| \cdot \mathbf{Pr}\left[M(C)) > \tfrac{1}{2}\epsilon m^2\right],$$

and using the upper bounds on grid graphs given in Theorem 4.6, we get

$$\pi(m) \leqslant 4m \cdot \mathbf{Pr}\left[|C| > (\epsilon m^2/28)^{2/3}\right].$$

By Theorem 5.1, the tail of a cluster size decays exponentially. So we have

$$\pi(m) \leqslant 4m \cdot \exp\left(-\rho(\epsilon m^2/28)^{2/3}\right) \leqslant 4\exp\left(-\rho\left(\tfrac{\epsilon}{28}\right)^{2/3} m^{1/3} + \ln m\right),$$

which implies that $\pi(m)$ tends to 0 as m tends to infinity. As ϵ is an arbitrary small positive real, we have

$$t_1(m) \xrightarrow{\mathrm{Pr}} 0. \tag{5.4}$$

Let us now consider $t_2(m)$. We have that $(M(C_x)/|C_x|)_{x \in \mathbb{Z}^2}$ is a collection of bounded functions of independent vertex-states, and is stationary under translations of the infinite grid $\mathbb{Z}^2$. It follows from the Ergodic theorem (see Theorem VII.6.9 in [79] and the proof of Theorem 4.2 in [109]) that

$$t_2(m) = \frac{1}{m^2} \sum_{x \in V_m} \frac{M(C_x)}{|C_x|} \xrightarrow{\mathrm{Pr}} \mathbf{E}\left[\frac{M(C)}{|C|}\right]. \tag{5.5}$$

As $M(\mathcal{L}_{m,p})/m^2 = t_1(m) + t_2(m)$, from (5.4) and (5.5) we have that

$$\frac{M(\mathcal{L}_{m,p})}{m^2} \xrightarrow{\text{Pr}} \mathbf{E}\left[\frac{M(C)}{|C|}\right].$$

Taking $\beta_{\text{LA}}(p) = \mathbf{E}\left[\text{MINLA}(C)/|C|\right]$, $\beta_{\text{MC}}(p) = \mathbf{E}\left[\text{MINMC}(C)/|C|\right]$ and $\beta_{\text{SC}}(p) = \mathbf{E}\left[\text{MINSC}(C)/|C|\right]$ the theorem is proved. □

5.2.3 Experimental determination of $\beta_{\text{LA}}(p)$

In the proof of Theorem 5.8, we characterized $\beta_{\text{LA}}(p)$, $\beta_{\text{MC}}(p)$ and $\beta_{\text{SC}}(p)$, but we did not give their exact value. The purpose of this subsection is to estimate empirically the value of $\beta_{\text{LA}}(p)$ for some values of p with $p < p_c$. The motivation behind is that Theorem 5.8 shows that for $\mathcal{L}_{m,p}$ graphs with $p \in (0, p_c)$, there exists a nonzero finite constant $\beta_{\text{LA}}(p)$ that, as m goes to infinity, $\text{MINLA}(\mathcal{L}_{m,p})/m^2$ converges in probability to $\beta_{\text{LA}}(p)$, where $\beta_{\text{LA}}(p) = \mathbf{E}[\text{MINLA}(C)/|C|]$. This result might be of importance in order to predict the expected value of a random grid graph $\mathcal{L}_{m,p}$, as for large values of m, we will have that $\text{MINLA}(\mathcal{L}_{m,p})$ behaves similarly to $\beta_{\text{LA}}(p) \cdot m^2$. The same holds for $\text{MINMC}(\mathcal{L}_{m,p})$ and $\text{MINSC}(\mathcal{L}_{m,p})$.

In order to estimate $\beta_{\text{LA}}(p)$, we must approximate the expectation of $\text{MINLA}(C)/|C|$, where C is a cluster obtained with site percolation with parameter p. For a given value $p \in (0, p_c)$, the setting of the basic experiment is obtained through the following procedure:

```
Generate at random a connected component C centered at the origin
    where the probability of a node to be open is p.
if |C| is "sufficiently small" then
  Compute exactly MINLA(C)
else
  Compute a lower bound of MINLA(C)
  Compute an upper bound of MINLA(C)
end if
```

To generate C, we have applied the following algorithm: We first draw a random number r in $(0, 1)$. If $r \leqslant 1-p$, we take C as the null graph. Otherwise, we start by including node $(0, 0)$ in C, marking it as "alive" and marking the rest of nodes as "waiting." Then, for each "alive" node, we process each of its "waiting" neighbors. Using a random number generator, with probability p, a "waiting" node is included in C and its mark changes to "alive;" and with probability $1-p$ its mark changes to "dead." This procedure is iterated until no "alive" nodes exist. Notice that this procedure correctly generates connected random grid graphs. The fact that we are dealing with the subcritical phase ensures termination.

When $|C| \leqslant 15$, we are able to compute the exact value of $\textsc{minla}(C)$ with a reasonable running time. Here, "reasonable time" means that the running time in order to perform the whole experiment on our machine was about two days. Otherwise, we use the Degree method to obtain a lower bound, and Randomized Successive Augmentation algorithms to compute an upper bound (see Sections 2.2.1 and 2.2.2). We have selected these techniques as they were the ones that obtained the best results and best running times in preliminary tuning tests.

This basic experiment of generation, lower bound and upper bound computation has been repeated 10,000 times, for several data points $p \in [0.1, 0.45]$. In this way, for each value of p, we have obtained an average lower bound and an average upper bound of $\mathbf{E}[\textsc{minla}(C)/|C|]$.

The results are shown in Figure 5.6. As it can be seen, the accuracy on the estimation of $\beta_{\text{LA}}(p)$ degrades as p increases, but for $p = 0.45$, the factor is less than 3. Unfortunately, the measured standard deviation, shown in Table 5.1, is quite big, so the results obtained may not be close to the mean value.

We conjecture that, for high values of p, the actual value of $\beta_{\text{LA}}(p)$ is closer to the upper bound than to the lower bound. There are two reasons for this conjecture: on one hand, the upper bounds obtained using the SS+SA heuristic presented in Section 2.5 are only slightly lower than the ones obtained by multiple runs of Successive Augmentation algorithms; on the other hand, our experience in Chapter 2 suggests that even if the Degree method is the method that delivers the higher lower bounds, these are usually far from the optimal values.

In any case, the bad results obtained with this approach do not encourage us to extend it for $\beta_{\text{SC}}(p)$ and $\beta_{\text{MC}}(p)$.

5.3 Connected random geometric graphs

In this section we study layout problems on random geometric graphs. Specifically, all through this section, we restrict our attention to the particular case where the radius of the random geometric graphs is of the form

$$r_n = \sqrt{\frac{a_n}{n}} \quad \text{where} \quad r_n \to 0 \quad \text{and} \quad a_n/\log n \to \infty.$$

It is important to recall that, almost surely, this choice results in the construction of connected graphs.

The problems considered here are Minimum Linear Arragement, Cutwidth, Edge Bisection, Vertex Bisection, Vertex Separation, Bandwidth and Sum Cut. Recall that geometric graphs are unit disk graphs and that Theorem 4.3 states that the Cutwidth, Vertex Separation and Bandwidth problems remain **NP**-hard, even for unit disk graphs.

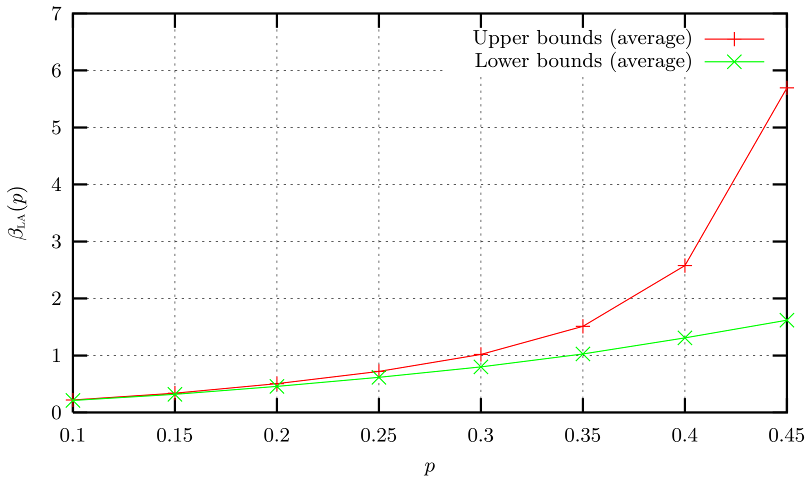


Figure 5.6: Experimental upper and lower bounds for the average value of $\beta_{\text{LA}}(p)$.

p	$\beta^{-}_{\text{LA}}(p)$	$\beta^{+}_{\text{LA}}(p)$	
0.10	0.093	0.106	Variance
	0.211	0.218	Average
0.15	0.145	0.185	Variance
	0.319	0.340	Average
0.20	0.204	0.299	Variance
	0.457	0.507	Average
0.25	0.272	0.467	Variance
	0.615	0.718	Average
0.30	0.343	0.817	Variance
	0.801	1.019	Average
0.35	0.421	0.666	Variance
	1.027	1.512	Average
0.40	0.493	0.185	Variance
	1.310	2.578	Average
0.45	0.583	26.330	Variance
	1.617	5.696	Average

Table 5.1: Experimental upper and lower bounds for the variance and average of $\beta_{\text{LA}}(p)$.

The first goal is to get lower bounds for the layout problems on random geometric graphs. Leading up to these, we will find some isoperimetric inequalities, which have some interest in their own right. The second goal is to derive asymptotics for the upper bounds obtained with two simple heuristics that we introduce. Combined with the lower bounds, we show that both heuristics are constant approximation algorithms for our layout problems, on the considered family of random geometric graphs. In the case of the Bandwidth and Vertex Separation problems, the solutions returned by either of our algorithms are asymptotically optimal. In these cases, our result is another analog of the BHH theorem. For the remaining problems, the approximation factor of the values provided by the two algorithms are tight. We emphasize that all our approximability results hold for random geometric graphs in the sense of convergence with probability 1.

We use our new results on random geometric graphs to give empirical evidence of the goodness of several well-known heuristics for layout and partitioning problems. These heuristics include global methods, such as Spectral, Multilevel and Greedy methods, and local methods, such as Simulated Annealing, Helpful Sets or Kernighan–Lin, as well as the approximation algorithms presented in this section.

5.3.1 Isoperimetric inequalities

To start, we will prove some isoperimetric inequalities that will be used latter on in this section. An isoperimetric inequality relates the size of a subgraph with the size of its boundary.

We start presenting an isoperimetric inequality on square grids with additional diagonal connections. Let (A, B) be a partition of $[m]^2$ for some integer m. Let $\partial_{A,B}$ be the number of elements of $A \times B$ that are neighbor pairs, including diagonal neighbors, that is, $\partial_{A,B} = |\{(x, y) \ : \ x \in A \wedge y \in B \wedge \|x - y\|_\infty = 1\}|$.

Proposition 5.1. For any integer $m \geqslant 3$ and any partition (A, B) of $[m]^2$, it holds that $\partial_{A,B} \geqslant 3 \min\left\{\sqrt{|A|}, \sqrt{|B|}\right\}$.

Proof. If A includes an entire row of elements, and B includes an entire row of elements, then each column includes a neighbor pair of elements, one from A and the other from B, which contributes at least 3 to $\partial_{A,B}$ except for the pair in the right-most column which contributes 1, so that $\partial_{A,B} \geqslant 3m - 2$ (see Figure 5.7). If B contains no entire row or column, and at least as many rows as columns have non-empty intersection with B, then there are at least $\sqrt{|B|}$ such rows, and each contains a neighbor pair from different sets which contributes at least 3 to $\partial_{A,B}$, so that $\partial_{A,B} \geqslant 3\sqrt{|B|}$. Applying similar arguments to the

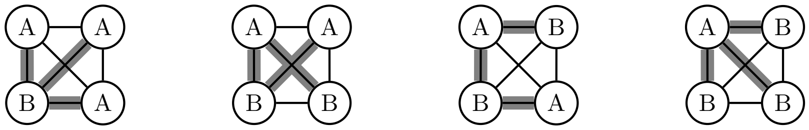


Figure 5.7: Illustration for the proof of Proposition 5.1.

other possible cases, we have

$$\partial_{A,B} \geqslant \min\left\{3\sqrt{|A|},\ 3\sqrt{|B|},\ 3m-2\right\},$$

and if $m \geqslant 3$ this minimum is always achieved at $3\sqrt{|A|}$ or at $3\sqrt{|B|}$. □

Let us present now an isoperimetric inequality for sets in $\mathbb{R}^2$. In the following, $|\cdot|$ denotes Lebesgue measure, $A+B$ denotes $\{\mathbf{x}+\mathbf{y} \ : \ \mathbf{x},\mathbf{y} \in A \times B\}$, and B_r denotes the l_∞ ball of radius r: $B_r = \{\mathbf{x} \in \mathbb{R}^2 \ : \ \|\mathbf{x}\|_\infty \leqslant r\}$. For $A \subset \mathbb{R}^2$, let $\partial_r A$ denote the set $(A + B_r) \setminus A$, and let $\partial_{-r} A$ denote the set $\partial_r(\mathbb{R}^2 \setminus A)$. Recall that a set A is compact if every open covering of A has a finite sub-covering.

Proposition 5.2. Suppose A is a compact subset of $\mathbb{R}^2$, and $r > 0$. Then,

$$|\partial_r A| \geqslant 4r\sqrt{|A|} \quad \text{and} \quad |\partial_{-r} A| \geqslant 4r\sqrt{|A|} - 16r^2.$$

Proof. By the Brunn–Minkowski inequality (see e.g. Section 1 in [166]),

$$|A + B_r| \geqslant (\sqrt{|A|} + \sqrt{|B_r|})^2 = (\sqrt{|A|} + 2r)^2 \geqslant |A| + 4r\sqrt{|A|},$$

so the first inequality follows.

Set $A^o = A \setminus \partial_{-r} A$. Then, $\partial_r(A^o) \subset \partial_{-r} A$ and hence

$$|\partial_{-r} A| \geqslant |\partial_r A^o| \geqslant 4\sqrt{|A^o|}r = 4r\sqrt{|A| - |\partial_{-r} A|}.$$

If $|\partial_{-r} A| \geqslant 4r\sqrt{|A|}$, we have the second inequality at once. Otherwise, we have

$$\begin{aligned} |\partial_{-r} A| &\geqslant 4r\sqrt{|A| - 4\sqrt{|A|}r} = 4r\sqrt{|A|}\sqrt{1 - 4r/\sqrt{|A|}} \\ &\geqslant 4r\sqrt{|A|}\left(1 - 4r/\sqrt{|A|}\right) = 4r\sqrt{|A|} - 16r^2, \end{aligned}$$

so the second inequality follows. □

We are interested here in subsets of the unit square $[0,1]^2$. For $A \subset [0,1]^2$, let A_r denote the set $(A + B_r) \cap [0,1]^2$ and let A^c denote the set $[0,1]^2 \setminus A$.

Proposition 5.3. Suppose A is a compact subset of $[0,1]^2$, and $r \in (0,1)$. Then,

$$|A_r \setminus A| \geqslant \min\left\{2r\sqrt{|A|}, 2r\sqrt{|A^c|} - 4r^2, r\right\}.$$

Proof. This proof has lots in common with the proof of Lemma 4.2, which can be regarded as its discrete analog.

Let A' be the set in $[0,1]^2$ obtained by "pushing each vertical section of A down as far as possible towards the x-axis;" formally, setting

$$S_x(A) = \{y : (x,y) \in A\}, \qquad \forall x \in [0,1],$$

let

$$A' = \bigcup_{\substack{x\in[0,1]\\ S_x(A)\neq\emptyset}} \{\{x\} \times [0, |S_x(A)|]\}.$$

We claim that $|A'| = |A|$: Let $f(x,y) = \big((x,y) \in A\big)$ and $g(x,y) = \big((x,y) \in A'\big)$. For all $x \in [0,1]$, let $f_x(y) = f(x,y)$ and $g_x(y) = g(x,y)$. By Fubini's theorem (see e.g. Chap. 12 of [217]),

$$\begin{aligned}|A| &= \int_{[0,1]^2} f(x,y)\,\mathrm{d}\Omega = \int_0^1\int_0^1 f_x(y)\,\mathrm{d}y\,\mathrm{d}x = \int_0^1 |S_x(A)|\,\mathrm{d}x\\ &= \int_0^1\int_0^1 g_x(y)\,\mathrm{d}y\,\mathrm{d}x = \int_{[0,1]^2} g(x,y)\,\mathrm{d}\Omega = |A'|.\end{aligned}$$

Moreover, we claim that $|A'_r| \leqslant |A_r|$. Indeed, consider a 1-dimensional setup. For any $W \subset [0,1]$, let W_r be the r-neighborhood of W in $[0,1]$ and let $|W|$ be the 1-dimensional measure of W. Then, if $W \neq \emptyset$, we have $|W_r| \geqslant \min\{|W| + r, 1\}$, because if $[0,1] \setminus W$ includes an interval of length at least r then $|W_r| \geqslant |W| + r$, and otherwise $W_r = [0,1]$ and so $|W_r| = 1$. Extending the argument to $A \subset [0,1]^2$, for $x \in [0,1]$,

$$S_x(A_r) = \bigcup_{\substack{u\in[0,1]\\ |u-x|\leqslant r}} (S_u(A))_r.$$

Assume $x \in [0,1]$ is such that $S_u(A) \neq \emptyset$ for some $u \in [0,1]$ with $|u - x| \leqslant r$. Then,

$$|S_x(A_r)| \geqslant \sup_{\substack{u\in[0,1]\\ |u-x|\leqslant r}} |(S_u(A))_r| \geqslant \sup_{\substack{u\in[0,1]\\ |u-x|\leqslant r}} \min\{|(S_u(A))| + r,\ 1\}.$$

With $x \in [0,1]$,

$$S_x(A'_r) = \cup_{u\in[0,1],\ |u-x|\leqslant r}[0, \min\{|S_u(A)| + r, 1\}].$$

Hence,

$$|S_x(A'_r)| \leqslant \sup_{\substack{u\in[0,1]\\ |u-x|\leqslant r}} \min\{|(S_u(A))| + r,\ 1\} \leqslant |S_x(A_r)|.$$

If $x \in [0,1]$ and $S_u(A) = \emptyset$ for all $u \in [0,1]$ with $|u-x| \leqslant r$, then $|S_x(A_r)| = 0$. Therefore, for all $x \in [0,1]$, $|S_x(A'_r)| \leqslant |S_x(A_r)|$. using again Fubini's theorem, we have

$$|A_r| = \int_0^1 |S_x(A_r)|\,\mathrm{d}x \quad \text{and} \quad |A'_r| = \int_0^1 |S_x(A'_r)|\,\mathrm{d}x,$$

and so, we get $|A'_r| \leqslant |A|$.

Let A'' be the set in $[0,1]^2$, obtained by "pushing each horizontal section of A' sideways as far as possible towards the y-axis," in an analogous manner to the construction of A' from A. Then $|A''| = |A'| = |A|$, and $|A''_r| \leqslant |A'_r| \leqslant |A_r|$. Moreover, A'' is a *down-set*, that is, it has the property that for any $\mathbf{x} \in A''$, all points of $[0,1]^2$ lying directly below or directly to the left of $\mathbf{x}$ are in A'.

From now on we assume that A is a down-set. Otherwise, it could be converted to a down-set A''. We consider four different cases:

Case 1: $(1-r, 0) \in A$ and $(0, 1-r) \notin A$. Then $S_x(A_r \setminus A)$ contains an interval of length at least r, for each $x \in [0,1]$, so that by Fubini's theorem, $|A_r \setminus A| \geqslant r$.

Case 2: $(1-r, 0) \notin A$ and $(0, 1-r) \in A$. Clearly in this case, $|A_r \setminus A| \geqslant r$ by an analogous argument to the one in Case 1.

Case 3: $(1-r, 0) \notin A$ and $(0, 1-r) \notin A$. In this case, set $A_1 = A$; let A_2 be the reflection of A in the x-axis. Let A_3 (respectively A_4) be the reflection of A_1 (respectively A_2) in the y-axis, and let $A_5 = \cup_{i=1}^4 A_i$. Then, by Lemma 5.2,

$$|A_r \setminus A| = (1/4)|\partial_r A_5| \geqslant \sqrt{|A_5|}r = 2r\sqrt{|A|}.$$

Case 4: $(1-r, 0) \in A$ and $(0, 1-r) \in A$. In this case, set $A_1 = A^c$; let A_2 be the reflection of A_1 in the line $y = 1$. Let A_3 (respectively A_4) be the reflection of A_1 (respectively A_2) in the line $x = 1$, and let $A_5 = \cup_{i=1}^4 A_i$. Then, by Lemma 5.2,

$$|A_r \setminus A| = (1/4)|\partial_{-r} A_5| \geqslant \sqrt{|A_5|} - 4r^2 = 2\sqrt{|A^c|}r - 4r^2.$$

Since one of the four cases considered above must occur, we get the lemma. $\square$

5.3.2 Lower bounds for MINEB, MINCW and MINLA

We seek now lower bounds for the optimal costs of the Edge Bisection, Cutwidth and Minimum Linear Arrangement problems on random geometric graphs whose radius is of the form $r_n = \sqrt{a_n/n}$, where $r_n \to 0$ and $a_n/\log n = b_n \to \infty$.

The following definition captures the property that vertices of a geometric graph are "nicely spread" on the unit square. The subsequent lemma is the only probabilistic result of this subsection.

Definition 5.5 (Nice graphs). Consider any set V_n of n points in $[0,1]^2$, which together with a radius r_n, induce a geometric graph $G = \mathcal{G}(V_n; r_n)$. Dissect the unit square into $4\lceil 1/r_n \rceil^2$ boxes of size $1/2\lceil 1/r_n \rceil \times 1/2\lceil 1/r_n \rceil$ placed packed in $[0,1]^2$ starting at $(0,0)$. Given $\epsilon \in (0,1)$, let us say that G is ϵ-*nice* if every box of this dissection contains at least $(1-\epsilon)\frac{1}{4}a_n$ points and at most $(1+\epsilon)\frac{1}{4}a_n$ points.[1]

Notice that in the above construction, all the boxes exactly fit in the unit square, and that any two points of V_n in neighboring boxes, including diagonals, will be connected by an edge in G because $1/2\lceil 1/r_n \rceil \leqslant \frac{1}{2}r_n$.

Lemma 5.3. Let $\epsilon \in (0, \frac{1}{5})$. Then, with probability 1, for all large enough n, random geometric graphs $\mathcal{G}(\mathcal{X}_n; r_n)$ are ϵ-nice.

Proof. Choose a box in the dissection and let Y be the random variable counting the number of points of $\mathcal{X}_n$ in this box. As the points in $\mathcal{X}_n$ are independently and uniformly distributed,

$$\mathbf{E}[Y] = \frac{n}{4\lceil 1/r_n \rceil^2} \sim \frac{nr_n^2}{4} = \tfrac{1}{4}a_n.$$

Using Chernoff's bounds we obtain

$$\begin{aligned}\mathbf{Pr}\left[Y \geqslant (1+\epsilon)\tfrac{1}{4}a_n\right] &\leqslant \mathbf{Pr}\left[Y \geqslant (1+\tfrac{1}{2}\epsilon)\mathbf{E}[Y]\right] \leqslant \exp\left(-\left(\tfrac{1}{2}\epsilon\right)^2 \mathbf{E}[Y]/3\right)\\ &\leqslant \exp\left(-\tfrac{1}{13}\epsilon^2\tfrac{1}{4}a_n\right) = n^{-\epsilon^2 b_n/52}\end{aligned}$$

and

$$\begin{aligned}\mathbf{Pr}\left[Y \leqslant (1-\epsilon)\tfrac{1}{4}a_n\right] &\leqslant \mathbf{Pr}\left[Y \leqslant (1-\tfrac{1}{2}\epsilon)\mathbf{E}[Y]\right] \leqslant \exp\left(-\left(\tfrac{1}{2}\epsilon\right)^2 \mathbf{E}[Y]/2\right)\\ &\leqslant \exp\left(-\tfrac{1}{9}\epsilon^2\tfrac{1}{4}a_n\right) = n^{-\epsilon^2 b_n/36} \leqslant n^{-\epsilon^2 b_n/52}.\end{aligned}$$

[1] To be completely rigorous, this definition should also state that no vertex of V_n falls in the boundary of a box. As in our study V_n will be a set of random points in the unit square, the probability that some vertex of V_n falls in the boundary of two or more boxes is zero. In the following, we will always make use of this assumption.

The number of boxes is certainly smaller than n, so by Boole's inequality, the probability that for some box the number of points in the box is less than $(1-\epsilon)\frac{1}{4}a_n$ or bigger than $(1+\epsilon)\frac{1}{4}a_n$, is bounded by $2n^{1-b_n\epsilon^2/52}$. As $b_n \to \infty$,

$$\sum_{n\in\mathbb{N}} 2n^{1-b_n\epsilon^2/52} < \infty.$$

The result follows by the Borel–Cantelli lemma. □

The following lemma is the basis of our lower bounds for nice graphs.

Lemma 5.4. Let $\epsilon \in (0, \frac{1}{5})$ and n large enough. Let G be any ϵ-nice geometric graph with n vertices and radius r_n and let φ be any layout of G. Then, for any integer i such that $\alpha = i/n \in (2\epsilon, 1-2\epsilon)$, it is the case that

$$\theta(i,\varphi,G) \geqslant \tfrac{3(1-5\epsilon)}{8} \min\left\{\sqrt{\alpha-2\epsilon}, \sqrt{1-\alpha-2\epsilon}\right\} n^2 r_n^3.$$

Proof. We assign colors to vertices and boxes: color the first i vertices in the ordering "red" and the other vertices "green;" color the boxes containing at most $\frac{1}{5}\epsilon a_n$ green vertices "red," the boxes containing at most $\frac{1}{5}\epsilon a_n$ red vertices "green," and the other boxes "yellow." Let Y_n be the number of yellow boxes.

Observe that $\theta(i,\varphi,G)$ is the total number of edges between opposite-color vertices. Let us refer to such edges as "within-box" if the vertices in question lie in the same box, or "between-box" otherwise. We consider two cases:

Case 1: $Y_n \geqslant 25\epsilon^{-2}\sqrt{n}/\sqrt{a_n}$. For each yellow box, the cost of within-box edges is at least $\epsilon^2 a_n^2/25$. Hence,

$$\theta(i,\varphi,G) \geqslant (\tfrac{1}{5}\epsilon a_n)^2 \cdot Y_n \geqslant n^{1/2}a_n^{3/2} = n^2 r_n^3.$$

Case 2: $Y_n < 25\epsilon^{-2}\sqrt{n}/\sqrt{a_n}$. In this case, we consider only between-box edges, i.e. between opposite colored vertices, which are between neighboring boxes, including diagonal neighbors. Consider a particular box containing a total of t vertices, r of them red. Suppose that the total number of red vertices in neighboring boxes is r' and the total number of green vertices in neighboring boxes is g'. Then, the total number of between-box edges of the type we are considering, involving vertices in that particular box, is

$$rg' + (t-r)r' = r(g'-r') + tr'.$$

Given t, r' and g', the equation above is a linear function of r and so it attains its minimum over the range $[0,t]$ either at $r=0$, at $r=t$ or at both. Hence, it is possible to change the vertices in that box to either all red or all green without increasing the total number of between-box edges of the type we are considering.

Let us modify the coloring of vertices by going through the yellow boxes in turn, successively changing the color of vertices in each box either to all red or to all green, whichever does not increase the total number of between-box edges of the type we are considering. When done, there will no longer be any yellow boxes! Let R_n be the number of red boxes and G_n the number of green boxes based on this modified coloring.

By niceness, the number of vertices whose color has been changed is at most

$$Y_n \cdot (1+\epsilon)\tfrac{1}{4}a_n \leqslant 25\epsilon^{-2}\sqrt{n}\sqrt{a_n} = 25\epsilon^{-2}nr_n.$$

Thus, for n so large that $25\epsilon^{-2}r_n \leqslant \epsilon$, the number of red vertices in the modified coloring is at least $(\alpha-\epsilon)n$.

By definition of "green box," the number of red vertices in green boxes is at most $\frac{1}{5}\epsilon a_n 4\lceil 1/r_n\rceil^2 \leqslant \epsilon n$. Thus, in the modified coloring, the total number of red vertices in red boxes is at least $(\alpha-2\epsilon)n$ for n large enough. By a similar argument, the number of green vertices in green boxes in the modified coloring is at least $(1-\alpha-2\epsilon)n$ for n large enough.

As, by ϵ-niceness, no box can contain more than $\frac{1}{4}(1+\epsilon)a_n$ vertices and there are at least $(\alpha-2\epsilon)n$ red vertices in red boxes and $(1-\alpha-2\epsilon)n$ green vertices in green boxes, we have

$$R_n \geqslant \frac{(\alpha-2\epsilon)n}{\frac{1}{4}(1+\epsilon)a_n} = \frac{4(\alpha-2\epsilon)}{(1+\epsilon)r_n^2}$$

and

$$G_n \geqslant \frac{(1-\alpha-2\epsilon)n}{\frac{1}{4}(1+\epsilon)a_n} = \frac{4(1-\alpha-2\epsilon)}{(1+\epsilon)r_n^2}.$$

Let ∂G denote the number of pairs of neighbor boxes of opposite colors in G with the modified coloring. By Proposition 5.1,

$$\partial G \geqslant 3\min\left\{\sqrt{R_n},\ \sqrt{G_n}\right\} \geqslant \frac{6}{r_n}\min\left\{\sqrt{\frac{\alpha-2\epsilon}{1+\epsilon}},\ \sqrt{\frac{1-\alpha-2\epsilon}{1+\epsilon}}\right\}.$$

By niceness and the definition of box coloring, each red box contains at least $(1-2\epsilon)\frac{1}{4}a_n$ red vertices, and each green box contains at least $(1-2\epsilon)\frac{1}{4}a_n$ green vertices. As a consequence,

$$\begin{aligned}\theta(i,\varphi,G) &\geqslant \partial G\tfrac{(1-2\epsilon)^2}{16}a_n^2 \geqslant \tfrac{3(1-2\epsilon)^2}{8\sqrt{1+\epsilon}}a_n^{3/2}\sqrt{n}\min\left\{\sqrt{\alpha-2\epsilon},\ \sqrt{1-\alpha-2\epsilon}\right\}\\ &\geqslant \tfrac{3(1-5\epsilon)}{8}n^2r_n^3\min\left\{\sqrt{\alpha-2\epsilon},\ \sqrt{1-\alpha-2\epsilon}\right\}.\end{aligned}$$

This lower bound is smaller than the one for Case 1 and thus it holds for both cases. □

The following result presents our lower bounds for the Edge Bisection, Cutwidth and Minimum Linear Arrangement problems on nice graphs. Latter on, we will show these lower bounds are sharp, since their order of magnitude match the upper bounds that we will obtain with two heuristics.

Theorem 5.9. Let $\epsilon \in (0, \frac{1}{5})$ and n large enough. Let G be an ϵ-nice geometric graph with n vertices and radius r_n. Then, the following lower bounds hold:

$$\text{MINEB}(G) \geqslant \tfrac{3(1-8\epsilon)}{8\sqrt{2}} \cdot n^2 r_n^3, \tag{5.6}$$

$$\text{MINCW}(G) \geqslant \tfrac{3(1-8\epsilon)}{8\sqrt{2}} \cdot n^2 r_n^3, \tag{5.7}$$

$$\text{MINLA}(G) \geqslant \tfrac{(1-42\sqrt{\epsilon})}{4\sqrt{2}} \cdot n^3 r_n^3. \tag{5.8}$$

Proof. The proofs of (5.6) and (5.7) are obtained from Lemma 5.4 by setting $i = \lfloor n/2 \rfloor$. To prove (5.8), take any layout φ of G. Then, by Lemma 5.4,

$$\begin{aligned}
\text{LA}(\varphi, G) &= \sum_{i=1}^{n} \theta(i, \varphi, G) \\
&\geqslant \sum_{2\epsilon n < i < (1-2\epsilon)n} \theta(i, \varphi, G) \\
&\geqslant \tfrac{3}{4}(1-5\epsilon) n^2 r_n^3 \sum_{2\epsilon n < i < n/2} \sqrt{i/n - 2\epsilon}.
\end{aligned}$$

Using the facts that $a > b$ implies $\sqrt{a-b} \geqslant \sqrt{a} - \sqrt{b}$ and that $\sum_{k=1}^{m} \sqrt{k} \geqslant \frac{2}{3} m^{3/2} + O(\sqrt{m})$ we obtain (5.8) by successive minorizations. □

5.3.3 Lower bounds for MINVS, MINSC, MINVB and MINBW

Lower bounds for the optimal costs of the Vertex Separation, Sum Cut and Bandwidth problems could also be obtained for nice graphs. However, we shall obtain tighter lower bounds taking a finer dissection:

Definition 5.6 (Good graphs). Again, we consider any set V_n of n points in $[0, 1]^2$, which together with a radius r_n induce a random geometric graph $G = \mathcal{G}(V_n; r_n)$. Given a constant $\epsilon \in (0, 1)$, G is said to be ϵ-*good* if setting

$$\gamma = \left(\left\lfloor \frac{2}{\epsilon r_n} \right\rfloor r_n \right)^{-1}$$

and dissecting the unit square into $(\gamma r_n)^{-2}$ boxes, each of size $\gamma r_n \times \gamma r_n$, every box contains at most $p_+ = (1+\gamma)\gamma^2 a_n$ points and at least $p_- = (1-\gamma)\gamma^2 a_n$ points.

Observe that $1/(\gamma r_n)$ is an integer and that $\gamma \to \epsilon/2$ so that $\epsilon/4 \leqslant \gamma \leqslant \epsilon$ for n large enough.

Lemma 5.5. Let $\epsilon \in (0,1)$. Then, with probability 1, for all large enough n, random geometric graphs $\mathcal{G}(\mathcal{X}_n; r_n)$ are ϵ-good.

Proof. Choose a box in the dissection and let Y be the random variable counting the number of points of $\mathcal{X}_n$ in this box. As the points in $\mathcal{X}_n$ are independent and uniformly distributed, we have $\mathbf{E}[Y] = \gamma^2 a_n$ where $\gamma = 1/\left(\left\lfloor \frac{2}{\epsilon r_n} \right\rfloor r_n\right)$. By Chernoff's bounds and Boole's inequality, used as in the proof of Lemma 5.3, the probability that some box has more than p_+ points or fewer than p_- points is bounded by

$$2\left(\frac{1}{\gamma r_n}\right)^2 \exp\left(-\gamma^2(\gamma^2 a_n)/3\right) = \frac{2n}{\gamma^2 a_n} n^{-\gamma^4 b_n/3} \leqslant \frac{2}{\gamma^2} n^{1-\gamma^4 b_n/3},$$

which is summable in n because $b_n \to \infty$ as $n \to \infty$. The result follows by the Borel–Cantelli lemma. □

The following lemma is the basis of our lower bounds for good graphs.

Lemma 5.6. Let $\epsilon \in (0,1)$ and n large enough. Let G be an ϵ-good geometric graph with n vertices and radius r_n. Let $i \in [n]$ and consider any ordering φ on G. Then,

$$\delta(i, \varphi, G) \geqslant (1 - 3\epsilon)\left(h(i/n) - 2\sqrt{\epsilon} - 4r_n\right) \cdot n r_n, \tag{5.9}$$

where for $x \in [0,1]$ we set $h(x) = \min\left\{2\sqrt{x},\ 2\sqrt{(1-x)},\ 1\right\}$.

Proof. With $[0,1)^2$ divided into boxes of side γr_n where $\gamma = 1/\left(\left\lfloor \frac{2}{\epsilon r_n} \right\rfloor r_n\right)$, we say that two boxes are *adjacent* if the l_∞ distance between their centers is at most $(1-\gamma)r_n$. Note that under the l_∞ norm, any two points in adjacent boxes are at distance at most r_n from each other.

Given a layout φ, let the first i points be denoted "red" and the others "green." Then $\delta(i, \varphi, G)$ is the number of red points of $\mathcal{X}_n$ having one or more green points within a distance r_n. Let $\delta'(i, \varphi, G)$ be number of red points X such that there is at least one green point lying either in the box containing X or in a box adjacent to the box containing X. Then $\delta'(i, \varphi, G) \leqslant \delta(i, \varphi, G)$. We shall show that the right side of (5.9) is a lower bound for $\delta'(i, \varphi, G)$.

Given φ, let boxes containing only red points be denoted "red," let boxes containing only green points be denoted "green," and let the other boxes be denoted "yellow." Note that $\delta'(i, \varphi, G)$ is the number of red points X for which the box containing X is either itself yellow, or has some non-red box adjacent to it.

We claim that there is an ordering φ on $\mathcal{X}_n$ minimizing $\delta'(i,\cdot,G)$ such that φ induces at most one yellow box. Indeed, given an ordering φ inducing more than one yellow box, choose an ordering on yellow boxes. It is then possible to modify φ to an ordering φ' which respects the chosen ordering on yellow boxes of φ, and which satisfies $\delta'(i,\varphi',G) \leqslant \delta'(i,\varphi,G)$. This can be done by successively swapping red and green points, with each swap not increasing δ'.

Thus, without loss of generality, we can assume that φ induces at most one yellow box. Set $\alpha = i/n$ and let N_R be the number of red boxes. Then, by goodness and the fact that there are αn red points,

$$\frac{\alpha}{(1+\gamma)(\gamma r_n)^2} - 1 \leqslant N_R \leqslant \frac{\alpha}{(1-\gamma)(\gamma r_n)^2}. \tag{5.10}$$

Let A_R be the union of the red boxes and let $A_G = [0,1]^2 \setminus A_R$, be the union of green and yellow boxes. Since each box has area $(\gamma r_n)^2$, by (5.10) we have

$$|A_R| \geqslant \frac{\alpha}{1+\gamma} - (\gamma r_n)^2 \quad \text{and} \quad |A_G| \geqslant 1 - \frac{\alpha}{1-\gamma}.$$

Let B be the union of red boxes that are adjacent to green or yellow boxes. Then $B = (A_G)_{(1-\gamma)r_n} \setminus A_G$, and therefore, by Proposition 5.3,

$$\begin{aligned}
|B| &\geqslant r_n \min\left\{2(1-\gamma)\sqrt{|A_G|},\ 2(1-\gamma)\sqrt{|A_R|} - 4((1-\gamma)r_n)^2,\ (1-\gamma)\right\}\\
&\geqslant (1-\gamma)r_n \min\left\{2\sqrt{1-\frac{\alpha}{1-\gamma}},\ 2\sqrt{\frac{\alpha}{1+\gamma} - (\gamma r_n)^2} - 4r_n,\ 1\right\}\\
&\geqslant (1-\gamma)r_n \left(\min\left\{2\sqrt{1-\frac{\alpha}{1-\gamma}},\ 2\sqrt{\frac{\alpha}{1+\gamma} - (\gamma r_n)^2},\ 1\right\} - 4r_n\right).
\end{aligned}$$

Using the fact that $a > b$ implies $\sqrt{a-b} \geqslant \sqrt{a} - \sqrt{b}$, we have

$$\sqrt{1-\frac{\alpha}{1-\gamma}} \geqslant \frac{\sqrt{1-\alpha}-\sqrt{\gamma}}{\sqrt{1-\gamma}} \geqslant \sqrt{1-\alpha} - \sqrt{\gamma} \geqslant (1-\gamma)\left(\sqrt{1-\alpha}-\sqrt{\gamma}\right).$$

As $r_n \to 0$, for n large enough, $r_n \leqslant /\sqrt{\gamma}$. Since $1/\sqrt{(1+\gamma)} \geqslant 1-\gamma$, we have

$$\begin{aligned}
\sqrt{\frac{\alpha}{1+\gamma} - (\gamma r_n)^2} &\geqslant \frac{\sqrt{\alpha - \gamma^2 r_n^2(1+\gamma)}}{\sqrt{1+\gamma}} \geqslant (1-\gamma)\sqrt{\alpha - \gamma^2 r_n^2 (1+\gamma)}\\
&\geqslant (1-\gamma)\left(\sqrt{\alpha} - \gamma r_n\sqrt{1+\gamma}\right) \geqslant (1-\gamma)\left(\sqrt{\alpha}-\sqrt{\gamma}\right).
\end{aligned}$$

Therefore,

$$\begin{aligned}
|B| \geqslant (1-\gamma)r_n \big(\min\big\{&2(1-\gamma)\left(\sqrt{1-\alpha}-\sqrt{\gamma}\right),\\
&2(1-\gamma)\left(\sqrt{\alpha}-\sqrt{\gamma}\right),\ 1\big\} - 4r_n\big)
\end{aligned}$$

$$\begin{aligned}
&\geqslant (1-\gamma)^2 r_n \left(\min\left\{2\sqrt{1-\alpha}-2\sqrt{\gamma},\ 2\sqrt{\alpha}-2\sqrt{\gamma},\ 1-2\sqrt{\gamma}\right\}-4r_n\right)\\
&\geqslant (1-\gamma)^2 r_n \left(\min\left\{2\sqrt{1-\alpha},\ 2\sqrt{\alpha},\ 1\right\}-2\sqrt{\gamma}-4r_n\right). \qquad (5.11)
\end{aligned}$$

Since B is a union of disjoint boxes, each one with area $\gamma^2 r_n^2$, the number of such boxes is their total area divided by $\gamma^2 r_n^2$. By goodness, the number of points lying in the region B is bounded below by $(1-\gamma)n$ times its area. Hence by (5.11) we obtain

$$\begin{aligned}
\delta'(i,\varphi,G) &\geqslant (1-3\gamma)nr_n(h(\alpha)-2\sqrt{\gamma}-4r_n)\\
&\geqslant (1-3\epsilon)nr_n(h(\alpha)-2\sqrt{\epsilon}-4r_n),
\end{aligned}$$

which proves the lemma. □

The following result presents our lower bounds for the Vertex Separation, Sum Cut, Vertex Bisection and Bandwidth problems on good graphs.

Theorem 5.10. Let $\epsilon\in(0,1)$ and n large enough. Let G be an ϵ-good geometric graph with n vertices and radius r_n. Then, the following lower bounds hold:

$$\textsc{minvb}(G) \geqslant (1-6\sqrt{\epsilon})\cdot nr_n \qquad (5.12)$$

$$\textsc{minvs}(G) \geqslant (1-6\sqrt{\epsilon})\cdot nr_n \qquad (5.13)$$

$$\textsc{minbw}(G) \geqslant (1-6\sqrt{\epsilon})\cdot nr_n \qquad (5.14)$$

$$\textsc{minsc}(G) \geqslant \left(\tfrac{5}{6}-6\sqrt{\epsilon}\right)\cdot n^2 r_n \qquad (5.15)$$

Proof. The proof of (5.12) is obtained directly from Lemma 5.6 by taking $i=\lceil n/2\rceil$ and the proof of (5.13) is obtained directly from Lemma 5.6 by taking i with $\frac{1}{4}\leqslant i/n\leqslant\frac{3}{4}$, so that $h(i/n)=1$.

The proof (5.14) follows from Lemma 1.1.

To prove (5.15), consider any layout φ of G. Then, using again Lemma 5.6, we have that for large enough n,

$$\textsc{sc}(\varphi,G)=\sum_{i=1}^{n}\delta(i,\varphi,G)\geqslant(1-3\epsilon)nr_n\sum_{i=1}^{n}\left(h(i/n)-2\sqrt{\epsilon}-4r_n\right).$$

As

$$\begin{aligned}
&\sum_{i=1}^{n}\left(h(i/n)-2\sqrt{\epsilon}-4r_n\right)\geqslant\\
&\qquad\geqslant\left(4\sum_{1\leqslant i\leqslant n/4}\sqrt{\tfrac{i}{n}}\right)+\left(\sum_{n/4<i<3n/4}1\right)-\left(\sum_{i=1}^{n}2\sqrt{\epsilon}+4r_n\right),
\end{aligned}$$

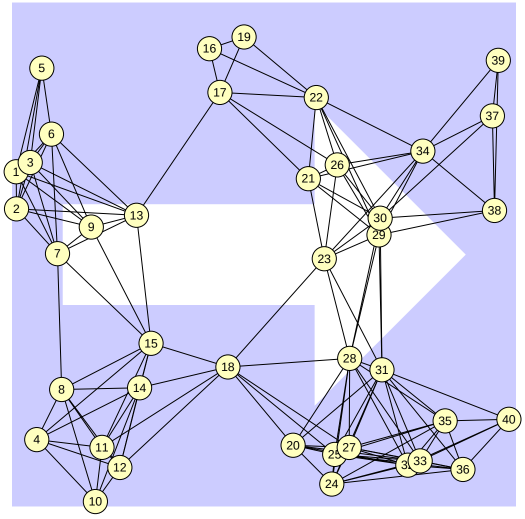


Figure 5.8: Illustration of the projection algorithm.

and $\sum_{k=1}^{m} \sqrt{k} \geqslant \frac{2}{3} m^{3/2} + O\left(\sqrt{m}\right)$, we obtain

$$\textsc{sc}(\varphi, G) \geqslant \left(\tfrac{5}{6} - 6\sqrt{\epsilon}\right) \cdot n^2 r_n,$$

which proves the theorem. $\square$

5.3.4 Approximation algorithms

We now present two heuristics, denoted Projection and Dissection, that compute feasible layouts of random geometric graphs. Using again the notion of ϵ-good graphs introduced in Definition 5.6, we show that the costs of these layouts are within a constant of the previous lower bounds. In the following, the coordinates of a point $u \in [0, 1]^2$ are denoted by $x(u)$ and $y(u)$.

Algorithm 5.1 (Projection). The *Projection algorithm* creates a layout by ordering the vertices according to their projection onto the x-axis. Another way to see this algorithm is to sweep a vertical line starting from $x = 0$ to $x = 1$, numbering vertices in the order the line touches them.

Figure 5.8 illustrates this algorithm.

The expected running time of the Projection algorithm to compute the projected layout of a random geometric graph is linear, as it requires only the ranking of n numbers distributed uniformly.

Theorem 5.11. Let $\epsilon \in (0,1)$ and n large enough. Let G be an ϵ-good geometric graph with n vertices and radius r_n. Then, the following upper bounds on the cost of the projected layout π of G hold:

$$\textsc{cw}(\pi,G) \leqslant (1+3\epsilon)^5 \cdot n^2 r_n^3, \tag{5.16}$$
$$\textsc{eb}(\pi,G) \leqslant (1+3\epsilon)^5 \cdot n^2 r_n^3, \tag{5.17}$$
$$\textsc{bw}(\pi,G) \leqslant (1+3\epsilon)^2 \cdot n r_n, \tag{5.18}$$
$$\textsc{vb}(\pi,G) \leqslant (1+3\epsilon)^2 \cdot n r_n, \tag{5.19}$$
$$\textsc{la}(\pi,G) \leqslant (1+3\epsilon)^5 \cdot n^3 r_n^3, \tag{5.20}$$
$$\textsc{vs}(\pi,G) \leqslant (1+3\epsilon)^2 \cdot n r_n, \tag{5.21}$$
$$\textsc{sc}(\pi,G) \leqslant (1+3\epsilon)^2 \cdot n^2 r_n. \tag{5.22}$$

Proof. Given a vertex u of G, let $\theta(u)$ denote the cut induced by the projected layout π on u, that is, the number of edges vw such that $x(v) \leqslant x(u)$ and $x(u) < x(w)$. Given an edge uv from G, let $\lambda(uv)$ denote the length induced by π on uv, that is, the number of vertices w such that $x(u) < x(w)$ and $x(w) < x(v)$. Set $\gamma = 1/\left(\left\lfloor \frac{2}{\epsilon r_n} \right\rfloor r_n\right)$ and $k = \lceil 1/\gamma \rceil \leqslant 1/\gamma + 1$. Recall that $\frac{1}{4}\epsilon \leqslant \gamma \leqslant \epsilon$. Every possible edge is between boxes of side γr_n, with centers at distance at most r_n. Thus,

$$\theta(u) \leqslant \sum_{0 \leqslant i \leqslant k} (k+1-i)\,(2k+1)\,\frac{1}{\gamma r_n} p_+^2 \leqslant (1+3\epsilon)^5 n^2 r_n^3. \tag{5.23}$$

On the other hand, observe that $\lambda(uv)$ is bounded above by the number of possible vertices in the columns of boxes between the column of u and the column of v. Thus,

$$\lambda(uv) \leqslant p_+\,(k+2)\,\frac{1}{\gamma r_n} \leqslant (1+3\epsilon)^2 n r_n. \tag{5.24}$$

Bounds (5.16), (5.17) and (5.18) follow directly from Equations (5.23) and (5.24). Bounds (5.20), (5.21), (5.19) and (5.22) hold because for any layout φ, it is the case that $\textsc{la}(\varphi,G) \leqslant n\textsc{cw}(\varphi,G)$, $\textsc{vb}(\varphi,G) \leqslant \textsc{vs}(\varphi,G) \leqslant \textsc{bw}(\varphi,G)$ and $\textsc{sc}(\varphi,G) \leqslant n\textsc{vs}(\varphi,G)$, as seen in Lemma 1.1. ☐

Next we give lower bounds on the costs of the layouts delivered by the Projection algorithm on good graphs:

Lemma 5.7. Let $\epsilon \in (0,1)$ and n large enough. Let G be an ϵ-good geometric graph with n vertices and radius r_n. Then, the following lower bounds on the cost of the projected layout π of G hold:

$$\textsc{vs}(\pi,G) \geqslant (1-3\epsilon) \cdot n r_n, \tag{5.25}$$

$$
\begin{aligned}
\text{VB}(\pi, G) &\geqslant (1-3\epsilon)\cdot nr_n, &(5.26)\\
\text{BW}(\pi, G) &\geqslant (1-3\epsilon)\cdot nr_n, &(5.27)\\
\text{SC}(\pi, G) &\geqslant (1-5\epsilon)\cdot n^2 r_n, &(5.28)\\
\text{CW}(\pi, G) &\geqslant (1-8\epsilon)\cdot n^2 r_n^3, &(5.29)\\
\text{EB}(\pi, G) &\geqslant (1-8\epsilon)\cdot n^2 r_n^3, &(5.30)\\
\text{LA}(\pi, G) &\geqslant (1-10\epsilon)\cdot n^3 r_n^3. &(5.31)
\end{aligned}
$$

Proof. Set $\gamma = 1/\left(\left\lfloor \frac{2}{\epsilon r_n} \right\rfloor r_n\right)$ and $k = \lceil 1/\gamma \rceil$. Let us prove (5.25). Consider any vertex u far enough from the square boundaries. All the vertices in the $k-2$ columns preceding the column of u must be connected to some vertex in the next column after the column of u. Therefore,

$$\text{VS}(\pi, G) \geqslant p_-(k-2)\frac{1}{\gamma r_n} \geqslant (1-3\epsilon)nr_n.$$

Bound (5.26) follows by the same argument, and bound (5.27) follows because of Lemma 1.1.

Next, let us prove (5.28). We can extend the previous proof to all the points which are away from the left and the right borders of the unit square:

$$
\begin{aligned}
\text{SC}(\pi, G) &\geqslant p_-\frac{1}{\gamma r_n}\left(\frac{1}{\gamma r_n} - 2k\right)\left(p_-(k-2)\frac{1}{\gamma r_n}\right)\\
&\geqslant (a_n^2/r_n^3)(1-\gamma)^3(1-2\gamma) \geqslant n^2 r_n(1-5\epsilon).
\end{aligned}
$$

We prove now (5.29) and (5.30). Take any vertex u away from the left and right borders of $[0,1]^2$. We have,

$$
\begin{aligned}
\text{CW}(\pi, G) &\geqslant \sum_{i=1}^{k-2} p_-^2(k-i-1)\left(\frac{1}{\gamma r_n} - 2k\right)(2k-3)\\
&\geqslant (1-2\gamma)^2(1-\gamma)^4 a_n^2/r_n \geqslant (1-8\epsilon)n^2 r_n^3.
\end{aligned}
$$

As the $\left\lfloor \frac{1}{2}n \right\rfloor$-th vertex of the projected layout must be away from the left and right borders of $[0,1]^2$, we have the same result for $\text{EB}(\pi, G)$.

Finally, let us prove (5.31). By the same argument we used for the cut-width,

$$
\begin{aligned}
\text{LA}(\pi, G) &\geqslant p_-\frac{1}{\gamma r_n}\left(\frac{1}{\gamma r_n} - 2k\right)\sum_{i=1}^{k-2} p_-^2(k-i-1)\left(\frac{1}{\gamma r_n} - 2k\right)(2k-3)\\
&\geqslant (1/r_n^2)(1-2\gamma)^2(1-6\epsilon)a_n^2/r_n \geqslant (1-10\epsilon)n^3 r_n^3.
\end{aligned}
$$

This concludes the proof. □

The behavior of the Projection algorithm on good graphs is characterized by the following consequence of Theorem 5.11 and Lemma 5.7:

Theorem 5.12. Let $\epsilon \in (0,1)$ and n large enough. Let G be an ϵ-good geometric graph with n vertices and radius r_n. Then, for any layout measure $f \in \{\text{BW}, \text{VS}, \text{VB}, \text{SC}, \text{CW}, \text{EB}, \text{LA}\}$, we have

$$(1-10\epsilon) \leqslant \frac{f(\pi, G)}{A_f} \leqslant (1+3\epsilon)^5,$$

where

$$\begin{array}{llll} A_{\text{BW}} = nr_n, & A_{\text{VS}} = nr_n, & A_{\text{VB}} = nr_n, & A_{\text{SC}} = n^2 r_n, \\ A_{\text{CW}} = n^2 r_n^3, & A_{\text{EB}} = n^2 r_n^3, & A_{\text{LA}} = n^3 r_n^3. & \end{array}$$

As a consequence of Lemmas 5.3 and 5.5, Theorems 5.9, 5.10 and 5.12, we obtain the following approximability result:

Theorem 5.13. Let $(r_i)_{i\geqslant 1}$ be a sequence of positive numbers with $r_n \to 0$ and $nr_n^2/\log n \to \infty$; let $(X_i)_{i\geqslant 1}$ be a sequence of independent and uniformly distributed random points in $[0,1]^2$. For all $n \in \mathbb{N}$, let $G_n = \mathcal{G}(\mathcal{X}_n; r_n)$; then, with probability 1:

$$\begin{array}{rcccccc}
 & & \displaystyle\lim_{n\to\infty} \frac{\text{MINBW}(G_n)}{nr_n} & = 1, & & & \\
 & & \displaystyle\lim_{n\to\infty} \frac{\text{MINVS}(G_n)}{nr_n} & = 1, & & & \\
 & & \displaystyle\lim_{n\to\infty} \frac{\text{MINVB}(G_n)}{nr_n} & = 1, & & & \\
1 & \geqslant & \displaystyle\limsup_{n\to\infty} \frac{\text{MINSC}(G_n)}{n^2 r_n} & \geqslant & \displaystyle\liminf_{n\to\infty} \frac{\text{MINSC}(G_n)}{n^2 r_n} & \geqslant & \frac{5}{6}, \\
1 & \geqslant & \displaystyle\limsup_{n\to\infty} \frac{\text{MINCW}(G_n)}{n^2 r_n^3} & \geqslant & \displaystyle\liminf_{n\to\infty} \frac{\text{MINCW}(G_n)}{n^2 r_n^3} & \geqslant & \frac{3}{8\sqrt{2}}, \\
1 & \geqslant & \displaystyle\limsup_{n\to\infty} \frac{\text{MINEB}(G_n)}{n^2 r_n^3} & \geqslant & \displaystyle\liminf_{n\to\infty} \frac{\text{MINEB}(G_n)}{n^2 r_n^3} & \geqslant & \frac{3}{8\sqrt{2}}, \\
1 & \geqslant & \displaystyle\limsup_{n\to\infty} \frac{\text{MINLA}(G_n)}{n^3 r_n^3} & \geqslant & \displaystyle\liminf_{n\to\infty} \frac{\text{MINLA}(G_n)}{n^3 r_n^3} & \geqslant & \frac{1}{4\sqrt{2}}.
\end{array}$$

From the previous theorem we obtain the following approximation result.

Theorem 5.14. Let $(r_i)_{i\geqslant 1}$ be a sequence of positive numbers with $r_n \to 0$ and $nr_n^2/\log n \to \infty$; let $(X_i)_{i\geqslant 1}$ be a sequence of independent and uniformly distributed random points in $[0,1]^2$. Then, with probability 1, and for all large enough n, the Projection algorithm is a constant approximation algorithm for the Bandwidth, Minimum Linear Arrangement, Minimum Cut, Minimum Sum Cut, Vertex Separation, Edge Bisection and Vertex Bisection problems on random geometric graphs $\mathcal{G}(\mathcal{X}_n; r_n)$. Moreover, for the Bandwidth, Vertex Bisection and Vertex Separation problems, the Projection algorithm is asymptotically optimal.

As seen in Section 5.1.3, Karp's analysis on the dissection algorithm for the TSP problem was an important idea in combinatorial optimization on the plane [146]. Let us adapt his algorithm to layout problems and random geometric graphs.

Algorithm 5.2 (Dissection). For a geometric graph with n vertices and radius r_n, our Dissection algorithm, parameterized by a constant $\kappa > 0$, is given by the following steps:

1. Dissect $[0,1]^2$ into boxes of size $r_n/\kappa \times r_n/\kappa$.
2. Enumerate the points, following the order of the boxes in lexicographic order: from bottom to top and from left to right. Enumerate arbitrarily the points in the same box.

Figure 5.9 shows an example of the application of this algorithm.

Using the same kind of arguments used for the Projection algorithm, we can prove the following result:

Theorem 5.15. Let $(r_i)_{i\geqslant 1}$ be a sequence of positive numbers with $r_n \to 0$ and $nr_n^2/\log n \to \infty$; let $(X_i)_{i\geqslant 1}$ be a sequence of independent and uniformly distributed random points in $[0,1]^2$. Then, with probability 1, for all large enough n, the Dissection algorithm is a constant approximation algorithm for the Bandwidth, Minimum Linear Arrangement, Minimum Cut, Minimum Sum Cut, Vertex Separation, Edge Bisection and Vertex Bisection problems random geometric graphs $\mathcal{G}(\mathcal{X}_n; r_n)$. Moreover, for the Bandwidth, Vertex Separation and Vertex Bisection problems, the Dissection algorithm is asymptotically optimal.

5.3.5 Experimental considerations

At this point, we have much more information of layout problems on random geometric graphs than in Chapter 2. It thus makes sense to think how the new results can be used in practice. In this subsection, we present several computational experiments we have performed in order to obtain a better knowledge of

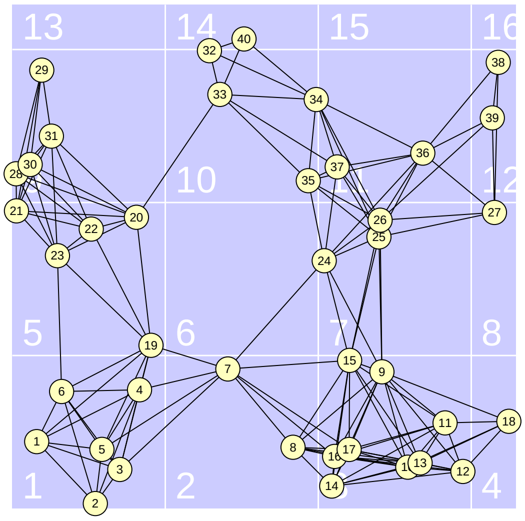


Figure 5.9: Illustration of the dissection algorithm.

layout problems on several random geometric graphs. The goal was to complement and expand the theoretical results presented in this section.

Behavior of the Projection and Dissection algorithms. According to Theorem 5.12, the Projection and Dissection algorithms exhibit a behavior such that for any measure $F \in \{\text{BW}, \text{VS}, \text{SC}, \text{CW}, \text{EB}, \text{LA}\}$, we have that $F(\pi, G)/A_F$ and $F(\psi, G)/A_F$ are as close to 1 as we wish. However, this result is asymptotic in nature. The following computational experiment was designed in order to compare the behavior of these two algorithms for the different measures. The aim was to compare between the algorithms, and to compare the experimental results with the predicted ones.

To conduct the experiment, we have generated random geometric graphs with $r_n = \sqrt{(\log n)(\log\log n)/n}$ with up to 200,000 vertices. For each each value of n, we computed the average of all the considered measures F using the Projection algorithm and the Dissection algorithm (with $\kappa \in \{1, 2, 4, 8, 16\}$) and normalized by their respective orders of magnitude, A_F, as given by Theorem 5.12.

Figures 5.10 and 5.11 show the obtained results. Note that the plots have neither the same scaling nor the same origin. The standard deviation was very low in all cases.

With regard to the competitive analysis, from the results of this exper-

	gai	far	spl	ine	spm	mul	pro
—	2.36	3.51	164.51	0.48	23.01	18.44	0.44
kl	11.58	10.28	170.71	8.26	29.12	21.97	6.49
hs	6.36	6.78	167.47	3.42	26.18	21.73	3.60

Table 5.2: Running times (in seconds) for bisecting a random geometric graph with 10000 vertices and $5,534,888$ edges.

iment, we can observe that, for all the measures except EB, the Projection algorithm is the one that obtains better approximations within the considered range of values of n. We also observe that the approximations of the Dissection algorithm improve as κ increases. In fact, Dissection with $\kappa = 16$, performs only slightly worse than Projection. For EB, both algorithms seem to perform similarly. However, we have observed that the execution time increases when κ increases. On the other hand, with regard to the predicted asymptotic behavior, we can observe that EB is the only measure that achieves it for $n < 200,000$. For the remaining measures, LA seems to tend to 1 quickly, whereas SC and VS remain far from 1 (about 5%), and BW and CW remain quite far from 1 (10% and 15% respectively).

The conclusions induced from this experiment are twofold: On one hand, the Projection algorithm has a better behavior than the Dissection algorithm, both in quality and time. On the other hand, for the SC, CW, VS and BW measures, the predicted convergence to 1 is far from being observed. This may be due to the fact that $b_n = \log\log n$ goes to infinity very slowly, but experiments with denser graph exhibit the same phenomenon, albeit reduced.

Bisection of random geometric graphs. As described in the Introduction, the problem of partitioning a graph into two pieces with the same number of vertices is fundamental in computer science. Many heuristics have been proposed for the Edge Bisection problem and many libraries implement them. Two outstanding libraries are Chaco [126], already introduced in Section 2.2.3, and Party, developed by Preis [212]. These libraries offer several global and local heuristics which can be combined between them. On the other hand, the bisection of geometric random graphs has already been considered in several experimental papers [23, 137, 165], but without the theoretical framework we have developed. In the following, we analyze the behavior of the Projection algorithm and of the heuristics included in the Chaco and Party libraries for the EDGEBIS problem on random geometric graphs.

We have compared the following global heuristics. pro: the Projection algorithm we have presented in Section 5.1; mul: the Multilevel method of

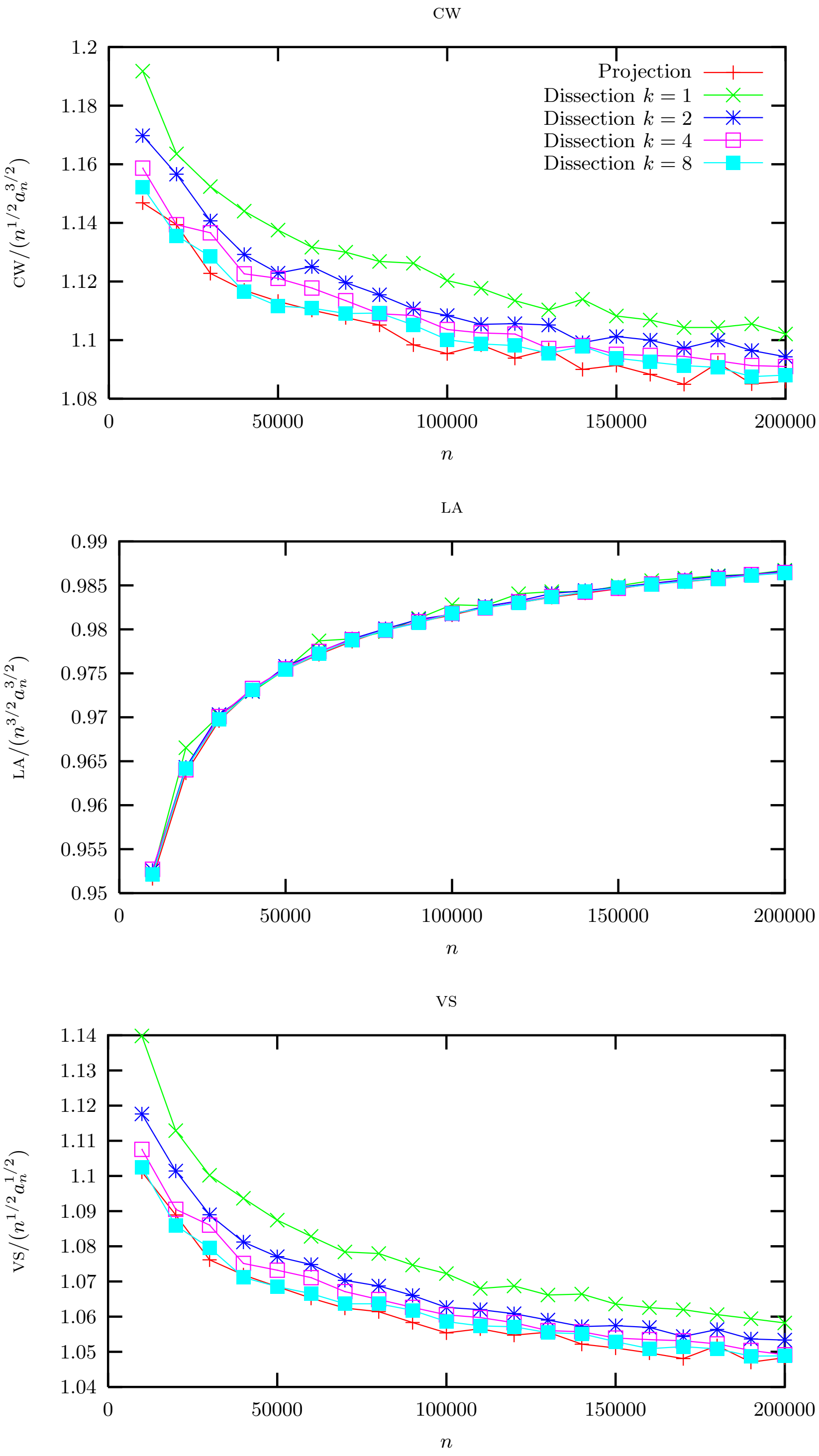


Figure 5.10: Normalized cost of each problem for the Projection and Dissection algorithms (Part 1).

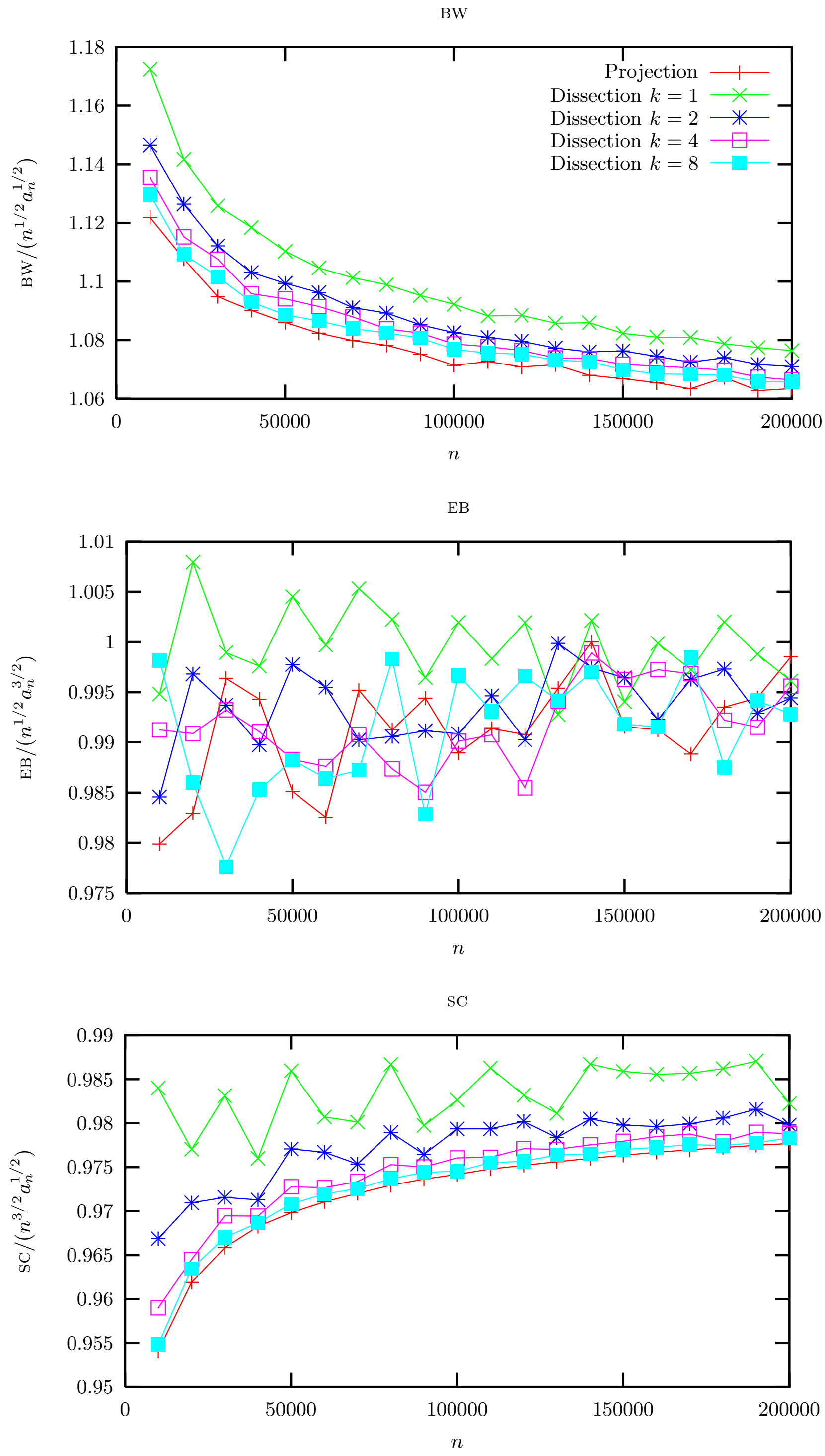


Figure 5.11: Normalized cost of each problem for the Projection and Dissection algorithms (Part 2).

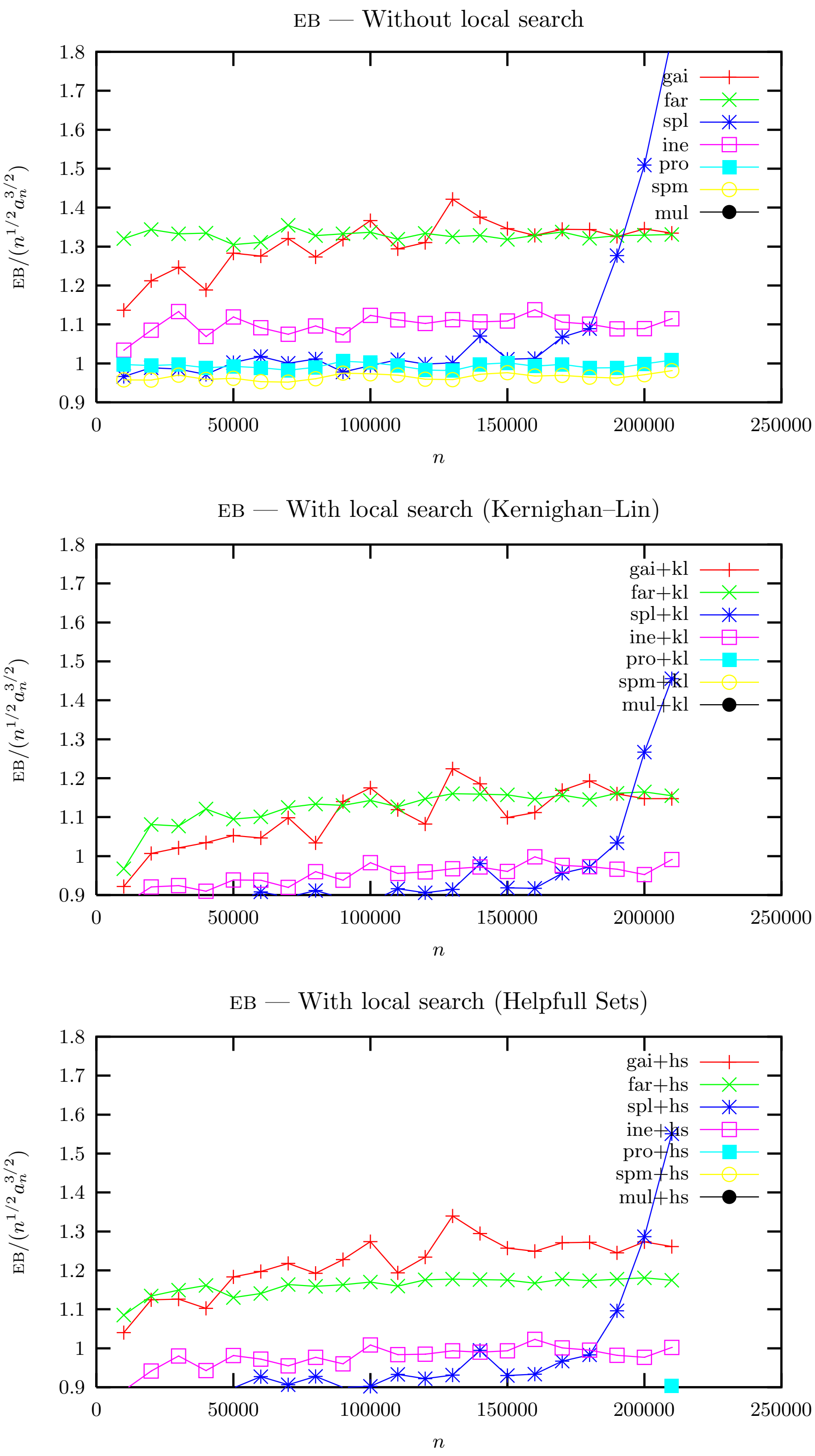


Figure 5.12: Bisection of random geometric graphs with different combinations of global and local heuristics (averages over 25 runs).

Hendrickson and Leland, which uses edge contraction schemes; ine: the Inertial method requires geometric information of the vertices in order to assimilate mass values to vertices; gai: the Gain method is based on a greedy strategy; far: the Farhat method is based on another greedy strategy; spl: an Spectral bisection algorithm using Lanczos eigen solver, spm: a combination of Multilevel and Spectral methods. More details on all these heuristics are available in [126, 212]. The solutions obtained with these methods have been refined with local search based on the Kernighan–Lin heuristic (kl) [152] or the Helpful Sets heuristic (hs) [76].

The measurement results are summarized in Figure 5.12 for graphs with up to 210,000 vertices and $r_n = \sqrt{(\log n)(\log \log n)/n}$. These results were produced by generating 25 different random geometric graphs for each data point and taking the average of their bisection normalized by $n^{1/2}((\log n)(\log \log n))^{3/2}$. Again, the standard deviation was very low. Table 5.2 shows the run time measurements needed by the different combinations of local and global heuristics to bisect a random geometric graph with $100,000$ vertices on a DEC Alpha Server 8400 machine.

From these experimental results and from the knowledge we have on the projection algorithm for EdgeBis on random geometric graphs, we can classify the global heuristics in two groups, according on weather they offer better or worse approximations than the Projection algorithm on the considered range of vertices. The global heuristics that perform worse than Projection are spl, far, gai and spm. The ones that perform better than pro are mul and spm. It is noteworthy that spl performs better than pro for $n \leqslant 100,000$ but abruptly gets worse afterwards.

If we consider the application of local search after the global heuristics, we can observe that all the methods except mul can be substantially improved by a factor around 20%. However, the quality of the improved solution depends directly on the quality of the global solution. Moreover, when fixing the global heuristic, we can observe that kl always returns slightly better solutions than hs. It is worth to remark that global solutions obtained by mul cannot be improved more than a 0.25%. On the other hand, Kernighan–Lin is the local search heuristic that better improves a given partition. The solutions delivered by the Helpful Sets heuristic are only slightly worse. However, the better solutions are always delivered by the Multilevel method, and these are difficult to improve even when using Kernighan–Lin.

In definitive, according to our experimental results, we infer that the Multilevel method is the best method to bisect the considered classes of random geometric graphs. Moreover, this method has the advantage of not using geometric information.

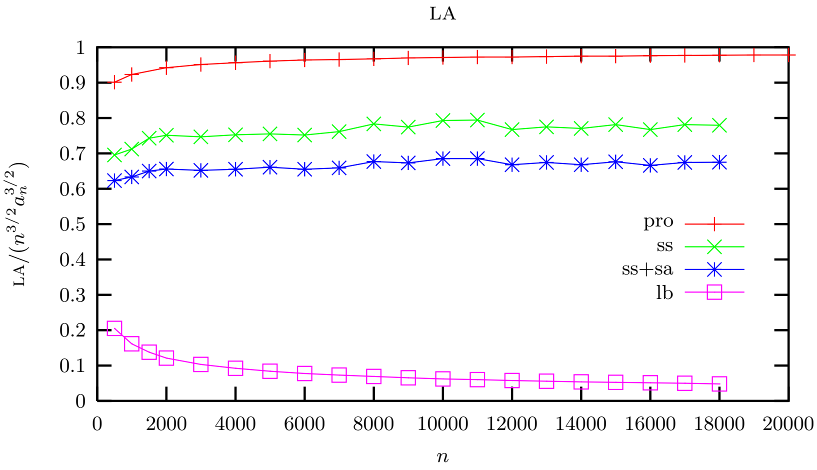


Figure 5.13: Comparison of upper and lower bounding techniques for the MinLA problem.

Minimum Linear Arrangement. In Section 2.5, we have presented and analyzed the SS+SA heuristic. This heuristic was delivering the best upper bounds for the MinLA problem for graphs with a geometric structure. In that chapter, we have also presented a lower bounding technique for MinLA based on the degree of the vertices (lb-degree). We consider now the evaluation of these techniques on random geometric graphs.

Figure 5.13 compares the Projection algorithm (pro) with the Spectral Sequencing (ss) and SS+SA (ss+sa) heuristics for the MinLA problem. The lower bounds computed by the Degree methods (lb-degree) are also shown. These results were again obtained by taking the average of 100 geometric random graphs for each n and normalizing by $n^{3/2}a_n^{3/2}$. In this case, we had to content ourselves with graphs up to 20,000 vertices, because Simulated Annealing consumes lots of time.

The obtained results show that SS+SA performs better than Spectral Sequencing alone. Also, both heuristics perform better than the Projection algorithm. This shows that the approximations obtained are within a factor of the optimal. On the other hand, we can also observe that the $0.176n^3r_n^3$ lower bound presented in Theorem 5.9 is better than the values obtained with the lb-degree lower bounding technique.

5.4 Subcritical random geometric graphs

In the previous section we have dealt with random geometric graphs whose radius was of the form $r_n = \sqrt{a_n/n}$ with $r_n \to 0$ and $n/\log n \to \infty$. These graphs were a particular case of supercritical random geometric graphs. The goal of the present section is to study subcritical random geometric graphs.

Recall that in the subcritical phase, random geometric graphs are highly disconnected, with at most $O(\log n)$ vertices in each connected component. In this case, using an exhaustive algorithm to find an optimal layout for each component would still need $O((\log n)!n/\log n)$ steps, which is not polynomial in n.

In what follows, we consider an infinite-volume analogue for random geometric graphs. Let $\mathcal{P}_\lambda$ denote an homogeneous Poisson process on $\mathbb{R}^2$ of intensity λ, and set $\mathcal{P}_{\lambda,0} = \mathcal{P}_\lambda \cup \{(0,0)\}$. For n large, after appropriate scaling and centering at a randomly chosen point of $\mathcal{X}_n$, the graph $\mathcal{G}(\mathcal{X}_n; r_n)$ looks locally like $\mathcal{G}(\mathcal{P}_{\lambda,0}; 1)$. The *continuum percolation probability* $\widetilde{\vartheta}(\lambda)$ is the probability that the added point at the origin lies in an infinite component of $\mathcal{G}(\mathcal{P}_{\lambda,0}; 1)$. Then $\widetilde{\vartheta}(\lambda)$ is nondecreasing in λ. Set $\lambda_c = \inf\{\lambda > 0 \,:\, \widetilde{\vartheta}(\lambda) > 0\}$. We deal with random geometric graphs satisfying the condition $nr_n^2 \to \lambda$, for λ in the subcritical phase. Under the probability measure $\mathbf{Pr}[\cdot]$, with corresponding expectation $\mathbf{E}[\cdot]$, let C be the component of $\mathcal{G}(\mathcal{P}_{\lambda,0}; 1)$ which includes the origin.

All through this section we consider random geometric graphs generated by subcritical limiting phases; that is, with $\lambda < \lambda_c$. First, we present convergence results for the EdgeBis and VertBis problems. Then, we give tight bounds for the Cutwidth and VertSep problems. Finally, we present a convergence result for the MinLA, ModCut and SumCut problems. This last result can be regarded as another analog of the BHH theorem for those problems.

5.4.1 Convergence results for mineb and minvb

Let us consider the behavior of the EdgeBis and VertBis problems. The subcritical phase for these problem is given by $\lambda < \lambda'_c$, where λ'_c is defined by $\lambda'_c = \inf\{\lambda > 0 \,:\, \widetilde{\vartheta}(\lambda) \geqslant \frac{1}{2}\}$.

For $k \in \mathbb{N}$, set $\pi_k = \mathbf{Pr}[|C| = k]$. It holds that $\pi_k > 0$. For $n \in \mathbb{N}$, let $N_n(k)$ denote the number of points of $G(\mathcal{X}_n; r_n)$ lying in clusters of size k. The following result will be of help.

Lemma 5.8. Suppose $nr_n^2 \to \lambda \in (0, \lambda'_c)$. Then, $N_n(k)/n \xrightarrow{\mathrm{Pr}} \pi_k$.

Proof. Let $p_k(\mathbf{x})$ denote the probability that when adding a point $\mathbf{x}$ to a set of $n-1$ uniformly distributed points, the new point will be in a cluster of size k.

Then,

$$\mathbf{E}\left[N_n(k)\right] = n \int_{[0,1]^2} p_k(\mathbf{x})\mathrm{d}\mathbf{x}. \tag{5.32}$$

For $\mathbf{x}$ not on the boundary of $[0,1]^2$, we claim that

$$p_k(\mathbf{x}) \to \pi_k. \tag{5.33}$$

This is because the question of whether $\mathbf{x}$ lies in a cluster of size k is determined by the configuration of points within a distance kr_n of $\mathbf{x}$, and for any bounded set $B \subset \mathbb{R}^2$, as a consequence of the standard Poisson approximation to the binomial distribution, the restriction to B of the point process $\{(X_i - \mathbf{x})/r_n\}_{i=1}^{n-1}$ converges in distribution to the restriction to B of $\mathcal{P}_\lambda$.

By the dominated convergence theorem (see Theorem 1.34 in [218]), from equations (5.32) and (5.33) we get $\mathbf{E}\left[N_n(k)/n\right] \to \pi_k$. To look at the variance, notice that since $N_n(k)(N_n(k)-1)$ is twice the number of pairs of points both in clusters of size k, if we denote by $p_k(\mathbf{x},\mathbf{y})$ the probability that when inserting points $\mathbf{x}$ and $\mathbf{y}$ into a set of $n-2$ uniform points they will both be in a cluster of size k, then

$$\mathbf{E}\left[N_n(k)(N_n(k)-1)\right] = n(n-1) \int_{[0,1]^2} \int_{[0,1]^2} p_k(\mathbf{x},\mathbf{y})\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y}.$$

For points $\mathbf{x}$ and $\mathbf{y}$ not on the boundary with $\mathbf{x} \neq \mathbf{y}$, by a similar argument to the proof of (5.33), we have that $p_k(\mathbf{x},\mathbf{y}) \to \pi_k^2$. Therefore, using again the dominated convergence theorem, we get $\mathbf{E}\left[(N_n(k)/n)^2\right] \to \pi_k^2$. Using Chebyschev's inequality, for any constant $\epsilon > 0$, we have

$$\begin{aligned} \mathbf{Pr}\left[\left|N_n(k)/n - \mathbf{E}\left[N_n(k)/n\right]\right| \geqslant \epsilon\right] &\leqslant \tfrac{1}{\epsilon}\mathbf{Var}\left[N_n(k)/n\right] \\ &= \tfrac{1}{\epsilon}\Big(\mathbf{E}\left[(N_n(k)/n)^2\right] - \mathbf{E}\left[N_n(k)/n)\right]^2\Big) \\ &\to 0, \end{aligned}$$

so that $N_n(k)/n \xrightarrow{\Pr} \pi_k$, which proves the lemma. □

The following theorem states that, with high probability, it is possible to bisect subcritical random geometric graphs with an empty cut.

Theorem 5.16. Suppose $nr_n^2 \to \lambda \in (0, \lambda_c')$. Then,

$$\lim_{n\to\infty} \mathbf{Pr}\left[\textsc{mineb}(\mathcal{G}(\mathcal{X}_n; r_n)) = 0\right] = \lim_{n\to\infty} \mathbf{Pr}\left[\textsc{minvb}(\mathcal{G}(\mathcal{X}_n; r_n)) = 0\right] = 1.$$

Proof. As $1 - \sum_k \pi_k = \widetilde{\vartheta}(\lambda) < \frac{1}{2}$, we can choose k_1 such that $\sum_{k \leqslant k_1} \pi_k > \frac{1}{2}$. This inequality, together with Lemma 5.8 implies that, with high probability,

$$\sum_{k > k_1} N_n(k) < \left\lfloor \tfrac{1}{2} n \right\rfloor,$$

and $N_n(k)$ are non-zero for $k \in [k_1]$.

We generate a bisection $(W, \mathcal{X}_n)$ as follows: First, take the union of all clusters of size greater than k_1. Then, add clusters of size k_1 until there are none left. Then add clusters of size $k_1 - 1$ until there are none left. Continue in this way. At some point, having just added a set of size i, there will be a set of size $\left\lfloor \frac{n}{2} \right\rfloor - m$ with $0 \leqslant m < i$. If $m = 0$, stop. Otherwise add a cluster of size m and stop. This gives a set $W \subset \mathcal{X}_n$, of size $\lfloor \frac{n}{2} \rfloor$, with no edges connecting W to $\mathcal{X}_n \backslash W$, as desired. ☐

We can transfer the previous high probability result to a convergence result:

Corollary 5.1. Suppose $nr_n^2 \to \lambda \in (0, \lambda_c')$. Then,

$$\textsc{mineb}(\mathcal{G}(\mathcal{X}_n; r_n)) \xrightarrow{\Pr} 0 \quad \text{and} \quad \textsc{minvb}(\mathcal{G}(\mathcal{X}_n; r_n)) \xrightarrow{\Pr} 0.$$

Proof. From the definitions,

$$\begin{aligned}
&\textsc{mineb}(\mathcal{G}(\mathcal{X}_n; r_n)) \xrightarrow{\Pr} 0 \\
\iff &\forall \epsilon > 0,\ \mathbf{Pr}\left[|\textsc{mineb}(\mathcal{G}(\mathcal{X}_n; r_n)) - 0| > \epsilon\right] \to 0 \\
\iff &\forall \epsilon > 0,\ \mathbf{Pr}\left[\textsc{mineb}(\mathcal{G}(\mathcal{X}_n; r_n)) > \epsilon\right] \to 0 \\
\iff &\mathbf{Pr}\left[\textsc{mineb}(\mathcal{G}(\mathcal{X}_n; r_n)) \neq 0\right] \to 0 \\
\iff &\mathbf{Pr}\left[\textsc{mineb}(\mathcal{G}(\mathcal{X}_n; r_n)) = 0\right] \to 1 \\
\iff &\textsc{mineb}(\mathcal{G}(\mathcal{X}_n; r_n)) = 0 \ \mathbf{whp},
\end{aligned}$$

and the same applies to $\textsc{minvb}(\mathcal{G}(\mathcal{X}_n; r_n))$. ☐

5.4.2 Order of growth of MINCW and MINVS

In the case of the CUTWIDTH and VERTSEP problems, we do not have convergence results, but are able to characterize their order of growth.

The next lemma establishes lower bounds for MINCW and MINVS on random geometric graphs given by a Poisson process in the subcritical phase. In the following, B_m denotes the box $[0, m)^2$.

Lemma 5.9. Suppose $nr_n^2 \to \lambda \in (0, \lambda_c)$ and that there exists a constant $c > 0$ such that $m_n = c/r_n$ for all n. Then,

$$\lim_{n \to \infty} \mathbf{Pr}\left[\textsc{minvs}(\mathcal{G}(\mathcal{P}_\lambda \cap B_{m_n}; 1)) \geqslant \log n / \log\log n\right] = 1$$

and

$$\lim_{n\to\infty} \mathbf{Pr}\left[\textsc{mincw}(\mathcal{G}(\mathcal{P}_\lambda \cap B_{m_n})) \geqslant \tfrac{1}{5}(\log n/\log\log n)^2\right] = 1.$$

Proof. Let us dissect B_{m_n} into boxes of size $\nu \times \nu$ where ν is chosen as big as possible so that all pairs of points in the same box are connected by an edge in $G = \mathcal{G}(\mathcal{P}_\lambda \cap B_{m_n}; 1)$. There are $b_n = \lfloor m_n/\nu \rfloor^2$ boxes that completely fall in B_{m_n}. For all $i \in [b_n]$, let N_i be the number of points of the process that fall in box i. By construction, we have that N_i follows a Poisson distribution with mean $\mu = \lambda\nu^2$, and that N_i is independent of N_j, for all $i \neq j$. Let $M_n = \max_{i=1}^{b_n} N_i$. As G contains at least a clique with M_n points, we have $\textsc{minvs}(G) \geqslant M_n - 1$ and $\textsc{mincw}(G) \geqslant \frac{1}{2}M_n(\frac{1}{2}M_n - 1) \geqslant \frac{1}{5}M_n^2$.

In the following, we show that, with high probability, some box contains at least $f(n) = \log n/\log\log n$ points. Fix a box i. Then,

$$\mathbf{Pr}\left[N_i > f(n)\right] = \sum_{k>f(n)} \frac{\mu^k e^{-\mu}}{k!} \geqslant e^{-\mu}\frac{\mu^{f(n)+1}}{(f(n)+1)!}.$$

Consider now all the boxes. As $\{N_i\}_{i\in[b_n]}$ are independent, we have

$$\begin{aligned}
\mathbf{Pr}\left[\bigcap_{i=1}^{b_n} N_i \leqslant f(n)\right] &= \prod_{i=1}^{b_n} \mathbf{Pr}\left[N_i \leqslant f(n)\right] \\
&\leqslant \left(1 - e^{-\mu}\frac{\mu^{f(n)+1}}{(f(n)+1)!}\right)^{b_n} \\
&\leqslant \left(1 - e^{-\mu}\frac{\mu^{f(n)+1}}{(f(n)+1)!}\right)^{4c^2n/\mu},
\end{aligned}$$

which goes to zero as n goes to infinity. □

In order to determine upper bounds with the same order of magnitude than the previous lower bounds, we will need two auxiliary results, not related with random graphs, which may be useful in other situations.

Definition 5.7. Given a graph $G = (V, E)$ and a natural s, the *s-explosion of G* is the graph $G' = (V', E')$ where $V' = \{(u, i) \ : \ u \in V,\ 1 \leqslant i \leqslant s\}$ and $E' = \big\{(u,i)(u,j) \ : \ u \in V,\ 1 \leqslant i < j \leqslant s\big\} \cup \big\{(u,i)(v,j) \ : \ uv \in E,\ 1 \leqslant i \leqslant j \leqslant s\big\}$.

Figure 5.14 illustrates this definition.

Lemma 5.10. Let G' be the s-explosion of a graph G. Then, $\textsc{mincw}(G') \leqslant s^2(\textsc{mincw}(G) + 1)$.

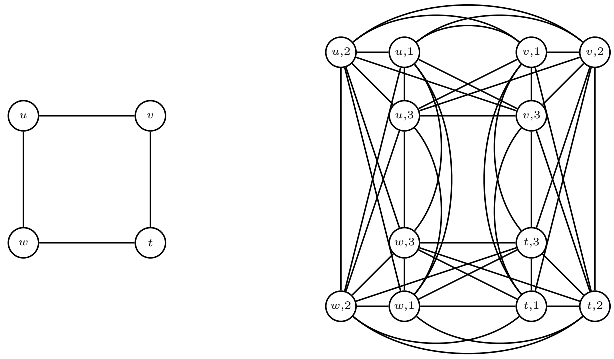


Figure 5.14: A graph and its 3-explosion.

Proof. Let $G = (V, E)$ and $G' = (V', E')$. Let φ be a layout of G with minimal cutwidth. Let φ' be a layout of G' defined by

$$\varphi'((u, i)) = (\varphi(u) - 1)s + i, \qquad \forall u \in V,\ \forall i \in [s].$$

In plain words, the exploded nodes are placed in φ' in the same relative order that their "parents" are in φ.

Let $i \in [|V|]$. As each original edge from E corresponds to s^2 edges in E', we have $\theta(si, \varphi', G') = s^2\theta(i, \varphi, G)$. Moreover, for all $j \in [s-1]$, we have $\theta(si + j, \varphi', G') \leqslant \theta(i + 1, \varphi, G) \cdot s^2 + s^2$, because we have to count s^2 edges in E' for each edge in E plus the edges inside a clique of size s. As a consequence,

$$\text{MINCW}(G') \leqslant \text{CW}(G', \varphi') \leqslant s^2(\text{CW}(G, \varphi) + 1) = s^2(\text{MINCW}(G) + 1),$$

which proves the stated result. □

Definition 5.8. Given two naturals m and l, a (m, l)-*mesh* is the graph $G = (V, E)$ with vertex set $V_m = \{0, \ldots, m-1\}^2$ and edge set $E = \{uv : u, v \in V,\ \|u - v\|_\infty \leqslant l\}$.

Lemma 5.11. Given an integer $l > 0$, there exists a constant $\kappa_l > 0$ such that for any node induced subgraph L of an (m, l)-mesh with n nodes, $\text{MINVS}(L) \leqslant \text{MINCW}(L) \leqslant \kappa_l\sqrt{n}$.

Proof. First assume that $l = 1$. An argument similar to the proof of the third bound in Theorem 4.3 shows that $\text{MINCW}(L) \leqslant \kappa_1\sqrt{n}$ for some constant $\kappa_1 > 0$. Notice that in this case, diagonal edges must be taken into account. Recall from Lemma 1.1 that the minimal vertex separation of a graph can never be greater than its minimal cutwidth. Therefore, the statement is proved for the case $l = 1$.

Assume now that $l > 1$. Let $m' = \lceil m/l \rceil$ and let H be a node induced subgraph of the $(m', 1)$-mesh where a potential node u with coordinates x_u and y_u belongs to H if and only if some node v from L with coordinates x_v and y_v satisfies $\lfloor x_v/l \rfloor = x_u$ and $\lfloor y_v/l \rfloor = y_u$. As H is a $(m', 1)$-mesh and $|H| \leqslant |L| = n$, we have $\textsc{mincw}(H) \leqslant \kappa_1\sqrt{n}$. Construct a graph G by s-exploding H with $s = l^2$. By Lemma 5.10, $\textsc{mincw}(G) \leqslant l^4(\kappa_1\sqrt{n} + 1)$. Now observe that, by construction, L is a subgraph of G. Then, by Lemmas 1.1 and 1.2,

$$\textsc{minvs}(L) \leqslant \textsc{mincw}(L) \leqslant \textsc{mincw}(G) \leqslant \kappa_l\sqrt{n},$$

where $\kappa_l = l^4(\kappa_1 + 1)$. □

The next lemma establishes upper bounds for MINCW and MINVS on random geometric graphs generated by a Poisson process in the subcritical phase.

Lemma 5.12. Suppose $nr_n^2 \to \lambda \in (0, \lambda_c)$ and that there exists a constant $c > 0$ such that $m_n = c/r_n$ for all $n \in \mathbb{N}$. Then, there exist constants $\gamma_1 > 0$ and $\gamma_2 > 0$ such that

$$\lim_{n\to\infty} \mathbf{Pr}\left[\textsc{minvs}(\mathcal{G}(\mathcal{P}_\lambda \cap B_{m_n}; 1)) \leqslant \gamma_1 \log n/\log\log n\right] = 1$$

and

$$\lim_{n\to\infty} \mathbf{Pr}\left[\textsc{mincw}(\mathcal{G}(\mathcal{P}_\lambda \cap B_{m_n}; 1)) \leqslant \gamma_2 (\log n/\log\log n)^2\right] = 1.$$

Proof. We only give the proof for the vertex separation, as the proof for the cutwidth result is similar. Let us denote $\mathcal{G}(\mathcal{P}_\lambda \cap B_{m_n}; 1)$ by G.

Let us dissect B_{m_n} in boxes of size $\nu \times \nu$, where ν is a constant that will be determined latter. We consider $b_n = \lceil m_n/\nu \rceil^2$ boxes. Let $H = H(P_\lambda)$ be the node induced subgraph of the $(\lceil m_n/\nu \rceil), 1/\nu)$-mesh whose nodes are present if and only if their corresponding box contains at least one point in $\mathcal{P}_\lambda$.

Our proof consists in three steps; let us first outline them. In Step 1, we prove that, with high probability, H does not contain "too large" connected components. Assuming that H does not contain any large connected component, in Step 2 we give an upper bound to $\textsc{minvs}(H)$ with a certain function $f(n)$, and show that if ψ is a layout such that $\textsc{vs}(\psi, H) \leqslant f(n)$, then $\textsc{minvs}(G)$ is bounded by M_n, a magnitude related to the number of points of $\mathcal{P}_\lambda$ in $f(n)$ boxes and to the layout ψ. Finally, in Step 3 we prove that, with high probability, $M_n \leqslant \nu \log n/\log\log n$. The combination of these steps shall imply the proposed result.

Step 1. For all $i \in [b_n]$, let N_i be the number of points of $\mathcal{P}_\lambda$ that fall in box i. By construction, if box i completely falls in B_{m_n}, we have that N_i

follows a Poisson distribution with mean $\mu = \lambda\nu^2$; otherwise N_i is bounded by a Poisson distribution with mean μ. Moreover, for all $i \neq j$, the variables N_i and N_j are independent.

We identify each box with a node of an infinite lattice with connections between boxes whose centers lie at distance not greater than $1/\nu$. We define on this lattice a percolation process where each node is open if its corresponding box contains at least one point from $\mathcal{P}_\lambda$. Observe that our lattice is a "general lattice" in the sense of Kesten [153, pp. 10–12] and Grimmett [109, p. 349]. Therefore, a percolation process on it must exhibit a phase transition at some critical probability, and a sufficiently small value of ν places the process in the subcritical phase (see Lemma 2 in [203]). As a consequence, there is an exponential decay in the size of a connected component C: for some constant α and all integer m,

$$\mathbf{Pr}\left[|C| > m\right] \leqslant \exp\left(-\alpha m\right).$$

Let $c_1 = 3/\alpha$. The probability that some connected component of H has size bigger than $c_1 \log n$ is

$$\mathbf{Pr}\left[\bigcup_{x \in V(H)} \{|C_x| > c_1 \log n\}\right] \leqslant b_n \cdot \exp\left(-\alpha c_1 \log n\right) \leqslant 4/n^2\mu,$$

which tends to zero as n tends to infinity.

Step 2. We now suppose that no component of H has more than $c_1 \log n$ points. By Lemmas 1.2 and 5.11, this implies that for some constant $\kappa > 0$, $\textsc{minvs}(H) \leqslant \kappa\sqrt{\log n}$.

Set $f(n) = \kappa\sqrt{\log n}$ and let ψ be a layout of H with $\textsc{vs}(\psi, G) \leqslant f(n)$. We use the layout ψ as the basis to build a layout φ for the points in $\mathcal{P}_\lambda \cap B$, taking the ordering of points within multiply occupied boxes in an arbitrary way. As $f(n)$ boxes separate any two connected boxes of H in ψ, we have that $\textsc{vs}(G, \varphi) \leqslant M_n$, where M_n is defined as the maximum number of points in $\mathcal{P}_\lambda$ that belong to the union of any $f(n)$ consecutive boxes according to layout ψ.

Step 3. In the following we show that, with high probability, $f(n)$ consecutive boxes contain at most $g(n) = \nu \log n / \log\log n$ points.

Recall that a random variable X is *stochastically dominated by* a random variable Y if $\mathbf{Pr}\left[X \leqslant t\right] \geqslant \mathbf{Pr}\left[Y \leqslant t\right]$ for all $t \in \mathbb{R}$.

The number of points in an open box follows a Poisson distribution with parameter μ, conditioned to be at least one. Using the previous definition and induction, it is easy to check that this is stochastically dominated by an unconditioned Poisson distribution with parameter μ plus one unit. Also, the number of points in $f(n)$ boxes follows the sum of $f(n)$ independent Poisson distributions each one with parameter μ and conditioned to be at least one. This is stochastically dominated by $f(n)$ plus the sum of $f(n)$ unconditioned Poisson distributions with parameter μ, which is the same as the variable

$$S = f(n) + \mathcal{P}(f(n)\mu) = \kappa\sqrt{\log n} + \mathcal{P}\left(\mu\kappa\sqrt{\log n}\right).$$

Since each connected component of H contains at most $c_1 \log n$ boxes, and H contains at most b_n connected components, the probability that any $f(n)$ consecutive boxes contain more than $g(n)$ points is at most

$$c_1 \log n \cdot b_n \cdot \mathbf{Pr}\left[S \geqslant g(n)\right].$$

In order to prove that the previous probability goes to zero as n goes to infinity, let Y be some random variable following a Poisson distribution with mean μ_n to be defined latter and let α_n be a sequence to be given latter such that $\mu_n = \omega(\alpha_n)$ and that $\mu_n = \omega(1)$. Then, we claim that

$$\mathbf{Pr}\left[Y \geqslant \alpha_n\right] \leqslant e^{-\mu_n}\mu_n^{\alpha_n}e^{\alpha_n}\alpha_n^{-\alpha_n},$$

and so

$$\log \mathbf{Pr}\left[Y \geqslant \alpha_n\right] \leqslant -\mu_n + \alpha_n \log \mu_n + \alpha_n - \alpha_n \log \alpha_n. \tag{5.34}$$

Take Y as a Poisson distribution with mean $\mu_n = \mu\kappa\sqrt{\log n}$ and take $\alpha_n = \gamma_1 \log n/\log\log n - \kappa\sqrt{\log n}$ for some constant γ_1. Let $\gamma_1' = \frac{3}{4}\gamma_1$ so that

$$\gamma_1' \log n/\log\log n < \alpha_n < \gamma_1 \log n/\log\log n.$$

Substituting in the right hand side of (5.34) and taking exponentials, we find

$$\mathbf{Pr}\left[Y \geqslant \alpha_n\right] \geqslant n^{-\gamma_1/4},$$

and thus

$$c_1 \log n \cdot b_n \cdot \mathbf{Pr}\left[Y \geqslant \alpha_n\right] \to 0,$$

provided that γ_1 is chosen big enough. □

Lemmas 5.9 and 5.12 together enable us to get the order of growth of MINVS and MINSC on subcritical random geometric graphs, holding with high probability. In order to relate the random geometric graphs in $[0,1]^2$ with the random geometric graphs in $\mathcal{P}_{\lambda,0}$, we use a coupling argument.

Theorem 5.17. Suppose $nr_n^2 \to \lambda \in (0, \lambda_c)$. Then, there exist constants $0 < c_3 < c_4$ and $0 < c_5 < c_6$ such that

$$\lim_{n\to\infty} \mathbf{Pr}\left[c_3 \leqslant \frac{\text{MINVS}(\mathcal{G}(\mathcal{X}_n; r_n))}{\log n/\log\log n} \leqslant c_4\right] = 1$$

and

$$\lim_{n\to\infty} \mathbf{Pr}\left[c_5 \leqslant \frac{\text{MINCW}(\mathcal{G}(\mathcal{X}_n; r_n))}{(\log n/\log\log n)^2} \leqslant c_6\right] = 1.$$

Proof. We first couple $\mathcal{X}_n$ to a Poisson processes with a slightly lower density of points, as follows: Take $\lambda_1 \in (0, \lambda)$ and set $m_n = \lceil 1/r_n \rceil$. Let M_n be a Poisson variable with mean $\lambda_1 m_n^2$. Then, $\mathbf{Pr}[M_n > n] \to 0$. Let us set $m_n\mathcal{X}_n = \{m_n X_i \ : \ i \in [n]\}$ and $\mathcal{P} = \{m_n X_i \ : \ i \in [M_n]\}$. Notice that $\mathcal{P}$ is a Poisson process on B_{m_n} with intensity λ_1. If $M_n \leqslant n$ then $\mathcal{G}(\mathcal{P}; 1)$ is a subgraph of $\mathcal{G}(m_n\mathcal{X}_n; m_n r_n)$, which by a suitable scaling is isomorphic to $\mathcal{G}(\mathcal{X}_n; r_n)$. By monotonicity,

$$\mathbf{Pr}[\text{VS}(\mathcal{G}(\mathcal{X}_n; r_n)) \geqslant \text{VS}(\mathcal{G}(\mathcal{P}; 1))] \to 1.$$

As a consequence, by Lemma 5.9, we have

$$\mathbf{Pr}[\text{VS}(\mathcal{G}(\mathcal{X}_n; r_n)) \geqslant \log n/\log\log n] \to 1. \tag{5.35}$$

We now couple $\mathcal{X}_n$ to a Poisson processes with a slightly higher density of points: Take $\lambda_2 \in (\lambda, \lambda_c)$ and set $m'_n = \lfloor 1/r_n \rfloor$. Let M'_n be a Poisson variable with mean $\lambda_2 (m'_n)^2$. Then, $\mathbf{Pr}[M'_n < n] \to 0$. Let $\mathcal{P}' = \{m'_n X_i \ : \ i \in [M'_n]\}$. Notice that $\mathcal{P}'$ is a Poisson processes on $B_{m'_n}$ with intensity λ_2. If $M'_n \leqslant n$ then $\mathcal{G}(\mathcal{P}; 1)$ is a super-graph of $\mathcal{G}(m_n\mathcal{X}_n; m_n r_n)$, which by a suitable scaling is isomorphic to $\mathcal{G}(\mathcal{X}_n; r_n)$. By monotonicity,

$$\mathbf{Pr}\left[\text{VS}(\mathcal{G}(\mathcal{X}_n; r_n)) \leqslant \text{VS}(\mathcal{G}(\mathcal{P}'; 1))\right] \to 1.$$

As a consequence, by Lemma 5.12, we have

$$\mathbf{Pr}[\text{VS}(\mathcal{G}(\mathcal{X}_n; r_n)) \leqslant \gamma_1 \log n/\log\log n] \to 1. \tag{5.36}$$

Combining equations (5.35) and (5.36) with Boole's inequality, and setting $c_3 = 1$ and $c_4 = \gamma_1$, we obtain

$$\mathbf{Pr}\left[c_3 \leqslant \frac{\text{MINVS}(\mathcal{G}(\mathcal{X}_n; r_n))}{\log n/\log\log n} \leqslant c_4\right] \to 1.$$

The proof for the cutwidth is similar. □

5.4.3 Convergence results for MINLA, MINMC and MINSC

We present now convergence results for the MinLA, ModCut and SumCut problems. We shall first prove that in the subcritical case, the expected values of MINLA, MINMC and MINSC on C are finite.

Lemma 5.13. Let $\lambda \in (0, \lambda_c)$ and $M \in \{\textsc{la}, \textsc{mc}, \textsc{sc}\}$. Then,

$$\mathbf{E}[\textsc{min}M(C)] \in (0, \infty).$$

Proof. For any graph G with n nodes, $\textsc{min}M(G) \leqslant n^3$. So, to prove the statement it is enough to show that $\mathbf{E}[|C|^3] \in (0, \infty)$. To show this, let $B(\rho)$ be the ball of radius ρ centered at the origin and let

$$\mathcal{P}_{\lambda,0}(B(\rho)) = |\{x \in \mathcal{P}_{\lambda,0} \ : \ x \in B(\rho)\}|.$$

Then, for any $m > 0$, the event $\{|C| \geqslant m^{1/3}\}$ is contained in the union of the events $\{\mathcal{P}_{\lambda,0}(B(m^{1/12}) \geqslant m^{1/3}\}$ and $\{\mathrm{diam}(C) \geqslant m^{1/12}\}$. Therefore using Boole's inequality we get

$$\begin{aligned}\mathbf{Pr}\left[|C| \geqslant m^{1/3}\right] \leqslant \mathbf{Pr}\left[\mathcal{P}_{\lambda,0}(B(m^{1/12})) \geqslant m^{1/3}\right] + \\ + \mathbf{Pr}\left[\mathrm{diam}(C) \geqslant m^{1/12}\right].\end{aligned}$$

The first term in the right hand side is summable in m by standard estimates of the Poisson distribution (see e.g. Prop. A.2.3 in [18]) and the second term is summable in m by Lemma 2 in [203]. Therefore the left hand side is summable and the statement follows. □

Next we prove a technical lemma that will be needed later. To explicitly indicate that C depends on λ, we will use the notations $C(\lambda)$ and $\mathbf{E}_\lambda[\cdot]$.

Lemma 5.14. For all $M \in \{\textsc{la}, \textsc{mc}, \textsc{sc}\}$, the function

$$\lambda \mapsto \mathbf{E}_\lambda\left[\frac{\textsc{min}M(C)}{|C|}\right]$$

is continuous in λ on $(0, \lambda_c)$.

Proof. We only give the proof for the case of MINLA, as the remaining cases are similar. Define coupled versions of the Poisson process $\mathcal{P}_\lambda$ in the following way: Let $\mathcal{P}$ be a Poisson process on $\mathbb{R}^2 \times [0, \infty)$ of rate 1, and let $\mathcal{P}_\lambda$ consist of the projections onto the 2 coordinates of the points of $\mathcal{P} \cap (\mathbb{R}^2 \times [0, \lambda])$. Using this coupling, write $C(\lambda)$ for the component including the origin of $\mathcal{G}(\mathcal{P}_{\lambda,0}; 1)$.

Suppose (λ_n) is a sequence with $\lambda_n \to \lambda \in (0, \lambda_c)$. With this coupling, with probability 1, for all large enough n the component $C(\lambda_n)$ tends to $C(\lambda)$. Hence by the dominated convergence theorem (see p. 160 in [110]),

$$\mathbf{E}_{n\lambda}\left[\frac{\textsc{minla}(C(\lambda_n))}{|C(\lambda_n)|}\right] \to \mathbf{E}_{\lambda}\left[\frac{\textsc{minla}(C(\lambda))}{|C(\lambda)|}\right],$$

which proves the lemma ☐

We give now asymptotics for the MINLA, MINMC and MINSC costs of graphs $\mathcal{G}(\mathcal{P}_\lambda \cap B_m; 1)$.

Lemma 5.15. Suppose $\lambda \in (0, \lambda_c)$, and let $M \in \{\textsc{la}, \textsc{mc}, \textsc{sc}\}$. Then, as $m \to \infty$,

$$\frac{\textsc{min}M(\mathcal{G}(\mathcal{P}_\lambda \cap B_m; 1))}{m^2} \xrightarrow{\Pr} \lambda\mathbf{E}\left[\frac{\textsc{min}M(C)}{|C|}\right].$$

Proof. We only sketch a proof for MINLA, as the arguments are similar to the proof of Theorem 5.8.

For each point x of $\mathcal{P}_\lambda \cap B_m$, let C_x^m denote the component of $\mathcal{G}(\mathcal{P}_\lambda \cap B_m; 1)$ that includes the point x, and let C_x denote the component of $\mathcal{G}(\mathcal{P}_\lambda; 1)$ that includes the point x. According to equation (5.3), it suffices to prove that

$$\mathbf{E}\left[\frac{1}{m^2}\sum_{x\in\mathcal{P}_\lambda\cap B_m}\left|\frac{\textsc{minla}(C_x)}{|C_x|} - \frac{\textsc{minla}(C_x^m)}{|C_x^m|}\right|\right] \to 0. \tag{5.37}$$

For $l > 0$, let $\partial_l B_m$ be the set of points $z \in B_m$ with $\|z-y\|_\infty \leqslant l$ for some $y \notin B_m$. The quantity inside the sum in (5.37) is at most $\textsc{minla}(C_x)\mathbf{(}C_x^m \neq C_x\mathbf{)}$. Hence the random variable inside the expectation in (5.37) is at most

$$\frac{1}{m^2}\sum_{x\in\mathcal{P}_\lambda\cap\partial_l B_m}\textsc{minla}(C_x) + \frac{1}{m^2}\sum_{x\in B_m\setminus\partial_l B_m}\textsc{minla}(C_x)\mathbf{(}\mathrm{diam}(C_x) > l\mathbf{)}.$$

The expectation of the first term tends to zero, while the expectation of the second term equals $\lambda\mathbf{E}[\textsc{minla}(C)\mathbf{(}|C| > l\mathbf{)}]$, which can be made arbitrarily small by the choice of l. Then equation (5.37) follows. ☐

We can now finally obtain a convergence result for the cost of the MINLA, MODCUT and SUMCUT problems on subcritical random geometric graphs.

Theorem 5.18. Suppose $nr_n^2 \to \lambda \in (0, \lambda_c)$. Then, there exist two constants $\tilde{\beta}_{\textsc{la}}(\lambda) > 0$ and $\tilde{\beta}_{\textsc{sc}}(\lambda) > 0$ such that, as $n \to \infty$,

$$\textsc{minla}(\mathcal{G}(\mathcal{X}_n; r_n))/n \xrightarrow{\Pr} \tilde{\beta}_{\textsc{la}}(\lambda),$$

$$\begin{aligned}
\textsc{minmc}(\mathcal{G}(\mathcal{X}_n; r_n))/n &\xrightarrow{\text{Pr}} \tilde{\beta}_{\text{MC}}(\lambda), \\
\textsc{minsc}(\mathcal{G}(\mathcal{X}_n; r_n))/n &\xrightarrow{\text{Pr}} \tilde{\beta}_{\text{SC}}(\lambda).
\end{aligned}$$

Proof. The same coupling that in the proof of Theorem 5.17 shows that

$$\mathbf{Pr}\left[\textsc{minla}(\mathcal{G}(\mathcal{P}_n; 1)) \leqslant \textsc{minla}(\mathcal{G}(\mathcal{X}_n; r_n)) \leqslant \textsc{minla}(\mathcal{G}(\mathcal{P}'_n; 1))\right] \to 1.$$

By Lemma 5.15,

$$\frac{\textsc{minla}(\mathcal{G}(\mathcal{P}_n; 1))}{m_n^2} \xrightarrow{\text{Pr}} \lambda_1 \mathbf{E}_{\lambda_1}\left[\frac{\textsc{minla}(C)}{|C|}\right],$$

so that

$$\frac{\textsc{minla}(\mathcal{G}(\mathcal{P}_n; 1))}{n} \xrightarrow{\text{Pr}} \left(\frac{\lambda_1}{\lambda}\right) \mathbf{E}_{\lambda_1}\left[\frac{\textsc{minla}(C)}{|C|}\right].$$

Similarly,

$$\frac{\textsc{minla}(\mathcal{G}(\mathcal{P}'_n; 1))}{n} \xrightarrow{\text{Pr}} \left(\frac{\lambda_2}{\lambda}\right) \mathbf{E}_{\lambda_2}\left[\frac{\textsc{minla}(C)}{|C|}\right].$$

Taking $\lambda_1 \uparrow \lambda$ and $\lambda_2 \downarrow \lambda$ and using Lemma 5.14,

$$\frac{\textsc{minla}(\mathcal{G}(\mathcal{X}_n; r_n))}{n} \xrightarrow{\text{Pr}} \mathbf{E}_{\lambda}\left[\frac{\textsc{minla}(C)}{|C|}\right].$$

The proof for the convergence of MINSC and MINMC is analogous. □

5.5 Conclusion

In this chapter we have pursued a probabilistic analysis of several layout problems on two random models of unit disk graphs: random grid graphs and random geometric graphs. Both models exhibit a differentiated behavior according their subcritical and supercritical phases, which are characterized by percolation theory. These models seem to be a relevant abstraction to model instances that occur in practice and, unlike binomial random graphs, having a knowledge on their optimal layouts is of interest for the evaluation of heuristics. In the following, we summarize our results and point out some conclusions.

First we have considered subcritical random grid graphs, generated by randomly selecting vertices of an $m \times m$ square grid with probability $p < p_c$. For the VERTSEP and CUTWIDTH problems, we have determined that, with probability 1, the order of magnitude of their minimal costs is $\Theta(\sqrt{\log m})$. For the MINLA, MODCUT and SUMCUT problems, we have even been able to prove that their optimal cost divided by m^2 converges in probability to a particular constant. In this sense, these results are analogous to the well-known Beardwood, Halton and Hammersley theorem for the TSP and to other convergence results for random geometric problems on $[0,1]^2$. Table 5.3 summarizes our results for the subcritical phase, and Table 5.4 summarizes Penrose's results for the supercritical phase. These tables make clear that the optimal costs for all problems are different in the supercritical and subcritical phases. This means that the optimal costs of the considered layout problems exhibit a transition phase on random grid graphs.

Afterwards, we have studied random geometric graphs, generated by scattering n vertices at random in the unit square, and joining them with an edge when their distance is small than r_n. Selecting $r_n = \sqrt{a_n/n} \to 0$ and $a_n/\log n \to \infty$ ensures connectivity while preserving sparsity. In this case, we have provided lower bounds on the cost of several layout problems: Bandwidth, MinLA, Cutwidth, SumCut, VertSep and EdgeBis. Much effort has been invested in order to obtain tight lower bounds. Owing to the probabilistic nature of random geometric graphs, these lower bounds hold, almost surely, for large enough graphs. Two simple and fast heuristics have been proposed to approximate layout problems. We have proved that these heuristics, while naive, are in fact constant approximation algorithms. Moreover, for the Bandwidth and the VertSep problems, the algorithms are asymptotically optimal. In this sense, our results are analogous to Karp's dissection technique to approximate the Euclidean TSP on $[0,1]^2$. Table 5.5 summarizes the obtained results on supercritical random geometric graphs.

It must be remarked that, in order to build a layout, our algorithms use the coordinates of the vertices. This simplification is an accepted common practice in the literature (see page 104). In any case, this is not a nuisance for our results: On one hand, the characterization of the algorithms in Corollary 5.12 is so tight, that we already know which solution they will deliver with high probability. On the other hand, now that we have analytical information on these algorithms, we do not view them as a practical way to find approximate solutions, but rather as a valuable tool with which to compare other heuristics.

Indeed, while several heuristic methods have been acclaimed as an effective way to find good approximate solutions, the assessment of their quality has only been shown empirically. We claim that our new analytical results on supercritical random geometric graphs offer more insight and give more informative ways to analyze and compare heuristics in an experimental setting. The reason is that for this class of graphs, we provide upper and lower bounds with a "guarantee of quality." For instance, many heuristics to approximate the Bisection problem are Local Search methods that accept an initial partitioning of the input graph and then attempt to improve it (i.e. the Kernighan–Lin heuristic [152], Simulated Annealing [137], Path Optimization [23] or Helpful Sets [76]). It is shown in [137] and [165] that initial partitioning created by a certain *Line* heuristic dramatically improve the performance of the Kernighan–Lin and Simulated Annealing heuristics on random geometric graphs. The Line heuristic uses geometric information to split the vertex set of a geometric graph into two equal sized halves with a line of randomly chosen slope. Thanks to our results on the Projection algorithm, which is close to the Line heuristic, we can now affirm that all these heuristics which perform so well in practice are, with high probability, constant approximation algorithms when applied to random geometric graphs.

Our experimental results on supercritical random geometric graphs show that there still remains a gap between theory and practice: for several layout problems, the asymptotic behavior of Corollary 5.12 is far from being achieved even with huge graphs with 200,000 vertices. On the positive side, our experimental comparison of several well-known heuristics for the Bisection problem with our constant approximation algorithms gives an informative benchmarking for all these heuristics.

Finally, we have also considered random geometric graphs in the subcritical phase. This phase is characterized by $nr_n^2 \to \lambda \in (0, \lambda_c)$. In this case, we have proved that the optimal costs of the MinLA, ModCut and SumCut problems, divided by the number of nodes of the graph converge in probability to another particular constant. For the VertBis, VertSep and Cutwidth problems, we have identified their probabilistic asymptotic order of magnitude. We also have shown how one can obtain, with high probability, an empty bisection. Table 5.6 summarizes the obtained results on subcritical random geometric graphs.

Some considerations on random geometric problems on $[0, 1]^2$ that had an influence in the research presented in this chapter were exposed at the *2nd. International Workshop on Randomization and Approximation Techniques in Computer Science*—RANDOM '98 *(Barcelona, 1998)* [67]. A preliminary version of the results on supercritical random geometric graphs, with much weaker results, was presented at the *25th International Workshop on Graph-Theoretic Concepts in Computer Science*—WG99 *(Ascona, 1999)* [63]. Results related to supercritical random geometric graphs will appear in *Journal of Algorithms* [66], and results related to subcritical random geometric graphs and subcritical random grid graphs have been published in *Combinatorics, Probability and Computing* [65].

Our results can be extended in several ways. First of all, notice that, for the sake of simplicity, we have concentrated only on the l_∞ norm. However, using Penrose's techniques [206], all the bounds also should hold, with different values of the constants, for any norm l_p, $p \in \mathbb{N}$. The reason is that all these norms only differ by a constant factor. Moreover, all through the chapter, we have only considered two-dimensional unit disk graphs, but we expect that similar results would also hold on higher dimensional spaces. These extensions are left as open problems. We also leave as an open problem to determine whether or not the Projection and Dissection algorithms are, in some probabilistic and asymptotic sense, optimal for layout problems other than BANDWIDTH, VERTBIS and VERTSEP. This would call for better lower bounds.

Two more challenging research problems also arise. The first of them is to investigate the value of the limiting constants in our BHH-like results. As with many other optimization problems in the plane that exhibit convergence phenomena, finding good methods to evaluate numerically the constants in Theorems 5.8 and 5.18 seems a difficult task. Our experimental tentative to characterize $\beta_{\text{LA}}(p)$ is not tight enough for $p > 0.25$. New empirical or analytical methods to estimate these constants seem necessary. The second problem consists in evaluating the performance of widely acclaimed heuristic on supercritical random geometric graphs. Kernighan–Lin, Spectral methods, Simulated Annealing or Helpful Sets are important candidates for this analysis. The results and techniques developed in this chapter are a first step towards this direction.

Subcritical random grid graphs
$\textsc{mincw}(\mathcal{L}_{m,p}) = \Theta(\sqrt{\log m})$, whp
$\textsc{minvs}(\mathcal{L}_{m,p}) = \Theta(\sqrt{\log m})$, whp
$\textsc{minla}(\mathcal{L}_{m,p})/m^2 \xrightarrow{\Pr} \beta_{\text{LA}}(p)$
$\textsc{minmc}(\mathcal{L}_{m,p})/m^2 \xrightarrow{\Pr} \beta_{\text{MC}}(p)$
$\textsc{minsc}(\mathcal{L}_{m,p})/m^2 \xrightarrow{\Pr} \beta_{\text{SC}}(p)$

Table 5.3: Overview of results for subcritical random grid graphs. We require $p < p_c$.

Supercritical random grid graphs
$\textsc{minsc}(\mathcal{L}_{m,p}) = \Theta(m^2)$
$\textsc{minla}(\mathcal{L}_{m,p}) = \Theta(m^2)$
$\textsc{minbw}(\mathcal{L}_{m,p}) = \Theta(m)$
$\textsc{minvs}(\mathcal{L}_{m,p}) = \Theta(m)$
$\textsc{mincw}(\mathcal{L}_{m,p}) = \Theta(m)$
$\textsc{mineb}(\mathcal{L}_{m,p}) = \Theta(m)$

Table 5.4: Overview of results for supercritical random grid graphs. We require $p > p_c$ for all rows, except the last one, where $p > p_c'$. All results hold with overwhelming probability.

Supercritical random geometric graphs
$\text{MINVS}(\mathcal{G}(\mathcal{X}_n; r_n)) = \Theta(nr_n)$
$\text{MINBW}(\mathcal{G}(\mathcal{X}_n; r_n)) = \Theta(nr_n)$
$\text{MINSC}(\mathcal{G}(\mathcal{X}_n; r_n)) = \Theta(n^2 r_n)$
$\text{MINEB}(\mathcal{G}(\mathcal{X}_n; r_n)) = \Theta(n^2 r_n^3)$
$\text{MINCW}(\mathcal{G}(\mathcal{X}_n; r_n)) = \Theta(n^2 r_n^3)$
$\text{MINLA}(\mathcal{G}(\mathcal{X}_n; r_n)) = \Theta(n^3 r_n^3)$

Table 5.5: Overview of results for supercritical random geometric graphs. The radius must satisfy $r_n = \sqrt{a_n/n} \to 0$ and $a_n/\log n \to \infty$. All results hold with probability 1 for all n large enough.

Subcritical random geometric graphs
$\text{MINCW}(\mathcal{G}(\mathcal{X}_n; r_n)) = \Theta((\log n/\log\log n)^2)$, whp
$\text{MINVS}(\mathcal{G}(\mathcal{X}_n; r_n)) = \Theta(\log n/\log\log n)$, whp
$\text{MINLA}(\mathcal{G}(\mathcal{X}_n; r_n))/n \xrightarrow{\Pr} \tilde{\beta}_{\text{LA}}(\lambda)$
$\text{MINMC}(\mathcal{G}(\mathcal{X}_n; r_n))/n \xrightarrow{\Pr} \tilde{\beta}_{\text{MC}}(\lambda)$
$\text{MINSC}(\mathcal{G}(\mathcal{X}_n; r_n))/n \xrightarrow{\Pr} \tilde{\beta}_{\text{SC}}(\lambda)$
$\text{MINEB}(\mathcal{G}(\mathcal{X}_n; r_n)) \xrightarrow{\Pr} 0$
$\text{MINVB}(\mathcal{G}(\mathcal{X}_n; r_n)) \xrightarrow{\Pr} 0$

Table 5.6: Overview of results for subcritical random geometric graphs. The radius must satisfy $nr_n^2 \to \lambda$, and we require $\lambda < \lambda_c$ for all rows, except the last two, where $\lambda < \lambda'_c$.

6

Faulty Random Geometric Networks

6.1 Introduction

The use of distributed computing in wireless networks is a computational model that is gaining increasing importance in computer science and telecommunications. In this setting, the processors, scattered geographically, communicate through transmitters, effectively forming a *wireless broadcast network*. The following setting arises in applications of wireless broadcast networks: A set of stations are located in some geographical area. These stations can compute, send and receive messages in synchronous steps. All the transmitters have the same power, but their effective broadcast range is limited to some specified distance r, that is, two stations can only communicate if their distance is at most r. Unit disk graphs provide a convenient way to model this setting: Recall from Definition 4.1 that a graph is a unit disk graph if each vertex can be mapped to a point in the plane in such a way that two points are adjacent if and only if their distance is at most some specified bound r. Several researchers have shown that some important problems on broadcast networks are, in fact, classic problems restricted to unit disk graphs (see [41]). However, as pointed by Clark, Colbourn and Johnson in [51], unit disk graphs assume that no interference from weather, mountains or other obstacles affects the communication between two stations. Also, this model does not take into account the possibility that individual stations go down because of problems with power supply, mechanical damages, sabotage, or other reasons.

The advent of mobile computing and of cellular phones introduces an uncertainty with respect to the positions of the stations. Assuming that the stations are homogeneously distributed in the plane is a simplified way to cope with that changing environment. Random geometric graphs (RGGs) come at hand to model such a situation: Recall from Definition 5.4 that a *random geometric graph* is a unit disk graph whose vertices are points uniformly distributed in the unit square.

In this chapter, we analyze some computational properties and parameters of a random geometric network in the presence of *random faults*. Both edge failures and vertex failures are taken into account, keeping in mind that edge failures can be interpreted as the inability to communicate between two stations because of the presence of some unexpected obstacle, and vertex failures can be interpreted as the inability to perform computation in inoperative stations. The three network properties and parameters we study are: Hamiltonicity, emulation and layout costs.

A *Hamiltonian cycle* is a cycle that visits once each vertex of a graph. If a graph has a Hamiltonian cycle, it is said to be *Hamiltonian*. This is an important issue, because if a graph has this property it is possible to build a path to perform distributed computations based on *end-to-end communication protocols*, which allow distributed algorithms to treat an unreliable network as a reliable channel [194]. Deciding whether or not a given graph is Hamiltonian is a well-known **NP**-complete problem [145]. The question whether a uniform or binomial random graph is Hamiltonian is well solved; see Chapter VIII of [34] for an extensive account. However, the Hamiltonian problem is **NP**-complete even when restricted to grid graphs, and thus it is **NP**-complete on unit disk graphs, too [132]. These considerations call for a probabilistic analysis of the Hamiltonicity properties on random geometric graphs. Intuitively, it seems likely that random geometric graphs with a sufficiently large radius have Hamiltonian cycles, but the question is what happens when dealing with faulty random geometric graphs.

In order to cope with vertex or edge faults, a reliable network should be able to emulate the non faulty network in the faulty network, with a minimal slowdown. In the case of binomial random graphs in the $\mathcal{G}_{n,p}$ model, a random network can be emulated with constant slowdown [103, 193]. Similarly, it would be quite desirable to know how reliable random geometric networks are; that is, how slowly they can be emulated when vertices or edges fail at random.

In the previous chapter we have analyzed several layout problems on random geometric graphs. As said in the Introduction, some relevant network measures can be stated as layout problems. Therefore, it would be interesting to know how edge and vertex faults produced randomly affect the optimal cost of these layout problems. In this chapter we will also consider this issue, restricting our study to the Bisection, Minimum Linear Arrangement and Cutwidth problems.

The chapter is organized as follows. In Section 6.2 we formalize the concept of faulty random geometric networks as previously described. We also define the emulation problem and fix the radius of the random geometric graphs that we shall consider in the following. After these necessary preliminaries, in Section 6.3 we deal with Hamiltonian cycles in random geometric networks and faulty random geometric networks. In Section 6.4, we analyze the emulation problem on random geometric networks with faulty vertices and faulty edges. Finally, in Section 6.5 we present some layout problems on faulty random geometric graphs.

All through this chapter, we will make use of techniques that have been developed in the previous chapter.

6.2 Preliminaries

In this chapter, a network is modeled by a graph, where processors correspond to vertices and communication channels correspond to edges. Let G be a graph and let F be a subgraph of G; G represents a fault-free graph and F represents the resulting graph after the occurrence of edge or vertex faults in G. We differentiate between edge faults and vertex faults: graphs with faulty edges are the result of removing edges from an original graph; graphs with faulty vertices are the result of removing vertices from an original graph together with the edges incident to the removed vertices. In graph theoretical terms, a graph with faulty vertices refers to a vertex induced subgraph of an original graph and a graph with faulty edges refers to a subgraph of an original graph. We assume that vertex faults happen independently and with a constant probability f. Likewise, we assume that edge faults happen independently and with a constant probability f.

Definition 6.1 (Faulty graphs). Let $G = (V, E)$ be a graph and let $f \in [0, 1]$ denote its vertex failure probability. Let V' be a subset of V where $\mathbf{Pr}[v \in V'] = 1 - f$, independently for all vertices $v \in V$. Then we say that the subgraph of G induced by V' is a graph with random faulty vertices. We denote by $\mathcal{FN}(G, f)$ the corresponding probability space with uniform probability.

Analogously, let $G = (V, E)$ be a graph and let $f \in [0, 1]$ denote its edge failure probability. Let E' be a subset of E where $\mathbf{Pr}[uv \in E'] = 1 - f$, independently for all edges $uv \in E$. Then we say that the subgraph of G induced by E' is a graph with random faulty edges. We denote by $\mathcal{FE}(G, f)$ the corresponding probability space with uniform probability.

In particular, a random geometric graph with faulty vertices can be generated by the outcome of the following experiment: First, generate a random geometric graph; then, with probability f, remove each of its vertices and all its incident edges. Likewise, a random geometric graph with faulty edges can be generated by the outcome of the following experiment: First generate a random geometric graph; then, with probability f, remove each of its edges.

The *emulation* of a fault-free network G on a faulty network F consists in performing the computations performed by the vertices of G in the vertices of F and ensuring that the direct communication across edges of G is simulated across paths in F. Formally, an emulation ϕ is an injective function that maps $V(G)$ to $V(F)$ and $E(G)$ to the set of paths of $E(F)$ such that for all $uv \in E(G)$, $\phi(uv) = \langle w_0w_1, w_1w_2, \ldots, w_{l-1}w_l \rangle$ where $w_0 = \phi(u)$, $w_l = \phi(v)$ and $w_{i-1}w_i \in E(F)$ for all $i \in [l]$.

The *slowdown* of the emulation ϕ of G in F is defined as the length of time it takes to the faulty graph F to emulate one communication step and one computation step of the original network G. The parameters to measure the quality of an emulation are: the *dilation*, defined as the length of the largest associated path; the *congestion*, defined as the maximal number of paths that share an edge of F; and the *load*, defined as the maximum number of vertices of G that are mapped to a vertex of F. We denote by $d(\phi)$, $c(\phi)$ and $l(\phi)$ the dilation, congestion and load of an emulation ϕ. It is well known that an emulation with dilation d, congestion c and load l has a slowdown of $O(d+c+l)$ [167].

Through this chapter we restrict our attention to the particular case of random geometric graphs in l_∞ whose radius is of the form

$$r_n = \sqrt{\frac{a_n}{n}} \quad \text{where} \quad r_n \to 0 \quad \text{and} \quad a_n/\log n \to \infty.$$

As seen in Section 5.1.2, this choice guarantees almost surely the construction of connected graphs—disconnected fault-free networks would not make much sense in our setting. As in the previous chapter, we set $b_n = a_n/\log n$.

Recall from Definition 5.5 that a geometric graph G with n vertices is ϵ-nice if, dissecting the unit square into $\kappa_n = 4\lceil 1/r_n \rceil^2$ square boxes of size $s_n \times s_n$ with $s_n = 1/2\lceil 1/r_n \rceil$, each box contains between $\frac{1}{4}(1-\epsilon)a_n$ and $\frac{1}{4}(1+\epsilon)a_n$ vertices. Recall from Lemma 5.3 that, with high probability, random geometric graphs are ϵ-nice.

All through this chapter, when we speak about boxes we refer to the above dissection. For any vertex u, let B_u be denote the box where u belongs to. Also, for any $i \in [\kappa_n]$, let $B(i)$ denote the i-th box in the dissection, according to some arbitrary but fixed order. Finally, let $\alpha(i)$ denote the number of vertices of G in box $B(i)$.

6.3 Hamiltonian cycles

In this section we deal with the existence of Hamiltonian cycles in random geometric graphs and faulty random geometric graphs. First, we consider random geometric graphs with random vertex faults, which include non-faulty random geometric graphs as a particular case. Afterwards, we consider random geometric graphs with random edge faults.

6.3.1 Hamiltonian cycles in RGGs with vertex faults

The following definition expresses the fact that vertices of nice graphs fail "appropriately" in the boxes of the dissection.

Definition 6.2 (Friendly graphs). Let $\epsilon \in (0, \frac{1}{5})$ and $f \in [0, 1)$ be two constants. Let G be an ϵ-nice geometric graph with n vertices and radius r_n and let F be a vertex induced subgraph of G_n. We say that F is (ϵ, f)-*friendly* if every box of this dissection contains at least $\frac{1}{4}(1-\epsilon)^2(1-f)a_n$ vertices of F and at most $\frac{1}{4}(1+\epsilon)^2(1-f)a_n$ vertices of F.

The following lemma states that, with high probability, nice geometric graphs with random faulty vertices are friendly.

Lemma 6.1. Let $\epsilon \in (0, \frac{1}{5})$ and $f \in [0, 1)$ be two constants. For all $n \in \mathbb{N}$, let G_n be an ϵ-nice graph with n vertices and radius r_n. Then,

$$\lim_{n\to\infty} \mathbf{Pr}\left[\mathcal{FN}(G_n, f) \text{ is } (\epsilon, f)\text{-friendly}\right] = 1.$$

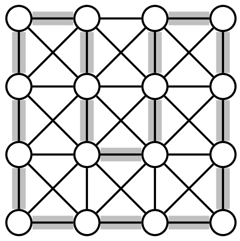
Proof. For any n, let F_n be drawn from $\mathcal{FN}(G_n, f)$. Choose any box $B(i)$ in the dissection; let $\alpha(i)$ be the number of vertices of G_n in this box and let $A(i)$ be the random variable counting the number of vertices of F_n in this box. As G_n is ϵ-nice, we have that $(1-\epsilon)\frac{1}{4}a_n \leqslant \alpha(i) \leqslant (1+\epsilon)\frac{1}{4}a_n$. On the other hand, as A_n is a sum of $\alpha(i)$ Bernoulli variables with parameter $1-f$,

$$(1-\epsilon)\tfrac{1}{4}a_n(1-f) \leqslant \mathbf{E}\left[A_n\right] = y_n(1-f) \leqslant (1+\epsilon)\tfrac{1}{4}a_n(1-f).$$

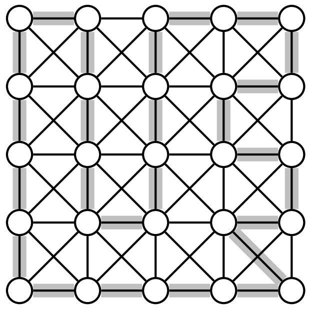
The result follows by the use of Chernoff's bounds and Boole's inequality (Theorems A.9 and A.1). $\square$

We are ready to prove that friendly geometric graphs are Hamiltonian. The proof of this statement includes an algorithm to construct a Hamiltonian cycle.

Lemma 6.2. Let $\epsilon \in (0, \frac{1}{5})$ and $f \in [0, 1)$ be two constants. Let G_n be any ϵ-nice geometric graph with n vertices and radius r_n and let F_n be an (ϵ, f)-friendly vertex induced subgraph of G_n. Then, F_n is Hamiltonian. Moreover, there exists an algorithm to construct the Hamiltonian cycle in $O(|E(F_n)|)$ steps.



(a) Even number of vertices



(b) Odd number of vertices

Figure 6.1: Hamiltonian cycles in grid graphs with diagonal edges.

Proof. First of all, notice that any square grid with diagonal edges has a Hamiltonian cycle (see Figure 6.1 for an example). Also, notice that in an (ϵ, f)-friendly graph, any pair of vertices in the same box or in neighboring boxes, including diagonals, are connected by an edge because $2s_n \leqslant r_n$. For each box in the dissection, construct a path visiting all the vertices of F_n in the box. This is always possible, as vertices in a box form a clique. Then, following the order given by a Hamiltonian cycle in the $\kappa_n \times \kappa_n$ square grid with diagonal edges, patch the last point of each path with the first point of the next path. This is always possible since the friendness of F_n ensures the existence of at least two vertices in each box, and the vertices in two neighboring boxes form a clique. This construction produces a Hamiltonian cycle for F_n. This algorithm can be implemented in $O(|E(F_n)|)$ steps. $\square$

Combining the previous lemmas, we have the following result:

Theorem 6.1. With high probability, random geometric graphs with radius $r_n = \sqrt{a_n/n}$, where $r_n \to 0$ and $a_n/\log n \to \infty$, and random vertex faults with probability $f \in [0, 1)$ are Hamiltonian.

Notice that taking $f = 0$ in the previous theorem implies that non-faulty random geometric graphs are also Hamiltonian with high probability.

6.3.2 Hamiltonian cycles in RGGs with edge faults

The following lemma proves that, with high probability, nice geometric graphs with random edge faults are Hamiltonian.

Lemma 6.3. Let $\epsilon \in (0, \frac{1}{5})$ and $f \in (0,1)$ be two constants. For all $n \in \mathbb{N}$, let G_n be an ϵ-nice geometric graph with n vertices and radius r_n. Then,

$$\lim_{n\to\infty} \mathbf{Pr}\left[\mathcal{FE}(G_n, f) \text{ is Hamiltonian}\right] = 1.$$

Proof. Let n be any natural. Let F_n be an edge faulty random geometric graph of G_n drawn from $\mathcal{FE}(G, f)$. For a box $B(i)$ and a vertex $u \in V(F_n)$ inside this box, let $\deg(u, B(i))$ denote the number of neighbors of u in F_n inside box $B(i)$.

Let us compute the probability π that, for some box $B(i)$ and some vertex $u \in V(F_n)$ inside $B(i)$, $\deg(u, B(i))$ is smaller than 2. Using Boole's inequality, we have:

$$\begin{aligned}\pi &= \mathbf{Pr}\left[\vee_{i=1}^{\kappa_n} \vee_{u\in B(i)} \deg(u, B(i)) < 2\right]\\ &\leqslant \sum_{i=1}^{\kappa_n} \sum_{u\in B(i)} \mathbf{Pr}\left[\deg(u, B(i)) < 2\right].\end{aligned}$$

As G_n is ϵ-nice and its edges are faulty independently with probability f in F_n, we have:

$$\begin{aligned}\pi &\leqslant \sum_{i=1}^{\kappa_n} \sum_{u\in B(i)} \Big(\mathbf{Pr}\left[\deg(u, B(i)) = 0\right] + \mathbf{Pr}\left[\deg(u, B(i)) = 1\right]\Big)\\ &\leqslant 4\left\lceil 1/r_n\right\rceil^2 \cdot (1+\epsilon)\tfrac{1}{4}a_n \cdot \left(f^{(1-\epsilon)\frac{1}{4}a_n - 1} + (1-f)f^{(1-\epsilon)\frac{1}{4}a_n-2}\right)\\ &= \left\lceil 1/r_n\right\rceil^2 (1+\epsilon)a_n f^{(1-\epsilon)\frac{1}{4}a_n-2}\\ &\leqslant (1+\epsilon)^3 n f^{(1-\epsilon)\frac{1}{4}b_n \log n - 2}\\ &\leqslant (1+\epsilon)^3 n f^{(1-\epsilon)\frac{1}{5}b_n \log n}.\end{aligned}$$

Set $t = -3/\log f$; then,

$$\pi \leqslant (1+\epsilon)^3 n f^{t\log n} = (1+\epsilon)^3 n^{1+t\log f} = (1+\epsilon)^3/n^2. \tag{6.1}$$

Inside a box, the vertices of G_n form a clique, whose edges fail with probability f independently in F_n. Therefore, the edges of F_n form a $\mathcal{G}_{\alpha(i),1-f}$ binomial random graph inside each box. The probability that such a binomial random graph has a Hamiltonian cycle corresponds with the probability that each of its vertices has at least degree 2 (see Theorem VIII.11 of [34]). Therefore, with probability greater than $1-(1+\epsilon)^3/n^2$, each box of the dissection contains a Hamiltonian cycle.

In order to get a Hamiltonian cycle for F_n, we patch the Hamiltonian cycles inside each box. We proceed as shown in Figure 6.2: The cycle will be

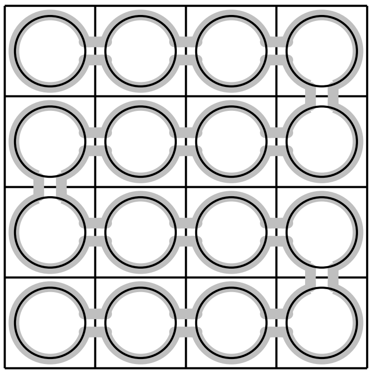


Figure 6.2: How to patch a big Hamiltonian cycle for the faulty graph using the small Hamiltonian cycles inside each box of the dissection.

made in a snake-like[1] way, removing two edges from each Hamiltonian cycle inside each box and joining the paths with the small Hamiltonian cycles of the previous and next boxes. The probability that this construction cannot be done is bounded above by

$$\left((f^2)^{\alpha(1)\alpha(2)}\right)\prod_{i=2}^{\kappa_n-1}(f^2)^{(\alpha(i)-1)\alpha(i+1)} \leqslant \left(f^{2(1-\epsilon)\frac{1}{4}a_n}\right)\left(f^{2(1-\epsilon)^2\frac{1}{16}a_n^2}\right)^{4\lceil 1/r_n\rceil^2-1}. \tag{6.2}$$

Therefore, the probability that F_n does not have a Hamiltonian cycle is smaller than the sum of the probabilities (6.1) and (6.2), which tends to zero as n tends to infinity. □

The previous lemma shows, with high probability, the existence of a Hamiltonian cycle in a random geometric graph with edge faults. Still, it would be desirable to have a polynomial time algorithm to construct such a Hamiltonian cycle. We now present an heuristic algorithm that, given a geometric graph with edge faults, either returns a Hamiltonian cycle or reports that none was found. This algorithm is inspired in the previous proof. Afterwards, we pursue a probabilistic analysis of this algorithm, assuming that the distribution of its inputs is $\mathcal{FE}(G_n, f)$ for nice graphs G_n.

Algorithm 6.1 (Algorithm FGeo-HAM). Let G_n be a geometric graph with n vertices and radius r_n. Let F_n be an edge induced subgraph of G_n. Given a

[1]The term “snake-like” comes after [167].

realization of F_n and r_n, the following algorithm either returns a Hamiltonian cycle in F_n or fails.

Dissect the unit square into $\kappa_n = 4\lceil 1/r\rceil^2$ boxes of side $s_n = 1/2\lceil 1/r_n\rceil$. For all $i \in [\kappa_n]$, let $B(i)$ be the i-the box of this dissection, with respect to the snake-like ordering.

According to the proof of Lemma 6.3, to obtain a Hamiltonian cycle for F, we have to

- find a Hamiltonian cycle for the vertices in each box, and
- patch the cycles in an snake-like way.

In order to perform the first step, we resort to the HAM algorithm of Bollobás, Fenner and Frieze [36]. For each $i \in [\kappa_n]$, the HAM algorithm either fails or finds a Hamiltonian cycle $\langle u_{i,0}, u_{i,1}, \ldots, u_{i,\alpha(i)-1}, u_{i,0}\rangle$ where $u_{i,j}$ is a vertex in $B(i)$ for all $j \in \{0, \ldots, \alpha(i)-1\}$. If HAM does not find a Hamiltonian cycle for some box $B(i)$, FGeo-HAM reports that no Hamiltonian cycle for F_n can be found.

To simplify notation, in the following, a subindex (i, j) must be understood as $(i, j \bmod \alpha(i))$.

In order to perform the second step, for each box $B(i)$, the algorithm needs to:

1. Decide in which direction will the cycle be traversed: $\langle u_{i,0}, u_{i,1}, \ldots, u_{i,\alpha(i)-1}\rangle$ or $\langle u_{i,\alpha(i)-1}, \ldots, u_{i,1}, u_{i,0}\rangle$.
2. Decide which edge in the cycle will be removed to patch with the cycle in box $B(i-1)$ (unless $i = 1$).
3. Decide which edge in the cycle will be removed to patch with the cycle in box $B(i+1)$ (unless $i = \kappa_n$).

Of course, these three decisions must be in agreement with the edges in $E(F_n)$. In order to take the above decisions, we use a directed multistage flow network. This network will contain $2\kappa_n$ stages, where stages $2i-1$ and $2i$ are defined according to $B(i)$:

- Stage 1 contains a source node s and stage $2\kappa_n$ contains a target node t.
- For all $j \in [\alpha(1)]$, the network contains two nodes $w^+_{1,j}$ and $w^-_{1,j}$ at stage 2.
- For all $j \in [\alpha(\kappa_n)]$, the network contains two nodes $v^+_{\kappa_n,j}$ and $v^-_{\kappa_n,j}$ at stage $2\kappa_n - 1$.
- For all $i \in \{2, \ldots, \kappa_n - 1\}$ and all $j \in [\alpha(i)]$, the network contains two nodes $v^+_{i,j}$ and $v^-_{i,j}$ at stage $2i-1$ and two nodes $w^+_{i,j}$ and $w^-_{i,j}$ at stage $2i$.

Figure 6.3 illustrates this construction.

The arcs in this network are given by the following rules, defined from stage $2i-1$ to stage $2i$ according to box $B(i)$, and from stage $2i$ to stage $2i+1$ according to boxes $B(i)$ and $B(i+1)$, where $x \to y$ means an arc connecting node x towards node y:

- For all $i \in \{2, \ldots, \kappa_n - 1\}$ and all $j, k \in [\alpha(i)]$, add $v^+_{i,j} \to w^+_{i,k}$ provided $k \neq j+1$.
- For all $i \in \{2, \ldots, \kappa_n - 1\}$ and all $j, k \in [\alpha(i)]$, add $v^-_{i,j} \to w^-_{i,k}$ provided $k \neq j-1$.
- The source s is connected towards all nodes in stage 2, and all nodes in stage $2\kappa_n - 1$ are connected towards the target t.
- For all $i \in [\kappa_n - 1]$, and for all $j \in [\alpha(i)]$ and all $k \in [\alpha(i+1)]$, do the following connections ($p \sim q$ means $pq \in E(F_n)$):
  - If $u_{i,j} \sim u_{i+1,k}$ and $u_{i,j+1} \sim u_{i+1,k-1}$, then add $w^+_{i,j} \to v^+_{i+1,k}$.
  - If $u_{i,j} \sim u_{i+1,k-1}$ and $u_{i,j+1} \sim u_{i+1,k}$, then add $w^+_{i,j} \to v^-_{i+1,k}$.
  - If $u_{i,j+1} \sim u_{i+1,k}$ and $u_{i,j} \sim u_{i+1,k-1}$, then add $w^-_{i,j} \to v^+_{i+1,k}$.
  - If $u_{i,j+1} \sim u_{i+1,k-1}$ and $u_{i,j} \sim u_{i+1,k}$, then add $w^-_{i,j} \to v^-_{i+1,k}$.

Figures 6.4 and 6.5 illustrate this construction.

Let $\sigma_h \in \{+, -\}$ for $h \in [\kappa_n]$. By construction, the above network has the following property: If

$$\left\langle\ s,\ \ w^{\sigma_1}_{1,j_1}, \ldots,\ v^{\sigma_h}_{h,i_h},\ w^{\sigma_h}_{h,j_h},\ v^{\sigma_{h+1}}_{h+1,i_{h+1}},\ w^{\sigma_{h+1}}_{h+1,j_{h+1}},\ \ldots,\ v^{\sigma_{\kappa_n}}_{\kappa_n,i_{\kappa_n}},\ t\ \right\rangle$$

is a valid path of nodes in the network, then F contains the Hamiltonian cycle determined by

$$\begin{aligned}
\Big\langle\quad & \overrightarrow{\pi}\left(s,\ w^{\sigma_1}_{1,j_1}\right), \\
& \qquad \ldots,\ \overrightarrow{\pi}\left(v^{\sigma_h}_{h,i_h},\ w^{\sigma_h}_{h,j_h}\right), \overrightarrow{\pi}\left(v^{\sigma_{h+1}}_{h+1,i_{h+1}},\ w^{\sigma_{h+1}}_{h+1,j_{h+1}}\right), \ldots, \\
& \overrightarrow{\pi}\left(v^{\sigma_{\kappa_n}}_{\kappa_n,j_{\kappa_n}},\ t\right), \\
& \qquad \ldots, \overleftarrow{\pi}\left(v^{\sigma_{h+1}}_{h+1,i_{h+1}},\ w^{\sigma_{h+1}}_{h+1,j_{h+1}}\right), \overleftarrow{\pi}\left(v^{\sigma_h}_{h,i_h},\ w^{\sigma_h}_{h,j_h}\right), \ldots, \\
& \overleftarrow{\pi}\left(s,\ w^{\sigma_1}_{1,j_1}\right) \\
\Big\rangle, &
\end{aligned}$$

where $\overrightarrow{\pi}$ is the "outward" path given by:

$$\begin{aligned}
& \overrightarrow{\pi}\ (s,\ w^+_{1,j_1}) = \langle u_{1,j_1} \rangle, \\
& \overrightarrow{\pi}\ (s,\ w^-_{1,j_1}) = \langle u_{1,j_1+1} \rangle,
\end{aligned}$$

$$\overrightarrow{\pi}\,(v^+_{h,i_h},\ w^+_{h,j_h}) = \langle u_{h,i_h}, u_{h,i_h+1}, \ldots, u_{h,j_h}\rangle,$$
$$\overrightarrow{\pi}\,(v^-_{h,i_h},\ w^-_{h,j_h}) = \langle u_{h,i_h-1}, u_{h,i_h-2}, \ldots, u_{h,j_h+1}\rangle,$$
$$\overrightarrow{\pi}\,(v^+_{\kappa_n,i_{\kappa_n}},\ t) = \langle u_{\kappa_n,i_{\kappa_n}}, u_{\kappa_n,i_{\kappa_n}+1}, \ldots, u_{\kappa_n,i_{\kappa_n}-1}\rangle,$$
$$\overrightarrow{\pi}\,(v^-_{\kappa_n,i_{\kappa_n}},\ t) = \langle u_{\kappa_n,i_{\kappa_n}-1}, u_{\kappa_n,i_{\kappa_n}-2}, \ldots, u_{\kappa_n,i_{\kappa_n}}\rangle,$$

and $\overleftarrow{\pi}$ is the "return" path given by:

$$\overleftarrow{\pi}\,(v^+_{h,i_h},\ w^+_{h,j_h}) = \langle u_{h,j_{h+1}}, u_{h,j_{h+2}}, \ldots, u_{h,i_{h-1}}\rangle,$$
$$\overleftarrow{\pi}\,(v^-_{h,i_h},\ w^-_{h,j_h}) = \langle u_{h,j_h}, u_{h,j_{h-1}}, \ldots, u_{h,i_h}\rangle,$$
$$\overleftarrow{\pi}\,(s,\ w^+_{1,j_1}) = \langle u_{1,j_1+1}, u_{1,j_1+2}, \ldots, u_{1,j_1}\rangle,$$
$$\overleftarrow{\pi}\,(s,\ w^-_{1,j_1}) = \langle u_{1,j_1}, u_{1,j_1-1}, \ldots, u_{1,j_1+1}\rangle.$$

These paths are illustrated in Figure 6.6.

In order to discover whether there is a path from s to t in the multistage network, we use a breadth first search algorithm: From stage 1 to stage $2\kappa_n$, mark all nodes that are reachable from s. If t is not reachable from s, FGeo-HAM reports that no Hamiltonian cycle for F_n can be found. Otherwise, it recovers a path from t to s by looking iteratively to the predecessor nodes from stage $2\kappa_n$ to stage 1 and returns the corresponding Hamiltonian cycle.

The following result characterizes the probabilistic asymptotic behavior of the FGeo-HAM algorithm on nice graphs with random edge faults.

Lemma 6.4. Let $\epsilon \in (0, \frac{1}{5})$ and $f \in (0, 1)$ be two constants. For all $n \in \mathbb{N}$, let G_n be an ϵ-nice geometric graph with n vertices and radius r_n. Then,

$$\lim_{n\to\infty} \mathbf{Pr}\left[\text{FGeo-HAM returns a Hamiltonian cycle in } \mathcal{FE}(G_n, f)\right] = 1$$

and

$$\lim_{n\to\infty} \mathbf{Pr}\left[T_n \leqslant \gamma n a_n^{3+\epsilon}\right] = 1$$

where $\gamma > 0$ is some constant and T_n is the random variable that measures the number of steps of applying algorithm FGeo-HAM to an input drawn from $\mathcal{FE}(G_n, f)$.

Proof. Let F_n be drawn from $\mathcal{FE}(G_n, f)$.

We first analyze the cost of the FGeo-HAM algorithm: As $(G_n)_{n\in\mathbb{N}}$ are ϵ-nice, $\max_{i\in[\kappa_n]} \alpha(i) = O(a_n)$. The cost of finding (or not) a "small" Hamiltonian cycle in a box with HAM is $O(a_n^{4+\epsilon})$ [36]. As there are $O(n/a_n)$ boxes, each with $O(a_n)$ vertices, the number of steps needed to compute all the small Hamiltonian

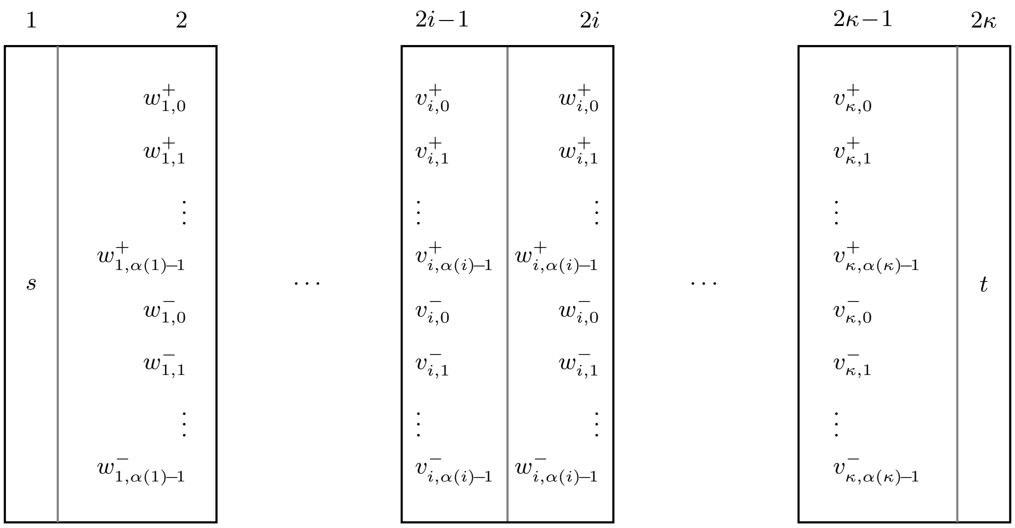


Figure 6.3: Boxes in the multistage network build by the FGeo-HAM algorithm.

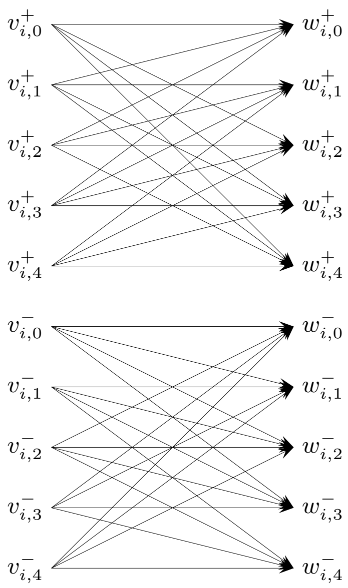
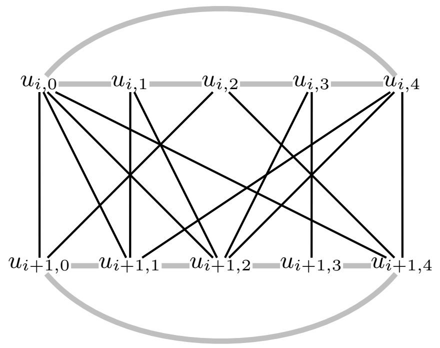
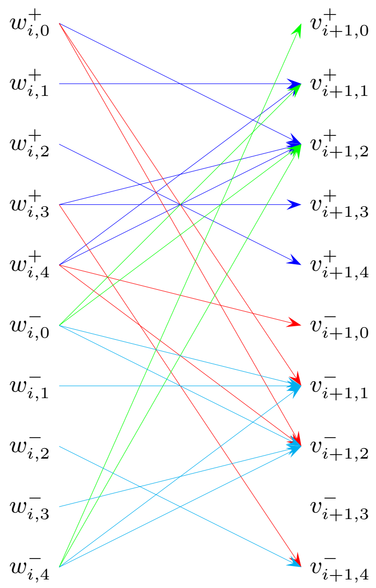


Figure 6.4: Connections in the multistage network build by the FGeo-HAM algorithm in a box $B(i)$. (Part 1)



(a) Thin dark lines represent edges between $B(i)$ and $B(i+1)$ and bold light lines represent the cycle in the two boxes



(b) Corresponding connections from stage $2i$ to stage $2i+1$. The four colors represent the four kinds of arcs.

Figure 6.5: Connections in the multistage network built by the FGeo-HAM algorithm for boxes $B(i)$ and $B(i+1)$. (Part 2)

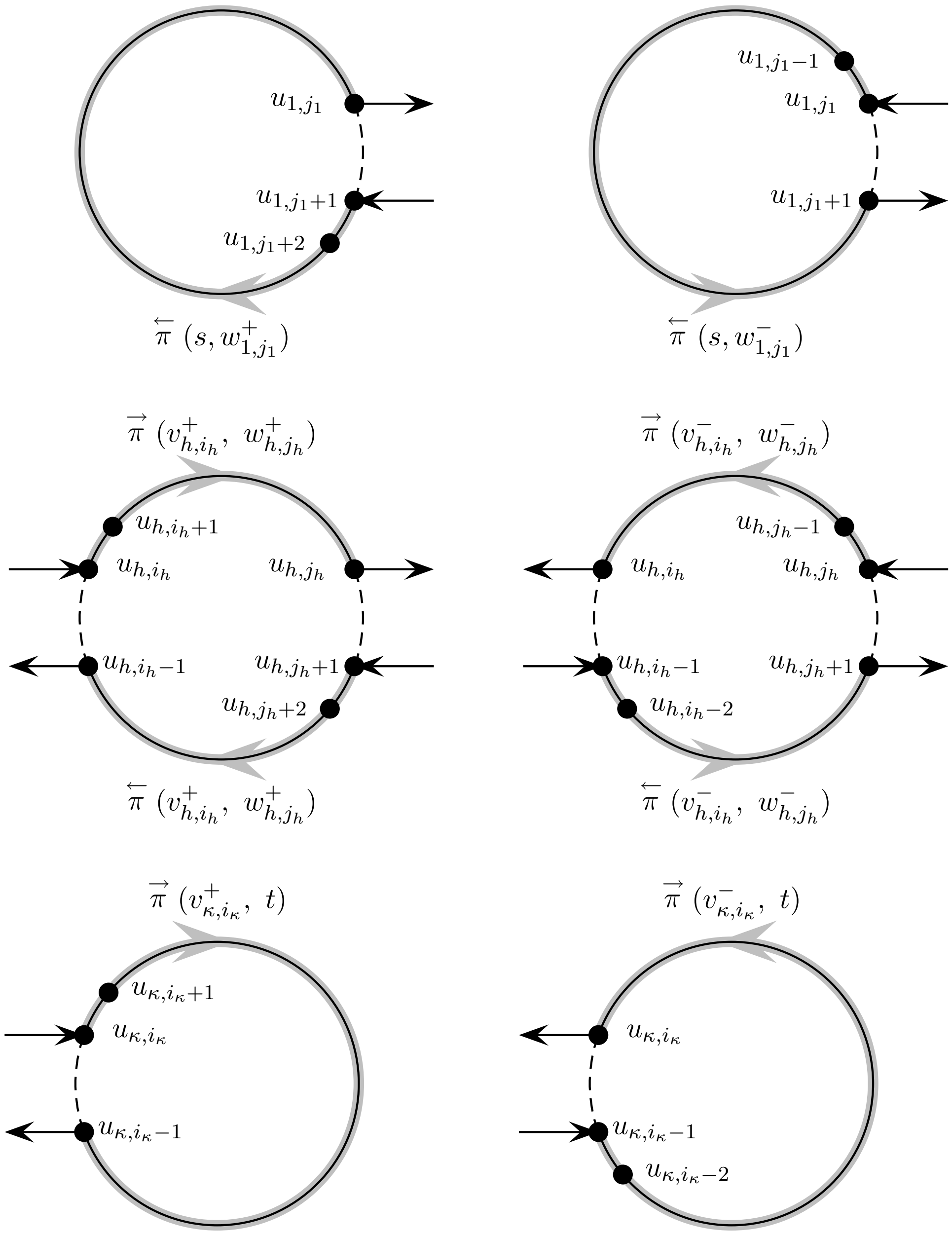


Figure 6.6: Illustration of the paths. Original cycles are drawn clockwise with a thin line, dashed at the edges that must be deleted. The bold gray line with an arrow shows the paths $\overrightarrow{\pi}$ and $\overleftarrow{\pi}$, together with their direction. The thin arrows show the vertices and direction where to patch the paths with the ones of the previous or following box. The dashed lines are the edges that are removed from the cycles.

cycles is $O(na_n^{3+\epsilon})$. Afterwards, we have to find a path in a multistage graph with $O(n/a_n)$ stages and $O(a_n)$ nodes per stage, which can be done in $O(a_n \cdot n/a_n) = O(n)$ steps. Thus, the total cost is $O(na_n^{3+\epsilon} + n) = O(na_n^{3+\epsilon})$.

We now analyze the failure probability of FGeo-HAM. There are two reasons for failure: no small Hamiltonian cycle is found for some box, or no path exists from s to t. The probability that HAM does not find a Hamiltonian cycle on a $\mathcal{G}_{n,p}$ graph is $o(2^{-n})$ [36]. So, the probability of not finding a small Hamiltonian cycle for some of the $O(n/a_n)$ boxes is $o(2^{-a_n} \cdot n/a_n) = o(n^{1-b_n}/b_n \log n)$, which tends to zero because b_n tends to infinity. The probability that no path exists from s to t is given by Equation (6.2), which tends to 0. Therefore, the probability that FGeo-HAM returns a Hamiltonian cycle for a graph drawn from $\mathcal{FE}(G_n, f)$ tends to 1 as n tends to infinity. □

Notice that if the algorithm is not able to return a Hamiltonian cycle, this means that, with high probability, no Hamiltonian cycle exists in the graph.

As a consequence of Lemmas 5.3 and 6.4, we have:

Theorem 6.2. With high probability, the FGeo-HAM algorithm returns in polynomial time a Hamiltonian cycle on random geometric graphs with radius $r_n = \sqrt{a_n/n}$, where $r_n \to 0$ and $a_n/\log n \to \infty$, and random edge faults, provided the failure probability is constant.

6.4 Emulations

In this section we first study the emulation of random geometric graphs in random geometric graphs with vertex faults. Then, we study the emulation of random geometric graphs in random geometric graphs with edge faults.

6.4.1 Emulation in RGGs with faulty vertices

We introduce an algorithm to compute an emulation of a geometric graph G on a vertex induced subgraph F of G. The basic ideas of this algorithm are: First, use a round-robin strategy to map all the vertices of G in each box to the vertices of F in the same box. Second, emulate edges between neighboring boxes by the mapping of their end vertices. Third, emulate edges between non neighboring boxes by selecting an auxiliary vertex (hop) in an intermediate box assigned with a round-robin strategy.

Algorithm 6.2 (One hop emulation). Let G_n be a geometric graph with n vertices and radius r_n and let F_n be a vertex induced subgraph of G_n. The *one hop emulation* ϕ of G_n in F_n is given by the following rules:

1. Dissect the unit square into $\kappa_n = 4\lceil 1/r \rceil^2$ boxes of side $s_n = 1/2\lceil 1/r_n \rceil$. Let $B(i)$ be the i-the box of this dissection, according to some ordering.

2. For any i, let $\alpha(i)$ and $\beta(i)$ denote the number of vertices of G_n and F_n respectively in $B(i)$. Let $u^i_0, \ldots, u^i_{\alpha(i)-1}$ be the vertices of G_n in $B(i)$, and let $v^i_0, \ldots, v^i_{\beta(i)-1}$ be the vertices of F_n in $B(i)$.

3. If $\beta(i) = 0$ for some i, abort.

4. For all i and for all $0 \leqslant j < \alpha(i)$, let $\phi(u^i_j) = v^i_{j \bmod \beta(i)}$.

5. For all i and for all $0 \leqslant j < k < \alpha(i)$, let $\phi(u^i_j u^i_k) = \langle \phi(u^i_j)\phi(u^i_k)\rangle$.

6. For all pair of neighboring boxes $B(i)$ and $B(j)$, for all $0 \leqslant k < \alpha(i)$, and for all $0 \leqslant l < \alpha(j)$, let $\phi(u^i_k u^j_l) = \langle \phi(u^i_k)\phi(u^j_l)\rangle$.

7. Let $B(i)$ and $B(j)$ be any different non neighboring boxes such that a box $B(k)$ is neighbor of both $B(i)$ and $B(j)$ and no box $B(l)$ is neighbor of both $B(i)$ and $B(j)$ for $l < k$. Then, for all $0 \leqslant p < \beta(i)$ and for all $0 \leqslant q < \beta(j)$, define $w(v^i_p, v^j_q) = v^k_{(p+q) \bmod \beta(k)}$. For all $0 \leqslant r < \alpha(i)$ and all $0 \leqslant s < \alpha(j)$, if $u^i_r u^j_s \in E(G_n)$, let $\phi(u^i_r u^j_s) = \langle \phi(u^i_r)w, w\phi(u^j_s)\rangle$, where $w = w(\phi(u^i_r), \phi(u^j_s))$.

The following lemma states that the one hop emulation is able to emulate a nice graph G_n on a friendly vertex induced subgraph of G_n with constant dilation, congestion and load.

Lemma 6.5. For any constants $\epsilon \in (0, \frac{1}{5})$ and $f \in (0, 1)$ and for all sufficiently large n, let G_n be an ϵ-nice geometric graph with n vertices and radius r_n. Also, let F_n be an (ϵ, f)-friendly vertex induced subgraph of G_n. Then, there exist two positive constants c_0 and l_0 independent of n such that the one hop algorithm computes an emulation ϕ of G_n on F_n with $d(\phi) = 2$, $c(\phi) \leqslant c_0$ and $l(\phi) \leqslant l_0$.

Proof. By the friendness of F_n, $\beta(i) > 0$ for all i. So, the one hop algorithm does not abort at Step 3, and a one hop emulation ϕ can be obtained. The computed emulation is correct, because the algorithm satisfies the conditions given in Section 6.2.

By the definition of the one hop algorithm, ϕ maps all edges in G_n to a path consisting of at most two edges in F_n; therefore $d(\phi) \leqslant 2$.

As G_n is ϵ-nice and F_n is (ϵ, f)-friendly, the round-robin mapping of vertices in Step 4 ensures that any vertex $v^i_j \in V(F_n)$ in any box i has load at most $1 + \lceil \alpha(i)/\beta(i) \rceil$, which is at most $2 + (1+\epsilon)(1-\epsilon)^{-2}(1-f)^{-1}$. Therefore, $l(\phi) \leqslant 2 + (1+\epsilon)(1-\epsilon)^{-2}(1-f)^{-1} = l_0$, which is a constant.

The congestion of an edge can be decomposed as the sum of the congestions produced by Steps 5, 6 and 7. The congestion of an edge $v^i_j v^i_k$ produced by Step 5 is at most the product of the loads of v_j^i and of v_k^i, and it is at most l_0^2. Likewise, the congestion of an edge produced by Step 6 is at most the product of the loads of its endpoints, so is at most l_0^2. The congestion of an edge produced by Step 7 is a bit more complicated to calculate: First notice that the number of configurations for which a box is neighbor of two boxes that are not neighbors is at most 40. Assume that k is the smallest integer such that box $B(k)$ is neighbor of two boxes $B(i)$ and $B(j)$. For all $0 \leqslant p < \beta(i)$ and for all $0 \leqslant q < \beta(j)$, vertex $w(v_p^i, v_q^j)$ is the hop to go from v_p^i to v_q^j. By the round-robin assignment of step 7, the number of paths that go from box $B(i)$ to box $B(j)$ using edges $v_p^i w(v_p^i, v_q^j)$ or $w(v_p^i, v_q^j) v_q^j$ is bounded by $\lceil (\alpha(i) + \alpha(j))/\beta(i) \rceil$, which is smaller than $1 + 2l_0$. So, the number of edges in $E(G_n)$ from box $B(i)$ to box $B(j)$ that use the edges $v_p^i w(v_p^i, v_q^j)$ or $w(v_p^i, v_q^j) v_q^j$ is bounded above by the product of the loads of the endpoints times $1 + 2l_0$. Overall, the maximal congestion is $l_0^2 + l_0^2 + 40 \cdot l_0^2 \cdot (1 + 2l_0) = c_0$, which is a constant. □

The combination of Lemmas 5.3, 6.1 and 6.5 implies our main result for random geometric graphs with vertex faults.

Theorem 6.3. With high probability, random geometric graphs with radius $r_n = \sqrt{a_n/n}$, with $r_n \to 0$ and $a_n/\log n \to \infty$, can be emulated with constant slowdown in the presence of random vertex faults, provided the failure probability is constant.

6.4.2 Emulation in RGGs with faulty edges

We associate now a box to each pair $u, v \in V_n$ for which B_u and B_v are two non neighboring boxes. If $B(i)$ is a neighbor of both B_u and B_v but, for $j < i$, no $B(j)$ is a neighbor of both B_u and B_v define $B_{uv} = B(i)$, otherwise define $B_{uv} = B_u$. Observe that, when defined, B_{uv} is a box such that all its vertices are connected to all the vertices in box B_u and to all the vertices in box B_v.

The following definition expresses the fact that edges of nice graphs fail "appropriately:"

Definition 6.3 (Kindly graphs). For any constants $\epsilon \in (0, \frac{1}{5})$ and $f \in (0, 1)$, let G be an ϵ-nice geometric graph with n vertices and radius r_n. Let F be an edge induced subgraph of G. Let us say that F is (ϵ, f)-*kindly* if, for all vertex $u \in V(F)$, $\deg_F(u) \geqslant (1-\epsilon)^3(1-f)a_n$ and if, for all edge $uv \in E(G_n) \setminus E(F)$, there exists a vertex t in B_{uv} such that $ut \in E(F)$ and $tv \in E(F)$.

The following result states that, with high probability, edge induced subgraphs of nice graphs are kindly, provided that $f < \frac{1}{2}$:

Lemma 6.6. For any constants $\epsilon \in (0, \frac{1}{5})$ and $f \in (0, \frac{1}{2})$, let $(G_n)_{n\in\mathbb{N}}$ be a sequence of ϵ-nice graphs with n vertices and radius r_n. Then,

$$\lim_{n\to\infty} \mathbf{Pr}\left[\mathcal{FE}(G_n, f) \text{ is } (\epsilon, f)\text{-kindly}\right] = 1.$$

Proof. Let n be any sufficiently large natural, let F_n be an edge faulty random geometric graph of G_n drawn from $\mathcal{FE}(G_n, f)$. We first compute the probability $\pi_1(n)$ that some vertex in F has degree less than $(1-\epsilon)^3(1-f)a_n$. Afterwards we compute the probability $\pi_2(n)$ that for some edge $uv \in E(G) \setminus E(F_n)$ no vertex $t \in B_{uv}$ satisfies $u \sim t$ and $t \sim v$.

Let u be a vertex in G_n. As G_n is ϵ-nice, each box contains at least $\frac{1}{4}(1-\epsilon)a_n$ vertices. As u is connected to all other vertices in its box and to all the vertices in its neighboring boxes, we have

$$\deg_{G_n}(u) \geqslant \tfrac{3}{4}(1-\epsilon)a_n + \tfrac{1}{4}(1-\epsilon)a_n - 1 \geqslant (1-\epsilon)^2 a_n.$$

As each edge of G_n fails independently with probability f, we have that $\mu = \mathbf{E}\left[\deg_{F_n}(u)\right] \geqslant (1-\epsilon)^2(1-f)a_n$. Using Chernoff's bounds we get

$$\mathbf{Pr}\left[\deg_{F_n}(u) < (1-\epsilon)\mu\right] \leqslant \exp(-\tfrac{1}{2}\epsilon^2\mu) \leqslant n^{-(1-f)(1-\epsilon)^2\epsilon^2 b_n/2}.$$

As the number of boxes is certainly bounded by n, applying Boole's inequality, we get $\pi_1(n) \leqslant n^{1-(1-f)(1-\epsilon)^2\epsilon^2 b_n/2}$, which implies $\pi_1(n) \to 0$.

Using Boole's inequality and de Morgan's laws, we have

$$\begin{aligned}
\pi_2(n) &= \mathbf{Pr}\left[\exists uv \in E(G_n) \ : \ (u \not\sim v \wedge \neg\exists t \in B_{uv} \ : \ (u \sim t \wedge t \sim v))\right] \\
&\leqslant \sum_{uv\in E(G_n)} \mathbf{Pr}\left[u \not\sim v \wedge \neg\exists t \in B_{uv} \ : \ (u \sim t \wedge t \sim v)\right] \\
&= \sum_{uv\in E(G_n)} \mathbf{Pr}\left[u \not\sim v\right] \cdot \mathbf{Pr}\left[\forall t \in B_{uv} \ : \ (u \not\sim t \vee t \not\sim v)\right] \\
&= \sum_{uv\in E(G_n)} f \prod_{t\in B_{uv}} \mathbf{Pr}\left[u \not\sim t \vee t \not\sim v\right] \\
&\leqslant f \sum_{uv\in E(G_n)} \prod_{t\in B_{uv}} \left(\mathbf{Pr}\left[u \not\sim t\right] + \mathbf{Pr}\left[t \not\sim v\right]\right) \\
&= f \sum_{uv\in E(G_n)} \prod_{t\in B_{uv}} 2f \\
&\leqslant f n^2 (2f)^{(1-\epsilon)^2 a_n/4} = f n^{\ln(2f)(1-\epsilon)^2 b_n/4+2}.
\end{aligned}$$

Since $\ln(2f)$ is negative, then $\pi_2(n) \to 0$.

So, by Boole's inequality, the probability that F_n is not (ϵ, f)-kindly is smaller than $\pi_1(n) + \pi_2(n)$, which tends to zero as n tends to infinity $\square$

We introduce now a randomized algorithm to compute an emulation of a geometric graph G on an edge induced subgraph of G. The basic idea is to map all vertices to themselves, map all alive edges to themselves, and map death edges to paths of two live edges with one hop, selected at random in some suitable box.

Algorithm 6.3 (Randomized one hop emulation). Let G_n be a geometric graph with n vertices and radius r_n, and let F_n be an edge induced subgraph of G_n. The *randomized one hop emulation* ϕ of G_n on F_n is given by the following three steps:

1. Dissect the unit square.
2. For all $u \in V(G_n)$, let $\phi(u) = u$ and for all $uv \in E(F_n)$, let $\phi(uv) = \langle uv \rangle$.
3. For all $uv \in E(G_n) \setminus E(F_n)$, let $\phi(uv) = \langle uw, wv \rangle$ where w is chosen at random among all vertices $t \in B_{uv}$ such that $ut \in E(F_n)$ and $tv \in E(F_n)$. If no such t exists, abort.

Given G_n and F_n, in the case that the above algorithm does not abort, it returns a random emulation ϕ; therefore, $l(\phi)$, $d(\phi)$ and $c(\phi)$ are random variables. The following theorem establishes their asymptotic probabilistic behavior.

Lemma 6.7. For any constants $\epsilon \in (0, \frac{1}{5})$ and $f \in (0, \frac{1}{2})$ and for all $n \in \mathbb{N}$, let G_n be an ϵ-nice geometric graph with n vertices and radius r_n, and let F_n be an (ϵ, f)-kindly edge induced subgraph of G_n. Then, the randomized one hop algorithm computes an emulation ϕ_n for G_n on F_n, such that

$$l(\phi_n) = 1, \quad d(\phi_n) \leqslant 2, \quad \text{and} \quad \lim_{n\to\infty} \mathbf{Pr}\left[c(\phi_n) \leqslant \log n / \log\log\log n\right] = 1.$$

Proof. The randomized one hop algorithm cannot abort because F_n is (ϵ, f)-kindly. So, the randomized algorithm computes an emulation ϕ_n. By the definition of our randomized one hop emulation, all vertices in G_n are mapped to themselves; thus $l(\phi_n) = 1$. Moreover, every edge in G_n is mapped to a path with at most two edges; thus $d(\phi_n) \leqslant 2$.

Let $c(\phi_n, uv)$ denote the congestion of an edge $uv \in E(F_n)$ in the emulation ϕ_n: $c(\phi_n, uv)$ denotes the number of edges in G_n mapped to uv by ϕ_n. Then, by linearity of expectation,

$$\begin{aligned}\mu &= \mathbf{E}\left[c(\phi_n, uv)\right] \\ &\leqslant 1 + \sum_{w \in B_{uv}} \mathbf{Pr}\left[w \not\sim v\right] \mathbf{Pr}\left[w \sim u\right] \mathbf{Pr}\left[\phi_n(wv) = \langle wu, uv \rangle\right]\end{aligned}$$

$$\leqslant 1 + (1+\epsilon)\tfrac{1}{4}a_n \cdot f \cdot (1-f) \cdot \frac{1}{(1-\epsilon)^3(1-f)a_n}$$
$$\leqslant 1 + \frac{f(1+\epsilon)}{4(1-\epsilon)^3} = C_{\epsilon,f}.$$

So, it holds that $1 \leqslant \mu \leqslant C_{\epsilon,f}$. Assume n large enough and take $d = \log n/\mu \log\log\log n - 1$. Using the general Chernoff's bounds (Theorem A.8), we have

$$\mathbf{Pr}\left[c(\phi_n, uv) > \frac{\log n}{\log\log\log n}\right] \leqslant \left(\frac{e^{\frac{\log n - \mu \log\log\log n}{\mu \log\log\log n}}}{\left(\frac{\log n}{\mu \log\log\log n}\right)^{\frac{\log n}{\mu \log\log\log n}}}\right)^{\mu}$$
$$\leqslant \left(\frac{e\mu}{\log n}\right)^{\log n/\log\log\log n}.$$

As the number of edges in F_n is less than n^2, using Boole's inequality we can affirm that the probability that some edge in F_n has congestion greater than $\log n/\log\log\log n$ is bounded above by

$$n^2\left(\frac{e\mu}{\log n}\right)^{\log n/\log\log\log n},$$

which tends to zero. □

Notice that the previous proof can be strengthen to allow any congestion growing as $\omega(\log n/\log\log n)$. In any case, the congestion is a sub-logarithmic function in the number of vertices.

The combination of Lemmas 5.3, 6.6 and 6.7 implies our main result for random geometric graphs with edge faults.

Theorem 6.4. With high probability, random geometric graphs with n vertices and radius $r_n = \sqrt{a_n/n}$, with $r_n \to 0$ and $a_n/\log n \to \infty$, can be emulated with slowdown $\log n/\log\log\log n$, in the presence of random edge faults, provided the failure probability is constant and smaller than $\frac{1}{2}$.

Observe that in the case of faulty edges we had to deal with a randomized emulation, so the slowdown is a bit larger than in the case of faulty vertices. We suspect that a deterministic algorithm could achieve a constant congestion. Also notice that in this case we required $f < \frac{1}{2}$, which was not necessary for faulty vertices.

6.5 Layout problems

In this section we deal with the EDGEBIS, CUTWIDTH and MINLA layout problems. We will develop the case of random geometric graphs with edge faults and finish by pointing out the results for the case of vertex faults.

The following definition captures the property that edges of a geometric graph are appropriately distributed in the unit square. This definition is similar to the definition of mixing graphs (Definition 3.3), in the sense that it relates the size of a cut between two sets of vertices with their size.

Definition 6.4 (Melting graphs). For any constants $\epsilon \in (0,1)$ and $f \in (0,1)$, let $G = (V, E)$ be an ϵ-nice geometric graph with n vertices and radius r_n. Let $F = (V, E')$ be an edge induced subgraph of G. Let us say that F is (ϵ, f)-*melting* if for every two disjoint vertex sets A and B such that $|A| \geqslant \epsilon a_n$, $|B| \geqslant \epsilon a_n$, A is inside a box B_A and B is inside a box B_B, and either $B_A = B_B$ or B_A is a neighbor of B_B, then

$$\theta'(A, B) \geqslant (1-\epsilon)(1-f)|A||B|$$

where $\theta'(A, B) = |\{uv \in E' \ : \ u \in A \text{ and } v \in B\}|$.

The following lemma shows that, with high probability, nice geometric graphs are melting when their edges fail:

Lemma 6.8. For any constants $\epsilon \in (0,1)$ and $f \in (0,1)$ and for all n, let G_n be an ϵ-nice geometric graph with n vertices and radius r_n. Then,

$$\lim_{n\to\infty} \mathbf{Pr}\left[\mathcal{FE}(G_n, f) \text{ is } (\epsilon, f)\text{-melting}\right] = 1.$$

Proof. For any $n \in \mathbb{N}$, let G_n be an ϵ-nice geometric graph with n vertices and radius r_n, and let F_n be an edge induced subgraph of G_n drawn from $\mathcal{FE}(G_n, f)$.

Let $A, B \subseteq V(G_n)$ be two disjoint vertex sets such that $|A|, |B| \geqslant \epsilon a_n$, A is inside a box B_A and B is inside a box B_B, and either $B_A = B_B$ or B_A is a neighbor of B_B. We have,

$$\mathbf{E}\left[\theta'(A, B)\right] = \sum_{u\in A}\sum_{v\in B} \mathbf{Pr}\left[uv \in E'\right] = \sum_{u\in A}\sum_{v\in B}(1-f) = |A||B|(1-f).$$

Using Chernoff's bounds, we get

$$\begin{aligned}
\mathbf{Pr}\left[\theta'(A, B) < (1-\epsilon)\mathbf{E}\left[\theta'(A, B)\right]\right] &\leqslant \exp\left(-\tfrac{1}{2}\epsilon^2 \mathbf{E}\left[\theta'(A, B)\right]\right) \\
&= \exp\left(-\tfrac{1}{2}\epsilon^2 (1-f)|A||B|\right) \\
&\leqslant \exp\left(-\tfrac{1}{2}\epsilon^2 (1-f)\epsilon^2 a_n^2\right) \\
&= n^{-(1-f)\epsilon^4 b_n^2 \ln n/2}.
\end{aligned}$$

The number of possible choices for B_A and B_B is certainly smaller than n. Once B_A and B_B are chosen, as G_n is ϵ-nice, there are at most $3^{2(1+\epsilon)a_n/4}$ choices for A and B. So, the probability that some pair of vertex sets A and B satisfying the hypotheses of the lemma have $\theta'(A,B) < (1-\epsilon)(1-f)|A||B|$ is bounded above by

$$n \cdot 3^{2(1+\epsilon)a_n/4} \cdot n^{-\epsilon^4(1-f)b_n^2 \ln n/2} \leqslant n^{1+b_n(-(1-f)\epsilon^4 b_n \ln n/2+2(1+\epsilon)\ln 3)},$$

which tends to zero as n tends to infinity. □

The following lemma is the basis of our lower bounds for melting graphs. Its statement and its proof are an extension of the proof of Lemma 5.4.

Lemma 6.9. For any constants $\epsilon \in (0, \frac{1}{5})$ and $f \in (0,1)$, let G be an ϵ-nice geometric graph with n vertices and radius r_n; let F be a (ϵ, f)-melting subgraph of G. Let φ be any layout of F. Then, for any integer i such that $\alpha = i/n \in (2\epsilon, 1-2\epsilon)$,

$$\theta(i,\varphi,F) \geqslant (1-\epsilon)(1-f)\frac{3(1-5\epsilon)}{8} \min\left\{\sqrt{\alpha-2\epsilon}, \sqrt{1-\alpha-2\epsilon}\right\} n^2 r_n^3.$$

Proof. We assign colors to vertices and boxes: color the first i vertices in the ordering "red" and the remaining vertices "green;" color the boxes containing at most $\frac{1}{5}\epsilon a_n$ green vertices "red," the boxes containing at most $\frac{1}{5}\epsilon a_n$ red vertices "green," and the remaining boxes "yellow." Let Y_n be the number of yellow boxes.

Observe that $\theta(i,\varphi,F)$ is the total number of edges between opposite-color vertices. Let us refer to such edges as "within-box" if the vertices in question lie in the same box, or "between-box" otherwise. We consider two cases:

Case 1: $Y_n \geqslant 25\epsilon^{-2} n^{1/2} a_n^{-1/2}$. As F is melting, for each yellow box, the cost of within-box edges is at least $(1-\epsilon)(1-f)\epsilon^2 a_n^2/25$. Hence,

$$\begin{aligned}\theta(i,\varphi,F) &\geqslant (1-\epsilon)(1-f)(\tfrac{1}{5}\epsilon a_n)^2 \cdot Y_n\\ &\geqslant (1-\epsilon)(1-f)n^{1/2}a_n^{3/2} = (1-\epsilon)(1-f)n^2 r_n^3.\end{aligned}$$

Case 2: $Y_n < 25\epsilon^{-2} n^{1/2} a_n^{-1/2}$. In this case, we consider only between-box edges that are between neighboring boxes, including diagonal neighbors. Consider a particular box containing a total of t vertices, of which r of them are red. Suppose that the total number of red vertices in neighboring boxes is r' and the total number of green vertices in neighboring boxes is g'. Then, the total number of between-box edges of the type we are considering, involving vertices in that particular box, is

$$(rg' + (t-r)r') = (r(g'-r') + tr').$$

Given t, r', and g', the above equation is a linear function of r and it attains its minimum at the interval $[0, t]$ at $r = 0$, at $r = t$ or at both of them. Hence, it is possible to change the vertices in that box to either all red or all green without increasing the total number of between-box edges of the type we are considering.

Let us modify the coloring of vertices by going through the yellow boxes in turn, successively changing the color of vertices in each box either to all red or to all green, whichever does not increase the total number of between-box edges of the type we are considering. When we are done, there will no longer be any yellow boxes. Let R_n be the number of red boxes and G_n the number of green boxes based on this modified coloring.

By niceness of G, the number of vertices whose color has been changed is at most

$$Y_n \cdot (1+\epsilon)\tfrac{1}{4}a_n \leqslant 25\epsilon^{-2}n^{1/2}a_n^{1/2} = 25\epsilon^{-2}nr_n.$$

Thus, for n so large that $25\epsilon^{-2}r_n \leqslant \epsilon$, the number of red vertices in the modified coloring is at least $(\alpha - \epsilon)n$.

By definition of "green box," the number of red vertices in green boxes is at most $\frac{1}{5}\epsilon a_n 4\lceil 1/r_n\rceil^2 \leqslant \epsilon n$. Thus, the total number of red vertices in red boxes in the modified coloring is at least $(\alpha - 2\epsilon)n$, for n large enough. By a similar argument, the number of green vertices in green boxes in the modified coloring is at least $(1-\alpha-2\epsilon)n$, for n large enough.

As G and F are ϵ-nice, no box can contain more than $\frac{1}{4}(1+\epsilon)a_n$ vertices and there are at least $(\alpha-2\epsilon)n$ red vertices in red boxes and $(1-\alpha-2\epsilon)n$ green vertices in green boxes, we have

$$R_n \geqslant \frac{(\alpha-2\epsilon)n}{\frac{1}{4}(1+\epsilon)a_n} = \frac{4(\alpha-2\epsilon)}{(1+\epsilon)r_n^2}$$

and

$$G_n \geqslant \frac{(1-\alpha-2\epsilon)n}{\frac{1}{4}(1+\epsilon)a_n} = \frac{4(1-\alpha-2\epsilon)}{(1+\epsilon)r_n^2}.$$

Let ∂F denote the number of pairs of neighbor boxes of opposite colors in F with the modified coloring. By Proposition 5.1,

$$\partial F \geqslant 3\min\left\{\sqrt{R_n},\ \sqrt{G_n}\right\} \geqslant \frac{6}{r_n}\min\left\{\sqrt{\frac{\alpha-2\epsilon}{1+\epsilon}},\ \sqrt{\frac{1-\alpha-2\epsilon}{1+\epsilon}}\right\}.$$

By niceness and the definition of box coloring, each red box contains at least $(1-2\epsilon)\frac{1}{4}a_n$ red vertices, and each green box contains at least $(1-2\epsilon)\frac{1}{4}a_n$ green vertices. As F is (ϵ, f)-melting,

$$\theta(i, \varphi, F) \geqslant \partial F(1-\epsilon)(1-f)\tfrac{(1-2\epsilon)^2}{16}a_n^2$$

$$\geqslant (1-\epsilon)(1-f)\tfrac{3(1-2\epsilon)^2}{8\sqrt{1+\epsilon}}a_n^{3/2}n^{1/2}\min\left\{\sqrt{\alpha-2\epsilon},\ \sqrt{1-\alpha-2\epsilon}\right\}$$
$$\geqslant (1-\epsilon)(1-f)\tfrac{3(1-5\epsilon)}{8}n^2r_n^3\min\left\{\sqrt{\alpha-2\epsilon},\ \sqrt{1-\alpha-2\epsilon}\right\}.$$

This lower bound is smaller than the one for Case 1 and thus holds for both cases. □

The following result presents lower bounds for the layout problems, considered on melting graphs:

Lemma 6.10 (Lower bounds). For any constants $\epsilon \in (0, \frac{1}{5})$ and $f \in (0,1)$ and for all n large enough, let G_n be an ϵ-nice geometric graph with n vertices and radius r_n; let F_n be a (ϵ, f)-melting subgraph of G_n. Then, the following lower bounds hold:

$$\textsc{mineb}(F_n) \geqslant (1-\epsilon)(1-f)\tfrac{3(1-8\epsilon)}{8\sqrt{2}}\cdot n^2r_n^3, \tag{6.3}$$
$$\textsc{mincw}(F_n) \geqslant (1-\epsilon)(1-f)\tfrac{3(1-8\epsilon)}{8\sqrt{2}}\cdot n^2r_n^3, \tag{6.4}$$
$$\textsc{minla}(F_n) \geqslant (1-\epsilon)(1-f)\tfrac{(1-42\sqrt{\epsilon})}{4\sqrt{2}}\cdot n^3r_n^3. \tag{6.5}$$

Proof. The proofs of (6.3) and (6.4) are obtained from Lemma 6.9 by setting $i = \lfloor n/2 \rfloor$. To prove (6.5), take any layout φ of F_n. Then, by Lemma 6.9,

$$\begin{aligned}\textsc{la}(\varphi, F_n) = \sum_{i=1}^{n}\theta(i,\varphi,F_n) &\geqslant \sum_{2\epsilon n<i<(1-2\epsilon)n}\theta(i,\varphi,F_n)\\ &\geqslant (1-\epsilon)(1-f)\tfrac{3(1-5\epsilon)}{4}n^2r_n^3\sum_{2\epsilon n<i<n/2}\sqrt{i/n-2\epsilon}.\end{aligned}$$

Using the facts that $a > b$ implies $\sqrt{a-b} \geqslant \sqrt{a}-\sqrt{b}$ and that $\sum_{k=1}^{m}\sqrt{k} \geqslant \frac{2}{3}m^{3/2} + O\left(\sqrt{m}\right)$ we obtain (6.5) by successive minorizations. □

To obtain upper bounds on the cost of the layout problems on faulty graphs, we make use of the upper bounds computed by the projection algorithm; see algorithm 5.1).

Lemma 6.11. For any constants $\epsilon \in (0, \frac{1}{5})$ and $f \in (0,1)$, let G be a ϵ-nice geometric graph vertices and radius r_n; let F be a (ϵ, f)-melting subgraph of G. Then with high probability, the following upper bounds on the cost of the projected layout π of F hold:

$$\begin{aligned}\textsc{cw}(\pi, F) &\leqslant (1+3\epsilon)^5\cdot n^2r_n^3,\\ \textsc{eb}(\pi, F) &\leqslant (1+3\epsilon)^5\cdot n^2r_n^3,\\ \textsc{la}(\pi, F) &\leqslant (1+3\epsilon)^5\cdot n^3r_n^3.\end{aligned}$$

Proof. By Lemma 1.3, the monotonicity characteristics of MINLA and MINCW suffice to prove that the $\text{LA}(\pi, F) \leqslant \text{LA}(\pi, G)$ and $\text{CW}(\pi, F) \leqslant \text{CW}(\pi, G)$. As far as the number of vertices remains unchanged, the Edge Bisection problem is also monotonic. So we also have $\text{EB}(\pi, F) \leqslant \text{EB}(\pi, G)$. The result is obtained using Lemma 5.11. □

As a consequence of Lemmas 6.8, 6.10 and 6.11, we get:

Theorem 6.5. With high probability, the Projection algorithm is a constant approximation algorithm for the Minimum Linear Arrangement, Cutwidth and Edge Bisection problems on random geometric graphs, with radius $r_n = \sqrt{a_n/n}$, where $r_n \to 0$ and $a_n/\log n \to \infty$, and random edge faults, provided that the failure probability is constant.

In the case of random geometric graphs with vertex faults, the resulting subgraph is also a random geometric graph. It is straightforward to see that a random geometric graph with vertex faults is ϵ-nice with a slight modification in the required number of vertices per box, just to incorporate the failure probability f. Therefore all the results in Lemma 5.11 concerning layout measures are also true for random geometric graphs with random vertex faults. In particular, the following result holds:

Theorem 6.6. With high probability, the Projection algorithm is a constant approximation algorithm for the Minimum Linear Arrangement, Cutwidth and Edge Bisection problems on random geometric graphs, with radius $r_n = \sqrt{a_n/n}$, where $r_n \to 0$ and $a_n/\log n \to \infty$, and random vertex faults, provided the failure probability is constant.

6.6 Conclusion

In this chapter we have analyzed the computational power of random geometric networks in the presence of random edge or vertex faults, considering several important network properties and parameters. The radius of the random geometric graphs has been chosen so that the non-faulty graph is connected almost surely.

We have first shown that, with high probability, faulty random geometric networks do have a Hamiltonian cycle, provided that the edge or vertex failure probability is constant. Such capability enables performing distributed computations based on end-to-end communication protocols. We have also presented algorithms that, with high probability, construct the Hamiltonian cycle.

Afterwards, we have analyzed how to emulate an original random geometric network G on a faulty network F. Theorem 6.3 states that, under the presence of some natural assumptions, random geometric networks can tolerate a constant vertex failure probability with a constant slowdown. In the case of constant edge failure probability, Theorem 6.4 states that the slowdown is sub-logarithmic in the number of vertices in the graph. We leave as an open problem to seek an emulation with constant slowdown.

We have also considered several network measures, stated as linear layout problems: Edge Bisection, Minimum Linear Arrangement and Cutwidth. Our results have shown that random geometric networks can tolerate a constant edge or vertex failure probability while maintaining the order of magnitude of the measures considered here.

An exposition of the results of Section 6.4 was presented at the *International Conference on Mathematical Foundation of Informatics (Hanoi,* 1999*)*. The results of Section 6.4 and Section 6.5 will appear in *Parallel Processing Letters* [68]. The results in Section 6.3 have been submitted for publication [211]. The case where both vertex and edge failure can happen is left as future work. Other parameters that have been analyzed in the non-faulty version remain open, for instance, k-connectivity or chromatic number [8, 9, 10, 206].

7

Communication Tree Problems

7.1 Introduction

In general, communication problems involve a set of locations with communication requirements between pairs of them. The goal is to establish a communication pattern, often a tree, optimizing some communication cost. Problems in which a communication tree has to be constructed arise in many applications. For instance, in phone communication, it is usual to have several locations with a known expected number of phone calls between each pair of locations. In this case, the goal is to design a network to handle these calls in an optimal way. In distributed or mobile computing, there are shared resources as disks, input, output devices, etc., and system requirements that force to establish an optimal point-to-point communication. In tree-structured computations, the computational activity often is limited to the leaves of the tree. In such a case, it is important not only to distribute the tasks evenly among the leaves but also to build an adequate computation tree taking into account the communication parameters. In all cases, it makes sense to restrict the maximum degree of the communication tree.

Given a collection of terminal sites where some pairs of them want to exchange information, a *communication tree* is a tree that contains the set of sites but that might not contain direct links between communicating pairs [139].

Following the nomenclature in [222], in the case where the terminal sites must appear as the leaves of the communication tree we speak of a *routing tree*. In the particular case that all internal nodes of a routing tree have degree 3, we speak of a *tree layout*. We model the input by a graph, whose vertices correspond to terminal sites and whose edges join each communicating pair. In this chapter we are interested in the problem of finding communication trees, routing trees or tree layouts minimizing different communication costs like *edge congestion*, *vertex congestion*, *dilation*, *load* and *total communication* (see Section 7.2 for the formal definitions of these terms). Observe that in the case that the maximum degree of the communication tree is 2, the tree becomes a path and the communication tree becomes a linear layout. In this case, the corresponding problems are *Cutwidth*, *Bandwidth*, *Vertex Separation* and *Minimum Linear Arrangement*.

From the previous paragraph, tree layouts are routing trees, and routing trees are communication trees. Thus, a natural question to ask is whether there exists any useful relation between the considered problems on these different types of trees.

Another interesting question is whether a randomly selected tree can provide a good approximation to any of the problems previously described. We answer this on the negative for some of the problems: In particular, we show that, for a given graph, the average cost might be far away from the optimum.

Very few hardness results are known for communication tree problems; Table 7.1 summarizes them. The *minimum dilation communication tree* problem is **NP**-hard even for trees [184]; the *minimum congestion communication tree* problem is **NP**-hard for planar graphs [228], in contrast with the *minimum congestion routing tree* problem, also known as the *carving-width* problem, which is solvable in polynomial time for planar graphs, but is **NP**-hard in general [222]. In [77] it is shown that there is a logarithmic gap between the minimum congestion and the minimum dilation of a given graph, where the minimum is taken over all routing trees with maximum degree 3. Also, the problem of solving the vertex congestion on communication trees and tree layouts is **NP**-complete [7]. In the case that we consider communication trees with unbounded maximum degree the problems become easier, finding a communication tree of minimum total communication cost is in **P** [130], and an analogous result for the case of a routing tree is given in [4].

Due to the lack of efficient or approximation algorithms for communication tree problems, it is interesting to ask for the approximability of these problems when the input graph is obtained from some probabilistic distribution representing the communication requirements. In this chapter, we deal with two models of random graphs: binomial random graphs and random geometric graphs; see Definitions 3.1 and 5.3 respectively.

Problem	Communication tree	Tree layout
Dilation	**NP**-complete for trees when $d = 3$ [184]	Open
Congestion	**NP**-complete when G is planar **P** if G is outerplanar for any $d \geqslant 3$ [228]	**NP**-complete **P** if G is planar for $d \geqslant 3$ [222]
V. Congestion	**NP**-complete [7]	**NP**-complete [7]
Comm. Load	Open	Open
Length	Open	Open

Table 7.1: Complexity results for communication problems.

The outline of this chapter is as follows: In Section 7.2 we define all the measures, costs and problems we are interested in. In Section 7.3, we relate the costs of the problems on communication trees, routing trees and tree layouts. From this point on, we only will have to deal with tree layouts. In Section 7.4 we analyze the average cost of a random tree layout. Then, in Section 7.5, we study the approximability of tree layout problems on binomial random graphs. Square meshes are considered in Section 7.6 as the base for building approximation algorithms for tree layout problems on random geometric graphs in Section 7.7. Finally, in Section 7.8 we present the conclusions of this chapter.

7.2 Problems and preliminaries

Let us introduce some notation used all through the chapter: For an undirected graph G, we denote by $\operatorname{diam}(G)$ its diameter. Given a graph G and an edge uv of $E(G)$, $G \setminus \{uv\}$ represents the graph $(V(G), E(G) \setminus \{uv\})$. Unless explicitly said all our trees are non-rooted. Any node of a tree with degree one will be called a *leaf*, any non-leaf node of a tree will be called an *internal* node. Let $L(T)$ denote the set of leaves of a tree T. Given a tree T and two nodes $x, y \in V(T)$, let $d_T(x, y)$ denote the distance between x and y in T, counted as the number of edges of the unique path joining x and y.

The following definition formally defines the general concept of communication tree for a given graph.

Definition 7.1 (Communication tree). Given a graph G, a *communication tree* for G is a tree T such that $V(G) \subseteq V(T)$.

A communication tree T for a graph G associates to each edge $uv \in E(G)$ the unique path $P_T(uv)$ that connects u and v in T. Notice that no relationship is required between the set of edges of the graph and the set of edges of the tree. Also remark that if T is a path and $V(T) = V(G)$, a communication tree represents two linear layouts φ and φ^R.

We are particularly interested in communication trees whose set of leaves coincide with the vertices of the graph:

Definition 7.2 (Routing tree). A *routing tree* for a graph G is a communication tree T such that $V(G) = L(T)$.

In the following, we will restrict our attention to routing trees with bounded degree, and specially to routing trees whose internal nodes have degree 3:

Definition 7.3 (Tree layout). A *tree layout* for a graph G is a routing tree such that every non-leaf node has exactly degree 3.

See Figure 7.1 for an illustration of the different communication trees.

Given a graph G and a communication tree T of G, we define the following measures:

- The *dilation* $\lambda(uv, T, G)$ of an edge uv in $E(G)$ is the distance from u to v in T.
- The *edge congestion* $\theta(xy, T, G)$ of an edge xy in $E(T)$ is the number of edges uv in $E(G)$ such that the path from u to v in T traverses xy.
- The *vertex congestion* $\vartheta(x, T, G)$ of a vertex x in $V(T)$ is the number of edges uv in $E(G)$ such that the path from u to v in T goes through x.
- The *communication load* $\delta(xy, T, G)$ of an edge xy in $E(T)$ is the number of vertices u in G such that some of its neighboring vertices in G lies in a different component of T after the removal of the edge xy.

Table 7.2 summarizes these definitions.

A function $F(T, G)$ mapping a communication tree T and a graph G to the positive numbers is called a *communication tree cost*. For any communication tree cost F and any graph G,

- d-MINC$F(G)$ is the minimum value of F over all communication trees T for G with $\Delta(T) \leqslant d$.
- d-MINR$F(G)$ is the minimum value of F over all routing trees T for G with $\Delta(T) \leqslant d$.

$P_T(uv)$	$=$	Path from u to v in T for $uv \in E(G)$
$d_T(u,v)$	$=$	Distance between u and v in T
$\lambda(uv,T,G)$	$=$	$d_T(u,v)$
$\theta(xy,T,G)$	$=$	$\|\{uv\ :\ uv \in E(G) \text{ and } P_T(uv) \text{ contains } xy\}\|$
$\vartheta(x,T,G)$	$=$	$\|\{uv\ :\ uv \in E(G) \text{ and } P_T(uv) \text{ contains } x\}\|$
$\delta(xy,T,G)$	$=$	$\|\{u\ :\ \exists uv \in E(G) \text{ such that } u \text{ and } v \text{ lie in different components of } T \setminus \{xy\}\}\|$

Table 7.2: Communication tree measures for a communication tree T of a graph G. In the table, $uv \in E(G)$, $u, v \in V(G)$, $xy \in E(T)$ and $x \in V(T)$.

Problem	Name	Cost
Min. Tree Dilation	MinTD	$\textsc{td}(T,G) = \max_{uv \in E(G)} \lambda(uv,T,G)$
Min. Tree Congestion	MinTC	$\textsc{tc}(T,G) = \max_{xy \in E(T)} \theta(xy,T,G)$
Min. Tree Vertex Congestion	MinTVC	$\textsc{tvc}(T,G) = \max_{x \in V(T)} \vartheta(x,T,G)$
Min. Tree Comm. Load	MinTCL	$\textsc{tcl}(T,G) = \max_{xy \in E(T)} \delta(xy,T,G)$
Min. Tree Length	MinTL	$\textsc{tl}(T,G) = \begin{cases} \sum_{uv \in E(G)} \lambda(uv,T,G) \\ \sum_{xy \in E(T)} \theta(xy,T,G) \end{cases}$

Table 7.3: Tree layout problems. Here G stands for a graph and T for a tree layout of G.

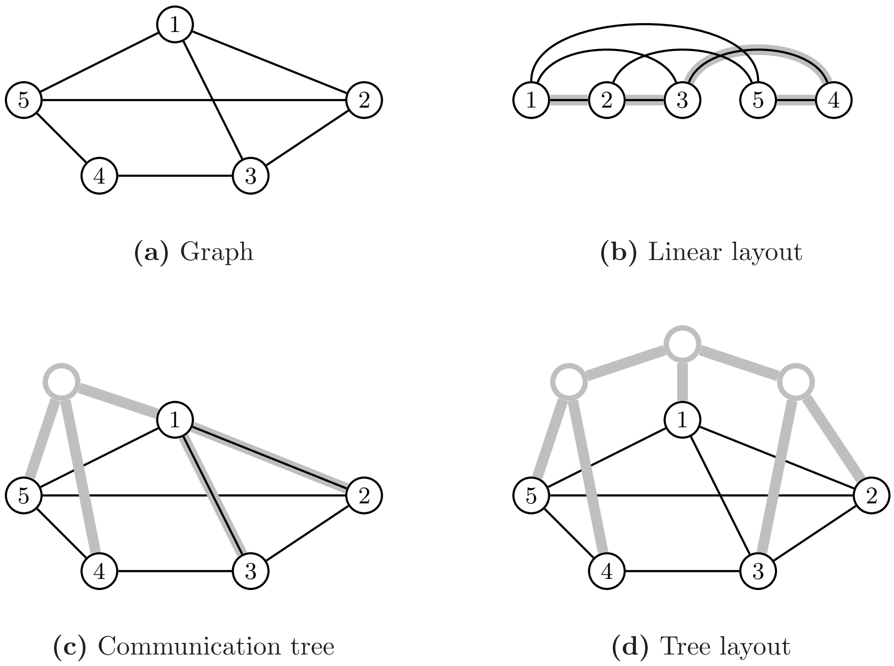


Figure 7.1: Different types of communication trees for a graph. Thin edges represent edges of the graph; bold edges represent edges of the communication tree.

- d-MIN$F(G)$ is the minimum value of F over all routing trees for G with internal nodes of degree exactly d.

We will omit the d- prefix when $d = 3$, so that, MIN$F(G)$ is the minimum value of F over all tree layouts.

Given a graph G, the tree layout problem for a cost F consists in finding an optimal tree layout T^* such that $F(T^*, G)$ is minimal, that is $F(T^*, G) =$ MIN$F(G)$. We are interested in the following problems:

- Minimum Tree Dilation (MINTD): Given a graph G, find a tree layout T^* such that $\text{TD}(T^*, G) = \text{MINTD}(G)$, where

$$\text{TD}(T, G) = \max_{uv \in E(G)} \lambda(uv, T, G).$$

- Minimum Tree Congestion (MINTC): Given a graph G, find a tree layout

T^* such that $\textsc{tc}(T^*, G) = \textsc{mintc}(G)$, where

$$\textsc{tc}(T, G) = \max_{xy \in E(T)} \theta(xy, T, G).$$

- Minimum Tree Vertex Congestion (MinTVC): Given a graph G, find a tree layout T^* such that $\textsc{tvc}(T^*, G) = \textsc{mintvc}(G)$, where

$$\textsc{tvc}(T, G) = \max_{x \in V(T)} \vartheta(x, T, G).$$

- Minimum Tree Communication Load (MinTCL): Given a graph G, find a tree layout T^* such that $\textsc{tcl}(T^*, G) = \textsc{mintcl}(G)$, where

$$\textsc{tcl}(T, G) = \max_{xy \in E(T)} \delta(xy, T, G).$$

- Minimum Tree Length (MinTL): Given a graph G, find a tree layout T^* such that $\textsc{tl}(T^*, G) = \textsc{mintl}(G)$, where

$$\textsc{tl}(T, G) = \sum_{uv \in E(G)} \lambda(uv, T, G) = \sum_{xy \in E(T)} \theta(xy, T, G).$$

The definitions of these tree layout problems are summarized in Table 7.3.

The above problems, defined for tree layouts, can easily be generalized to communication tree and routing problems, both restricted to trees with bounded degree.

The following basic upper bounds on the cost of a tree layout will be useful.

Lemma 7.1. Let G be any graph with n nodes and m edges. Let T be any tree layout of G. Then,

$$\begin{aligned}
\textsc{tc}(T, G) &\leqslant m, \\
\textsc{tvc}(T, G) &\leqslant m, \\
\textsc{tcl}(T, G) &\leqslant n, \\
\textsc{td}(T, G) &\leqslant \mathrm{diam}(T), \\
\textsc{tl}(T, G) &\leqslant m \cdot \mathrm{diam}(T).
\end{aligned}$$

As we always have a tree layout with n leaves and diameter $\log n + 1$, the previous lemma implies that $\textsc{td}(T, G) \leqslant \log n + 1$ and $\textsc{tl}(T, G) \leqslant m \log n + m$. On the other hand, $\textsc{tc}(T, G)$ and $\textsc{tvc}(T, G)$ can be $\Theta(n^2)$ and $\textsc{tcl}(T, G)$ can be $\Theta(n)$, for instance in the case that G is a complete graph on n vertices.

The following basic inequalities follow from the definitions.

Lemma 7.2. Let G be any graph with n nodes and m edges and let T be any tree layout for G. Then, $\text{TCL}(T,G) \leqslant \text{TC}(T,G) \leqslant \text{TVC}(T,G) \leqslant 3\text{TC}(T,G)$.

Let us finish this section with a result on trees that we will need later. We say that an edge in a tree T is a *s-splitter* if its removal splits T in two rooted trees, each one with at least $\lfloor s \rfloor$ leaves. The following lemma is similar to a result referred in [173].

Lemma 7.3. Let T be any tree layout with n leaves. Then, T always contains a $\frac{1}{3}n$-splitter edge.

Proof. We prove it by induction on the number of leaves. The base case is $n = 3$. There only exists a tree layout with three leaves, which satisfies the property. Assume that any tree layout with n' leaves always contains a $\frac{1}{3}n'$-splitter edge for some $n' \geqslant 3$. Let T be any tree with $n = n' + 1$ leaves.

Take any node with two adjacent leaves from T and substitute this group by a marked leaf m, obtaining a new tree T' with $n' = n - 1$ leaves. By the induction hypothesis, T' contains a $\frac{1}{3}n'$-splitter edge uv that splits T' into two rooted trees T'_u and T'_v rooted at u and v respectively. Also, uv splits T into two rooted trees T_u and T_v rooted at u and v respectively. Let a' be the number of leaves in T'_u and let b' be the number of leaves in T'_v. Also, let a be the number of leaves in T_u, and let b be the number of leaves in T_v. Without loss of generality, assume $a' \leqslant b'$. This situation is illustrated in Figure 7.2(a).

As uv is a $\frac{1}{3}n'$-splitter edge for T', we have

$$\tfrac{1}{3}n' \leqslant a' \leqslant \tfrac{1}{2}n' \leqslant b' \leqslant \tfrac{2}{3}n' \quad \text{and} \quad a' + b' = n'.$$

We break the proof in several cases:

Case 1: m belongs to T'_u and $\frac{1}{3}n' \leqslant a' < \frac{1}{2}n'$. As $a = a' + 1$, we have $\frac{1}{3}n \leqslant a \leqslant \frac{1}{2}n$ and so uv is a $\frac{1}{3}n$-splitter edge for T.

Case 2: m belongs to T'_u and $a' = \frac{1}{2}n'$. In this case, n must be even, $b' = a'$ and $a = a' + 1$. So, $\frac{1}{2}n \leqslant a = \frac{1}{2}(n+1) \leqslant \frac{2}{3}n$ and $\frac{1}{3}n \leqslant b = \frac{1}{2}(n-1) \leqslant \frac{1}{2}n$. This proves that uv is a $\frac{1}{3}n$-splitter edge for T.

Case 3: m belongs to T'_v and $\frac{1}{2}n' \leqslant b' < \frac{2}{3}n'$. As $b = b' + 1$, we have $\frac{1}{2}n \leqslant b \leqslant \frac{2}{3}n$ and so uv is a $\frac{1}{3}n$-splitter edge for T.

Case 4: m belongs T'_v and $b' = \frac{2}{3}n'$. In this case, n' is a multiple of 3, $a' = \frac{1}{3}n$ and $b = b' + 1$. Let uw and ut be the two edges adjacent to v in T'_v. Let T'_w the subtree of T'_v rooted at w and let T'_t the subtree of T'_v rooted at t. Also, let b'_1 be the number of leaves in T'_w and let b'_2 be the number of leaves in T'_t. Assume, without loss of generality, that $b'_1 \leqslant b'_2$. Notice that $1 \leqslant b'_1 \leqslant \frac{1}{3}n \leqslant b'_2 \leqslant \frac{2}{3}n$. This situation is illustrated in Figure 7.2(b). Several new sub-cases must be considered:

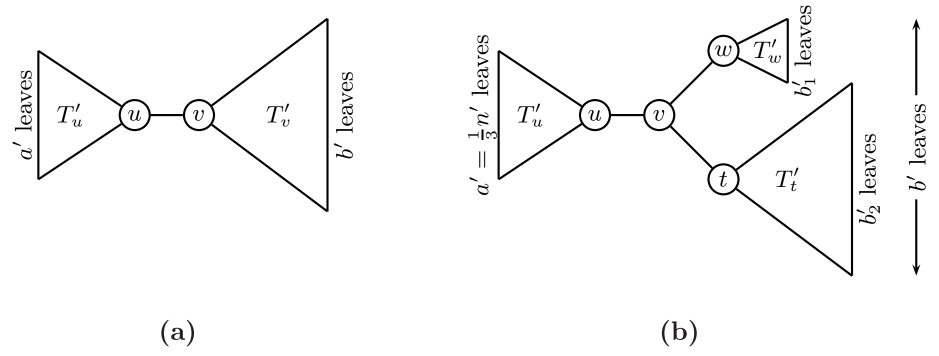


Figure 7.2: Illustration for the proof of Lemma 7.3.

Case 4.1: m belongs T'_v. In this case, let T_1 be the tree rooted at v obtained by splitting T with vt. Let c be the number of leaves in T_1. Then, $c = a' + b'_1$ and so $\frac{1}{3}n \leqslant c \leqslant \frac{2}{3}n$. Therefore, vt is a $\frac{1}{3}n$-splitter edge for T.

Case 4.2: m belongs T'_w and $b'_1 < \frac{1}{3}n'$. As in the previous case, let T_1 be the tree rooted at v obtained by splitting T with vt. Let c be the number of leaves in T_1. Then, $c = a' + b'_1 + 1$ and so $\frac{1}{3}n \leqslant c \leqslant \frac{2}{3}n$. Therefore, vt is also a $\frac{1}{3}n$-splitter edge for T.

Case 4.3: m belongs T'_w and $b'_1 = \frac{1}{3}n'$. In this case, let T_2 be the tree rooted at w obtained by splitting T with vw. Let d be the number of leaves in T_2. Then, $d = b'_1 + 1$ and so $\frac{1}{3}n \leqslant d \leqslant \frac{2}{3}n$. Therefore, vw is a $\frac{1}{3}n$-splitter edge for T.

As there are no other possible cases, the induction step is proved and the lemma follows. □

7.3 Relationships between tree layouts, routing trees and communication trees

In this section we relate the costs of communication tree problems defined on tree layouts to the costs of communication tree problems on routing trees and communication trees. In this section we assume that the maximal degree of a communication tree is bounded by a constant.

Any tree layout is a routing tree, and every routing tree is a communication tree. Therefore we have:

Lemma 7.4. For any cost $F \in \{\text{TD}, \text{TC}, \text{TVC}, \text{TCL}, \text{TL}\}$, any fixed $d \geqslant 3$ and any graph G, we have $d\text{-MINC}F(G) \leqslant d\text{-MINR}F(G) \leqslant d\text{-MIN}F(G)$.

Taking into account that any communication or routing tree T is a valid communication or routing tree for any $d \geqslant \Delta(T)$ we have:

Lemma 7.5. For any cost $F \in \{\text{TD}, \text{TC}, \text{TVC}, \text{TCL}, \text{TL}\}$, any fixed $d \geqslant 3$ and any graph G we have $(d+1)$-MINCF(G) $\leqslant$ d-MINCF(G) and $(d+1)$-MINRF(G) $\leqslant$ d-MINRF(G).

The next theorem relates the optimal costs of the communication tree problems on tree layouts and communication trees.

Theorem 7.1. For any fixed $d \geqslant 3$ and any non-empty graph G,

$$\begin{aligned}
\text{MINTD}(G) &\leqslant c_1 \cdot d\text{-MINCTD}(G),\\
\text{MINTC}(G) &\leqslant c_2 \cdot d\text{-MINCTC}(G),\\
\text{MINTVC}(G) &\leqslant c_3 \cdot d\text{-MINCTVC}(G),\\
\text{MINTCL}(G) &\leqslant c_4 \cdot d\text{-MINCTCL}(G),\\
\text{MINTL}(G) &\leqslant c_5 \cdot d\text{-MINCTL}(G).
\end{aligned}$$

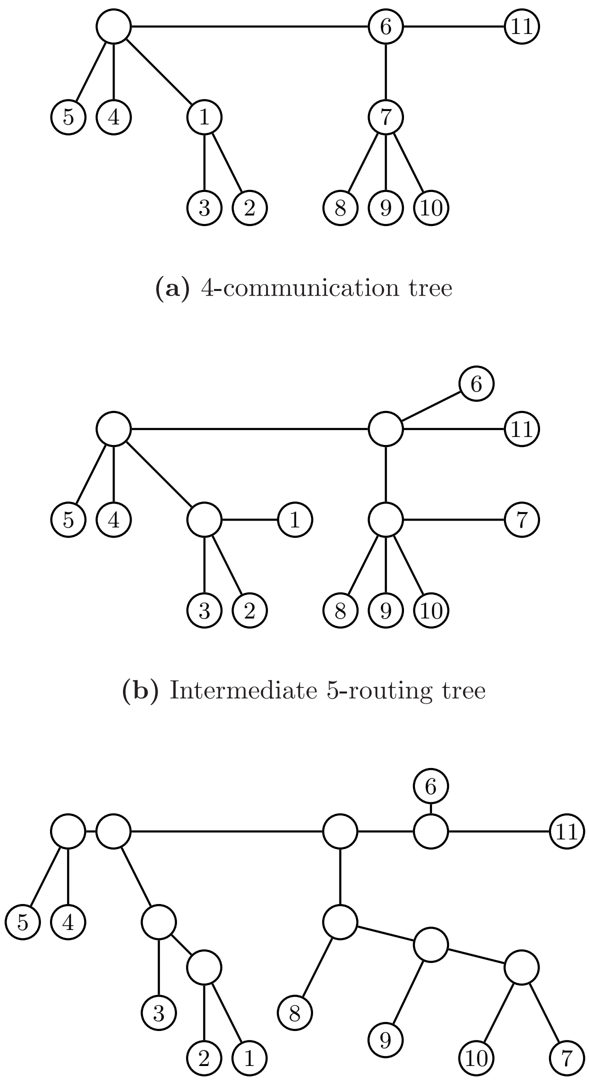
where $c_1, \ldots, c_5$ are positive constants that depend only on d.

Proof. The proof considers a way of constructing a tree layout starting from a communication tree. In Figure 7.3 we give a sketch of the construction. Assume that T is a communication tree for G with $\Delta(T) \leqslant d$. Construct a routing tree T' in the following way: for every vertex $u \in V(G)$, let $t(u)$ be the corresponding node in T. If $t(u)$ is not a leaf, add a new leaf connected to $t(u)$ with label u. The resulting tree is a routing tree T' for G with $\Delta(T') \leqslant d+1$. Then we have,

$$\begin{aligned}
\text{TD}(T', G) &\leqslant \text{TD}(T, G) + 2 \leqslant 3 \cdot \text{TD}(T, G),\\
\text{TC}(T', G) &\leqslant \max\{\text{TC}(T, G), \Delta(G)\} \leqslant d \cdot \text{TC}(T, G),\\
\text{TVC}(T', G) &\leqslant \text{TVC}(T, G),\\
\text{TCL}(T', G) &\leqslant \text{TCL}(T, G),\\
\text{TL}(T', G) &\leqslant \text{TL}(T, G) + 2|E(G)| \leqslant 3 \cdot \text{TL}(T, G).
\end{aligned}$$

To reduce the degree in the routing tree T' we construct a new routing tree T'' for G, as follows: replace every node x in $V(T')$ with degree $\deg(x) > 3$ by a worm layout with $\deg(x)$ leaves. Notice that as $\deg(x) \leqslant d$, the number of nodes in such a caterpillar is a constant that is different for each value of d. Then,

$$\begin{aligned}
\text{TD}(T'', G) &\leqslant d \cdot \text{TD}(T', G),\\
\text{TC}(T'', G) &\leqslant (d-2) \cdot \text{TC}(T', G),\\
\text{TVC}(T'', G) &\leqslant \text{TVC}(T', G),
\end{aligned}$$



(a) 4-communication tree

(b) Intermediate 5-routing tree

(c) Final tree layout

Figure 7.3: Transforming a 4-communication tree into a tree layout.

$$\text{TCL}(T'', G) \leqslant \text{TCL}(T, G'),$$
$$\text{TL}(T'', G') \leqslant d \cdot \text{TL}(T', G).$$

Observe that T'' is a tree layout. Considering as starting point an optimal tree according to each of the costs, the result follows. □

The relationship between the tree layout and congestion presented in the previous theorem can be strengthened for the vertex congestion.

Theorem 7.2. For any graph G, $\text{MINTVC}(G) = \text{MINRTVC}(G) = \text{MINCTVC}(G)$.

Proof. Assume that we have a communication tree T for G with $\text{TVC}(T, G) \geqslant 3$. We will produce a tree layout T' for G with the same cost.

For every vertex $x \in V(G)$ let $t(x)$ be the corresponding node in T. If $t(x)$ has degree 2, we add a new leaf connected to $t(x)$ with label x. If $t(x)$ has degree 3, split one of the three edges out of $t(x)$ by adding a new internal vertex t' and a new leaf connected to t' with label x.

The vertex congestion of the new nodes introduced by the previous construction is less than or equal to the vertex congestion of the node $t(x)$ in the original graph. Therefore $\text{TVC}(T', G) \leqslant \text{TVC}(T, G)$. □

7.4 Average

In this section we study the average costs of the MinTD and MinTL problems over all possible tree layouts for a fixed graph. Our main task here is to identify these trees.

Let us define the basic nomenclature and notation we are going to use through this section.

Recall that two non-rooted trees are different as *plane trees* if one cannot be obtained from the other by continuous operations in the plane [118]. Following Knuth [161], an *ordered tree* is defined recursively as formed by a root and an ordered sequence (possibly empty) of ordered trees, called subtrees of the root. A *non-ordered tree* is defined recursively as formed by a root and a multi-set (possibly empty) of non-ordered trees, called also subtrees of the root. Notice that a non-ordered tree can have multiple representations using ordered trees. The degree $\deg(x)$ of a node x in a non-rooted tree is defined as the number of adjacent nodes to x, whereas the degree $\deg(x)$ of a node x in a rooted tree is defined as the number of subtrees of x. Therefore, a leaf in a rooted tree has degree 0. Let $l(T)$ denote the number of leaves in T. A *k-ary tree* is a rooted tree such that each internal node has degree k.

Given a non-ordered tree T let us consider, for each node $x \in V(T)$, the multi-set of subtrees hanging from x. In this multi-set, subtrees can have

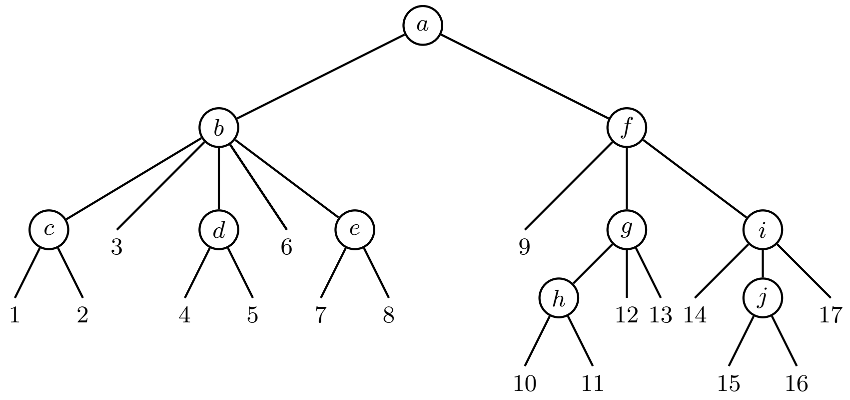


Figure 7.4: A rooted-tree T in which $\beta_2(T) = 9$ (in nodes $b, c, d, e, f, g, h, i, j$), $\beta_3(T) = 1$ (in node b), $\beta_i(T) = 0$ for $i \geqslant 4$.

different degrees of multiplicity. Let $\eta_i(x)$ be the number of different subtrees that have degree of multiplicity exactly i. Let $\eta_i(T) = \sum_{x \in V(T)} \eta_i(x)$; see Figure 7.4.

Given a non-ordered tree T, let $\mathfrak{T}(T)$ be the set of all ordered trees that are representations of T and let $c(T) = |\mathfrak{T}(T)|$. In order to construct the set $\mathfrak{T}(T)$ for T, we have to

- permute in all possible ways all the subtrees of each node $x \in V(T)$,
- and remove the duplicated trees.

From this observation, we obtain an expression for $c(T)$:

Lemma 7.6. The number of ordered trees representing a given non-ordered tree T is

$$c(T) = \frac{\prod_{x \in V(T)} \deg(x)!}{\prod_{i > 1} (i!)^{\eta_i(T)}}.$$

Proof. The total number of permutations of the subtrees is given by

$$\prod_{x \in V(T)} \deg(x)!. \tag{7.1}$$

On the other hand, the number of repetitions obtained by performing the construction described above to obtain $\mathfrak{T}(T)$ is

$$\prod_{i > 1} (i!)^{\eta_i(T)} \tag{7.2}$$

because the repetitions are associated with the existence of identical trees hanging from the same node, and, for every constructed tree, each time that i identical subtrees occur in a node, there are $i!$ identical constructions on the same tree.

As the number of repetitions is the same for every element in $\mathfrak{T}(T)$, the result follows from equations (7.1) and (7.2). $\square$

Given a non-ordered tree T, let $\mathfrak{E}(T)$ be the set on non-ordered trees obtained by labelling the leaves in T with labels in $[l(T)]$ and let $e(T) = |\mathfrak{E}(T)|$. Starting from T we can construct the set $\mathfrak{E}(T)$ by

- considering all the $l(T)!$ possible labellings,
- and removing the duplicated trees.

From this observation, we can also obtain an expression for $e(T)$:

Lemma 7.7. The number of different non-ordered labelled trees obtained by labelling the leaves of a given non-ordered tree T is

$$e(T) = \frac{l(T)!}{\prod_{i>1}(i!)^{\eta_i(T)}}.$$

Proof. The number of labellings obtained by performing the construction described above to obtain $\mathfrak{E}(T)$ is $l(T)!$. As in the proof of Lemma 7.6, the number of repetitions obtained by the above construction to obtain $\mathfrak{E}(T)$ is $\prod_{i>1}(i!)^{\eta_i(T)}$. $\square$

If T is a k-ary tree with n internal nodes, then the number of leaves is $l(T) = (k-1)n+1$. So, from the two previous lemmas, we obtain the following corollary.

Corollary 7.1. For any non-ordered k-ary tree T, $e(T)/c(T)$ is independent on the shape of T and depends only on the size of T. If T has n internal nodes, we get

$$\frac{e(T)}{c(T)} = \frac{((k-1)n+1)!}{k!^n}.$$

Our next goal is to find the average distance between any two leaves, the average being taken among all possible tree layouts.

Definition 7.4 (NLN, RLN and Catalan trees). Let n-NLN denote the set of trees that are non-rooted, non-plane, with n internal nodes, with $n+2$ labeled leaves, and such that each of its n internal nodes has degree 3. Also, let n-RLN denote the set of trees that are rooted, non-ordered, with $n+1$ labelled leaves, and such that each of its n internal nodes has outgoing degree 2. Finally, define the n-Catalan trees as the set of ordered, non-labeled, rooted binary trees with n internal nodes.

Our strategy is to use the well know results on n-Catalan trees to obtain results for n-NLN trees, that is, for tree layouts.

Lemma 7.8. The set of n-NLN trees is isomorphic to the set of n-RLN trees.

Proof. Let us define the following isomorphism between the n-NLN trees and the n-RLN trees: given a n-NLN tree, suppress the leaf with label $n+2$ and make its neighbor the root of the new n-RLN tree. $\square$

The isomorphism just described, will allow us to interchange the study of both families of trees.

Given a Catalan tree T and a property function f on T, as for example internal path length, height, etc, we say that f is *order invariant* if the value of f is the same for all Catalan trees that are equivalent to T as non-ordered trees. In the same manner, given a RLN tree T and a property function f on T, we say that f is *order invariant* if the value of f is the same for all RLN trees that as non-labelled trees are equivalent to T.

Lemma 7.9. For any order invariant property function, its average value is the same on n-RLN trees and on n-Catalan trees.

Proof. Let $\mathfrak{B}_n$ denote the set of all n-Catalan trees, $\mathfrak{E}_n$ the set of all n-RLN trees and $\mathfrak{C}_n$ the set of all non-ordered, rooted and non-labeled binary trees with n internal nodes. Also, let $f(T)$ be an order invariant property function. Do the following decompositions,

$$\mathfrak{B}_n = \bigcup_{T\in\mathfrak{C}_n} \mathfrak{T}(T) \quad \text{and} \quad \mathfrak{E}_n = \bigcup_{T\in\mathfrak{C}_n} \mathfrak{E}(T),$$

where $\mathfrak{T}(T)$ and $\mathfrak{E}(T)$ are restricted to binary trees. As the considered trees have internal nodes with degree 2, and the only possible multiplicities are 1 or 2, using Lemmas 7.6 and 7.7 we have

$$c(T) = |\mathfrak{T}(T)| = \frac{2^n}{2^{\eta_2(T)}} \quad \text{and} \quad e(T) = |\mathfrak{E}(T)| = \frac{(n+1)!}{2^{\eta_2(T)}}.$$

On the other hand,

$$\sum_{T'\in\mathfrak{B}_n} f(T') = \sum_{T\in\mathfrak{C}_n}\sum_{T'\in\mathfrak{T}(T)} f(T') = \sum_{T\in\mathfrak{C}_n} c(T)f(T),$$

and

$$\sum_{T''\in\mathfrak{E}_n} f(T'') = \sum_{T\in\mathfrak{C}_n}\sum_{T''\in\mathfrak{E}(T)} f(T'') = \sum_{T\in\mathfrak{C}_n} e(T)f(T).$$

Using Corollary 7.1 with $k = 2$, and the previous equations, we get

$$\begin{aligned}\sum_{T''\in\mathfrak{C}_n} f(T'') &= \sum_{T\in\mathfrak{C}_n} f(T)e(T)\\ &= \sum_{T\in\mathfrak{C}_n} f(T)c(T)\frac{e(T)}{c(T)}\\ &= \frac{(n+1)!}{2^n}\sum_{T\in\mathfrak{C}_n} f(T)c(T)\\ &= \frac{(n+1)!}{2^n}\sum_{T'\in\mathfrak{B}_n} f(T')\\ &= \frac{e(n)}{c(n)}\sum_{T'\in\mathfrak{B}_n} f(T').\end{aligned}$$

In particular, taking 1 as order invariant property function,

$$\sum_{T''\in\mathfrak{C}_n} 1 = \frac{e(n)}{c(n)}\sum_{T'\in\mathfrak{B}_n} 1.$$

As a consequence,

$$\frac{\sum_{T''\in\mathfrak{C}_n} f(T'')}{\sum_{T''\in\mathfrak{C}_n} 1} = \frac{\sum_{T'\in\mathfrak{B}_n} f(T')}{\sum_{T'\in\mathfrak{B}_n} 1},$$

and we can conclude that the average value of $f(T'')$ for all $T'' \in \mathfrak{C}_n$ is the same that the average value of $f(T')$ for all $T' \in \mathfrak{B}_n$. □

The average distance between two different leaves among all n-Catalan trees is known to be one unit more than the average depth of a leaf, which is $4^n/\binom{2n}{n} - 1 = \sqrt{\pi n} - 1 + o(\sqrt{n})$ (see Section 2.3.4.5 of [161]). This result, together with Lemma 7.9, implies the following theorem.

Theorem 7.3. Given a graph $G = (V, E)$ with $|V| = n$ and $|E| = m \geqslant 1$ the average length for G is $\Theta(m\sqrt{n})$, and the average dilation for G is $\Theta(\sqrt{n})$; these averages being taken over all possible $(n-2)$-NLN trees.

The previous theorem says that using a random $(n-2)$-NLN as tree layout for a graph with n nodes, will provide communication costs far away from the optimal ones, as by Lemma 7.1, selecting a tree layout of diameter $2\log n + 2$ will do better than a randomly selected routing tree. Note however that a tree layout with logarithmic diameter does not always provide the optimum. In particular, when the graph is a line or a cycle, a "worm" layout (caterpillar with hair length 1) gives the optimum (see Figure 7.5).

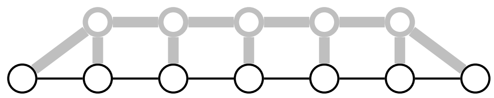


Figure 7.5: A worm (caterpillar tree) layout for a line.

7.5 Binomial random graphs

In this section we consider tree layout problems on binomial random graphs $\mathcal{G}_{n,p_n}$ (see Definition 3.1). We show that all the tree layout problems defined in Section 7.2 are approximable within a constant on mixing graphs (see Definition 3.3). More precisely, our results establish that the cost of any balanced tree layout for a mixing graph is within a constant of the optimal cost. Recall from Lemma 3.2 that, for suitable p_n, with overwhelming probability, $\mathcal{G}_{n,p_n}$ graphs are mixing.

The following result will prove helpful. For any node u in a given tree T and any integer i, let $L_{>i}(T,u)$ denote the set of leaves of T at distance greater than i from u.

Lemma 7.10. Let $\alpha, \beta \in (0,1)$. Let T be a tree with n leaves and with internal nodes of degree 3. Then, for any node u in T, it holds that $L_{>\alpha \log n}(T,u) \geqslant \beta n$ for sufficiently large n.

Proof. Starting at a vertex u, consider a breadth first search process in T. At iteration i, all nodes at distance i from u have been marked and there can be at most $3 \cdot 2^{i-1}$ such nodes. Therefore,

$$L_{>\alpha \log n}(u) \geqslant n - \sum_{i=0}^{\alpha \log n} 3 \cdot 2^{i-1} \geqslant n - 3n^{\alpha} + \tfrac{3}{2} \geqslant \beta n$$

by the assumption that n is large enough. $\square$

Using a balanced tree, it is possible to obtain a constant approximation for the MinTC, MinTD, MinTL and MinTCL problems on mixing graphs:

Lemma 7.11. Let $\epsilon \in (0, \frac{1}{9})$, $\gamma \in (0,1)$. Consider a sequence $(c_n)_{n\in\mathbb{N}}$ such that $C_{\epsilon,\gamma} \leqslant c_n \leqslant n$ for all $n \geqslant n_0$ for some natural n_0. Let G be any (ϵ,γ,c_n)-mixing graph with n nodes where n is large enough. Let T_b be a balanced tree layout of G. Then,

$$\begin{aligned}
\textsc{tc}(T_b,G)/\textsc{mintc}(G) &\leqslant 2(1-\gamma)\epsilon^2/\,(1+\gamma)\,,\\
\textsc{td}(T_b,G)/\textsc{mintd}(G) &\leqslant (1-\gamma)^2/\,(1+\gamma)\,,
\end{aligned}$$

$$\text{TL}(T_b, G)/\text{MINTL}(G) \leqslant (1-\gamma)^3 2\epsilon^2/(1+\gamma)^2,$$
$$\text{TCL}(T_b, G)/\text{MINTCL}(G) \leqslant 3/2(1-7\epsilon^2).$$

Proof. To prove this result, we present lower and upper bounds to each of the considered problems. The lower bounds hold for any tree layout, while the upper bounds are obtained through a balanced tree layout.

Lower bound for MINTC(G)*:* Consider any tree layout T of G. Let uv be a $n\sqrt{\epsilon}$-splitter edge of T that separates T into two binary trees T_u and T_v rooted at u and v respectively. Such an edge must exist by Lemma 7.3. Let $\alpha, \beta \in (0,1)$ be two parameters to be determined latter. By Lemma 7.10, there exists a set of leaves L_u of T_u such that for all $x \in L_u$, $d_{T_u}(x,u) \geqslant \alpha \log(n\sqrt{\epsilon})$ and $|L_u| \geqslant \beta n\sqrt{\epsilon}$. Also, there exists a set of leaves L_v of T_v such that for all $y \in L_v$, $d_{T_v}(y,v) \geqslant \alpha \log(n\sqrt{\epsilon})$ and $|L_v| \geqslant \beta n\sqrt{\epsilon}$. Setting $\beta = \sqrt{\epsilon}$, we have $|L_u| \geqslant \epsilon n$ and $|L_v| \geqslant \epsilon n$. As G is (ϵ, γ, c_n)-mixing, we have $\theta(L_u, L_v) \geqslant (1-\gamma)|L_u||L_v|c_n/n \geqslant (1-\gamma)\epsilon^2 n c_n$. Thus, $\theta(uv, T, G) \geqslant (1-\gamma)\epsilon^2 n c_n$. So, $\text{TC}(T,G) \geqslant (1-\gamma)\epsilon^2 n c_n$ and as T is arbitrary we get $\text{MINTC}(G) \geqslant (1-\gamma)\epsilon^2 n c_n$.

Lower bounds for MINTD(G) *and* MINTL(G)*:* Observe that for all $x \in L_u$ and all $y \in L_v$, $d_T(x,y) \geqslant 2\alpha \log(n\sqrt{\epsilon}) + 1$. Setting $\alpha = 1-\gamma$, we have

$$\text{TD}(T,G) \geqslant 2\alpha \log(n\sqrt{\epsilon}) + 1 \geqslant (1-\gamma)^2 2 \log n$$

and

$$\text{TL}(T,G) \geqslant (1-\gamma)\epsilon^2 n c_n (2\alpha \log(n\sqrt{\epsilon}) + 1) \geqslant (1-\gamma)^3 2\epsilon^2 c_n n \log n.$$

As T is arbitrary, $\text{MINTD}(G) \geqslant (1-\gamma)^2 2 \log n$ and $\text{MINTL}(G) \geqslant 2(1-\gamma)^3 \epsilon^2 c_n n \log n$.

Lower bound for MINTCL(G)*:* Let us say that a graph G with n nodes satisfies the *dispersion* property if, for any two disjoint subsets A and B of $V(G)$ with $|A| \geqslant \epsilon n$ and $|B| \geqslant \epsilon n$, it is the case that there is at least one edge between A and B. From Definition 3.3 we get $\theta(A,B) \geqslant (1-\gamma)\epsilon^2 n^2$, which implies $\theta(A,B) \geqslant 1$ for n large enough. Therefore mixing graphs satisfy the dispersion property.

Let xy be a $\lfloor \frac{1}{3} n \rfloor$-splitter edge of T separating T into two binary trees T_x and T_y rooted at x and y respectively. Let L_x and L_y denote the leaves of T_x and T_y respectively. For n large enough, $|L_x| \geqslant (1-\epsilon)\frac{1}{3}n$ and $|L_y| \geqslant (1-\epsilon)\frac{1}{3}n$. Let L_x^1 be a subset of size $\lceil \epsilon n \rceil$ of L_x and let L_y^1 be a subset of the same size of L_y. Because of dispersion, there must be at least one edge in $E(G)$ connecting a node from L_x^1 to a node in L_y^1. Let $u_x^1 u_y^1$ be such edge and let v_x^1 be a node in $L_x \setminus L_x^1$ and let v_y^1 be a node in $L_y \setminus L_y^1$

Now we will construct recursively two sequences of sets L_x^i and L_y^i for $1 < i \leqslant (1-\epsilon)\frac{1}{3}n - (1+\epsilon)\epsilon n$: Let $L_x^i = (L_x^{i-1} \setminus \{u_x^{i-1}\}) \cup \{v_x^{i-1}\}$, and let $L_y^i = (L_y^{i-1} \setminus \{u_y^{i-1}\}) \cup \{v_y^{i-1}\}$.

As, the two sets have size $\lceil \epsilon n \rceil$, by dispersion, there must be at least one edge in $E(G)$ connecting a node from L_x^i to a node in L_y^i. Call $u_x^i u_y^i$ the endpoints of such an edge.

Let v_x^i be a node in $L_x \setminus (L_x^i \cup \{u_x^j : 1 \leqslant j \leqslant i\})$ and similarly let v_y^i be a node in $L_y \setminus (L_y^i \cup \{u_y^j : 1 \leqslant j \leqslant i\})$, notice that such nodes must exist.

By construction, all nodes in $\{u_x^i : 1 \leqslant i \leqslant (1-\epsilon)\frac{1}{3}n - (1+\epsilon)\epsilon n\}$ are connected in G to some node in L_y and, likewise, all nodes in $\{u_y^i : 1 \leqslant i \leqslant (1-\epsilon)\frac{1}{3}n - (1+\epsilon)\epsilon n\}$ are connected in G to some node in L_x. Therefore,

$$\textsc{tcl}(T,G) \geqslant 2 \cdot ((1-\epsilon)\tfrac{1}{3}n - (1+\epsilon)\epsilon n) \geqslant (1-7\epsilon^2)\tfrac{2}{3}n.$$

As T is arbitrary, we have $\textsc{mintcl}(G) \geqslant (1-7\epsilon^2)\frac{2}{3}n$.

Upper bounds: Let m denote the number of edges of the graph G. Using Lemma 7.1, we have $\textsc{tcl}(T_b,G) \leqslant n$. Moreover, as G is mixing, we also obtain $\textsc{tc}(T_b,G) \leqslant m \leqslant (1+\gamma)\frac{1}{2}nc_n$. As T_b is a balanced tree of G, its height is at most $\lceil \log n \rceil \leqslant (1+\gamma)\log n$. Therefore, we have $\textsc{td}(T_b,G) \leqslant 2(1+\gamma)\log n$ and $\textsc{tl}(T_b,G) \leqslant 2m(1+\gamma)\log n \leqslant (1+\gamma)^2 nc_n \log n$. □

As a consequence of Lemmas 3.2, 7.11 and 7.2, we get our main result on the approximability of tree layout problems on binomial random graphs:

Theorem 7.4. Let $\epsilon \in (0, \frac{1}{9})$, $\gamma \in (0,1)$ and define $C_{\epsilon,\gamma} = 3(1+\ln 3)(\epsilon\gamma)^{-2}$. Consider a sequence $(c_n)_{n\in\mathbb{N}}$ such that $C_{\epsilon,\gamma} \leqslant c_n \leqslant n$ for all $n \geqslant n_0$ for some natural n_0 and let $p_n = c_n/n$. Then, with overwhelming probability, the problems MinTD, MinTC, MinTVC, MinTCL and MinTL can be approximated within a constant on binomial random graphs $\mathcal{G}_{n,p_n}$ using a balanced tree layout. Moreover, in the case of the MinTD, the approximation factor can be made as small as desired.

7.6 Square grid graphs

In this section we study tree layout problems on square grid graphs. This is intended as an intermediate step to treat random geometric graphs on the next section. Recall from Section 4.1 that L_m denotes a $m \times m$ square grid with $V(L_m) = \{0, \dots, m-1\}^2$ and $E(L_m) = \{uv : u,v \in V(L_m) \wedge \|u-v\|_2 = 1\}$.

Let (A,B) be a partition of $V(L_m)$. Let $\theta(A,B)$ denote the number of edges between A and B in L_m. The following proposition will be of help. It is similar to Proposition 5.1, but without diagonal edges.

Proposition 7.1. For any partition (A,B) of $V(L_m)$ it holds that

$$\theta(A,B) \geqslant \min\left\{\sqrt{|A|}, \sqrt{|B|}\right\}.$$

Proof. If A includes an entire row of nodes, and B includes an entire row of nodes, then each column includes an edge with one endpoint in A and the other in B, which contributes 1 to $\theta(A,B)$, so that $\theta(A,B) \geqslant m$. If B contains no entire row or column, and at least as many rows as columns have non-empty intersection with B, then there are at least $\sqrt{B}$ such rows, and each contains a cutting edge which contributes 1 to $\theta(A,B)$, so that $\theta(A,B) \geqslant \sqrt{B}$. Applying similar arguments to the other possible cases, we have

$$\theta(A,B) \geqslant \min\left\{\sqrt{|A|},\ \sqrt{|B|},\ m\right\}$$

but this minimum is always achieved at $\sqrt{|A|}$ or at $\sqrt{|B|}$, proving the result. □

The next lemma presents lower bounds of the costs of several tree layout problems on square grids. Notice that despite the fact that MinTC belongs to **P** for planar graphs, the exact value of $\textsc{mintc}(L_m)$ is unknown.

Lemma 7.12. Let m be a sufficiently large natural. Then,

$$\begin{aligned}
\textsc{mintc}(L_m) &\geqslant \tfrac{1}{2}m,\\
\textsc{mintd}(L_m) &\geqslant \log m,\\
\textsc{mintcl}(L_m) &\geqslant \tfrac{\sqrt{6}}{3}m,\\
\textsc{mintl}(L_m) &\geqslant 6m^2 - 8m + 1.
\end{aligned}$$

Proof. Let T be any tree layout of L_m. Let uv be a $\left\lfloor \frac{1}{3}m^2 \right\rfloor$-splitter edge of T. As uv determines a partition (A,B) of L_m with $|A|,|B| \geqslant \left\lfloor \frac{1}{3}m^2 \right\rfloor$, by Proposition 7.1 the congestion of edge uv is at least equal to $\min\{\sqrt{|A|},\ \sqrt{|B|}\} \geqslant \sqrt{\lfloor m^2/3 \rfloor}$. Therefore, $\textsc{tc}(T, L_m) \geqslant \sqrt{\lfloor m^2/3 \rfloor} \geqslant \sqrt{m^2/4} = \frac{1}{2}m$. As T is arbitrary, the $\textsc{mintc}$ result follows.

Recall from Lemma 4.2 that for any linear layout φ on L_m and any $k \in [m^2]$, it is the case that $\delta(k,\varphi,L_m) \geqslant \delta(k,\varphi_D,L_m)$, where ϕ_D stands for the diagonal layout of L_m. Set

$$q_m = \left\lfloor \tfrac{1}{6}\sqrt{9+24m^2} - \tfrac{1}{2} \right\rfloor .$$

Then,

$$\sum_{i=1}^{q_m} i \leqslant \left\lfloor m^2/3 \right\rfloor .$$

So, $\delta(\left\lfloor \frac{1}{3}m^2 \right\rfloor, \varphi, L_m) \geqslant q_m$ (recall Figure 4.11).

To prove the $\textsc{mintd}$ and the $\textsc{mintcl}$ lower bounds, let T be any tree layout of L_m and let uv be a $\left\lfloor \frac{1}{3}m^2 \right\rfloor$-splitter edge of T. As there are at least q_m

leaves from one subtree connected in L_m to at least one other leave in the other subtree, we have $\text{TD}(T, L_m) \geqslant \log q_m + 2$ and $\text{TCL}(T, G) \geqslant \delta(\frac{1}{3}m^2, \varphi, L_m)$. As T is arbitrary, an for m sufficiently large, $\text{MINTD}(L_m) \geqslant \log q_m + 2 \geqslant \log m$ and $\text{MINTCL}(L_m) \geqslant \frac{\sqrt{6}}{3}m$.

To prove the MINTL result, let $G = (V, E)$ be any graph. Observe that in any tree layout of G no edge can have length 0 or 1. Also, observe that, at most, only $\frac{1}{2}|V|$ edges can have length 2, which happens in a balanced tree. Furthermore, at most $|V|-1$ edges can have length 3, which happens in a worm. Finally, observe that all edges that do not have length 2 or 3 must have, at least, length 4. In the case of L_m with $|V| = m^2$ nodes and $|E| = 2m^2 - 2m$ edges, we get

$$\begin{aligned} \text{MINTL}(L_m) &\geqslant 2(\tfrac{1}{2}m^2) + 3(m^2 - 1) + 4((2m^2 - 2m) - \tfrac{1}{2}m^2 - (m^2 - 1)) \\ &\geqslant 6m^2 - 8m + 1, \end{aligned}$$

which ends the proof. □

In order to get upper bounds, we shall use a recursive algorithm to produce a tree layout for L_m. We start describing the algorithm for the case that m is a power of two.

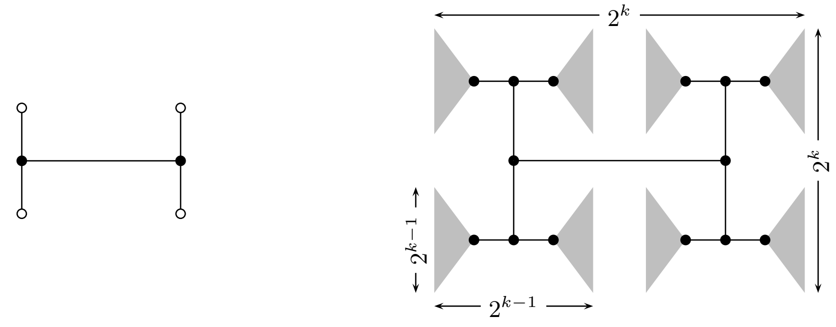
Algorithm 7.1 (The recursive algorithm). Let $m = 2^k$ for some integer $k \in \mathbb{N}$. The *recursive algorithm* generates a tree layout of L_m according to the following two rules:

- If $k = 1$: form a tree layout by joining the four nodes of L_m as shown in Figure 7.6(a).
- If $k > 1$: divide L_m in four $L_{m/2}$ sub-grids (top/left, bottom/left, top/right and bottom/left); recursively create a tree layout for each one of the sub-grids; join the four tree layouts in one tree layout as shown in Figure 7.6(b).

This construction generates an H-tree. Figure 7.7 illustrates the tree layout and problem costs produced by the recursive algorithm on L_2, L_4 and L_8. Observe that the recursive algorithm generates balanced tree layouts and produces a (2^{k-1})-splitting edge, which we call the *top* edge. We will also assume that the so obtained tree layout is rooted at the top edge. The following lemma states the costs computed by the recursive algorithm.

Lemma 7.13. Assume $k \in \mathbb{N}$. Let T_{2^k} be the tree layout of L_{2^k} computed by the recursive algorithm. Then,

$$\begin{aligned} &\text{TC}(T_{2^k}, L_{2^k}) = 2^k, \\ &\text{TD}(T_{2^k}, L_{2^k}) = 4k - 1, \end{aligned}$$



(a) Base case: $k = 1$ **(b)** Recursive case

Figure 7.6: Recursive algorithm to build a tree layout for a L_{2^k} square grid.

$$\textsc{tcl}(T_{2^k}, L_{2^k}) = 2^{k+1},$$
$$\textsc{tl}(T_{2^k}, L_{2^k}) = 14 \cdot 4^k - 8 \cdot 2^k k - 15 \cdot 2^k.$$

Proof. The proof for TC and TCL is obtained showing by induction on k that the maximal congestion and the maximal separation are reached at the top edge and the edges that are incident to its endpoints.

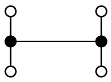
In order to compute $\textsc{td}(T_{2^k}, L_{2^k})$, observe that T_{2^k} is made of four tree layouts $T_{2^{k-1}}$ for which the distance from any leaf to their respective top edge is $2k - 3$. But to construct T_{2^k} one node is inserted in these top edges and to connect a left sub-grid with a right sub-mesh three edges are added. So,

$$\textsc{td}(T_{2^k}, L_{2^k}) = 2((2k - 3) + 1) + 3 = 4k - 1.$$

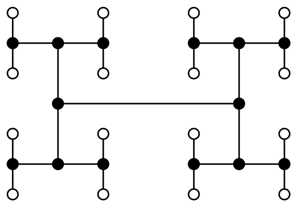
In the following, let $f(k) = \textsc{tl}(T, L_{2^k})$; note that $f(1) = 10$. For $k \geqslant 2$, in order to compute $f(k)$ observe that T_{2^k} is made of four tree layouts $T_{2^{k-1}}$, each one containing $2^{k-1} \cdot 2^{k-1}$ nodes and whose height from the top edge is $2k - 3$. We obtain the following recurrence:

$$\begin{cases} f(k) = 4f(k-1) + 4 \cdot 2^{k-1} + 2 \cdot 2^{k-1}(2(2k-3) + 4) + \\ \qquad\qquad + 2 \cdot 2^{k-1}(2(2k-3) + 5), \\ f(1) = 10. \end{cases}$$

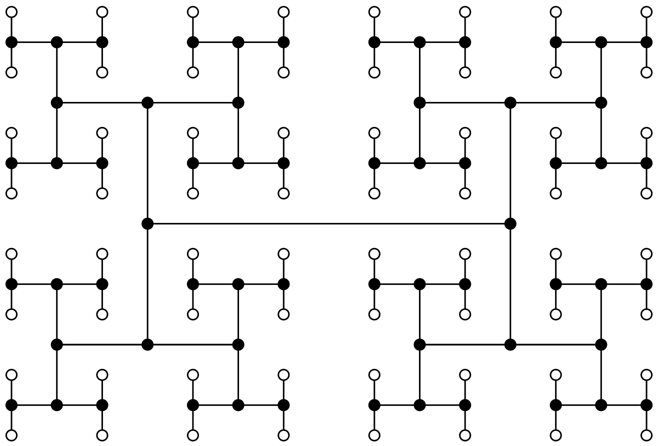
The first term of $f(k)$ comes from the cost of the four recursive tree layouts; the second term comes from the lengthening of the four recursive tree layouts due to the addition of a new node on its top edge; the third term comes from the cost of the length of the horizontal edges between the two top trees and the two



(a) L_2: TC $= 2$, TD $= 3$, TL $= 10$, TCL $= 4$.



(b) L_4: TC $= 4$, TD $= 7$, TL $= 100$, TCL $= 8$.



(c) L_8: TC $= 8$, TD $= 11$, TL $= 584$, TCL $= 16$.

Figure 7.7: Illustration of tree layouts computed by the recursive algorithm.

bottom trees; the fourth term comes from the cost of the length of the vertical edges between the two left trees and the two right trees.

The resolution of the recurrence yields the result. □

We now generalize the recursive algorithm to handle general square grids, when their side is not a power of two.

Algorithm 7.2 (The generalized recursive algorithm). Let $m \in \mathbb{N}$. Let k be the integer such that $m \leqslant 2^k < 2m$ and let T_{2^k} be the tree computed by the recursive algorithm on L_{2^k} rooted at the top edge. The *generalized recursive algorithm* generates a tree layout T_m of L_m applying iteratively the following transformation for all node $u \in V(L_{2^k}) \setminus V(L_m)$:

- let p_1 be the parent of u, let v be the sibling of u and let p_2 be the parent of p_1;
- remove the nodes u and p_1 from T together with its three incident edges;
- add the edge $p_2 v$ to T

Figure 7.8 shows the above transformation. Figure 7.9 illustrates the application of the generalized recursive algorithm to a 3×3 square grid.

The following theorem states that the generalized recursive algorithm is a constant approximation algorithm for the MinTD, MinTC, MinTVC, MinTCL and MinTL tree layout problems on square grids:

Theorem 7.5. Let m be a sufficiently large natural; let L_m be a $m \times m$ square grid and let T_m be its tree layout computed by the generalized recursive algorithm. Then,

$$\begin{aligned}
\textsc{td}(T_m, L_m)/\textsc{mintd}(L_m) &< 4,\\
\textsc{tc}(T_m, L_m)/\textsc{mintc}(L_m) &< 4,\\
\textsc{tvc}(T_m, L_m)/\textsc{mintvc}(L_m) &< 12,\\
\textsc{tcl}(T_m, L_m)/\textsc{mintcl}(L_m) &< \sqrt{6},\\
\textsc{tl}(T_m, L_m)/\textsc{mintl}(L_m) &< 10.
\end{aligned}$$

Proof. Let k be the natural such that $m \leqslant 2^k < 2m$, and let T_{2^k} be the tree computed by the recursive algorithm on L_{2^k}. Observe that the iterative deletion of a leaf by the generalized recursive algorithm cannot increase the congestion or separation at an edge of the tree layout. Also, the iterative deletion of a leave by the generalized recursive algorithm cannot increase the length of a graph edge in the tree layout. Therefore, using Lemmas 7.13 and 7.2, we get

$$\textsc{td}(T_m, L_m) \leqslant \textsc{td}(T_{2^k}, L_{2^k}) \leqslant 4k - 1 < 4 \log m - 3,$$

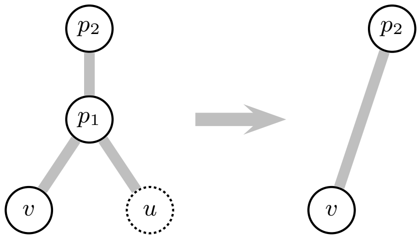


Figure 7.8: Deleting a leaf with the generalized recursive algorithm.

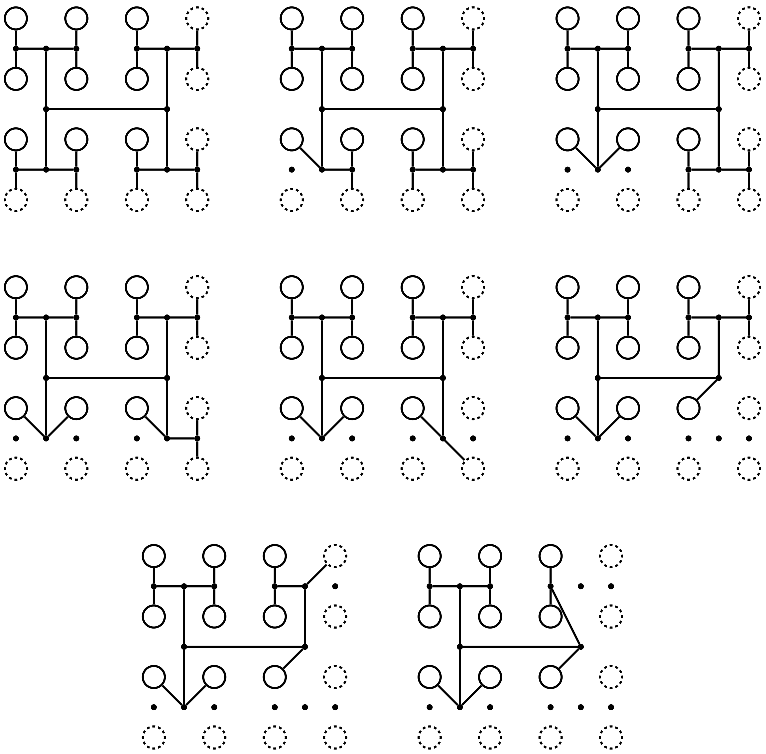


Figure 7.9: Application of the generalized recursive algorithm to a 3×3 square grid.

$$\begin{aligned}
\text{TC}(T_m, L_m) &\leqslant \text{TC}(T_{2^k}, L_{2^k}) \leqslant 2^k < 2m, \\
\text{TVC}(T_m, L_m) &\leqslant 3 \cdot \text{TC}(T_{2^k}, L_{2^k}) \leqslant 3 \cdot 2^k < 6m, \\
\text{TCL}(T_m, L_m) &\leqslant \text{TCL}(T_{2^k}, L_{2^k}) \leqslant 2^k < 2m, \\
\text{TL}(T_m, L_m) &\leqslant \text{TL}(T_{2^k}, L_{2^k}) \leqslant 14{\cdot}4^k - 8{\cdot}2^k{\cdot}k - 15{\cdot}2^k \leqslant 14{\cdot}4^k \leqslant 56m^2.
\end{aligned}$$

The statement of the theorem follows from these upper bounds, together with the lower bounds of Lemma 7.12. □

7.7 Random geometric graphs

In this section, we are concerned with the approximability of the MinTD, MinTC, MinTVC, MinTCL and MinTL tree layout problems on random geometric graphs.

Recall from Definition 5.3 that, given a set V of points in the unit square and a positive real r, the *geometric graph* $\mathcal{G}(V;r)$ is the graph $G = (V, E)$ where $E = \{uv \;:\; u, v \in V \wedge 0 < \|u - v\| \leqslant r\}$. All through this chapter, $\|\cdot\|$ will denote the l_∞ norm. Let $(r_n)_{n\in\mathbb{N}}$ be a sequence of positive numbers and let $X = (X_n)_{n\in\mathbb{N}}$ be a sequence of independently and uniformly distributed random points in $[0,1]^2$. According to Definition 5.4, for any $n \in \mathbb{N}$, we write $\mathcal{X}_n = \{X_1, \ldots, X_n\}$ and call $\mathcal{G}(\mathcal{X}_n; r_n)$ the *random geometric graph* of n vertices on X with radius r_n. As we did in Section 5.3 and in Chapter 6, in the remaining of this section we shall restrict our attention to almost surely connected random geometric graphs, that is, random geometric graphs whose radius is of the form

$$r_n = \sqrt{\frac{a_n}{n}} \quad \text{where} \quad r_n \to 0 \quad \text{and} \quad a_n/\log n \to \infty.$$

The rationale behind this choice was explained in Section 5.3.

An easy adaptation of the proofs of Lemmas 5.9 and 5.10 suffices to prove the following lower bounds for some tree layout problems on random geometric graphs. Recall that $(\text{MINTD}(\mathcal{G}(\mathcal{X}_n; r_n)))_{n\in\mathbb{N}}$ and alike are sequences of random variables.

Lemma 7.14. Let $(r_n)_{n\in\mathbb{N}}$ be a sequence of positive numbers with $r_n \to 0$ and $nr_n^2/\log n \to \infty$; let $(X_n)_{n\in\mathbb{N}}$ be a sequence of independently and uniformly distributed random points in $[0,1]^2$. Then, with probability 1,

$$\begin{aligned}
\text{MINTD}(\mathcal{G}(\mathcal{X}_n; r_n)) &= \Omega(\log n), \\
\text{MINTC}(\mathcal{G}(\mathcal{X}_n; r_n)) &= \Omega(n^2 r_n^3), \\
\text{MINTCL}(\mathcal{G}(\mathcal{X}_n; r_n)) &= \Omega(n r_n), \\
\text{MINTL}(\mathcal{G}(\mathcal{X}_n; r_n)) &= \Omega(n^2 r_n^2 \log n).
\end{aligned}$$

We introduce now a subclass of geometric graphs that captures the properties we need to bound our tree layout costs on random geometric graphs.

Definition 7.5 (Well behaved graphs). Consider any set V_n of n points in $[0,1]^2$, which together with a radius r_n, induce a geometric graph $G_n = \mathcal{G}(V_n; r_n)$. Dissect the unit square into $4\lfloor 1/r_n \rfloor^2$ boxes of size $1/2\lfloor 1/r_n \rfloor \times 1/2\lfloor 1/r_n \rfloor$ placed packed in $[0,1]^2$ starting at $(0,0)$. By construction, all the boxes exactly fit in the unit square, and any two points of V_n connected by an edge in G_n will be in the same or neighboring boxes (including diagonals) because $1/2\lfloor 1/r_n \rfloor \geqslant r_n/2$. Given $\epsilon \in (0,1)$, let us say that G_n is *ϵ-well behaved* if every box of this dissection contains at least $(1-\epsilon)\frac{1}{4}a_n$ points and at most $(1+\epsilon)\frac{1}{4}a_n$ points.

Notice that well behaved graphs are not exactly the same as nice graphs, the dissections of Definitions 7.5 and 5.5 are slightly different. But well behaved graphs share with nice graphs the fact that almost every random geometric graph is well behaved. The proof of the following lemma is similar to the proof of Lemma 5.3.

Lemma 7.15. Let $\epsilon \in (0, \frac{1}{5})$. Then, with probability 1, random geometric graphs $\mathcal{G}(\mathcal{X}_n; r_n)$ are ϵ-well behaved.

In order to handle geometric graphs, we present now a modification to the recursive algorithm defined in the previous section. This algorithm uses the same dissection as the definition of well behaved graphs and exploits the fact that the vertices inside a box form a clique, whose best tree layout is a balanced one (use Theorem 7.4 with $\mathcal{G}_{n,1}$). Figure 7.10 illustrates the algorithm.

Algorithm 7.3 (The boxed recursive algorithm). Let G_n be a well behaved graph with n vertices and radius r_n. Dissect the unit square into $4\lfloor 1/r_n \rfloor^2$ boxes of size $1/2\lfloor 1/r_n \rfloor \times 1/2\lfloor 1/r_n \rfloor$ placed packed in $[0,1]^2$ starting at $(0,0)$. The *boxed recursive algorithm* generates a tree layout T of G in the following way:

- All points in the same box are the leaves of a balanced tree layout.
- The generalized recursive tree layout is used to form a tree layout for all the graph, taking as its leaves a node that is inserted at the top edge of each of the balanced trees for each box.

The following lemma presents upper bounds on the cost of tree layout problems on well behaved graphs. These upper bounds are obtained with the boxed recursive algorithm and they match the order of growth of the lower bounds in Lemma 7.14.

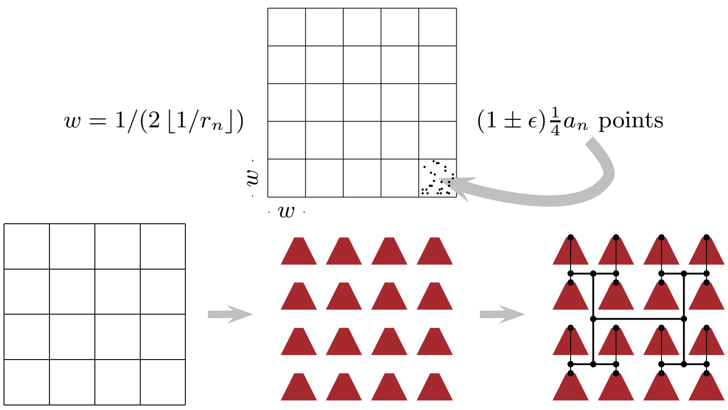


Figure 7.10: Illustration of tree layouts computed by the boxed recursive algorithm.

Lemma 7.16. Let $\epsilon \in (0, \frac{1}{5})$. For all $n \in \mathbb{N}$, let G_n be any ϵ-well behaved geometric graph with n vertices and radius r_n and let T_n be the tree layout produced by the boxed recursive algorithm for G_n. Then,

$$\begin{aligned} \text{TC}(T_n, G_n) &= O(n^2 r_n^3), \\ \text{TD}(T_n, G_n) &= O(\log n), \\ \text{TCL}(T_n, G_n) &= O(n r_n), \\ \text{TL}(T_n, G_n) &= O(n^2 r_n^2 \log n). \end{aligned}$$

Proof. As in the case of the square grid, the maximal congestion and separation are located at the top of T_n. In this place we have an edge which hosts the edges of two rows of $\sqrt{n/a_n}$ boxes, each with at most $(1+\epsilon)a_n$ points and connected to at most 3 neighbors. So, we have

$$\text{TC}(T_n, G_n) \leqslant 3 \cdot (1+\epsilon)a_n^2 \cdot \sqrt{n/a_n} = O\left(a_n \sqrt{a_n n}\right) = O(n^2 r_n^3)$$

and

$$\text{TCL}(T_n, G_n) \leqslant 2 \cdot \sqrt{n/a_n} = O(n r_n).$$

The diameter of the tree layout T obtained by the boxed recursive algorithm is upper bounded by $\lceil \log((1+\epsilon)a_n) \rceil + 1 + \lceil \log(4 \lfloor 1/r_n^2 \rfloor) \rceil = O(\log n)$. So, applying Lemma 7.1, we get that $\text{TD}(T_n, G_n) = O(\log n)$.

According to the boxed recursive algorithm, we can analyze the cost of the edges that appear at each level of the grid-like construction. At level 0, we consider all the edges that form a clique in each of the boxes. The total number of levels is

$$l = \sqrt{\log(4 \lfloor 1/r \rfloor^2)}.$$

Let us define h_i as the height of the subtree at level i. We have $h_0 = \log((1 + \epsilon)\frac{1}{4}a_n)$ and $h_{i+1} = h_i + 2$. Let t_i be the contribution of the edges taken into account in level i. We have $t_0 = ((1+\epsilon)\frac{1}{4}a_n)^2\, h_0\, 4 \lfloor 1/r \rfloor^2$ and $t_{i+1} = 48\; 2^i\, ((1 + \epsilon)\frac{1}{4}a_n)^2\, h_{i+1}\, 4 \lfloor 1/r \rfloor^2\; 4^{1-i}$. Using Maple to compute $\sum_{i=1}^{l} t_i$, we get the result. $\square$

The combination of Lemmas 7.14, 7.15, 7.16 and 7.2 leads to our main result on tree layouts for random geometric graphs:

Theorem 7.6. Let $(r_n)_{n\in\mathbb{N}}$ be a sequence of positive numbers with $r_n \to 0$ and $nr_n^2/\log n \to \infty$; let $(X_n)_{n\in\mathbb{N}}$ be a sequence of independently and uniformly distributed random points in $[0,1]^2$. Then, with probability 1, the problems MinTD, MinTC, MinTVC, MinTCL and MinTL can be approximated within a constant on random geometric graphs $\mathcal{G}(\mathcal{X}_n; r_n)$ using the boxed recursive algorithm.

7.8 Conclusion

In this chapter we have considered communication tree problems. These are a kind of problems that, appearing in some routing settings, extend the linear layout problems we have been considering in the previous chapters. Communication tree problems should not be confused with Steiner tree problems. General communication tree problems are difficult, and only a few results exist for them.

Our results show that the optimal costs of the edge congestion, vertex congestion, dilation, load and total communication problems on tree layouts, routing trees, and communication trees are related by a constant factor, provided that the maximal degree of the trees is bounded. Furthermore, we have proved that the optimal costs of the vertex congestion coincide for the three types of trees. Our results also show that selecting a random tree layout will not provide a good cost for the MinTD and MinTL problems on a given graph.

Using the same tools we introduced in Chapters 3 and 5, we have analyzed the minimal costs of the MinTC, MinTD, MinTL, MinTVC and MinTCL problems on binomial random graphs and random geometric graphs. We have also presented algorithms that, with high probability, find tree layouts whose cost is within a constant a factor of the optimal. Using Theorem 7.1, these approximation results on tree layouts can can be extended to communication trees and routing trees with bounded degree.

A preliminary version with the probabilistic results within Sections 7.4 to 7.7 has been presented at the workshop *Approximation and Randomized Algorithms in Communication Networks*—ARACNE2000 *(Geneva,* 2000*)* [6]. These results together with the ones in Section 7.3 and some considerations on complexity hardness have been sent to publication [7]. Our results can be extended in several ways: Results on square grids can be formulated in higher dimensions, and results on random geometric graphs can be analyzed for a different family of radii. We leave these as open problems.

A

Appendix

This appendix describes some of the notations used in this thesis and gives some background of probability theory.

A.1 Notation

Table A.1 lists several notations and abbreviations that might be unfamiliar to some readers.

If S is any Boolean statement, the parenthesized notation (S) stands for 1 if S, 0 if $\neg S$. Also, an expression of the form a/bc means the same as $a/(bc)$. These notations follow the ones of Graham, Knuth and Patashnik [108].

Given a number n, the notation $[n]$ denotes the set $\{1, \ldots, \lfloor n \rfloor\}$, when the context makes it clear it is not a reference.

The asymptotic notations $O(f(n))$, $\Omega(f(n))$, $\Theta(f(n))$, $o(f(n))$ and $\omega(f(n))$ are used as defined in the book of Cormen, Leiserson and Rivest [53].

Norms are defined by their associated distance functions in $\mathbb{R}^d$: Norm l_p is defined by the distance function $\|\cdot\|_p \colon \mathbb{R}^d \to \mathbb{R}$ where for any $p \in \mathbb{N}$,

$$\|x\|_p = \left(\sum_{i=1}^{d} |x_i|^p \right)^{1/p}.$$

The distance function for l_∞ is given by

$$\|x\|_\infty = \max_{i \in [d]} |x_i|.$$

Notation	Description
(B)	1 if B is true, 0 if B is false
$\mathbb{N}$	Set of natural numbers $\{1, 2, \dots\}$
$\mathbb{Z}$	Set of integer numbers $\{\dots, -1, 0, 1, \dots\}$
$\mathbb{Z}^+$	Set of positive integer numbers $\{0, 1, \dots\}$
$\mathbb{R}$	Set of real numbers
S^c	Complement of a set S
$\mathcal{P}(S)$	Power set of a set S
$\lvert S\rvert$	Cardinality of a set S
$[n]$	Set of the first n natural numbers $\{1, 2, \dots, n\}$
$\lvert x\rvert$	Absolute value of a number x
$\ln x$	Natural logarithm of x: $\log_e x$
$\log x$	Binary logarithm of x: $\log_2 x$
$\lfloor x \rfloor$	Floor of x: $\max\{n \in \mathbb{Z} \ : \ n \leqslant x\}$
$\lceil x \rceil$	Ceiling of x: $\min\{n \in \mathbb{Z} \ : \ n \geqslant x\}$
$\mathbf{Pr}\left[E\right]$	Probability of the event E
$\mathbf{E}\left[X\right]$	Expectation of the random variable X
$\mathbf{Var}\left[X\right]$	Variance of the random variable X
$X \xrightarrow{\mathrm{Pr}} Y$	Convergence in probability of X to Y
$X \xrightarrow{\mathrm{as}} Y$	Convergence almost surely of X to Y
a.a.	Almost always
i.o.	Infinitely often
w.h.p.	With high probability
w.o.p.	With overwhelming probability

Table A.1: Notation.

The l_1 norm is usually referred to as the *Manhattan norm*, the l_2 norm as the *Euclidean norm*, and the l_∞ norm as the *Maximum coordinate* or *Chebyshev* norms.

A.2 Background of probability theory

This section reviews some basic definitions and facts from probability theory that will be used in the body of this thesis. For a more complete presentation, refer to any standard reference [49, 88, 89, 110].

A.2.1 Basics

A *sample space* Ω is the set of possible outcomes of an experiment. Any element $\omega \in \Omega$ is called an *elementary event* and any subset $E \subseteq \Omega$ is called an *event*. In probability theory, it is customary to write ω instead of $\{\omega\}$. A *sigma field* is a nonempty set of subsets $\mathcal{F}$ of Ω satisfying:

1. $\emptyset \in \mathcal{F}$;
2. if $E \in \mathcal{F}$ then $E^c \in \mathcal{F}$; and
3. the countable union of elements in $\mathcal{F}$ also is in $\mathcal{F}$.

The event $\emptyset \in \mathcal{F}$ is called the *impossible event* and the event $\Omega \in \mathcal{F}$ is called the *sure event*. For a sequence of events $(E_n)_{n\in\mathbb{N}}$, define

$$\limsup E_n = \bigcap_{n\in\mathbb{N}} \bigcup_{k\geqslant n} E_k \quad \text{and} \quad \liminf E_n = \bigcup_{n\in\mathbb{N}} \bigcap_{k\geqslant n} E_k.$$

By the definition of sigma field, these sets are events: $\limsup E_n \in \mathcal{F}$ and $\liminf E_n \in \mathcal{F}$. It is said that $\limsup E_n$ is the event that E_n occurs *infinitely often* (written E_n **i.o.**), and that $\liminf E_n$ is the event that E_n occurs *almost always* (written E_n **a.a.**). By de Morgan's laws, $(E_n \textbf{ i.o.})^c = (E_n^c \textbf{ a.a.})$.

A *probability measure* $\mathbf{Pr}[\cdot]\colon \mathcal{F} \to \mathbb{R}$ is a function that satisfies the following conditions:

1. $\mathbf{Pr}[E] \geqslant 0$, for any $E \in \mathcal{F}$;
2. $\mathbf{Pr}[\Omega] = 1$; and
3. $\mathbf{Pr}[\cup_{n\in\mathbb{N}} E_n] = \sum_{n\in\mathbb{N}} \mathbf{Pr}[E_n]$ for any sequence $(E_n)_{n\in\mathbb{N}}$ of events in $\mathcal{F}$ such that $E_n \cap E_m = \emptyset$ for $n \neq m$.

A *probability space* is a triple $(\Omega, \mathcal{F}, \mathbf{Pr})$ where Ω is a sample space, $\mathcal{F}$ is a sigma field and $\mathbf{Pr}$ is a probability measure.

An event E is called *null* if $\mathbf{Pr}[E] = 0$ and is called *certain* if $\mathbf{Pr}[E] = 1$. Null events and the impossible event $\emptyset$ are different things, as the impossible event cannot happen but null events can. Also, certain events and the sure event Ω are different.

Boole's inequalities, also known as Bonferroni's inequalities, are useful to bound the probability that, given a set of events, at least one of them or all of them happen:

Theorem A.1 (Boole's inequalities). Let $(E_n)_{n\in\mathbb{N}}$ be a collection of events. Then,

$$\mathbf{Pr}\left[\bigcup_{n\in\mathbb{N}} E_n\right] \leqslant \sum_{n\in\mathbb{N}} \mathbf{Pr}\left[E_n\right] \quad \text{and} \quad \mathbf{Pr}\left[\bigcap_{n\in\mathbb{N}} E_n\right] \geqslant 1 - \sum_{n\in\mathbb{N}} \mathbf{Pr}\left[E_n^c\right].$$

From now on, assume that a probability space $(\Omega, \mathcal{F}, \mathbf{Pr})$ is fixed. A collection of events $(E_i)_{i=1}^n$ are *independent* if $\mathbf{Pr}\left[\cap_{i=1}^n E_i\right] = \prod_{i=1}^n \mathbf{Pr}\left[E_i\right]$.

A *random variable* is a function $X\colon \Omega \to \mathbb{R}$ such that $\{\omega \in \Omega \ :\ X(\omega) < x\} \in \mathcal{F}$ for any $x \in \mathbb{R}$. The *distribution function* of X is $F_X\colon \mathbb{R} \to [0,1]$ given by $F_X(x) = \mathbf{Pr}\left[X \leqslant x\right]$. A random variable is *discrete* if the set $X(\Omega)$ is discrete. Given a discrete random variable X, the function $f_X\colon \mathbb{R} \to [0,1]$ defined by $f_X(x) = \mathbf{Pr}\left[X = x\right]$ is called the *mass function* of X. A random variable X is *continuous* if $F_X(x) = \int_{-\infty}^{x} f_X(u)\mathrm{d}u$ for some integrable *density function* $f_X\colon \mathbb{R} \to [0,\infty)$.

A sequence of random variables $(X_i)_{i=1}^n$ are *independent* if, for all $x_1, \ldots, x_n$, the collection of events $(\{X_i \leqslant x_i\})_{i=1}^n$ are independent.

Given two random variables X and Y their *joint distribution* is defined by $f_{X,Y}\colon \mathbb{R}^2 \to [0,1]$ given by $f_{X,Y}(x,y) = \mathbf{Pr}\left[\{X = x\} \cap \{Y = y\}\right]$.

Given a discrete random variable X, its *expectation* is defined as

$$\mathbf{E}\left[X\right] = \sum_{x\in X(\Omega)} x\,\mathbf{Pr}\left[X = x\right],$$

For a continuous random variable Y, the expectation is

$$\mathbf{E}\left[X\right] = \int_{-\infty}^{+\infty} x f_X(x)\mathrm{d}x,$$

whenever this integral exists.

The *variance* of a random variable X is defined as

$$\mathbf{Var}\left[X\right] = \mathbf{E}\left[(X - \mathbf{E}\left[X\right])^2\right]$$

and its *standard deviation* as $\sigma_X = \sqrt{\mathbf{Var}\left[X\right]}$.

The following theorem presents some basic properties of expectations and variances, including the linearity of the expectation.

Theorem A.2. For any random variables X and Y and any $c \in \mathbb{R}$,

$$\begin{aligned}\mathbf{E}[cX+Y] &= c\mathbf{E}[X] + \mathbf{E}[Y],\\ \mathbf{Var}[X] &= \mathbf{E}\left[X^2\right] - \mathbf{E}[X]^2,\\ \mathbf{Var}[cX] &= c^2\,\mathbf{Var}[X].\end{aligned}$$

Moreover, if X and Y are independent,

$$\begin{aligned}\mathbf{E}[XY] &= \mathbf{E}[X]\,\mathbf{E}[Y],\\ \mathbf{Var}[X+Y] &= \mathbf{Var}[X] + \mathbf{Var}[Y].\end{aligned}$$

Let us now recall the definition and properties of some probability distributions. A random variable X has the *Bernoulli distribution* with parameter $p \in [0,1]$ if $\mathbf{Pr}[X=1] = p$ and $\mathbf{Pr}[X=0] = 1-p = q$. This distribution has expectation p and variance pq. When $(X_i)_{i=1}^n$ is a sequence of n independent Bernoulli random variables, the random variable $S_n = \sum_{i=1}^n X_i$ has the *binomial distribution* with parameters n and p. Its expectation is np and its variance npq. A *Poisson* variable with parameter $\lambda > 0$ is a random variable $\mathcal{P}(\lambda)$ whose mass function is $f(k) = \lambda^k e^{-\lambda}/k!$, for all $k \in \mathbb{N}$. Its expectation is λ.

In this thesis we consider the Poisson process on $\mathbb{R}^2$ with intensity λ, which formalizes the idea of scattering points at random in $\mathbb{R}^2$ so that the average number of points per unit volume is λ. To do so, each measurable subset B of $\mathbb{R}^2$ is associated with a random variable $X(B)$ such that

i) $X(B)$ has the Poisson distribution with parameter $\lambda|B|$, and
ii) if $B_1, \dots, B_n$ are disjoint, then $X(B_1), \dots, X(B_n)$ are independent and $X(B_1 \cup B_2) = X(B_1) + X(B_2)$.

A.2.2 Convergence

It is often necessary to state that a sequence of random variables converges to some limit or that the probabilities of a sequence of events tend to 1. There exist several ways to do this.

Let X be a random variable and $(X_n)_{n\in\mathbb{N}}$ a sequence of random variables. We say that X_n *converges in distribution to* X, written $X_n \xrightarrow{\mathrm{D}} X$, if

$$\lim_{n\to\infty} \mathbf{Pr}[X_n \leqslant x] = \mathbf{Pr}[X \leqslant x],$$

for all x where $\mathbf{Pr}[X \leqslant x]$ is continuous. X_n *converges in probability to* X, written $X_n \xrightarrow{\mathrm{Pr}} X$, if

$$\lim_{n\to\infty} \mathbf{Pr}[|X_n - X| > \epsilon] = 0, \qquad \forall \epsilon > 0.$$

Moreover, X_n *converges almost surely to* X, written $X_n \xrightarrow{\text{as}} X$, if

$$\mathbf{Pr}\left[\{\omega \in \Omega \ : \ X_n(\omega) \to X(\omega)\}\right] = 1.$$

This type of convergence is also called *convergence with probability 1*.

The following theorem states that convergence almost surely implies convergence in probability, which implies convergence in distribution:

Theorem A.3. Let X be a random variable and $(X_n)_{n\in\mathbb{N}}$ a sequence of random variables. If $X_n \xrightarrow{\text{as}} X$ then $X_n \xrightarrow{\text{Pr}} X$. Moreover, if $X_n \xrightarrow{\text{Pr}} X$ then $X_n \xrightarrow{\text{D}} X$.

The converse implications of the above theorem are false in general (see e.g. Lemma 7.10 of [110] for counterexamples). The following result is a way to prove almost surely convergence.

Theorem A.4. Let X be a random variable and $(X_n)_{n\in\mathbb{N}}$ a sequence of random variables. If, for all $\epsilon > 0$, $\sum_{n\in\mathbb{N}} \mathbf{Pr}\left[|X_n - X| > \epsilon\right] < \infty$, then $X_n \xrightarrow{\text{as}} X$.

A sequence of events $(E_n)_{n\in\mathbb{N}}$ occurs *with high probability*, written E_n **w.h.p.**, if

$$\lim_{n\to\infty} \mathbf{Pr}\left[E_n\right] = 1.$$

In the case $\mathbf{Pr}\left[E_n\right] \geqslant 1 - 2^{-cn}$ for some constant $c > 0$ and all large enough n, it is said that $(E_n)_{n\in\mathbb{N}}$ occurs *with overwhelming probability*, written E_n **w.o.p.**. Of course, with overwhelming probability is stronger than with high probability.

For all $n \in \mathbb{N}$ and all $w \in \Omega$, let $I_n(\omega) = \big(\omega \in E_n\big)$ be the indicator random variable of E_n. Using the above definitions, $I_n \xrightarrow{\text{Pr}} 1$ holds if and only if $(E_n)_{n\in\mathbb{N}}$ occurs with high probability. Moreover, using Theorem A.4, when $(E_n)_{n\in\mathbb{N}}$ occurs with overwhelming probability, then $I_n \xrightarrow{\text{as}} 1$.

The following result, know as the Borel–Cantelli Lemma, is similar to Theorem A.4, but for a sequence of events rather than a sequence of random variables.

Theorem A.5 (Borel–Cantelli Lemma). Let $(E_n)_{n\in\mathbb{N}}$ be a sequence of events. If $\sum_{n\in\mathbb{N}} \mathbf{Pr}\left[E_n\right] < \infty$ then $\mathbf{Pr}\left[E_n \ \mathbf{i.o.}\right] = 0$ and $\mathbf{Pr}\left[E_n^c \ \mathbf{a.a.}\right] = 1$.

As an example of the application of these concepts, consider a sequence of events $(E_n)_{n\in\mathbb{N}}$ where $\mathbf{Pr}\left[E_n\right] = 1/n^2$ for all n. It is the case that $\neg E_n$ holds with high probability, but $\neg E_n$ does not hold with overwhelming probability. On the other hand, as $\sum_{n\in\mathbb{N}} \mathbf{Pr}\left[E_n\right] = \sum_{n\in\mathbb{N}} 1/n^2 = \frac{1}{6}\pi^2$, using the Borel–Cantelli Lemma, we get $\mathbf{Pr}\left[E_n^c \ \mathbf{a.a.}\right] = 1$; that is, with probability 1, there exists a finite random N_0 such that, for all $n \geqslant N_0$, E_n does not hold.

A.2.3 Concentration bounds

Let X be a random variable and let x be a large positive real number. The functions $\mathbf{Pr}[X > x]$ and $\mathbf{Pr}[X < x]$ are called the upper and lower tails of X, respectively. Concentration bounds give upper bounds for the tails of the random variable $X - \mathbf{E}[X]$. A survey of some of the concentration bounds that are used in the field of combinatorics and theoretical computer science can be found in [69].

The following classical bound on the deviation from the expected value states a well known bound attributed to Markov.

Theorem A.6 (Markov inequality). Let X be a positive random variable. Then, for all $t > 0$, $\mathbf{Pr}[X > t\mathbf{E}[X]] \leqslant 1/t$.

Another classical bound, attributed to Chebyshev, relates $|X - \mathbf{E}[X]|$ with $\mathbf{Var}[X]$:

Theorem A.7 (Chebyshev inequality). Let X be a random variable. Then, for all $t > 0$, $\mathbf{Pr}\left[|X - \mathbf{E}[X]| > t\sqrt{\mathbf{Var}[X]}\right] \leqslant 1/t^2$.

The following result, due to Chernoff, gives bounds for the concentration around the expected value of a sum of independent Bernoulli variables.

Theorem A.8 (General Chernoff's bounds). Let $(X_i)_{i=1}^n$ be independent random variables corresponding to Bernoulli experiments, each variable X_i with probability p_i of success. Let $X = \sum_{i=1}^n X_i$ be a random variable with expected value $\mu = \sum_{i=1}^n p_i$. Then,

$$\mathbf{Pr}[X > (1+d)\mu] \leqslant \left(e^d/(1+d)^{1+d}\right)^{\mu}, \qquad \forall d > 0.$$

Moreover,

$$\mathbf{Pr}[X < (1-d)\mu] \leqslant \left(e^{-d}/(1-d)^{1-d}\right)^{\mu}, \qquad \forall d \in (0,1).$$

The previous theorem is the basis to develop customized Chernoff's bounds, which usually are weaker but easier to apply:

Theorem A.9 (Chernoff's bounds). Let $(X_i)_{i=1}^n$ be independent random variables corresponding to Bernoulli experiments, each variable with probability p_i of success. Let $X = \sum_{i=1}^n X_i$ be a random variable with expectation $\mu = \sum_{i=1}^n p_i$. Then, for any $d \in (0,1)$,

$$\mathbf{Pr}[X < (1-d)\mu] \leqslant \exp\left(-\tfrac{1}{2}d^2\mu\right),$$
$$\mathbf{Pr}[X > (1+d)\mu] \leqslant \exp\left(-\tfrac{1}{3}d^2\mu\right),$$
$$\mathbf{Pr}[|X-\mu| > d\mu] \leqslant 2\exp\left(-\tfrac{1}{3}d^2\mu\right).$$

The proofs of these concentration bounds can be found, for instance, in [69].

Bibliography

[1] E. Aarts and J. K. Lenstra, editors. *Local search in combinatorial optimization*. Wiley, New York, 1997.

[2] D. Adolphson. Single machine job sequencing with precedence constraints. *SIAM Journal on Computing*, 6:40–54, 1977.

[3] D. Adolphson and T. C. Hu. Optimal linear ordering. *SIAM Journal on Applied Mathematics*, 25(3):403–423, 1973.

[4] S. Agarwal, A. K. Mittal, and P. Sharma. Constrained optimum communications trees and sensitivity analysis. *SIAM Journal on Computing*, 13:315–328, 1984.

[5] N. Alon, J. H. Spencer, and P. Erdős. *The probabilistic method*. Wiley, New York, 1992.

[6] C. Àlvarez, R. Cases, J. Díaz, J. Petit, and M. Serna. Routing trees for random graphs. In J. Rolim, editor, *ICALP Workshops 2000*, volume 8 of *Proceedings in Informatics*, pages 99–110, Canada, 2000. Carleton Scientific.

[7] C. Àlvarez, J. Díaz, R. Cases, J. Petit, and M. Serna. Communication tree problems. Technical report, Universitat Politècnica de Catalunya, Departament de Llenguatges i Sistemes Informàtics, http://www.lsi.upc.es/∼jpetit, 2000.

[8] M. J. B. Appel and R. P. Russo. The connectivity of a graph on uniform points in $[0, 1]^d$. Technical report, University of Iowa, 1996.

[9] M. J. B. Appel and R. P. Russo. The maximum vertex degree of a graph on uniform points in $[0, 1]^d$. *Advances in Applied Probability*, 29(3):567–581, 1997.

[10] M. J. B. Appel and R. P. Russo. The minimum vertex degree of a graph on uniform points in $[0, 1]^d$. *Advances in Applied Probability*, 29(3):582–594, 1997.

[11] S. Arora, A. Frieze, and H. Kaplan. A new rounding procedure for the assignment problem with applications to dense graphs arrangements. In *37th Annual Symposium on Foundations of Computer Science*, pages 21–30. IEEE Computer Society Press, 1996.

[12] S. Arora, D. Karger, and M. Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. In *Proceedings of the 27th Annual ACM Symposium on the Theory of Computing*, pages 284–293, 1995.

[13] S. F. Assman, G. W. Peck, M. M. Syslo, and J. Zak. The bandwidth of caterpillars with hair of lengths 1 and 2. *SIAM Journal on Algebraic and Discrete Methods*, 2:387–393, 1981.

[14] J. E. Atkins, E. G. Boman, and B. Hendrickson. A spectral algorithm for seriation and the consecutive ones problem. *SIAM Journal on Computing*, 28(1):297–310, 1999.

[15] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and approximation*. Springer–Verlag, Berlin, 1999.

[16] F. Avram and D. Bertsimas. The minimum spanning tree constant in geometric probability and under the independent model: A unified approach. *The Annals of Applied Probability*, 2:113–130, 1992.

[17] R. Bar-Yehuda, G. Even, J. Feldman, and S. Naor. Computing an optimal orientation of a balanced decomposition tree for linear arrangement problems. Technical report, http://www.eng.tau.ac.il/~guy/Projects/Minla, 2001.

[18] A. D. Barbour, L. Holst, and S. Janson. *Poisson approximation*. Oxford University Press, New York, 1992.

[19] S. T. Barnard, A. Pothen, and H. Simon. A spectral algorithm for envelope reduction of sparse matrices. *Numerical Linear Algebra with Applications*, 2(4):317–334, 1995.

[20] S. T. Barnard and H. D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency: Practice and Experience*, 6:101–107, 1994.

[21] R. Battiti and A. Bertossi. Greedy, prohibition, and reactive heuristics for graph partitioning. *IEEE Transactions on Computers*, 2001. To appear. http://rtm.science.unitn.it/~battiti/archive/gp.ps.gz.

[22] J. Beardwood, J. Halton, and J. M. Hammersley. The shortest path through many points. *Mathematical Proceedings of the Cambridge Philosophy Society*, 55:299–327, 1959.

[23] J. W. Berry and M. K. Goldberg. Path optimization for graph partitioning problems. *Discrete Applied Mathematics*, 90:27–50, 1999.

[24] S. L. Bezrukov. Edge isoperimetric problems on graphs (a survey). In L. Lovasz, A. Gyarfas, G. O. H. Katona, A. Recski, and L. Szekely, editors, *Graph theory and combinatorial biology*, volume 7, pages 157–197. János Bolyai Math. Soc., Budapest, 1999.

[25] S. L. Bezrukov, R. Elsässer, B. Monien, R. Preis, and J.-P. Tillich. New spectral lower bounds on the bisection width of graphs. In U. Brandes and D. Wagner, editors, *Graph-Theoretic Concepts in Computer Science*, volume 1928 of *Lecture Notes in Computer Science*, pages 23–34, Berlin, 2000. Springer–Verlag.

[26] S. N. Bhatt and F. T. Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28:300–343, 1984.

[27] G. Blache, M. Karpinski, and J. Wirtgen. On approximation intractability of the bandwidth problem. Technical report TR98-014, Electronic Colloquium on Computational Complexity, 1998. http://cs.uni-bonn.de/~marek/publications/85182-CS.ps.Z.

[28] A. Blum, G. Konjevod, R. Ravi, and S. Vempala. Semi-definite relaxations for minimum-bandwith and other vertex-ordering problems. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing*, pages 284–293, 1998.

[29] H. Bodlaender, M. R. Fellows, and M. T. Hallet. Beyond NP-completeness for problems of bounded width: hardness for the W-hierarchy. In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing*, pages 449–458, 1994.

[30] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11(1-2):1–21, 1993.

[31] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[32] H. L. Bodlaender, T. Kloks, and D. Kratsch. Treewidth and pathwidth of permutation graphs. *SIAM Journal on Discrete Mathematics*, 8(4):606–616, 1995.

[33] H. L. Bodlaender and R. H. Möhring. The pathwidth and treewidth of cographs. *SIAM Journal on Discrete Mathematics*, 6(2):181–188, 1993.

[34] B. Bollobás. *Random graphs*. Academic Press, London, 1985.

[35] B. Bollobás. The isoperimetric number of random regular graphs. *European Journal of Combinatorics*, 9:241–244, 1988.

[36] B. Bollobás, T. I. Fenner, and A. M. Frieze. An algorithm for finding Hamilton paths and cycles in random graphs. *Combinatorica*, 7(4):327–341, 1987.

[37] B. Bollobás and I. Leader. Compressions and isoperimetric inequalities. *Journal of Combinatorial Theory Series A*, 56:47–62, 1991.

[38] R. Boppana. Eigenvalues and graph bisection: An average case analysis. In *28th Annual Symposium on Foundations of Computer Science*, pages 280–285. IEEE Computer Society Press, 1987.

[39] R. A. Botafogo. Cluster analysis for hypertext systems. In R. Korfhage, E. M. Rasmussen, and P. Willett, editors, *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 116–125. ACM, 1993.

[40] G. Brassard and P. Bratley. *Algorithmique. Conception et analyse*. Masson, Paris, 1987.

[41] H. Breu. *Algorithmic aspects of constrained unit disk graphs*. PhD thesis, University of British Columbia, 1996.

[42] H. Breu and D. G. Kirkpatrick. Unit disk graph recognition is NP-hard. *Computational Geometry. Theory and Applications*, 9(1-2):3–24, 1998.

[43] T. Bui, S. Chaudhuri, T. Leighton, and M. Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7:171–191, 1987.

[44] T. N. Bui and A. Peck. Partitioning planar graphs. *SIAM Journal on Computing*, 21(2):203–215, 1992.

[45] R. E. Burkard, S. E. Karisch, and F. Rendl. QAPLIB — A quadratic assignment problem library. *Journal of Global Optimization*, 391(10):391–403, 1997.

[46] P. Chinn, J. Chvátalová, A. Dewdney, and N. Gibbs. The bandwidth problem for graphs and matrices—A survey. *Journal of Graph Theory*, 6:223–254, 1982.

[47] S. Chirravuri, S. M. Bhandarkar, and J. Arnold. Parallel computing of physical maps—A comparative study in SIMD and MIMD parallelism. *Journal of Computational Biology*, 3(4):503–528, 1996.

[48] F. R. K. Chung. Labelings of graphs. In L. Beineke and R. Wilson, editors, *Selected topics in graph theory, 3*, pages 151–168. Academic Press, San Diego, 1988.

[49] K. L. Chung. *A course in probability theory*. Academic Press, New York, second edition, 1974.

[50] M. Chung, F. Makedon, I. H. Sudborough, and J. Turner. Polynomial time algorithms for the min cut problem on degree restricted trees. In *23th Annual Symposium on Foundations of Computer Science*, pages 262–271. IEEE Computer Society Press, 1982.

[51] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990.

[52] A. Condon and R. M. Karp. Algorithms for graph partitioning on the planted bisection model. *Random Structures & Algorithms*, 2001. To appear. http://www.cs.ubc.ca/∼condon/papers/karp99.ps.

[53] T. H. Cormen, C. Leiserson, and R. Rivest. *Introduction to algorithms*. The MIT Press, Cambridge, 1989.

[54] P. Crescenzi, R. Silvestri, and L. Trevisan. To weight or not to weight: where is the question? In *Israel Symposium on Theory of Computing and Systems*, pages 68–77. IEEE Computer Society Press, 1996.

[55] E. H. Cuthill and J. Mckee. Reducing the bandwidth of sparse symmetric matrices. In *24th ACM National Conference*, pages 157–172, 1969.

[56] F. Darema, S. Kirkpatrick, and V. A. Norton. Parallel algorithms for chip placement by simulated annealing. *IBM Journal of Research and Development*, 31:391–402, 1987.

[57] L. Davis. *Handbook of genetic algorithms*. Van Nostrand Reinhold, New York, 1991.

[58] N. Deo, M. S. Krishnamoorthy, and M. A. Langston. Exact and approximate solutions for the gate matrix layout problem. *IEEE Transactions on Computer Aided Design*, 6(1):79–84, 1987.

[59] J. Díaz. The δ-operator. In L. Budach, editor, *Fundamentals of Computation Theory*, pages 105–111. Akademie–Verlag, 1979.

[60] J. Díaz. Graph layout problems. In I. M. Havel and V. Koubek, editors, *Mathematical Foundations of Computer Science 1992*, volume 629 of *Lectures Notes in Computer Sciences*, pages 14–23, Berlin, 1992. Springer–Verlag.

[61] J. Díaz, A. Gibbons, G. Pantziou, M. Serna, P. Spirakis, and J. Torán. Parallel algorithms for the minimum cut and the minimum length tree layout problems. *Theoretical Computer Science*, 181(2):267–288, 1997.

[62] J. Díaz, A. M. Gibbons, M. S. Paterson, and J. Torán. The Minsumcut problem. In F. Dehen, R. J. Sack, and N. Santoro, editors, *Algorithms and Data Structures*, volume 519 of *Lecture Notes in Computer Science*, pages 65–79, Berlin, 1991. Springer–Verlag.

[63] J. Díaz, M. Penrose, J. Petit, and M. Serna. Linear ordering of random geometric graphs. In P. Wiedmayer and G. Neyer, editors, *Graph Theoretic Concepts in Computer Science*, volume 1665 of *Lecture Notes in Computer Science*, Berlin, 1999. Springer–Verlag.

[64] J. Díaz, M. D. Penrose, J. Petit, and M. Serna. Layout problems on lattice graphs. In T. Asano, H. Imai, D. T. Lee, S. Nakano, and T. Tokuyama, editors, *Computing and Combinatorics*, volume 1627 of *Lecture Notes in Computer Science*, pages 103–112, Berlin, 1999. Springer–Verlag.

[65] J. Díaz, M. D. Penrose, J. Petit, and M. Serna. Convergence theorems for some layout measures on random lattice and random geometric graphs. *Combinatorics, Probability and Computing*, 9(6):489–511, 2000.

[66] J. Díaz, M. D. Penrose, J. Petit, and M. Serna. Approximating layout problems on random geometric graphs. *Journal of Algorithms*, 39(1):78–116, 2001.

[67] J. Díaz, J. Petit, and M. Serna. Random geometric problems on $[0,1]^2$. In J. Rolim, M. Luby, and M. Serna, editors, *Randomization and Approximation Techniques in Computer Science*, volume 1518 of *Lecture Notes in Computer Science*, pages 294–306, Berlin, 1998. Springer–Verlag.

[68] J. Díaz, J. Petit, and M. Serna. Faulty random geometric networks. *Parallel Processing Letters*, 10(4):343–357, 2001.

[69] J. Díaz, J. Petit, and M. Serna. A guide to concentration bounds. In P. Pardalos and J. Rolim, editors, *Handbook on Randomized Computing*. Kluwer Press, New York, 2001. To Appear.

[70] J. Díaz, J. Petit, M. Serna, and L. Trevisan. Approximating layout problems on random graphs. *Discrete Mathematics*, 235(1–3):245–253, 2001.

[71] J. Díaz, M. Serna, P. Spirakis, and J. Torán. *Paradigms for fast parallel approximability*. Cambridge University Press, Cambridge, 1997.

[72] R. Dieckmann and B. Monien. A local graph partitioning heuristic meeting bisection bounds. In *8th SIAM Conference of Parallel Processing in Scientific Computing*, 1997.

[73] R. Diekmann, R. Lüling, and B. Monien. Communication throughput of interconnection networks. In I. Privara, B. Rovan, and P. Ruzicka, editors, *Mathematical Foundations of Computer Science 1994*, volume 841 of *Lecture Notes in Computer Science*, pages 72–86, Berlin, 1994. Springer–Verlag.

[74] R. Diekmann, R. Lüling, B. Monien, and C. Spräner. Combining helpful sets and parallel simulated annealing for the graph-partitioning problem. *Parallel Algorithms and Applications*, 8:61–84, 1996.

[75] R. Diekmann, R. Lüling, and J. Simon. A general purpose distributed implementation of simulated annealing. In *4th IEEE Symposium on Parallel and Distributed Processing*, pages 94–101. IEEE Computer Society Press, 1992.

[76] R. Diekmann, B. Monien, and R. Preis. Using helpful sets to improve graph bisections. In D. F. Hsu, A. L. Rosenberg, and D. Sotteau, editors, *Interconnection Networks and Mapping and Scheduling Parallel Computations*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pages 57–73. AMS, 1995.

[77] G. Ding and B. Oporowski. Some results on tree decomposition of graphs. *Journal of Graph Theory*, 20(4):481–499, 1995.

[78] R. G. Downey and M. R. Fellows. *Parameterized complexity.* Springer–Verlag, New York, 1999.

[79] N. Dunford and J. Schwartz. *Linear operators. Part I: general theory.* Interscience Publisher, New York, 1958.

[80] J. Ellis, I. H. Sudborough, and J. Turner. The vertex separation and search number of a graph. *Information and Computation*, 113:50–79, 1979.

[81] U. Elsner. Graph partitioning—A survey. Technical report 393, Technische Universität Chemnitz, http://www.tu-chemnitz.de/sfb393/Files/PS/sfb97-27.ps.gz, 1997.

[82] P. Erdős and A. Rényi. On random graphs—I. *Publicationes Matematicae*, 6:290–297, 1959.

[83] G. Even, J. Naor, S. Rao, and B. Schieber. Divide-and-conquer approximation algorithms via spreading metrics. In *36th Annual Symposium on Foundations of Computer science*, pages 62–71. IEEE Computer Society Press, 1995.

[84] S. Even and Y. Shiloach. NP-completeness of several arrangements problems. Technical Report TR-43, The Technion, Haifa, 1978.

[85] U. Feige. Approximating the bandwidth via volume respecting embeddings. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 23–26, 1998.

[86] U. Feige and R. Krauthgamer. Improved performance guarantees for bandwidth minimization heuristics (draft). Technical report, http://www.wisdom.weizmann.ac.il/~robi/papers, 1998.

[87] U. Feige, R. Krauthgamer, and K. Nissim. Approximating the minimum bisection size. In *Proceedings of the 32nd Annual ACM Symposium on the Theory of Computing*, pages 530–536, 2000.

[88] W. Feller. *An Introduction to Probability Theory and its Applications. Volume I.* Wiley, New York, 1957.

[89] W. Feller. *An Introduction to Probability Theory and its Applications. Volume II.* Wiley, New York, 1966.

[90] M. R. Fellows and M. A. Langston. Layout permutation problems and well-partially-ordered sets. In *Advanced research in VLSI*, pages 315–327. MIT Press, Cambridge, MA, 1988.

[91] M. R. Fellows and M. A. Langston. On well-partial-order theory and its application to combinatorial problems of VLSI design. *SIAM Journal on Discrete Mathematics*, 5(1):117–126, 1992.

[92] M. R. Fellows and M. A. Langston. On search, decision, and the efficiency of polynomial-time algorithms. *Journal of Computer and System Sciences*, 49(3):769–779, 1994.

[93] P. Fishburn, P. Tetali, and P. Winkler. Optimal linear arrangement of a rectangular grid. *Discrete Mathematics*, 213:123–139, 2000.

[94] L. R. Ford and D. R. Fulkerson. *Flows in networks*. Princeton University Press, 1962.

[95] A. Frieze and R. Kannan. The regularity lemma and approximation schemes for dense problems. In *37th Annual Symposium on Foundations of Computer Science*, pages 12–20. IEEE Computer Society Press, 1996.

[96] A. Frieze and C. McDiarmid. Algorithmic theory of random graphs. *Random Structures & Algorithms*, 10(1-2):5–42, 1997.

[97] M. R. Garey, R. L. Graham, D. S. Johnson, and D. Knuth. Complexity results for bandwidth minimization. *SIAM Journal on Applied Mathematics*, 34:477–495, 1978.

[98] M. R. Garey and D. S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness*. Freeman and Company, 1979.

[99] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[100] F. Gavril. Some NP-complete problems on graphs. In *11th Conference on Information Sciences and Systems*, pages 91–95, John Hopkins University, Baltimore, 1977.

[101] N. E. Gibbs, W. G. Poole, Jr., and P. K. Stockmeyer. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM Journal on Numerical Analysis*, 13(2):236–250, 1976.

[102] F. Glover and M. Laguna. *Tabu Search*. Kluwer, Boston, second edition, 1997.

[103] A. Goerdt and M. Molloy. Analysis of edge deletion processes on faulty random regular graphs. In G. H. Gonnet, D. Panario, and A. Viola, editors, *LATIN 2000: Theretical Informatics*, volume 1776 of *Lecture Notes in Computer Science*, Berlin, 2000. Springer–Verlag.

[104] M. K. Goldberg and I. A. Klipker. An algorithm for minimal numeration of tree vertices. *Sakharth. SSR Mecn. Akad. Moambe*, 81(3):553–556, 1976. In Russian.

[105] P. W. Goldberg, M. C. Golumbic, H. Kaplan, and R. Shamir. Four strikes against physical mapping od DNA. *Journal of Computational Biology*, 2(1):139–152, 1995.

[106] M. C. Golumbic. *Algorithmic graph theory and perfect graphs.* Academic Press, New York, 1980.

[107] R. E. Gomory and T. C. Hu. *Multi-terminal flows in a network*, volume 11 of *Studies in Mathematics*, pages 172–199. Mathematical Association of America, 1975.

[108] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete mathematics.* Addison–Wesley, Reading, second edition, 1994.

[109] G. R. Grimmett. *Percolation.* Springer–Verlag, Heidelberg, second edition, 1999.

[110] G. R. Grimmett and D. R. Stirzaker. *Probability and random processes.* Oxford University Press, New York, second edition, 1992.

[111] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI.* The MIT Press, Massachussets, 2 edition, 1995.

[112] E. Gurari and I. H. Sudborough. Improved dynamic programming algorithms for the bandwidth minimization and the mincut linear arrangement problem. *Journal of Algorithms*, 5:531–546, 1984.

[113] J. Gustedt. On the pathwidth of chordal graphs. *Discrete Applied Mathematics*, 45(3):233–248, 1993.

[114] M. D. Hansen. Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems. In *30th Annual Symposium on Foundations of Computer Science*, pages 604–609. IEEE Computer Society Press, 1989.

[115] J. Haralambides and F. Makedon. Approximation algorithms for the bandwidth minimization problem for a large class of trees. *Theory of Computing Systems*, 30:67–90, 1997.

[116] J. Haralambides, F. Makedon, and B. Monien. Bandwidth minimization: an approximation algorithm for cartepillars. *Mathematical Systems Theory*, (24):169–177, 1991.

[117] F. Harary. Problem 16. In M. Fiedler, editor, *Graph Theory and Computing*, page 161, Prague, 1967. Czechoslovak Academy Sciences.

[118] F. Harary and E. M. Palmer. *Graphical enumeration*. Academic Press, New York, 1973.

[119] L. H. Harper. Optimal assignments of numbers to vertices. *Journal of SIAM*, 12(1):131–135, 1964.

[120] L. H. Harper. Optimal numberings and isoperimetric problems on graphs. *Journal of Combinatorial Theory*, 1(3):385–393, 1966.

[121] L. H. Harper. Chassis layout and isoperimetric problems. Technical report SPS 37–66, vol II, Jet Propulsion Laboratory, 1970.

[122] L. H. Harper. Stabilization and the edgesum problem. *Ars Combinatoria*, 4:225–270, Dec. 1977.

[123] R. Heckmann, R. Klasing, B. Monien, and W. Unger. Optimal embedding of complete binary trees into lines and grids. In G. Schmidt and R. Berghammer, editors, *Graph-theoretic Concepts in Computer Science*, volume 570 of *Lectures Notes in Computer Sciences*, pages 25–35, Berlin, 1992. Springer–Verlag.

[124] C. Helmberg, F. Rendl, B. Mohar, and S. Poljak. A spectral approach to bandwidth and separator problems in graphs. *Linear and Multilinear Algebra*, 39(1-2):73–90, 1995.

[125] B. Hendrickson and R. Leland. Multidimensional spectral load balancing. *6th SIAM Conf. Parallel Proc. Sci. Comput.*, 1993.

[126] B. Hendrickson and R. Leland. The Chaco user's guide: version 2.0. Technical report SAND94–2692, Sandia National Laboratories, 1997. ftp://ftp.cs.sandia.gov/pub/papers/bahendr/guide.ps.gz.

[127] N. Hooker. Needed: An empirical science of algorithms. *Operations Research*, 42:201–212, 1994.

[128] N. Hooker. Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1:33–42, 1996.

[129] J. Hromkovič and B. Monien. The bisection problem for graphs of degree 4 (configuring transputer systems). In *Informatik*, pages 215–233. Teubner, Stuttgart, 1992.

[130] T. C. Hu. Optimum communication spanning trees. *SIAM Journal on Computing*, 3:188–195, 1974.

[131] H. B. Hunt, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. NC-approximation schemes for NP- and PSPACE- hard problems for geometric graphs. *Journal of Algorithms*, 26(2):238–274, 1998.

[132] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.

[133] S. Janson, T. Łuczak, and A. Rucinski. *Random graphs.* Wiley, New York, 2000.

[134] M. Jerrum and G. Sorkin. The Metropolis algorithm for graph bisection. *Discrete Applied Mathematics*, 82(1-3):155–175, 1998.

[135] D. S. Johnson. Local optimization and the traveling salesman problem. In M. S. Paterson, editor, *Automata, languages and programming*, volume 443 of *Lectures Notes in Computer Sciences*, pages 25–35, Berlin, 1990. Springer–Verlag.

[136] D. S. Johnson. A theoretician's guide to the experimental analysis of algorithms. http://www.research.att.com/~dsj/papers/exper.ps, 1996.

[137] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning. *Operations Research*, 37(6):865–892, 1989.

[138] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning. *Operations Research*, 39(3):378–405, 1991.

[139] D. S. Johnson, J. K. Lenstra, and A. H. G. R. Kan. The complexity of the network design problem. *Networks*, 8(4):279–285, 1978.

[140] D. S. Johnson, L. A. McGeoch, and E. E. Rothberg. Asymptotic experimental analysis for the Held–Karp traveling salesman bound. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 341–350. ACM/SIAM, 1996.

[141] A. Juels. *Topics in black-box combinatorial optimization*. PhD thesis, University of California at Berkeley, 1996.

[142] M. Juvan and B. Mohar. Optimal linear labelings and eigenvalues of graphs. *Discrete Applied Mathematics*, 36(2):153–168, 1992.

[143] P. Kadluczka and K. Wala. Tabu search and genetic algorithms for the generalized graph partitioning problem. *Control and cybernetics*, 24(4):459–476, 1995.

[144] D. R. Karger. A randomized fully polynomial approximation scheme for all terminal network realiability problem. In *Proceedings of the 27th Annual ACM Symposium on the Theory of Computing*, pages 11–17. ACM, 1996.

[145] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972. To Appear.

[146] R. M. Karp. Probabilistic analysis of partitioning algorithms for the travelling-salesman problem in the plane. *Mathematics of Operation Research*, 2:209–224, 1977.

[147] R. M. Karp. Mapping the genome: some combinatorial problems arising in molecular biology. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 278–285. ACM, 1993.

[148] R. M. Karp. Mapping the genome: some combinatorial problems arising in molecular biology. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 278–285, 1993.

[149] M. Karpinski, J. Wirtgen, and A. Zelikovsky. An approximating algorithm for the bandwidth problem on dense graphs. Technical report TR 97-017, Electronic Colloquium on Computational Complexity, http://cs.uni-bonn.de/ marek/publications/85164-CS.ps.Z, 1997.

[150] G. Karypis. Metis's home page. Web page, 2001. http://www-users.cs.umn.edu/~karypis/metis.

[151] D. G. Kendall. Incidence matrices, interval graphs, and seriation in archeology. *Pacific Journal of Mathematics*, 28:565–570, 1969.

[152] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, (49):291–307, 1970.

[153] H. Kesten. *Percolation theory for mathematicians*. Birkhäuser Boston, Massachussets, 1982.

[154] N. G. Kinnersley. The vertex separation number of a graph equals its path-width. *Information Processing Letters*, 42(6):345–350, 1992.

[155] S. Kirkpatrik, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[156] L. M. Kirousis and C. H. Papadimitriou. Searching and pebbling. *Theoretical Computer Science*, 47:205–216, 1986.

[157] R. Klasing. The relationship between the gossip complexity in vertex-disjoint paths mode and the vertex bisection width. *Discrete Applied Mathematics*, 83:229–246, 1998.

[158] K. Klohs. parSA library user manual (version 2.1a), 1999. http://www.uni-paderborn.de/cs/ag-monien/SOFTWARE/PARSA/PUBLICATIONS/ParSALibDoc.ps.

[159] T. Kloks, H. L. Bodlaender, J. R. Gilbert, and H. Hafsteinsson. Approximating treewidth, pathwidth, and minimum elimination tree height. *Journal of Algorithms*, 18:238–255, 1995.

[160] T. Kloks, D. Kratsch, and H. Müller. Bandwidth of chain graphs. *Information Processing Letters*, 68(6):313–315, 1998.

[161] D. E. Knuth. *The art of computer programming: fundamental algorithms*. Addison–Wesley, Reading, third edition, 1997.

[162] V. Kumar, A. Grama, A. Gupta, and G. Karypis, editors. *Introduction to parallel computing*. Benjamin Cummings, Redwood City, 1994.

[163] D. Kuo and G. J. Chang. The profile minimization problem in trees. *SIAM Journal on Computing*, 23:71–81, 1994.

[164] Y.-L. Lai and K. Williams. A survey of solved problems and applications on bandwidth, edgesum, and profile of graphs. *Journal of Graph Theory*, 31(2):75–94, 1999.

[165] K. Lang and S. Rao. Finding near-optimal cuts: An empirical evaluation. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 212–221. ACM/SIAM, 1993.

[166] M. Ledoux. Isoperimetry and Gaussian analysis. In P. Groeneboom, R. Dobrushin, and M. Ledoux, editors, *Lectures in Probability Theory*, volume 1648 of *Lectures Notes in Mathematics*, pages 165–294, Berlin, 1994. Springer–Verlag.

[167] F. T. Leighton. *Introduction to parallel algorithms and architectures: arrays, trees, hypercubes*. Morgan Kaufmann, San Mateo, 1993.

[168] F. T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problem with applications to approximation algorithms. In *29th Annual Symposium on Foundations of Computer Science*, pages 422–431. IEEE Computer Society Press, 1988.

[169] C. E. Leiserson. Area-efficient graph layouts (for VLSI). In *21st Annual Symposium on Foundations of Computer Science*, pages 270–281. IEEE Computer Society Press, 1980.

[170] R. Leland and B. Hendrickson. An empirical study of static load balancing algorithms. In *Scalable High-Performance Computing Conference*, pages 682–685. IEEE Computer Society Press, 1994.

[171] T. Lengauer. Black-white pebbles and graph separation. *Acta Informatica*, 16:465–475, 1981.

[172] T. Lengauer. Upper and lower bounds on the complexity of the min-cut linear arrangements problem on trees. *SIAM Journal on Algebraic and Discrete Methods*, 3:99–113, 1982.

[173] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36:177–189, 1979.

[174] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8:261–277, 1988.

[175] M. L. Luczak and C. McDiarmid. Bisecting sparse random graph. *Random Structures & Algorithms*, 18:31–38, 2001.

[176] F. Makedon, C. H. Papadimitriou, and I. H. Sudborough. Topological bandwidth. *SIAM Journal on Algebraic and Discrete Methods*, 6(3):418–444, 1985.

[177] F. Makedon and I. H. Sudborough. On minimizing width in linear layouts. *Discrete Applied Mathematics*, 23(3):243–265, 1989.

[178] C. C. McGeoch. Toward an experimental method for algorithm simulation. *INFORMS Journal on Computing*, 8(1):1–15, 1996.

[179] K. Mehlhorn and S. Näher. *LEDA — A platform for combinatorial and geometric computing*. Cambridge University Press, 1999.

[180] M. V. Menshikov, S. A. Molchanov, and A. F. Sidorenko. Percolation theory and some applications. *Journal of Soviet Mathematics*, 42:1766–1810, 1988.

[181] W. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[182] G. Mitchison and R. Durbin. Optimal numberings of an $n \times n$ array. *SIAM Journal on Algebraic and Discrete Methods*, 7(4):571–582, 1986.

[183] B. Mohar and S. Poljak. Eigenvalues in combinatorial optimization. In R. A. Brualdi, S. Friedland, and V. Klee, editors, *Combinatorial and Graph-Theoretical Problems in Linear Algebra*, volume 50 of *IMA Volumes in Mathematics and its Applications*, pages 107–151, Berlin, 1993. Springer–Verlag.

[184] B. Monien. The complexity of embedding graphs into binary trees. In L. Budach, editor, *Fundamentals of computation theory*, volume 199 of *Lecture Notes in Computer Science*, pages 300–309, Berlin, 1985. Springer–Verlag.

[185] B. Monien. The bandwidth minimization problem for caterpillars with hair length 3 is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 7(4):505–512, 1986.

[186] B. Monien and H. Sudborough. Embedding one interconnection network in another. In G. Tinhofer, E. Mayr, H. Noltemeier, and M. M. Syslo, editors, *Computational graph theory*, volume 7 of *Computing Supplementa*, pages 257–282, Berlin, 1990. Springer–Verlag.

[187] B. Monien and I. H. Sudborough. Min Cut is NP-complete for edge weighted trees. *Theoretical Computer Science*, 58(1-3):209–229, 1988.

[188] B. M. E. Moret. Towards a discipline of experimental algorithms. In C. C. McGeoch and D. S. Johnson, editors, *DIMACS*, DIMACS Monograph Series. AMS Press, 2001. To appear. http://www.cs.unm.edu/~moret/dimacs_algorithmics.ps.

[189] D. O. Muradyan and T. E. Piliposjan. Minimal numberings of vertices of a rectangular lattice. *Akad. Nauk. Armjan. SRR*, 1(70):21–27, 1980. In Russian.

[190] P. Mutzel. A polyedral approach to planar augmentation and related problems. In P. Spirakis, editor, *Algorithms — ESA'95*, volume 979 of *Lecture Notes in Computer Science*, pages 497–507, Berlin, 1995. Springer–Verlag.

[191] K. Nakano. Linear layouts of generalized hypercubes. In J. van Leeuwen, editor, *Graph-Theoretic Concepts in Computer Science*, volume

790 of *Lecture Notes in Computer Science*, pages 364–375, Berlin, 1994. Springer–Verlag.

[192] L. Niepel and P. Tomasta. Elevation of a graph. *Czechoslovak Mathematical Journal*, 31(106)(3):475–483, 1981.

[193] S. Nikoletseas, K. Palem, P. Spirakis, and M. Yung. Short vertex disjoint paths and multiconnectivity in random graphs: reliable network computing. In S. Abiteboul and E. Shamir, editors, *Automata, Languages and Programming*, volume 820 of *Lecture Notes in Computer Science*, pages 508–519, Berlin, 1994. Springer–Verlag.

[194] S. Nikoletseas and P. Spirakis. Efficient communication establishment in adverse communication environments. In J. Rolim, editor, *ICALP Workshops 2000*, volume 8 of *Proceedings in Informatics*, pages 215–226, Canada, 2000. Carleton Scientific.

[195] J. K. Ousterhout. *Tcl and the Tk Toolkit*. Adisson–Wesley, 1994.

[196] J. Pach, F. Shahrokhi, and M. Szegedy. Applications of the crossing number. *Algoritmica*, 16:111–117, 1996.

[197] C. Papadimitriou. The NP-completeness of the bandwidth minimization problem. *Computing*, 16:263–270, 1976.

[198] C. Papadimitriou. *Computational complexity*. Addison–Wesley, Reading, 1994.

[199] C. Papadimitriou and K. Steiglitz. *Combinatorial optimizations, algorithms and complexity*. Prentice–Hall, Englewood Cliffs, 1982.

[200] C. H. Papadimitriou. The probabilistic analysis of matching heuristics. In *15th Annual Conference on Communication Control Computing*, pages 368–378, 1978.

[201] C. H. Papadimitriou and M. Sideri. The bisection width of grid graphs. *Mathematical Systems Theory*, 29(2):97–110, 1996.

[202] B. Parlett, H. Simmon, and L. Stringer. Estimating the largest eigenvalue with the Lanczos algorithm. *Mathematics of Computation*, 38(157):153–165, 1982.

[203] M. Penrose. Single linkage clustering and continuum percolation. *Journal of Multivariate Analysis*, 53:94–109, 1995.

[204] M. Penrose. The longest edge of the random minimal spanning tree. *The Annals of Applied Probability*, 7:340–361, 1997.

[205] M. Penrose. Vertex ordering and partitioning problems for random spatial graphs. *The Annals of Applied Probability*, 10:517–538, 2000.

[206] M. D. Penrose. On k-connectivity for a geometric random graph. *Random Structures & Algorithms*, 15(2):145–164, 1999.

[207] J. Petit. Combinning spectral sequencing with simulated annealing for the MinLA problem: sequential and parallel heuristics. Report de recerca LSI-97-46-R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, http://www.lsi.upc.es/~jpetit, 1997.

[208] J. Petit. Approximation heuristics and benchmarkings for the MinLA problem. In R. Battiti and A. Bertossi, editors, *Alex '98 — Building bridges between theory and applications*, pages 112–128, http://rtm.science.unitn.it/alex98/book/jpetit.ps.gz, 1998. Università di Trento.

[209] J. Petit. Lower bounds for the minimum linear arrangement problem. In R. Fleischer, B. Moret, and E. M. Schmidt, editors, *Experimental Algorithms*, number 00371 in Dagstuhl Seminar, page 9, ftp://ftp.dagstuhl.de/pub/Reports/00/00371.ps.gz, 2000.

[210] J. Petit. Experiments for the MinLA problem. Report de recerca LSI-R01-7-R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, http://www.lsi.upc.es/~jpetit, 2001.

[211] J. Petit. Hamiltonian cycles in faulty random geometric networks. Report de recerca LSI-01-8-R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, http://www.lsi.upc.es/~jpetit, 2001.

[212] R. Preis and R. Diekmann. The Party partitioning library user guide. Technical report TR-RSFB-96-024, Universität Paderborn, 1996. http://www.uni-paderborn.de/fachbereich/AG/monien/RESEARCH/PART/party.html.

[213] S. Rao and A. W. Richa. New approximation techniques for some ordering problems. In *9th ACM-SIAM Symposium on Discrete Algorithms*, pages 211–218, 1998.

[214] R. Ravi, A. Agrawal, and P. Klein. Ordering problems approximated: single-processor scheduling and interval graph completition. In J. Leach, B. Monien, and M. Rodriguez, editors, *Automata, Languages and Programming*, volume 510 of *Lecture Notes in Computer Science*, pages 751–762, Berlin, 1991. Springer–Verlag.

[215] G. Reinelt. TSPLIB, 1995. http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95.

[216] N. Robertson and P. D. Seymour. Graph minors—a survey. In *Surveys in combinatorics*, pages 153–171. Cambridge University Press, 1985.

[217] H. L. Royden. *Real Analysis*. MacMillan, New York, 1963.

[218] W. Rudin. *Real and Complex Analysis*. McGraw-Hill, New York, 1966.

[219] Y. Saad. *Iterative methods for sparse linear systems*. PWS Publishing Company, Boston, 1996.

[220] A. Sangiovanni-Vincentelli. Automatic layout of integrated circuits. In A. De Michelli, Sangiovanni-Vincentelli, editor, *Design Systems for VLSI circuits*, pages 113–195, Dordrecht, 1987. Martinus Nijhoff Publishers.

[221] J. B. Saxe. Dynamic-programming algorithms for recognizing small-bandwidth graphs in polynomial time. *SIAM Journal on Algebraic Discrete Methods*, 1(4):363–369, 1980.

[222] P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

[223] F. Shahrokhi, O. Sykora, L. A. Szekely, and I. Vrto. On bipartite drawings and the linear arrangement problem. *SIAM Journal on Computing*, 2001. To appear. ftp://ifi.savba.sk/pub/imrich/siam2.ps.gz.

[224] M. I. Shamos and D. Hoey. Closest-point problems. In *16th Annual Symposium on Foundations of Computer Science*, pages 151–162. IEEE Computer Society Press, 1975.

[225] Y. Shiloach. A minimum linear arrangement algorithm for undirected trees. *SIAM Journal on Computing*, 8(1):15–32, 1979.

[226] M. T. Shing and T. C. Hu. Computational complexity of layout problems. In T. Ohtsuki, editor, *Layout design and verification*, volume 4 of *Advances in CAD for VLSI*, pages 267–294, Amsterdam, 1986. North–Holland.

[227] H. D. Simon and S.-H. Teng. How good is recursive bisection? *SIAM Journal on Scientific Computing*, 18(5):1436–1445, 1997.

[228] S. Simonson. A variation on the min cut linear arrangement problem. *Mathematical Systems Theory*, 20(4):235–252, 1987.

[229] K. Skodinis. Computing optimal linear layouts of trees in linear time. In M. Paterson, editor, *Algorithms — ESA 2000*, volume 1879 of *Lecture Notes in Computer Science*, pages 403–414, Berlin, 2000. Springer–Verlag.

[230] G. Skorobohatyj. Code for finding a minimum cut between all node pairs in an undirected graph, 1994. ftp://ftp.zib.de/pub/Packages/mathprog/mincut/all-pairs/index.html.

[231] D. A. Spielman and S.-H. Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *37th Annual Symposium on Foundations of Computer Science*, pages 96–105. IEEE, 1996.

[232] J. M. Steele. Subadditive Euclidean functional and nonlinear growth in geometric probability. *Annals of Probability*, 9:365–376, 1981.

[233] J. M. Steele. Growth rates of Euclidean minimal spanning trees with power weighted edges. *Annals of Probability*, 16:1767–1787, 1988.

[234] J. M. Steele. *Probability theory and combinatorial optimization.* Regional Conference Series in Applied Mathematics. SIAM, 1997.

[235] D. M. Thilikos, M. J. Serna, and H. L. Bodlaender. A constructive linear time algorithm for small cutwidth. Report de recerca LSI-00-48-R, Universitat Politècnica de Catalunya, Departament de Llenguatges i Sistemes Informàtics, http://www.lsi.upc.es/dept/techreps/ps/R01-4.ps.gz, 2001.

[236] D. M. Thilikos, M. J. Serna, and H. L. Bodlaender. A polynomial time algorithm for the cutwidth of bounded degree graphs with small treewidth. Report de recerca LSI-01-4-R, Universitat Politècnica de Catalunya, Departament de Llenguatges i Sistemes Informàtics, http://www.lsi.upc.es/dept/techreps/ps/R01-4.ps.gz, 2001.

[237] J. S. Turner. On the probable performance of heuristics for bandwidth minimization. *SIAM Journal on Computing*, 15(2):561–580, 1986.

[238] W. Unger. The complexity of the approximation of the bandwidth problem. In *37th Annual Symposium on Foundations of Computer Science*, pages 82–91. IEEE Computer Society Press, 1998.

[239] L. G. Valiant. Universality considerations in VLSI circuits. *IEEE Transactions on Computers*, 30(2):135–140, 1981.

[240] P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. Kluwer, New York, 1987.

[241] M. Yannakakis. A polynomial algorithm for the min cut linear arrangement of trees. In *24th Annual Symposium on Foundations of Computer Science*, pages 274–281. IEEE Computer Society Press, 1983.

[242] J. Yuan, Y. Lin, Y. Liu, and S. Wang. NP-completeness of the profile problem and the fill-in problem on cobipartite graphs. *Journal of Mathematical Study*, 31(3):239–243, 1998.

[243] J. E. Yukich. *Probability theory of classical Euclidian optimization problems*. Springer–Verlag, Heidelberg, 1998.

Remark: When possible, references to unpublished documents, such as technical reports or software packages, include an uniform resource locator (URL) to ease the reader obtaining them. Although these URLs were correct at the time this bibliography was made, the author cannot guarantee their future behavior, their contents, nor the legality of their contents.