New lower bounds on the cutwidth of graphs

Jean-Claude Bermond, Michel Cosnard, David Coudert, Frédéric Havet

▶ **To cite this version:**

Jean-Claude Bermond, Michel Cosnard, David Coudert, Frédéric Havet. New lower bounds on the cutwidth of graphs. Computers and Operations Research, 2025, 182, pp.107130. ⟨10.1016/j.cor.2025.107130⟩. ⟨hal-05083322⟩

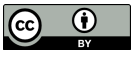
HAL Id: hal-05083322

https://hal.science/hal-05083322v1

Submitted on 24 May 2025

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

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



Distributed under a Creative Commons CC BY 4.0 - Attribution - International License

New lower bounds on the cutwidth of graphs

Jean-Claude Bermond[a], Michel Cosnard[a], David Coudert[a,*], Frédéric Havet[a]

[a] *Université Côte d'Azur, Inria, CNRS, I3S, Sophia Antipolis, France*

Abstract

Cutwidth is a parameter used in many layout problems. Determining the cutwidth of a graph is an NP-complete problem, but it is possible to design efficient branch-and-bound algorithms if good lower bounds are available for cutting branches during exploration. Knowing how to quickly evaluate good bounds in each node of the search tree is therefore crucial.

In this article, we give new lower bounds based on different graph density parameters such as the minimum, the average and the maximum average degree. Our main result is a new bound using the notion of traffic grooming on a path network, which appear to be in many cases better than bounds in the literature. Furthermore, the bound based on grooming can be computed quickly, in $\mathcal{O}(\log n)$ time, and so is of interest to design faster branch-and-bound algorithms. Through extensive experiments, we show that this bound behaves very well compared to other bounds. Furthermore, we show how to obtain even better results when combining it with heuristics for finding dense subgraphs.

Keywords: Cutwidth, lower bounds, traffic grooming, density

⋆This work has been supported by the French government, through the UCA$^{\text{JEDI}}$ Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-15-IDEX-0001

⋆⋆A preliminary version of this paper appears in [8].

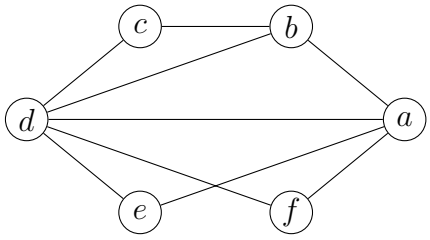
*Corresponding author

Email addresses: jean-claude.bermond@inria.fr (Jean-Claude Bermond), michel.cosnard@inria.fr (Michel Cosnard), david.coudert@inria.fr (David Coudert), frederic.havet@inria.fr (Frédéric Havet)

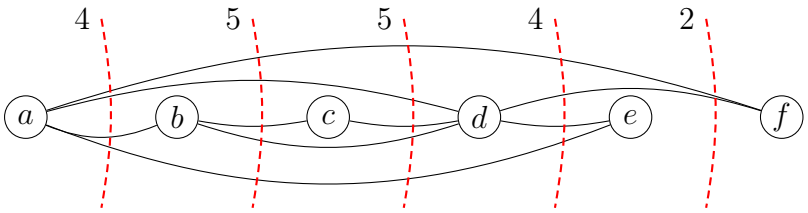
1. Introduction

Graph layout problems are a particular class of combinatorial optimization problems whose goal is to find a layout of an input graph in such a way that a certain objective cost is optimized. A *layout* of a graph $G = (V, E)$ is a linear ordering $\sigma = (v_1, \ldots, v_n)$ of the vertices of G. A large amount of relevant problems in different areas can be formulated as graph layout problems (see the survey [21] and references therein).

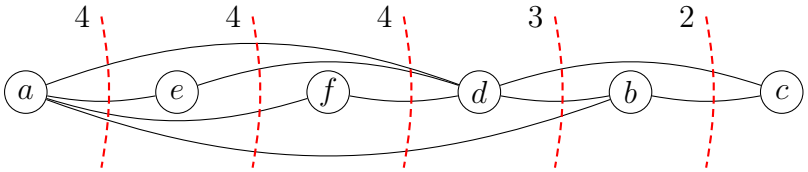
The *edge-cut* of a set $X \subseteq V$ is the set of edges with one end-vertex in X and the other in $\overline{X} = V \setminus X$. Its size is denoted by $\mathrm{ec}(X, \overline{X})$. Let $X_i = \{v_1, \ldots, v_i\}$. The *width* $w(G, \sigma)$ of a graph G with respect to the layout σ is the maximum of $\mathrm{ec}(X_i, \overline{X_i})$ over all $i \in \{1, \ldots, n-1\}$. The *cutwidth* of a graph G, denoted by $\mathrm{cw}(G)$, is the minimum width of a layout, that is $\mathrm{cw}(G) = \min\{w(G, \sigma) \mid \sigma \text{ layout of } G\}$.



(a) A graph G of cutwidth $\mathrm{cw}(G) = 4$.



(b) Layout $\sigma_1 = (a, b, c, d, e, f)$ of G such that $w(G, \sigma_1) = 5$.



(c) Layout $\sigma_2 = (a, e, f, d, b, c)$ of G such that $w(G, \sigma_2) = 4$.

Figure 1: Examples of layouts of the graph G of Figure 1a. We size the value of each edge-cut $\mathrm{ec}(X_i, \overline{X_i})$, identified with dotted lines.

To illustrate the definitions of layouts, edge-cuts and cutwidth, let us consider the example of Figure 1. The graph G of Figure 1a has cutwidth $\mathrm{cw}(G) = 4$. With the layout $\sigma_1 = (a, b, c, d, e, f)$ reported in Figure 1b, we have $\mathrm{ec}(\{a\}, \{b, c, d, e, f\}) = 4$, $\mathrm{ec}(\{a, b\}, \{c, d, e, f\}) = 5$, $\mathrm{ec}(\{a, b, c\}, \{d, e, f\}) = 5$, $\mathrm{ec}(\{a, b, c, d\}, \{e, f\}) = 4$ and $\mathrm{ec}(\{a, b, c, d, e\}, \{f\}) = 2$. Thus, we have $w(G, \sigma_1) = 5$. With the layout $\sigma_2 = (a, e, f, d, b, c)$ reported in Figure 1c, we have $w(G, \sigma_2) = 4$, and this is the best possible value for this graph.

The cutwidth was first used as a theoretical model for the number of channels in an optimal layout of a circuit in the seventies [3, 23]. In general, the cutwidth of a graph times its number of vertices gives a measure of the area needed to represent the graph in a VLSI (very-large-scale integration) layout when vertices are laid out in a row [22]. More recent applications of this problem include network reliability [20], automatic graph drawing [24], information retrieval [5], scheduling series-parallel tasks [35], and as a subroutine for the cutting plane algorithm to solve the Traveling Salesman Problem [19]. Due to its natural definition, the cutwidth has various applications in computer science: whenever data is expected to be roughly linearly ordered and dependencies or connections are local, the cutwidth of the corresponding graph is expected to be small. However, computing the cutwidth of graphs is NP-hard [21] in general and cannot be approximated within a constant factor under the Small Set Expansion Hypothesis [51]. Exact exponential-time algorithms using dynamic programming have been proposed in [10], one with time and space complexity in $\mathcal{O}^*(2^n)$, and another with time complexity in $\mathcal{O}^*(4^n)$ and polynomial space complexity (the $\mathcal{O}^*$ notation hides factors that are polynomial in n). Fixed-parameter algorithms have also been designed [29, 47] and a $\mathcal{O}(\log^{1+o(1)} n)$ approximation algorithm has been proposed in [4]. Furthermore efficient branch-and-bound algorithms have been proposed [39, 43] but they need to have good lower bounds for the cutwidth, and in particular bounds that can be quickly evaluated at each node of the branch-and-bound tree. These bounds are used to cut (prune) branches. The quality of these bounds have therefore a strong impact on the performances of the branch-and-bound algorithms. Finally, let us mention the recently proposed methods for computing lower bounds on the cutwidth of a graph based on semidefinite programming (SDP) relaxation [27], which are able to produce good lower bounds but at a high computational cost.

Interestingly, for any graph of order $n \leq 31$, an optimal solution for cutwidth can be obtained within seconds using the implementation available in SageMath [46] of the dynamic programming algorithm proposed in [10]

with time and space complexity in $\mathcal{O}^*(2^n)$. For larger graphs, one can use for instance integer linear programming formulations [18], the branch-and-bound algorithm proposed in [39] or the methods based on semidefinite programming proposed in [27]. However, these methods usually require hours of computations for graphs with 50 vertices. For these methods, lower bounds are not only used to cut the search space, but also to measure the optimality gap of the valid solutions found during the executions of the algorithms.

Our results. In this paper, we first review previous lower bounds on the cutwidth of a graph (Section 2). Next, we introduce new lower bounds based on the minimum or maximum average degree of G (Section 3). Note that all these parameters can be computed in polynomial time. Moreover, the bounds are complementary, in the sense as depending on the graph the bound giving the largest value changes. We then present in Section 4 a new lower bound based on the results obtained for the *Maximum All request Path Grooming problem* [9], a problem also known as the *Call Control Problem in Path Networks* in [2]. This new bound indicates the minimum cutwidth of any graph with n vertices and m edges, independently from the structure of the graph, and it can be computed in $\mathcal{O}(\log n)$ time. Furthermore, we prove that this new bound is better than bounds based on the average degree of the graph. Finally, we report in Section 5 experimental comparisons of these bounds on various graph classes. In particular, we exhibit different behaviours on Erdős-Rényi random graphs [25] and on random (proper) interval graphs [45]. Furthermore, we show how to obtain even better results when combining the bound based on traffic grooming with heuristics for finding dense subgraphs.

Definitions and notations. Notations and definitions follow [12]. All the graphs considered in this paper are finite, undirected, unweighted and simple (without loops nor multiple edges). The graph $G = (V, E)$ with vertex set V and edge set E has *order* $n = |V|$ and *size* $m = |E|$. Let $N(v)$ denotes the set of neighbours of $v \in V$, that is $N(v) = \{u \mid uv \in E\}$. The *degree* of a vertex v, denoted by $d(v)$, is its number $|N(v)|$ of neighbours. Let $\delta(G)$ and $\Delta(G)$ denote respectively the minimum and maximum degree of G. A graph G is *k-degenerate* if each of its subgraphs has a vertex of degree at most k. The *degeneracy* $\delta^*(G)$ of the graph G is the least integer k such that G is k-degenerate. It is easy and well-known that a graph is k-degenerate if and only if it has an elimination ordering $(v_1, \ldots, v_n)$ of the vertices such that v_i has at most k neighbours in $(v_{i+1}, \ldots, v_n)$ for all $1 \leq i \leq n-1$.

We will use the following property of cutwidth.

Property 1. *For any subgraph H of a graph G, we have* $\mathrm{cw}(G) \geq \mathrm{cw}(H)$.

2. Previous lower bounds

In this section, we survey the lower bounds on the cutwidth of graphs that have previously been proposed, and which are based on minimal edge-cuts [21], on the algebraic connectivity [34], on the maximum degree or on the degeneracy of the graph [37].

Bound based on minimal edge-cuts. From the definition of cutwidth, it is obvious that the minimum size of an edge-cut between any pair u, v of vertices of the graph, which we denote by $\kappa'(u, v)$, is a lower bound on its cutwidth (see [21] for more details).

Proposition 1 ([21])**.** *For any graph G, let $\kappa'(u, v)$ denote the minimum size of an edge-cut between vertices u and v of G. We have*

$$\mathrm{cw}(G) \geq LB_{\kappa'}(G) = \max\{\kappa'(u, v) \mid u, v \in V,\ u \neq v\} \tag{1}$$

This bound can be computed in $\mathcal{O}(m^{1+o(1)})$ time using the ground-breaking algorithm proposed in [1] for building the Gomory-Hu tree [31] of the graph, which uses the algorithm proposed in [16] for computing a maximum flow in $\mathcal{O}(m^{1+o(1)})$ time.

Bound based on the algebraic connectivity. Let λ_2 denote the second smallest eigenvalue of the Laplacian matrix associated with the graph. The following lower bound is due to [34]:

Proposition 2 ([34])**.** *For any graph G, let λ_2 denote the second smallest eigenvalue of the Laplacian matrix associated with a graph G. We have*

$$\mathrm{cw}(G) \geq LB_{\lambda_2}(G) = \frac{\lambda_2}{n} \left\lfloor \frac{n}{2} \right\rfloor \left\lceil \frac{n}{2} \right\rceil \tag{2}$$

The time complexity of this bound is due to the computation of λ_2, which requires $\mathcal{O}(n^\omega)$ time [44], with $\omega \leq 2.371552$ [49], but fast numerical methods can be used to compute eigenvalues in practice [38].

Note that λ_2 is known as the *algebraic connectivity* of a graph. It reflects how well the overall graph is connected (the larger the better). For instance, for the complete graph K_n, we have $\lambda_2(K_n) = n$, while for the hypercube Q_n of dimension $n \geq 2$, we have $\lambda_2(Q_n) = 2$. The algebraic connectivity

of any non-clique graph G is upper bounded by its vertex-connectivity $\nu(G)$ and its edge-connectivity $\kappa(G)$, i.e., $\lambda_2 \leq \nu(G) \leq \kappa(G)$ [26]. Furthermore, it is lower bounded by $\frac{1}{n\,\mathrm{diam}(G)}$ [41], where $\mathrm{diam}(G)$ denotes the diameter of the graph. Hence, it is equal to 0 if the graph is not connected and is larger than 0 otherwise.

Bound based on the maximum degree. Let $\Delta(G)$ denote the maximum degree of G, let v_i be a vertex with degree $\Delta(G)$ and consider a layout $(v_1, \ldots, v_n)$ of G. One set among $\{v_1, \ldots, v_{i-1}\}$ and $\{v_{i+1}, \ldots, v_n\}$ contains at least half of the neighbours of v_i. Consequently,

Proposition 3 (folklore)**.** *For any graph G with maximum degree $\Delta(G)$, we have*

$$\mathrm{cw}(G) \geq LB_{\Delta}(G) = \left\lceil \frac{\Delta(G)}{2} \right\rceil \tag{3}$$

Depending on the data structure used for storing the graph, this bound can be computed for instance in $\mathcal{O}(n)$ time when the degree of a vertex can be accessed in constant time, or in $\mathcal{O}(n+m)$ time when it is necessary to scan all the edges to first determine the degrees of the vertices.

Bound based on the degeneracy. A graph G is *k-degenerate* if every subgraph H of G has minimum degree at most k. Equivalently, G is k-degenerate, if there exists a layout $(v_1, \ldots, v_n)$ such that for all $i \in \{2, \ldots, n\}$ every vertex v_i has at most k neighbours in $\{v_1, \ldots, v_{i-1}\}$. The *degeneracy* of G, denoted by $\delta^*(G)$, is the smallest k such that G is k-degenerate.

The following lower bound is proved in [37].

Proposition 4 ([37])**.** *For any graph G with degeneracy $\delta^*(G)$, we have*

$$\mathrm{cw}(G) \geq LB_{\delta^*}(G) = \frac{1}{4}\delta^*(G)(\delta^*(G)+2) \tag{4}$$

The degeneracy of a graph can be computed in $\mathcal{O}(n+m)$ time [40] and so does this bound.

Other lower bounds can be obtained using linear programming [4] and SDP relaxation [27]. However, they cannot be computed in a reasonable time for graphs with more than 50 vertices.

3. New lower bound based on degrees

In this section, we propose new lower bounds on the cutwidth of a graph based on the minimum and the average degree.

Bound based on the minimum degree.

Proposition 5. *For any graph G with minimum degree $\delta(G)$, we have*

$$\mathrm{cw}(G) \geq LB_\delta(G) = (\lfloor \delta(G)/2 \rfloor + 1)\lceil \delta(G)/2 \rceil \geq \frac{1}{4}\delta(G)(\delta(G)+2) \quad (5)$$

Proof. Let $\delta(G)$ denote the minimum degree of G. Let $\sigma = (v_1, \ldots, v_n)$ be a layout of G. Set $k = \lfloor \delta(G)/2 \rfloor + 1$ and let $X_k = \{v_1, \ldots, v_k\}$. Every vertex in X_k has at most $k-1 = \lfloor \delta(G)/2 \rfloor$ neighbours in X_k and so at least $\lceil \delta(G)/2 \rceil$ neighbours in $\overline{X_k}$. Hence $\mathrm{ec}(X_k, \overline{X_k}) \geq k\lceil \delta(G)/2 \rceil = (\lfloor \delta(G)/2 \rfloor + 1)\lceil \delta(G)/2 \rceil$. Furthermore, we have that $(\lfloor \delta(G)/2 \rfloor + 1)\lceil \delta(G)/2 \rceil \geq \frac{1}{4}\delta(G)(\delta(G)+2)$. □

As $\delta^*(G) \geq \delta(G)$, the bound of Ineq. (4) is better than the bound of Ineq. (5). However, by definition of degeneracy, there is a subgraph H of G such that $\delta(H) = \delta^*(G)$. Since $\mathrm{cw}(G) \geq \mathrm{cw}(H)$, Ineq. (5) implies the bound of Ineq. (4) (with a simpler proof than the original one [37]).

The time complexity for computing the bound based on the minimum degree is the same as for the bound based on the maximum degree.

Bound based on the average degree.

Proposition 6. *Let $\sigma = (v_1, \ldots, v_n)$ be a layout of G and let $X_i = \{v_1, \ldots, v_i\}$. For each vertex v_i, let $d^+(v_i)$ be the number of edges $v_iv_j \in E(G)$ with $j > i$ (forward edges from v_i). We have*

$$w(G, \sigma) \geq \frac{1}{2(n-1)} \sum_{i=1}^{n-1} d^+(v_i)(d^+(v_i)+1)$$

Proof. We use a classical trick in (traffic) grooming. We first compute the total load of G on $P_\sigma = v_1v_2 \ldots v_n$, also called *sum cut* of the layout σ [21], which is $L(G, \sigma) = \sum_{i=1}^{n-1} l(v_iv_{i+1}) = \sum_{i=1}^{n-1} \mathrm{ec}(X_i, \overline{X_i})$, where $l(v_iv_{i+1})$ denotes the "load" of the edge v_iv_{i+1} of P_σ, that is the number of edges $v_jv_\ell \in E(G)$ such that $j \leq i < \ell$. To compute $L(G, \sigma)$, we note that the load of the forward edges of G from v_i is at least $d^+(v_i)$ on the edge v_iv_{i+1}, at least $d^+(v_i) - 1$ on the edge $v_{i+1}v_{i+2}$ and so on. Therefore the load due to

the edges from v_i is at least $\frac{1}{2}d^+(v_i)(d^+(v_i)+1)$ and the total load is at least $\frac{1}{2}\sum_{i=1}^{n-1} d^+(v_i)(d^+(v_i)+1)$. Then we note that, as the path has $n-1$ edges, there exists an edge with load at least $\frac{L(G,\sigma)}{n-1}$ and so $w(G,\sigma) \geq \frac{L(G,\sigma)}{n-1}$. □

The *average degree* of a graph G with m edges and n vertices is $\mathrm{Ad}(G) = 2m/n$.

Proposition 7. *For any graph G with n vertices and m edges, we have*

$$\begin{aligned}\mathrm{cw}(G) \geq LB_{\mathrm{Ad}}(G) &= \frac{1}{2}\frac{m}{n-1}\left(\frac{m}{n-1}+1\right)\\ &= \frac{1}{8}\left(1+\frac{1}{n-1}\right)^2 \mathrm{Ad}(G)\left(\mathrm{Ad}(G)+2-\frac{2}{n}\right) \quad (6)\end{aligned}$$

Proof. If a function $f(x)$ is convex and if $\sum_{i=1}^{n-1} x_i = m$, then the minimum of $\sum_{i=1}^{n-1} f(x_i)$ is attained when all the x_i are equal to $m/(n-1)$. Applying this argument to the convex function $x \mapsto x(x+1)$ with $x_i = d^+(v_i)$ and using the fact that $\sum_{i=1}^{n-1} d^+(v_i) = m$, we get

$$\begin{aligned}\frac{1}{2(n-1)}\sum_{i=1}^{n-1} d^+(v_i)(d^+(v_i)+1) &\leq \frac{1}{2(n-1)}\sum_{i=1}^{n-1}\frac{m}{n-1}\left(\frac{m}{n-1}+1\right)\\ &\leq \frac{1}{2}\frac{m}{n-1}\left(\frac{m}{n-1}+1\right)\end{aligned}$$

□

Clearly, this bound can be computed in $\mathcal{O}(1)$ time.

None of the bounds given by Ineq. (4) and (6) is always better than the other. Indeed, on the one hand, there exist graphs with bounded average degree and degeneracy as large as we want. For example, consider the graph H_k^i obtained by identifying one vertex of the clique K_k of order k with an extremity of the path P_i of order i. Then the degeneracy is $k-1$ but the average degree tends to 2 when i tends to infinity. So for i large, the bound of Ineq. (4) is better. In Table 1, we have given the value of the lower bounds for the graphs H_k^i with $k = 50$ and $0 \leq i \leq 6$.

On the other hand, consider the graph $G(s,n)$ with n vertices where vertex v_i is joined to the s vertices v_{i+h}, with $1 \leq h \leq s$ and $i+h \leq n$. Then its degeneracy is s (consider the layout $(v_1,\ldots,v_n)$), but we have $m =$

$sn - s(s+1)/2$ and so for n large the bound of Ineq. (6) is better. In Table 2, we have given the value of the lower bounds for the graphs $G(s,n)$ with $s = 1, 5, 10, 15, 20$ and $n = 2s + 1$.

Bound based on the maximum average degree. The *maximum average degree* of G is $\mathrm{Mad}(G) = \max\{\mathrm{Ad}(H) \mid H \text{ subgraph of } G\}$. Observe that this parameter can be computed in polynomial time [13, 14, 17, 28, 30, 36].

Proposition 8. *For any graph G, we have*

$$\mathrm{cw}(G) \geq LB_{\mathrm{Mad}}(G) = \max\{LB_{\mathrm{Ad}}(H) \mid H \text{ subgraph of } G \text{ such that } \mathrm{Ad}(H) = \mathrm{Mad}(G)\} \quad (7)$$

Proof. By definition of Mad, there is at least one subgraph H of G such that $\mathrm{Ad}(H) = \mathrm{Mad}(G)$. Since $\mathrm{cw}(G) \geq \mathrm{cw}(H)$, Ineq. (6) directly implies the result. □

Proposition 9. *Let n_{H_0} be the number of vertices of the smallest subgraph H_0 such that $\mathrm{Ad}(H_0) = \mathrm{Mad}(G)$, and let m_{H_0} be its number of edges. We have*

$$\mathrm{cw}(G) \geq LB_{\mathrm{Mad}}(G) = \frac{1}{2}\frac{m_{H_0}}{n_{H_0}-1}\left(\frac{m_{H_0}}{n_{H_0}-1}+1\right) \quad (8)$$

Proof. The subgraph H_0 maximizes $m_{H_0}/(n_{H_0}-1)$ among all the subgraphs of G with average degree $\mathrm{Mad}(G)$. Then, the results follows from Propositions 7 and 8. □

The following tight inequalities are well-known (see e.g. Proposition 3.1 of [42]): $\delta^*(G) \leq \mathrm{Mad}(G) < 2\delta^*(G)$. Thus none of the bounds given by Ineq. (4) and (8) is always better than the other. The lower bound of Ineq. (8) is better than the bound of Ineq. (4) when $\mathrm{Mad}(G) > \sqrt{2}\delta^*(G)$ and worse in the other case.

It has been shown in [28, 30] that Mad can be determined in $\mathcal{O}(nm\log(n^2/m))$ time using a maximum-flow computation. Furthermore, efficient approximation algorithms have been proposed, such as a $(1-\epsilon)$-approximation algorithm with time complexity in $\tilde{\mathcal{O}}(m/\epsilon)$ [13, 15] (the $\tilde{\mathcal{O}}$ notation hides logarithmic factors in n).

4. New lower bounds using the grooming on the path

In this section, we present a new lower bound on the cutwidth of a graph based on the results obtained for the *Maximum All request Path Grooming problem* [9], a problem also known as the *Call Control Problem in Path Networks* in [2]. This bound, that we explain in Section 4.1, uses a closed formula proved in [9] and recalled in Section 4.2 that gives the maximum number of requests that can be satisfied on the path network with n nodes and capacity C, or equivalently on the maximum number of edges of a graph of order n and cutwidth C. Next, we show in Section 4.3 that this new bound is always better than the bound $\mathrm{LB}_{\mathrm{Ad}}$, and in Section 4.4 that it can be computed in $\mathcal{O}(\log n)$ time. We finally explain in Section 4.5 how to combine this bound with methods for finding dense subgraphs.

4.1. Lower bound based on the grooming on the path, $C^*(m, n)$

Consider the path $P_\sigma = v_1 v_2 \cdots v_n$ associated to the layout $\sigma = (v_1, v_2, \ldots, v_n)$. For each edge $e \in E(G)$, the *request associated to e by σ*, denoted by $R_\sigma(e)$, is the subpath of P_σ whose end-vertices are those of e. Let $\mathcal{R}_\sigma(G)$ be the set of requests associated by σ to the edges of G. Observe that the load of the edge $v_i v_{i+1}$ in P_σ under $\mathcal{R}_\sigma(G)$ denoted $l(v_i v_{i+1})$ is the number of edges $v_j v_\ell$ of G with $j \leq i < \ell$, that is $\mathrm{ec}(X_i, \overline{X_i})$. Hence $w(G, \sigma)$ is the maximum load of an edge of P_σ under $\mathcal{R}_\sigma(G)$.

The *grooming factor* C is the maximum load that is allowed on the edges of P. Given a grooming factor C, the Maximum All request Path Grooming problem consists in finding the maximum number of requests denoted by $T(C, n)$ that can be satisfied (or groomed) together on P such that the load of any edge is at most C.

We note that the value $T(C, n)$ is nothing else than the maximum number of edges of a simple graph of order n with cutwidth C and we emphasize this observation as a proposition.

Proposition 10. *Let n and C be positive integers. The maximum number m of edges of any simple graph G of order n and cutwidth $\mathrm{cw}(G) \leq C$ is $m = T(C, n)$.*

It follows that for a given value C, any graph of order n with $m = T(C, n)$ edges has cutwidth at least C. We define $C^*(m, n) = \min\{C \mid m \leq T(C, n)\}$.

Proposition 11. *For any graph G with n vertices and m edges, we have*

$$\mathrm{cw}(G) \geq C^*(m, n) = \min\{C \mid m \leq T(C, n)\} \tag{9}$$

Proof. Let σ be a layout of G such that $w(G,\sigma) = \mathrm{cw}(G)$. As $w(G,\sigma)$ is the maximum load of an edge of P_σ under $\mathcal{R}_\sigma(G)$, we have $T(\mathrm{cw}(G), n) \geq m$. The result follows. □

Note that the bound is reached by taking as edges of G the pairs of end-vertices of the $T(C^*(m,n),n)$ requests that can be groomed on a path of n vertices with grooming factor $C^*(m,n)$. This set of requests can be found in polynomial time using the greedy algorithm proposed in [9]. In other words, one can find in polynomial time a graph with n vertices, $m = T(C^*(m,n),n)$ edges and cutwidth $\mathrm{cw}(G) = C^*(m,n)$.

We will see later (Proposition 13) that we can efficiently compute the value of $C^*(m,n)$ thanks to the results of [9] where a closed formula for $T(C,n)$ has been given.

4.2. Previous results on the computation of $T(C,n)$ (see [9])

First, we note that if $C \geq \lfloor n^2/4 \rfloor$, then we can satisfy all requests and so $T(C,n) = n(n-1)/2$. If $C = 1$, the determination of $T(1,n)$ is easy as an optimal solution consists in taking the $n-1$ requests of length 1 (that are the requests $(i, i+1)$ for $1 \leq i \leq n-1$) and so $T(1,n) = n-1$. In [7] it is also proven that for $C = 2$, we have $T(2,n) = \lfloor (3n-3)/2 \rfloor$. The optimum can be easily found for $C \leq 6$; in particular for $C = 3$ (respectively, $C = 6$ and $n \geq 6$) the maximum is obtained by considering all requests of length 1 and 2 (respectively 1, 2 and 3).

Based on these results, it was conjectured that the optimum can be obtained by taking the greedy solution consisting of all the requests with the smallest length (this was presented as a result in [39]). However, this is true only for $C \leq 9$ and it appears that the conjecture is false. In particular, one could have expected that, for $C = C_s = s(s+1)/2$, an optimal solution would be obtained by taking all requests of length at most s in number $sn - C_s$ (the graph of requests being the graph $G(s,n)$ used in Section 3 to show that the bound of Ineq. (6) is better than that of Ineq. (4)). But it is not true for $n = 11$ and $C = 10$ ($s = 4$). The solution with all requests of length at most 4 has $4 \times 11 - 10 = 34$ requests. The maximum load is 10, but it is reached only for the edges of the path $v_1 \cdots v_{11}$ of the form $v_i v_{i+1}$ with $4 \leq i \leq 8$. So we can delete the request of length 4 $v_4 v_8$ and add the two requests of length 5 $v_1 v_6$ and $v_6 v_{11}$ to get a solution with 35 requests. We summarize the results of [9] giving the value of $T(C,n)$:

Theorem 1 ([9])**.** *Let n and C be fixed positive integers.*

- *If* $C \geq \left\lfloor \frac{n^2}{4} \right\rfloor$, *then* $T(C,n) = \frac{n(n-1)}{2}$;
- *If* n *is even and* $\frac{n(n-2)}{8} < C \leq \left\lfloor \frac{n^2}{4} \right\rfloor$, *then* $T(C,n) = \frac{n(n-2)}{4} + C$;
- *If* n *is odd and* $\frac{n^2-1}{8} < C \leq \left\lfloor \frac{n^2}{4} \right\rfloor$, *then* $T(C,n) = \frac{(n-1)^2}{4} + C$.
- *Otherwise, let* s *be an integer such that* $C_{s-1} < C \leq C_s$ *with* $C_s = \frac{s(s+1)}{2}$; $d = C_s - C$; $n = qs + r$ *with* $0 \leq r < s$; $r = aq + \alpha$ *with* $0 \leq \alpha < q$; $s - r = b(q+1) + \beta$ *with* $0 \leq \beta \leq q$; $A(C,n) = ar - \frac{a(a+1)}{2}q$ *and* $B(C,n) = (b+1)(s-r) - \frac{b(b+1)}{2}(q+1)$. *We have*

$$T(C,n) = sn - C_s - dq + \min\{A(C,n) + d, B(C,n)\}$$

We will now show (Proposition 12) that the bound of Ineq. (9) is better than that of Ineq. (6). For that, we use the upper bound on $T(C,n)$ given in [9].

Theorem 2 (Theorem 17 in [9])**.** *Let* n *and* C *be fixed positive integers. Let* s *be an integer such that* $C_{s-1} < C \leq C_s$ *with* $C_s = \frac{s(s+1)}{2}$, *let* $d = C_s - C$ *and let* $n = qs + r$ *with* $0 \leq r < s$.

If $C \leq 9$, *then* $T(C,n) \leq sn - C_s - d(q-1)$.

If $C \geq 10$, *then* $T(C,n) \leq sn - C_s - d(q-1) + (5 - 2\sqrt{6})C$.

4.3. Proof that $C^*(m,n) \geq LB_{\text{Ad}}(m,n)$

We now show that the new lower bound C^* is better than the bound LB_{Ad}, i.e., that we always have $C^*(m,n) \geq \text{LB}_{\text{Ad}}(m,n)$.

Proposition 12. *Let* $n \geq 2$ *and* $m > 0$ *be integers. Then*

$$C^*(m,n) \geq LB_{\text{Ad}}(m,n).$$

Proof. Recall that $\text{LB}_{\text{Ad}}(m,n) = \frac{1}{2}\frac{m}{n-1}\left(\frac{m}{n-1} + 1\right)$. Note that the proposition is true if

$$T(\lfloor \text{LB}_{\text{Ad}}(m,n) \rfloor, n) < m \tag{10}$$

Indeed, this inequality implies $C^*(m,n) > \lfloor \text{LB}_{\text{Ad}}(m,n) \rfloor$ and so $C^*(m,n) \geq \text{LB}_{\text{Ad}}(m,n)$.

If $m \leq n-1$, then Proposition 12 is obviously true as $\text{LB}_{\text{Ad}}(m,n) \leq 1$ and $C^*(1,n) = \cdots = C^*(n-1,n) = 1$.

If $n-1 < m < 3(n-1)/2$, then we have $\mathrm{LB}_{\mathrm{Ad}}(m,n) < 15/8$ and so $\lfloor \mathrm{LB}_{\mathrm{Ad}}(m,n) \rfloor = 1$. But we also have $T(1,n) = n-1 < m$. Consequently, Inequality (10) holds and so does Proposition 12.

If $3(n-1)/2 \leq m < 2(n-1)$, then we have $\mathrm{LB}_{\mathrm{Ad}}(m,n) < 3$ and so $\lfloor \mathrm{LB}_{\mathrm{Ad}}(m,n) \rfloor = 2$. But we also have $T(2,n) \leq 3(n-1)/2 \leq m$. Consequently, (10) holds and so does Proposition 12.

More generally, let $(s-1+\delta/s) \leq m/(n-1) < s-1+(\delta+1)/s$, where s and δ are integers satisfying $s \geq 3$ and $0 \leq \delta \leq s-1$. We have:
$\mathrm{LB}_{\mathrm{Ad}}(m,n) < \frac{1}{2}[s-1+(\delta+1)/s][s+(\delta+1)/s]$, that is
$\mathrm{LB}_{\mathrm{Ad}}(m,n) < s(s-1)/2 + \delta + 1 - [(\delta+1)/s - (\delta+1)^2/s^2]/2$.
As $0 \leq (\delta+1)/s \leq 1$, we have $(\delta+1)/s - (\delta+1)^2/s^2 \geq 0$ and so

$$\lfloor \mathrm{LB}_{\mathrm{Ad}}(m,n) \rfloor \leq s(s-1)/2 + \delta$$

We now distinguish two cases:

- Case 1: $\delta = 0$. On the one hand, we have $(s-1)(n-1) \leq m$. On the other hand, we get by Theorem 2 applied with $s-1$, $d=0$ and $C = C_{s-1} = s(s-1)/2$:

$$T(\lfloor \mathrm{LB}_{\mathrm{Ad}}(m,n) \rfloor, n) \leq T(C_{s-1}, n) \leq (s-1)n - (4-2\sqrt{6})C_{s-1}$$

  Since $(2-\sqrt{6}) \geq 1$, for $s \geq 3$ we get $(4-2\sqrt{6})C_{s-1} > s-1$. Therefore, $T(\lfloor \mathrm{LB}_{\mathrm{Ad}}(m,n) \rfloor, n) < (s-1)n - (s-1) = (s-1)(n-1) \leq m$.

  So, when $\delta = 0$, Inequality (10) holds and so does Proposition 12.

- Case 2: $1 \leq \delta \leq s-1$. Let $d = s - \delta$, so $1 \leq d \leq s-1$.

  We have $m \geq (s-1+\delta/s)(n-1) = (s-d/s)(n-1) = sn - s - d(n-1)/s$.
  As $r \leq s-1$, $n-1 = qs + r - 1 \leq qs + s - 2$, we get:

$$m \geq sn - s - (q+1)d + 2d/s$$

  We also have $\lfloor \mathrm{LB}_{\mathrm{Ad}}(m,n) \rfloor \leq s(s-1)/2 + \delta = C = C_s - d$.

- Sub-case 2.1: $s \leq 4$. In this case, we have $C \leq 9$ and so, by Theorem 2, $T(C_s - d, n) \leq sn - C_s - d(q-1)$. Hence, Inequality (10) is satisfied if

$$sn - C_s - d(q-1) \leq sn - s - (q+1)d + 2d/s$$

which is equivalent to

$$C_s \geq s + 2d - 2d/s \geq 3s - 2 - 2(s-1)/s \quad (\text{as } d \leq s-1)$$

That is verified for $s = 4$ as $10 > 10 - 6/4$ and for $s = 3$ as $6 \geq 7 - 4/3$.

- Sub-case 2.2: $s \geq 5$. In this case, we have $C \geq 10$ and by Theorem 2, we have $T(C_s - d, n) \leq sn - C_s - d(q-1) + (5 - 2\sqrt{6})C$. As $C < C_s$, Inequality (10) is satisfied if

$$(2\sqrt{6} - 4)C_s \geq s + 2d - 2d/s \geq 3s - 2 - 2(s-1)/s \quad (\text{as } d \leq s-1)$$

That is clearly true for $s = 5$ as $30(\sqrt{6} - 2) \geq 13.48 > 13 - 8/5$. So it is also true for $s \geq 5$

In both cases, Inequality (10) holds and so does Proposition 12. □

The gap between the bounds can be significant. According to Theorem 1, for $C = C_s$ and $n = 2s + 1$, we have $T(C, n) = sn - C_s$; indeed $d = 0$ and $A(C, n) = 0$ as $a = 0$. For example, for $s = 5$ and $n = 11$, we have $T(15, 11) = 40$; but Ineq. (6) gives $\mathrm{LB}_{\mathrm{Ad}}(40, 11) = 10 < 15$. More generally, for $C = C_s$ and $n = 2s + 1$ we get $T(C, n) = sn - C_s = (3s^2 + s)/2$ and $\mathrm{LB}_{\mathrm{Ad}}(m, n) = \mathrm{LB}_{\mathrm{Ad}}((3s^2 + s)/2, 2s + 1) = (9s^2 + 18s + 5)/32$, and so the ratio $C_s/\mathrm{LB}_{\mathrm{Ad}}(m, n)$ tends to $16/9$ when s increases. See also the examples reported in Table 2.

4.4. Computational complexity of determining $C^*(m, n)$

In this section we show that, since the value of $T(C, n)$ can be computed in $\mathcal{O}(1)$ time, the bound $C^*(m, n)$ of Ineq. (9) can be computed efficiently in $\mathcal{O}(\log n)$ time.

Proposition 13. *Given two integers n and m, the bound $C^*(m, n)$ can be computed in $\mathcal{O}(\log n)$ time.*

Proof. Obviously, when $n < 2$ or $m = 0$, then the bound is 0, and when $1 \leq m \leq n-1$, it is 1. When $m = n(n-1)/2$, then by Theorem 1, $C^*(m, n) = \lfloor n^2/4 \rfloor$. It remains to consider the case in which $n - 1 < m < n(n-1)/2$. Since the function C^* increases with m (i.e., $C^*(m + 1, n) \geq C^*(m, n)$), we can find $C^*(m, n)$ using binary search on the interval $[C_1 = 1, C_2 = \lfloor n^2/4 \rfloor]$. Indeed, we can compute $T(C_p, n)$, with $C_p = \lfloor (C_1 + C_2)/2 \rfloor$, and compare

this value to m to decide which subinterval to consider next and so on. The number of iterations of the binary search is in $\mathcal{O}(\log n)$ and each computation of $T(C_p, n)$ requires $\mathcal{O}(1)$ time. This concludes the proof.

Note that we can start the search with a shorter interval. Indeed, by Theorem 2 when $C = C_s$ we have $sn - C_s \leq T(C_s, n) \leq sn - C_s + (5 - 2\sqrt{6})C_s$. So, if we let s_1 be the real number such that $m = s_1 n - s_1(s_1+1)/2$, we have $C^*(m, n) \leq C_{\lceil s_1 \rceil}$. If s_2 is such that $m = s_2 n - (2\sqrt{6} - 4)s_2(s_2 + 1)/2)$, we have $C^*(m, n) \geq C_{\lfloor s_2 \rfloor}$. It follows that we can start the binary search with the interval $\left[C_{\lfloor s_2 \rfloor}, C_{\lceil s_1 \rceil}\right]$. □

4.5. Extensions

Lower bounds in branch-and-bound algorithms can be quickly computed by using $C^*(m, n)$ (or a good approximation) or the bounds of Ineq. (6) or Ineq. (5). We can also use the better bound of Ineq. (8) but it needs the computation of $\mathrm{Mad}(G)$ which is polynomial but long in practice. We can also use an improvement of Ineq. (9) by noting that $\mathrm{cw}(G) \geq \mathrm{cw}(H)$ for any subgraph H of G. So, if we consider like for Ineq. (8) the subgraph H_0 with the minimum number n_{H_0} of vertices such that $\mathrm{Ad}(H_0) = \mathrm{Mad}(G)$, and let m_{H_0} be its number of edges, we have:

$$\mathrm{cw}(G) \geq C^*_{\mathrm{Mad}}(G) = C^*(m_{H_0}, n_{H_0}) \tag{11}$$

The design of a fast algorithm for computing the bounds of Ineq. (11) seems, however more challenging and we let it as an interesting open problem. However, one can use existing exact and approximate algorithms for finding dense subgraphs to get a good lower bound (see, e.g., [13, 14, 48]). For instance, we can use the method proposed in [14] for finding an approximation of the maximum average degree of a graph. This leads to the bounds C^*_g and C^*_{g++} that we define in Section 5.1.

5. Experimental evaluation

In this section, we compare the different lower bounds presented in this paper on synthetic graphs (Section 5.2), Erdős-Rényi random graphs (Section 5.3) and random (proper) interval graphs (Section 5.4). We also compare these lower bounds with the values obtained in [27] using SDP relaxation on some random graphs (Section 5.5). We start by presenting our experimental settings in Section 5.1.

5.1. Experimental settings

We have implemented all the bounds mentioned in this paper using SageMath [46], Cython [6], SciPy [50] (for computing the eigenvalue λ_2) and Cplex [32] (for solving (integer) linear programs). All reported computations have been performed on a computer equipped with a 3.70GHz Intel Core i9-10900K processor, 64GB of RAM, and with operating system Fedora 39. Recall that we denote by:

- $\mathrm{LB}_{\kappa'}$ the lowed bound based on the minimal edge-cuts (Ineq. (1)),
- LB_{λ_2} the lower bound based on the second smallest eigenvalue of the Laplacian matrix of the graph (Ineq. (2)),
- LB_{Δ} the bound based on the maximum degree of the graph (Ineq. (3)),
- LB_{δ^*} the bound based on the degeneracy of the graph (Ineq. (4)),
- LB_{δ} the bound based on the minimum degree of the graph (respectively Ineq. (5)),
- $\mathrm{LB}_{\mathrm{Ad}}$ the bound based on the average degree of the graph (Ineq. (6)),
- $\mathrm{LB}_{\mathrm{Mad}}$ the bound based on the maximum average degree (Ineq. (8)),
- $C^*(m,n)$, denoted shortly C^*, the bound based on grooming (Ineq. (9)),
- C^*_{Mad} the bound C^* computed on a subgraph of maximum average degree (Ineq. (11)).

Furthermore, we introduce two lower bounds obtained by using the method proposed in [14] for finding an approximation of the maximum average degree of a graph. Let $\sigma_{\delta^*} = (v_1, v_2, \ldots, v_n)$ be the elimination ordering of the vertices produced when computing the degeneracy $\delta^*(G)$ of a graph, and let $\overline{X_i} = (v_i, v_{i+1}, \ldots, v_n)$ for $1 \leq i \leq n$ and $G[\overline{X_i}]$ be the subgraph of G induced by $\overline{X_i}$. We have:

$$\mathrm{cw}(G) \geq C^*_g(m,n) = \max\left\{C^*(|E_i|,|V_i|) \mid G[\overline{X_i}] = (V_i,E_i), 1 \leq i < n\right\} \quad (12)$$

This bound can be computed in $\mathcal{O}(m + n\log n)$ time as the elimination ordering of the vertices, and so the degeneracy δ^* of a graph, can be computed

in $\mathcal{O}(n+m)$ time and we call the function C^* after each elimination of a vertex.

Finally, we denote by C^*_{g++} the maximum value obtained when computing C^* on each subgraph created during the execution of the greedy++ algorithm proposed in [13] to obtain a 2-approximation of the maximum average degree. Roughly, this algorithm performs $\log n$ times the procedure for finding the elimination ordering of the degeneracy, but it changes at each iteration the initial weight of the vertices, and so the priorities in the elimination ordering. Let $f^i(v)$ be the weight of vertex v at the beginning of iteration i (initially, $f^1(v) = d_G(v)$). At the beginning of iteration i, the algorithm makes a copy H of the graph G and feeds a min-heap data structure Q with the weights f^i of the vertices. Then it iteratively extracts the vertex v of minimum weight from Q, set $f^{i+1}(v) = f^i(v) + d_H(v)$, where $d_H(v)$ is the degree of vertex v in the remaining graph H, removes vertex v from H and reduces the weight of its neighbours in Q. When H is empty, it proceeds with iteration $i+1$. This algorithm has time complexity in $\mathcal{O}((m + n\log n)\log n)$ when using a min-heap data structure and running $\log n$ iterations of its main loop, and it performs overall $n \log n$ calls to C^* (one per extraction of a vertex from Q).

5.2. Analysis of the bounds on graphs H^i_k and $G(s,n)$

In this section, we analyze the evolution of the lower bounds on two families of graphs, defined below, for which previously known lower bounds behave very differently. For one of them, H^i_k, LB_{δ^*} behaves very well while LB_{λ_2} gives very low bounds. For the other family of graphs, $G(s,n)$, the bound LB_{λ_2} performs much better than the bound LB_{δ^*}.

Let H^i_k denote the graph obtained by identifying one vertex of the clique K_k of order k with an extremity of the path P_i of order i. Hence, H^0_k is the clique K_k, H^1_k is a clique plus a pending edge, H^2_k is a clique with a pending path of length 2, and so on. The graph H^i_k has $k+i$ vertices, $k(k-1)/2+i$ edges, diameter $i+1$, and cutwidth $\mathrm{cw}(H^i_k) = \mathrm{cw}(K_k) = \lceil k^2/4 \rceil$. We have reported in Table 1 the evolution of the lower bounds for H^i_{50} when $i = 0 \dots 6$. We note that $\mathrm{cw}(H^i_{50}) = 625$ for all i.

A first observation is that LB_{λ_2} decreases rapidly to 1 when i increases. Indeed, $\lambda_2(G)$ is upper bounded by the edge-connectivity of the graph, and so by its minimum degree $\delta(G)$. Since we have that $\delta(H^i_k) = k$ for $i = 0$ and 1 when $i \geq 1$, it follows that $\lambda_2(H^i_k) \leq 1$ when $i \geq 1$.

Another observation is that bounds that search for a dense subgraph ($\mathrm{LB}_{\mathrm{Mad}}$, LB_{δ^*}, C^*_{Mad}, C^*_g, C^*_{g++}) are independent from i. Indeed, we have

for all $i \geq 0$ that $\mathrm{LB}_{\mathrm{Mad}}(H_k^i) = \mathrm{LB}_{\mathrm{Mad}}(K_k) = \mathrm{LB}_{\mathrm{Ad}}(K_k) = k(k+2)/8$ and $C^*_{\mathrm{Mad}}(H_k^i) = C^*(k(k-1)/2, k) = (k^2 - \varepsilon)/4$, where $\varepsilon = 1$ when k is even and 0 otherwise.

Furthermore, we observe as expected that the bound C^* decreases with i. However, it is better than many other bounds that require much more computation time (LB_{λ_2}, $\mathrm{LB}_{\kappa'}$, LB_{Δ},LB_{δ}, $\mathrm{LB}_{\mathrm{Mad}}$). Recall that C^* can be computed in $\mathcal{O}(\log n)$ time and, as can be seen in Table 1b, can be computed orders of magnitude faster than other bounds (except $\mathrm{LB}_{\mathrm{Ad}}$ which can be computed in $\mathcal{O}(1)$ time).

	Previous bounds				Our bounds						
Graph	LB_{λ_2}	$\mathrm{LB}_{\kappa'}$	LB_{Δ}	LB_{δ^*}	LB_{δ}	$\mathrm{LB}_{\mathrm{Ad}}$	$\mathrm{LB}_{\mathrm{Mad}}$	C^*	C^*_{Mad}	C^*_g	C^*_{g++}
K_{50}	**625**	49	25	**625**	**625**	325	325	**625**	**625**	**625**	**625**
H^1_{50}	13	49	25	**625**	1	313	325	601	**625**	**625**	**625**
H^2_{50}	6	49	25	**625**	1	302	325	577	**625**	**625**	**625**
H^3_{50}	3	49	25	**625**	1	291	325	552	**625**	**625**	**625**
H^4_{50}	2	49	25	**625**	1	281	325	527	**625**	**625**	**625**
H^5_{50}	2	49	25	**625**	1	271	325	501	**625**	**625**	**625**
H^6_{50}	1	49	25	**625**	1	262	325	475	**625**	**625**	**625**

(a) Evolution of the lower bounds for the graphs H^i_{50}, $i = 0 \cdots 6$, with $\mathrm{cw}(H^i_{50}) = 625$. Best found lower bounds are highlighted in bold.

	Previous bounds				Our bounds						
Graph	LB_{λ_2}	$\mathrm{LB}_{\kappa'}$	LB_{Δ}	LB_{δ^*}	LB_{δ}	$\mathrm{LB}_{\mathrm{Ad}}$	$\mathrm{LB}_{\mathrm{Mad}}$	C^*	C^*_{Mad}	C^*_g	C^*_{g++}
K_{50}	2.66	323.03	0.021	0.365	0.018	0.0035	28.62	0.0074	27.69	0.488	4.00
H^1_{50}	2.36	320.39	0.021	0.370	0.018	0.0033	27.71	0.0057	27.88	0.484	4.04
H^2_{50}	2.34	322.46	0.012	0.359	0.019	0.0034	27.73	0.0057	28.21	0.483	4.07
H^3_{50}	2.39	330.51	0.021	0.353	0.018	0.0070	28.94	0.0060	29.25	0.488	4.18
H^4_{50}	2.48	331.72	0.021	0.358	0.019	0.0034	28.31	0.0060	27.95	0.497	4.05
H^5_{50}	2.45	322.55	0.021	0.358	0.018	0.0032	28.11	0.0055	28.27	0.498	4.08
H^6_{50}	2.44	323.90	0.021	0.362	0.019	0.0034	27.90	0.0054	28.06	0.512	4.20

(b) Running times in milliseconds (ms) for computing the bounds reported in Table 1a.

Table 1: Evolution of the lower bounds (Table 1a) for the graphs H^i_{50}, $i = 0 \cdots 6$, with $\mathrm{cw}(H^i_{50}) = 625$. The running times are reported in Table 1b.

Let us now consider the graph $G(s, n)$ with n vertices, defined in Section 3, where vertex v_i is joined to the s vertices v_{i+h}, with $1 \leq h \leq s$ and $i + h \leq n$. This graph has $m = sn - s(s+1)/2$ edges, and by construction, we have that $\mathrm{cw}(G(s, n)) = s(s+1)/2$ when $n \geq 2s+1$. We have reported in Table 2

the evolution of the lower bounds for $G(s,n)$ when $s = 1, 5, 10, 15, 20$ and $n = 2s + 1$.

Recall that we have seen in Section 3 that $\mathrm{LB}_{\mathrm{Ad}}$ (the bound of Ineq.(6)) has a large value than LB_{δ^*} (the bound of Ineq. (4)) for $G(s,n)$ when n is large enough. This is confirmed for $n = 2s + 1$. Furthermore, we observe in Table 2 that the bounds based on grooming perform much better than all other bounds on these graphs, and in fact, we have $C^*(m,n) = \mathrm{cw}(G(s,n))$ when $n \geq 2s + 1$. Furthermore, these bounds can be computed efficiently, in particular C^*, as can be seen from Table 2b.

	Previous bounds				Our bounds						
$G(s,n)$	LB_{λ_2}	$\mathrm{LB}_{\kappa'}$	LB_{Δ}	LB_{δ^*}	LB_{δ}	$\mathrm{LB}_{\mathrm{Ad}}$	$\mathrm{LB}_{\mathrm{Mad}}$	C^*	C^*_{Mad}	C^*_g	C^*_{g++}
$G(1,3)$	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
$G(5,11)$	10	9	5	9	9	10	11	**15**	**15**	**15**	**15**
$G(10,21)$	34	19	10	30	30	34	34	**55**	**55**	**55**	**55**
$G(15,31)$	74	29	15	64	64	72	72	**120**	**120**	**120**	**120**
$G(20,41)$	130	39	20	110	110	124	124	**210**	**210**	**210**	**210**

(a) Evolution of the lower bounds for the graphs $G(s,n)$ for $s = 1, 5, 10, 15, 20$ and $n = 2s + 1$. Best found lower bounds are highlighted in bold.

	Previous bounds				Our bounds						
$G(s,n)$	LB_{λ_2}	$\mathrm{LB}_{\kappa'}$	LB_{Δ}	LB_{δ^*}	LB_{δ}	$\mathrm{LB}_{\mathrm{Ad}}$	$\mathrm{LB}_{\mathrm{Mad}}$	C^*	C^*_{Mad}	C^*_g	C^*_{g++}
$G(1,3)$	0.91	0.53	0.010	0.024	0.007	0.0028	0.48	0.0058	0.34	0.028	0.04
$G(5,11)$	0.79	5.29	0.012	0.048	0.009	0.0030	1.32	0.0048	1.29	0.065	0.13
$G(10,21)$	1.00	24.29	0.014	0.112	0.011	0.0030	3.78	0.0051	3.77	0.142	0.50
$G(15,31)$	1.36	67.99	0.016	0.230	0.014	0.0031	7.94	0.0052	7.87	0.242	1.07
$G(20,41)$	1.81	149.03	0.019	0.328	0.017	0.0035	14.32	0.0055	14.30	0.387	2.13

(b) Running times in milliseconds (ms) for computing the bounds reported in Table 2a.

Table 2: Evolution of the lower bounds (Table 2a) for the graphs $G(s,n)$ for $s = 1, 5, 10, 15, 20$ and $n = 2s + 1$. The running times are reported in Table 1b.

5.3. Erdős-Rényi random graphs

In this section, we compare the different bounds presented in this paper on Erdős-Rényi random graphs [25] (See e.g., [11]). We start with graphs of order $n = 30$ for which we can also compute the optimal cutwidth (cw). We then consider graphs of order $n = 50$ for which the optimal value is already unreachable.

For $n = 30, 50$ and for each density in $p \in [0.05, 0.1, \ldots, 0.95]$ (or equivalently for each probability $p \in [0.05, 0.1, \ldots, 0.95]$ for the existence of an edge), we have generated 100 Erdős-Rényi random graphs, ensuring that all graphs have the same number $pn(n-1)/2$ of edges. Observe that the *density* of a graph G is measured as $2m/n(n-1)$, and so is different from its average degree $\mathrm{Ad}(G) = 2m/n$. We have then computed all the bounds and reported in Figures 2 and 3 the average of the computed bounds and running times.

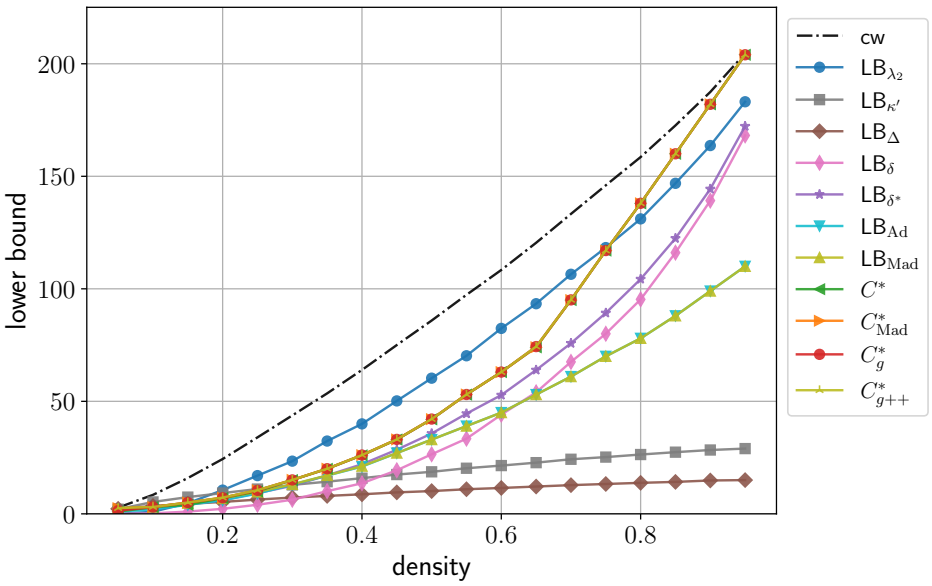
In Figure 2a, we observe that LB_{λ_2} dominates all other lower bounds when $p \leq 0.75$. Recall that the connectivity threshold for Erdős-Rényi random graphs is $\log n/n$. That is, if $p = p_0 \log n/n$, the graph is asymptotically almost surely connected when $p_0 > 1$, and disconnected when $p_0 < 1$. Hence, when $p_0 < 1$, we have asymptotically almost surely $\lambda_2 = 0$, and so $\mathrm{LB}_{\lambda_2}(G) = 0$. Since in our experiments, we have $n = 30$, and so $\log 30/30 \simeq 0.11$, most of the graphs we have tested are connected and exhibit the good properties of Erdős-Rényi random graphs for connectivity, small diameter, etc. Furthermore, it has been proved in [33] that when $p_0 > 1$ and for any $\varepsilon > 0$, we have $\lambda_2 = pn + o(n^{\frac{1}{2}+\varepsilon})$ in probability. In other words, the algebraic connectivity λ_2 of Erdős-Rényi random graphs increases with the (average) degree pn. It follows that LB_{λ_2} increases similarly to $pn^2/4$, and this is roughly what we observe in Figure 2a.

We then observe that all the bounds based on grooming (C^*, C^*_{Mad}, C^*_g, C^*_{g++}) dominate all other bounds when $p \geq 0.8$ and finally reach the optimal value of the cutwidth. Moreover, we observe that methods based on grooming provide the same bounds for these graphs. This is due to the properties of Erdős-Rényi random graphs which are such that a subgraph exhibit the same properties as the graph in terms of density, average degree, etc. We will see in Section 5.5 that the situation is different for other families of graphs.

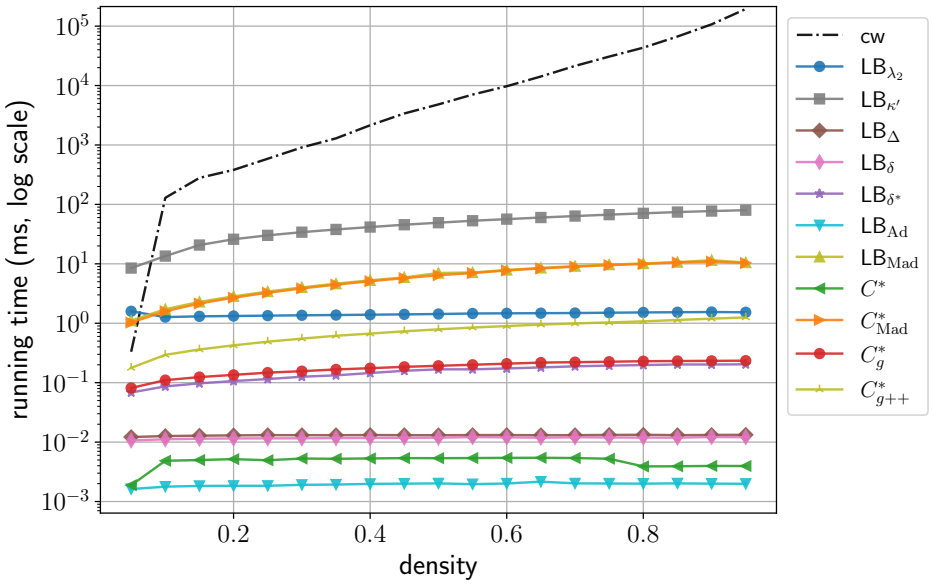
Non surprisingly, we observe that $\mathrm{LB}_{\kappa'}$ not only provides very small lower bounds compared to others, but also requires orders of magnitude more computation time. We also observe that the bounds $\mathrm{LB}_{\mathrm{Mad}}$ and C^*_{Mad}, both based on the maximum average degree, do not provide better bounds than $\mathrm{LB}_{\mathrm{Ad}}$ and C^* respectively, while requiring orders of magnitude more computation time. Hence, bounds relying on the resolution of linear programs seem of little interest for Erdős-Rényi random graphs.

Observe also that we experimentally confirm Proposition 12, that is that $C^*(m, n) \geq \mathrm{LB}_{\mathrm{Ad}}(m, n)$.

The running times reported in Figure 2b reflect the time complexity of



(a) Average lower bounds on the cutwidth of Erdős-Rényi random graphs with $n = 30$ vertices and $pn(n-1)/2$ edges for $p \in [0.05, 0.1, \ldots, 0.95]$. For each density p, we have reported the average values over 100 random graphs.



(b) Average running times (log scale). Average running times (log scale) for computing the lower bounds reported in Figure 2a.

Figure 2: Lower bounds and running times (log scale) on Erdős-Rényi random graphs ($n = 30$). For each density in $p \in [0.05, 0.1, \ldots, 0.95]$, we have generated 100 Erdős-Rényi random graphs with $pn(n-1)/2$ edges, and averaged the computed values and running times for each bound and for the exact cutwidth.

the methods: $\mathcal{O}(1)$ for $\mathrm{LB}_{\mathrm{Ad}}$; $\mathcal{O}(\log n)$ for C^*; $\mathcal{O}(n)$ for LB_{Δ} and LB_{δ}; $\mathcal{O}(m + n \log n)$ for LB_{δ^*} and C^*_g; $\mathcal{O}((m + n \log n) \log n)$ for C^*_{g++}; $\mathcal{O}(n^\omega)$ for LB_{λ_2}; and much more for $\mathrm{LB}_{\mathrm{Mad}}$, C^*_{Mad} and $\mathrm{LB}_{\kappa'}$. Moreover, we observe that the time needed to compute the cutwidth of the graphs increases exponentially with the density, and so with the number of edges. This is due to the way the dynamic programming algorithm is implemented in SageMath, as it first searches for the existence of a solution with width w before searching for a solution with width $w + 1$. Furthermore, it starts partitioning the graph into connected components as we have $\mathrm{cw}(G) = \max\{\mathrm{cw}(G[c]) \mid c \text{ is a connected component of } G\}$. This explains the behaviour for low densities since the graphs are not connected.

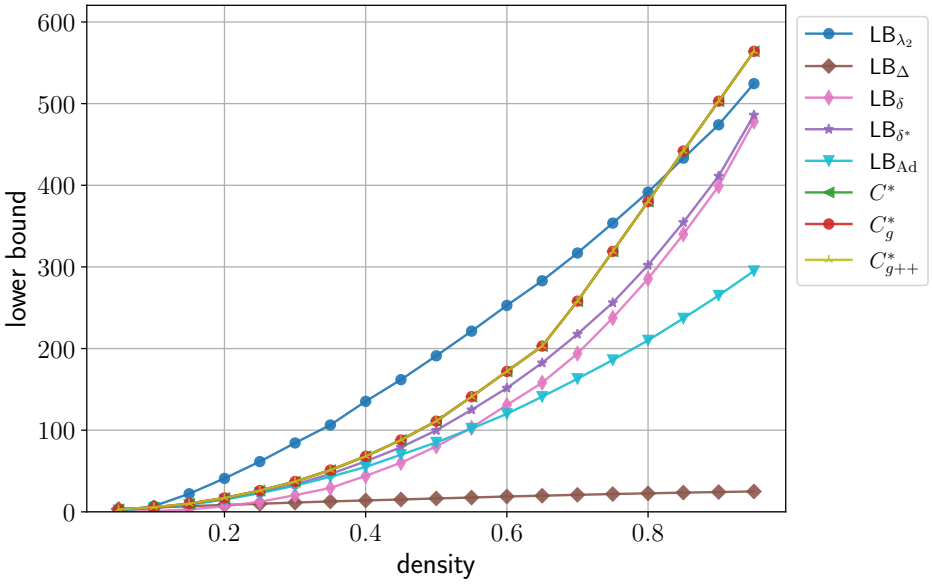
In Figure 3, we have reported for $n = 50$ the bounds on running times for the fastest methods only. We observe the same behaviours than in Figure 2 for the relative qualities of the bounds, except that now LB_{λ_2} dominates all other bounds when $p \leq 0.8$ instead of 0.75 for $n = 30$.

5.4. Interval graphs

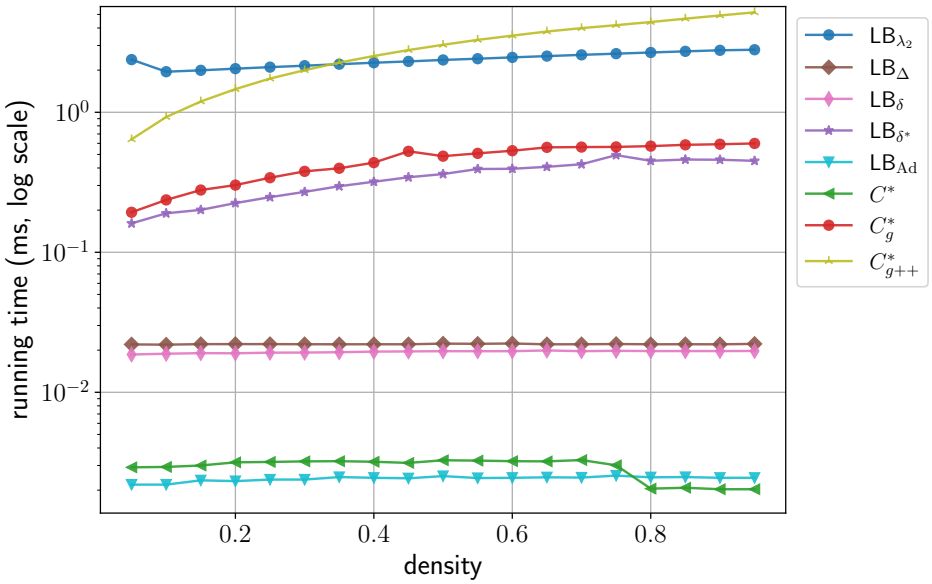
We now evaluate our bounds on random (proper) interval graphs [45]. Consider a family $\mathcal{S} = \{S_i \mid i = 1, 2, \ldots, n\}$ of intervals on the real line. An interval graph is obtained from $\mathcal{S}$ by creating a vertex v_i for each interval S_i and connecting two vertices v_i and v_j by an edge if the corresponding intervals intersect. An interval graph is *proper* if not interval in $\mathcal{S}$ contains another interval. That is, if l_i and r_i are the endpoints of interval S_i with $l_i < r_i$, and similarly $l_j < r_j$ are the extremities of S_j, we have either $l_j < l_i$ and $r_j < r_i$, or $l_j > l_i$ and $r_j > r_i$.

Using the uniform random generators of (proper) interval graphs available in SageMath, we generated 1 000 graphs of order $n = 100$ on which we have computed all the bounds. Then, we have sorted the graphs by increasing number of edges, group them by groups of 5 consecutive graphs, computed the average of the bounds for these groups of 5 graphs and reported the resulting values in Figure 4.

We observe very different results for random (proper) interval graphs than for Erdős-Rényi random graphs. In particular, the bound LB_{λ_2} which is very good for Erdős-Rényi random graphs provides poor bounds on (proper) interval graphs. Indeed, these graphs contain vertices with small degree, as highlighted by the very low values of LB_{δ}, and LB_{λ_2} suffers from the presence of vertices with small degree which have a strong impact on the value of the

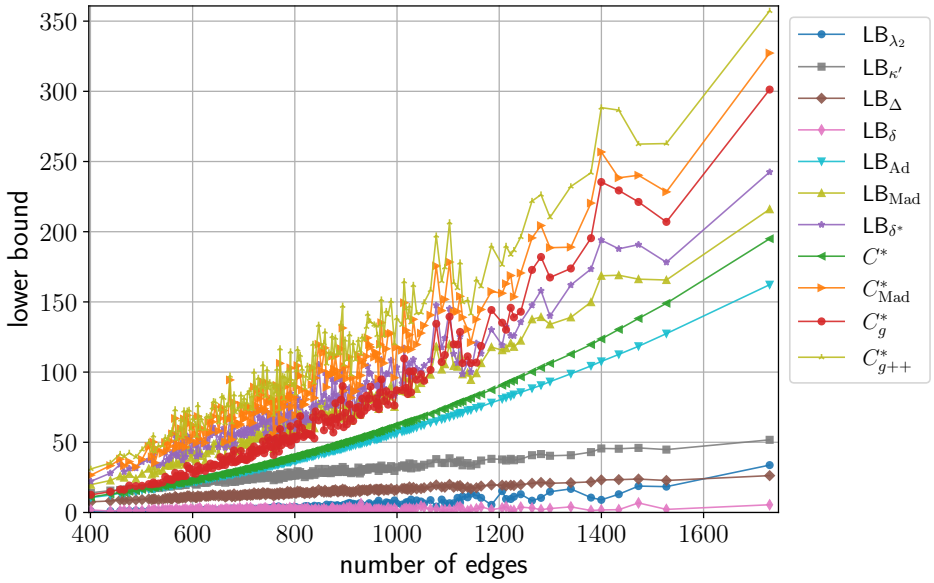


(a) Average lower bounds on the cutwidth of Erdős-Rényi random graphs with $n = 50$ vertices and $pn(n-1)/2$ edges for $p \in [0.05, 0.1, \ldots, 0.95]$. For each density p, we have reported the average values over 100 random graphs.

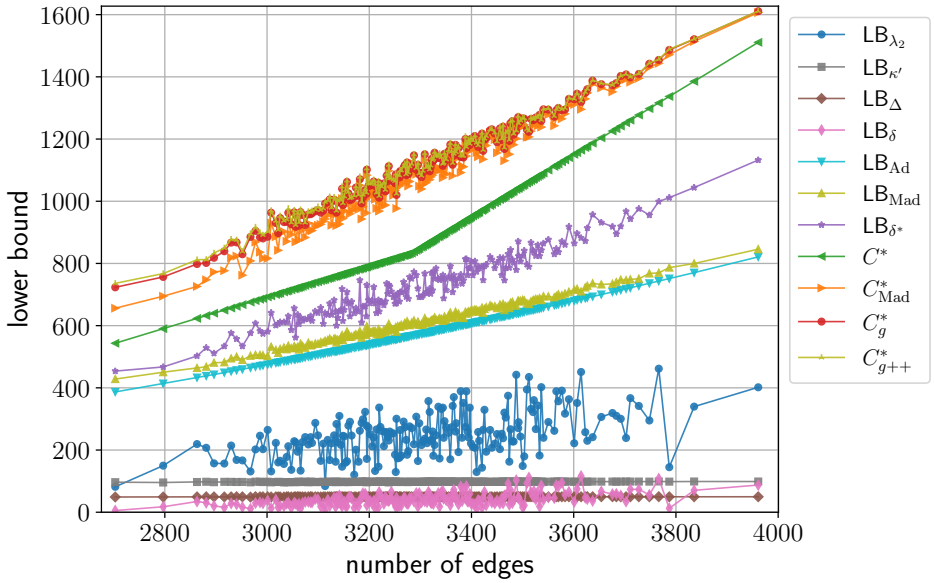


(b) Average running times (log scale). Average running times (log scale) for computing the lower bounds reported in Figure 3a.

Figure 3: Lower bounds and running times (log scale) on Erdős-Rényi graphs ($n = 50$). For each density in $p \in [0.05, 0.1, \ldots, 0.95]$, we have generated 100 Erdős-Rényi random graphs with $pn(n-1)/2$ edges, and averaged the computed values and running times for the lower bounds that are the fastest to compute.



(a) Lower bounds of random proper interval graphs with $n = 100$ nodes.



(b) Lower bounds of random interval graphs with $n = 100$ nodes.

Figure 4: Lower bounds for random proper interval graphs (Figure 4a) and random interval graphs (Figure 4b) with $n = 100$ nodes. The graphs are ordered by increasing number of edges and each dot represents the average value over 5 consecutive graphs.

algebraic connectivity, as explained in Section 5.2, that is upper bounded by the edge-connectivity and so by the minimum degree of the graph.

For proper interval graphs (Figure 4a), the best bounds are obtained using $\mathrm{LB}_{\mathrm{Mad}}$, LB_{δ^*}, C^*_{Mad}, C^*_g and C^*_{g++}, that is using the methods searching for a dense subgraph. The bound C^*_{g++} seems to offer the best trade-off between the quality of the bound and the time complexity for these graphs.

For interval graphs (Figure 4b), the best bounds are obtained using C^*_{Mad}, C^*_g and C^*_{g++}. Furthermore, we observe that C^* behaves very well while needing a low computation time. This can be explained by the fact that random interval graphs are generally very dense. Here, the bound C^*_g seems to offer the best trade-off between the quality of the bound and the time complexity for these graphs.

5.5. Comparisons with the random graphs used in [27]

We now compare the bounds presented in this paper with the lower bounds obtained in [27] using SDP relaxation. To do so, we have computed our bounds on the graphs used in [27]. These graphs are Erdős-Rényi random graphs and geometric random graphs (i.e., a set of vertices randomly placed on a square of side 1, and two vertices are connected by an edge if at euclidean distance less than d).

We have reported in Table 3 the best lower and upper bounds obtained in [27] on Erdős-Rényi random graphs, as well as the corresponding computation time (in seconds). We have also reported the best lower bound found using the methods presented in this paper with the list of methods reaching this bound and the required computation time (in milliseconds). The first two columns of the table corresponds to the number of nodes and the probability p of the existence of an edge. Table 4 presents similarly the results for the geometric random graphs.

For the Erdős-Rényi random graphs (Table 3), we first observe that the results for the bounds of this paper are consistent with the results presented in Section 5.3. More precisely, LB_{λ_2} gives very good bounds, but when the density of the graphs is very large, bounds based on grooming (C^*, C^*_{Mad}, C^*_g, C^*_{g++}) are much better. The main observation is that we obtain excellent lower bounds, and most of the time better bounds than those of [27], while the running times of the methods presented in this paper are several orders of magnitude smaller than those of [27].

For the random geometric graphs (Table 4), the methods based on grooming, and in particular C^*_{g++}, give the best lower bounds among all the methods

presented in this paper. Moreover, these bounds are most of the time better than the bounds of [27] and obtained up to 6 orders of magnitude faster.

Let us mention that an objective of [27] was to propose new methods for getting good lower bounds on dense graphs, but when $p \geq 0.5$, the bounds LB_{λ_2}, C^*, C^*_{Mad}, C^*_g and C^*_{g++} are better and can be computed more quickly.

6. Conclusion

In this article we have given new lower bounds for the cutwidth of a graph which are of interest to design faster branch-and-bound algorithms. In particular, we have given bounds using the notion of traffic grooming on a path, which in many cases appear to be better than bounds in the literature. Furthermore, bounds based on grooming can be computed quickly. This raises the question whether better bounds can be obtained using for instance the methods proposed in [27], using semidefinite programming, if given as input the bounds presented in this paper.

References

[1] Abboud, A., Krauthgamer, R., Li, J., Panigrahi, D., Saranurak, T., Trabelsi, O., 2022. Breaking the cubic barrier for all-pairs max-flow: Gomory-hu tree in nearly quadratic time, in: 63rd IEEE Annual Symposium on Foundations of Computer Science (FOCS), IEEE. pp. 884–895. doi:10.1109/FOCS54457.2022.00088.

[2] Adamy, U., Ambühl, C., Anand, R.S., Erlebach, T., 2007. Call control in rings. Algorithmica 47, 217–238. doi:10.1007/s00453-006-0187-4.

[3] Adolphson, D.L., Hu, T.C., 1973. Optimal linear ordering. SIAM Journal on Applied Mathematics 25, 403–423. doi:10.1137/0125042.

[4] Bansal, N., Katzelnick, D., Schwartz, R., 2024. On approximating cutwidth and pathwidth, in: 65th IEEE Annual Symposium on Foundations of Computer Science (FOCS), IEEE. pp. 713–729. doi:10.1109/FOCS61266.2024.00051.

[5] Barth, D., Pellegrini, F., Raspaud, A., Roman, J., 1995. On bandwidth, cutwidth, and quotient graphs. RAIRO-Theoretical Informatics and Applications 29, 487–508. doi:10.1051/ita/1995290604871.

			Results from [27]			This paper		
n	m	p	UB	LB	Time (sec)	Best LB	Algorithms	Time (ms)
20	60	0.3	20	**14.27**	62.62	12	LB_{λ_2}	1.3940
20	81	0.4	32	**21.54**	61.83	17	LB_{λ_2}	1.3888
20	85	0.5	31	**21.92**	63.94	16	LB_{δ^*}, C^*_{Mad}, C^*_g, C^*_{g++}	[0.1760, 3.1173]
20	124	0.6	52	36.40	63.38	**39**	LB_{λ_2}	1.4482
20	130	0.7	56	38.65	63.59	**41**	C^*_g, C^*_{g++}	[0.2275, 0.6764]
20	149	0.8	68	46.59	62.43	**59**	C^*, C^*_{Mad}, C^*_g, C^*_{g++}	[0.0172, 5.1172]
20	177	0.9	88	59.47	59.05	**87**	C^*, C^*_{Mad}, C^*_g, C^*_{g++}	[0.0184, 5.3825]
30	131	0.3	44	**30.69**	365.83	25	LB_{λ_2}	1.7538
30	166	0.4	60	**41.79**	370.88	40	LB_{λ_2}	1.4968
30	222	0.5	87	60.55	376.59	**69**	LB_{λ_2}	1.7805
30	253	0.6	101	70.84	400.35	**76**	LB_{λ_2}	1.5504
30	305	0.7	135	91.86	379.24	**96**	C^*_g, C^*_{g++}	[0.2944, 1.4577]
30	353	0.8	161	110.58	401.19	**143**	C^*, C^*_{Mad}, C^*_g, C^*_{g++}	[0.0196, 12.4459]
30	376	0.9	176	119.93	392.83	**166**	C^*, C^*_{Mad}, C^*_g, C^*_{g++}	[0.0193, 12.7959]
40	246	0.3	92	**60.12**	1 505.80	46	LB_{λ_2}	2.0754
40	325	0.4	126	**85.52**	1 597.97	81	LB_{λ_2}	1.9107
40	365	0.5	155	**99.75**	1 495.82	95	LB_{λ_2}	2.0809
40	458	0.6	195	130.84	1 593.94	**149**	LB_{λ_2}	2.2204
40	524	0.7	235	115.30	1 691.67	**194**	LB_{λ_2}	2.0056
40	610	0.8	281	187.95	1 723.79	**230**	C^*, C^*_{Mad}, C^*_g, C^*_{g++}	[0.0212, 21.0202]
40	704	0.9	346	227.74	1 703.14	**324**	C^*, C^*_{Mad}, C^*_g, C^*_{g++}	[0.0567, 24.4861]
50	372	0.3	159	**91.92**	5 310.93	85	LB_{λ_2}	95.2239
50	506	0.4	211	133.61	5 404.70	**156**	LB_{λ_2}	2.3170
50	647	0.5	286	182.44	5 313.71	**198**	LB_{λ_2}	2.4786
50	754	0.6	338	221.14	5 511.51	**271**	LB_{λ_2}	2.7552
50	850	0.7	382	253.98	5 552.34	**309**	LB_{λ_2}	2.7530
50	998	0.8	468	312.20	6 303.39	**398**	C^*, C^*_{Mad}, C^*_g, C^*_{g++}	[0.0403, 33.3912]
50	1 092	0.9	525	350.87	6 033.16	**492**	C^*, C^*_{Mad}, C^*_g, C^*_{g++}	[0.0200, 36.4516]

Table 3: Comparison of the lower bounds based on SDP relaxation obtained in [27] with the bounds obtained using the methods presented in this paper on the Erdős-Rényi random graphs used in [27]. We report the bounds and the running time (in seconds) from [27]. We report the best lower bound that can be obtained using the methods presented in this paper, list the algorithms reaching this bound and report the range of running times (in milliseconds) of these algorithms (algorithm C^* is always the fastest and C^*_{Mad} the slowest). A single running time is reported when a single algorithm reaches the bound.

			Results from [27]			This paper		
n	m	d	UB	LB	Time (sec)	Best LB	Algorithms	Time (ms)
20	34	0.3	7	5.24	42.55	**7**	C^*_{Mad}, C^*_{g++}	$[0.3238, 1.6108]$
20	54	0.4	12	8.28	42.69	**12**	LB_{δ^*}, C^*_{Mad}, C^*_{g++}	$[0.1600, 2.1939]$
20	80	0.5	22	**16.50**	61.18	15	C^*_{g++}	0.4959
20	84	0.6	21	16.56	57.48	**18**	C^*_{g++}	0.4826
20	137	0.7	55	39.83	58.37	**53**	C^*_{Mad}, C^*_g, C^*_{g++}	$[0.2458, 4.7052]$
20	173	0.8	83	56.75	62.43	**83**	C^*, C^*_{Mad}, C^*_g, C^*_{g++}	$[0.0191, 5.2438]$
20	158	0.9	69	48.57	59.91	**68**	C^*, C^*_{Mad}, C^*_g, C^*_{g++}	$[0.0172, 5.1551]$
30	83	0.3	13	10.46	293.96	**13**	C^*_{g++}	0.6435
30	154	0.4	45	**32.72**	337.93	30	C^*_{g++}	0.9339
30	210	0.5	72	51.65	346.9	**67**	C^*_{Mad}, C^*_{g++}	$[1.1430, 7.3059]$
30	327	0.6	126	92.20	373.08	**123**	C^*_g, C^*_{g++}	$[0.3982, 1.5669]$
30	304	0.7	119	84.98	395.81	**97**	C^*_{g++}	1.4546
30	408	0.8	200	134.97	391.51	**198**	C^*, C^*_{Mad}, C^*_g, C^*_{g++}	$[0.0207, 12.2125]$
30	417	0.9	207	139.47	381.37	**207**	C^*, C^*_{Mad}, C^*_g, C^*_{g++}	$[0.0200, 12.1496]$
40	167	0.3	31	**22.48**	1 253.51	22	C^*_{g++}	1.3340
40	243	0.4	50	**39.98**	1 389.39	35	C^*_{g++}	1.7180
40	340	0.5	92	**69.41**	1 353.95	58	C^*_{g++}	2.1372
40	431	0.6	126	97.37	1 416.42	**107**	C^*_{g++}	2.4588
40	576	0.7	238	165.83	1 650.37	**205**	C^*_{g++}	3.1359
40	654	0.8	294	201.33	1 669.65	**274**	C^*, C^*_{Mad}, C^*_g, C^*_{g++}	$[0.0205, 20.6606]$
40	742	0.9	366	245.12	1 758.20	**362**	C^*, C^*_{Mad}, C^*_g, C^*_{g++}	$[0.0196, 21.9421]$
50	314	0.3	63	48.00	3 534.43	**57**	C^*_{g++}	2.2986
50	467	0.4	131	**96.25**	4 139.52	94	C^*_{g++}	3.0117
50	580	0.5	175	**127.56**	4 433.95	121	C^*_{g++}	3.4204
50	749	0.6	266	**189.41**	4 604.08	182	C^*_{g++}	4.2415
50	936	0.7	398	272.53	5 064.41	**347**	C^*_g, C^*_{g++}	$[0.8366, 5.0154]$
50	963	0.8	416	283.43	5 512.39	**368**	C^*_g, C^*_{g++}	$[0.8130, 5.0299]$
50	1 136	0.9	556	368.44	6 983.07	**536**	C^*, C^*_{Mad}, C^*_g, C^*_{g++}	$[0.0226, 35.0571]$

Table 4: Comparison of the lower bounds based on SDP relaxation obtained in [27] with the bounds obtained using the methods presented in this paper on random geometric graphs. We report the bounds and the running time (in seconds) from [27]. We report the best lower bound that can be obtained using the methods presented in this paper, list the algorithms reaching this bound and report the range of running times of these algorithms (in milliseconds). A single running time is reported when a single algorithm reaches the bound.

[6] Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D.S., Smith, K., 2011. Cython: The best of both worlds. Computing in Science Engineering 13, 31 –39. doi:10.1109/MCSE.2010.118.

[7] Bermond, J.C., Braud, L., Coudert, D., 2007. Traffic Grooming on the Path. Theoretical Computer Science 384, 139–151. doi:10.1016/j.tcs.2007.04.028.

[8] Bermond, J.C., Cosnard, M., Coudert, D., Havet, F., 2023. Groupage sur le chemin pour borner la largeur de coupe, in: AlgoTel 2023 - 25èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, Cargese, France. p. 4. URL: https://inria.hal.science/hal-04076999.

[9] Bermond, J.C., Cosnard, M., Coudert, D., Perennes, S., 2024. Maximum number of requests on a path with a given grooming factor. Technical Report. hal-04736088. URL: https://inria.hal.science/hal-04736088. a preliminary version was presented at the Advanced International Conference on Telecommunications (AICT), Le Gosier, Guadeloupe, France, 2006, IEEE.

[10] Bodlaender, H.L., Fomin, F.V., Koster, A.M.C.A., Kratsch, D., Thilikos, D.M., 2012. A note on exact algorithms for vertex ordering problems on graphs. Theory of Computing Systems 50, 420–432. doi:10.1007/S00224-011-9312-0.

[11] Bollobás, B., 2001. Models of Random Graphs. Cambridge University Press. Cambridge Studies in Advanced Mathematics, p. 34–59. doi:10.1017/CBO9780511814068.

[12] Bondy, J.A., Murty, U.S.R., 2008. Graph Theory. Graduate Texts in Mathematics. 1st ed., Springer. URL: https://link.springer.com/book/9781846289699.

[13] Boob, D., Gao, Y., Peng, R., Sawlani, S., Tsourakakis, C.E., Wang, D., Wang, J., 2020. Flowless: Extracting densest subgraphs without flow computations, in: The Web Conference (WWW), ACM / IW3C2. pp. 573–583. doi:10.1145/3366423.3380140.

[14] Charikar, M., 2000. Greedy approximation algorithms for finding dense components in a graph, in: 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX), Springer. pp. 84–95. doi:10.1007/3-540-44436-X_10.

[15] Chekuri, C., Quanrud, K., Torres, M.R., 2022. Densest subgraph: Supermodularity, iterative peeling, and flow, in: ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM. pp. 1531–1555. doi:10.1137/1.9781611977073.6.

[16] Chen, L., Kyng, R., Liu, Y.P., Peng, R., Probst Gutenberg, M., Sachdeva, S., 2022. Maximum flow and minimum-cost flow in almost-linear time, in: 63rd IEEE Annual Symposium on Foundations of Computer Science (FOCS), IEEE. pp. 612–623. doi:10.1109/FOCS54457.2022.00064.

[17] Cohen, N., 2011. Three years of graphs and music : some results in graph theory and its applications. Ph.D. thesis. University of Nice Sophia Antipolis, France. URL: https://theses.fr/2011NICE4043.

[18] Coudert, D., 2016. A note on Integer Linear Programming formulations for linear ordering problems on graphs. Research Report hal-01271838. Inria ; I3S ; Universite Nice Sophia Antipolis ; CNRS. URL: https://inria.hal.science/hal-01271838.

[19] Cuthill, E., McKee, J., 1969. Reducing the bandwidth of sparse symmetric matrices, in: ACM'69: Proceedings of the 1969 24th national conference, pp. 157–172. doi:10.1145/800195.805928.

[20] Dewdney, A.K., 1976. The bandwidth of a graph: Some recent results, in: Proceedings of the Seventh Southeastern Conference on Combinatorics, Graph Theory, and Computing, pp. 273–288.

[21] Díaz, J., Petit, J., Serna, M., 2002. A survey on graph layout problems. ACM Computing Surveys 34, 313–356. doi:10.1145/568522.568523.

[22] Díaz, J., Serna, M., Spirakis, P., Torán, J., 1997. Paradigms for fast parallel approximability. Cambridge University Press. doi:10.1017/CBO9780511666407.

[23] Diks, K., Djidjev, H.N., Sykora, O., Vrťo, I., 1993. Edge separators for planar graphs and their applications. Journal of Algorithms 14, 258–279. doi:10.1006/jagm.1993.1013.

[24] Elsner, U., 1997. Graph Partitioning: A Survey. Preprintreihe des Chemnitzer SFB 393/97-27, Technishe Universität Chemnitz. URL: https://www.tu-chemnitz.de/sfb393/Files/PDF/sfb97-27.pdf.

[25] Erdős, P., Rényi, A., 1959. On random graphs I. Publicationes Mathematicae 6, 290–297. doi:10.5486/PMD.1959.6.3-4.12.

[26] Fiedler, M., 1973. Algebraic connectivity of graphs. Czechoslovak Mathematical Journal 23, 298–305. URL: http://eudml.org/doc/12723.

[27] Gaar, E., Puges, D., Wiegele, A., 2024. Strong SDP based bounds on the cutwidth of a graph. Computers & Operations Research 161, 106449. doi:10.1016/j.cor.2023.106449.

[28] Gallo, G., Grigoriadis, M.D., Tarjan, R.E., 1989. A fast parametric maximum flow algorithm and applications. SIAM Journal on Computing 18, 30–55. doi:10.1137/0218003.

[29] Giannopoulou, A.C., Pilipczuk, M., Raymond, J.F., Thilikos, D.M., Wrochna, M., 2019. Cutwidth: Obstructions and algorithmic aspects. Algorithmica 81, 557–588. doi:10.1007/s00453-018-0424-7.

[30] Goldberg, A.V., 1984. Finding a maximum density subgraph. University of California Berkeley URL: https://www2.eecs.berkeley.edu/Pubs/TechRpts/1984/CSD-84-171.pdf.

[31] Gomory, R.E., Hu, T.C., 1961. Multi-terminal network flows. Journal of the Society for Industrial and Applied Mathematics 9, 551–570. doi:10.1137/0109047.

[32] IBM ILOG, 2022. CPLEX Optimization Studio 22.1.1.

[33] Juhász, F., 1991. The asymptotic behaviour of Fiedler's algebraic connectivity for random graphs. Discrete Mathematics 96, 59–63. doi:10.1016/0012-365X(91)90470-M.

[34] Juvan, M., Mohar, B., 1992. Optimal linear labelings and eigenvalues of graphs. Discrete Applied Mathematics 36, 153–168. doi:10.1016/0166-218X(92)90229-4.

[35] Kayaaslan, E., Lambert, T., Marchal, L., Uçar, B., 2018. Scheduling series-parallel task graphs to minimize peak memory. Theoretical Computer Science 707, 1–23. doi:10.1016/j.tcs.2017.09.037.

[36] Khuller, S., Saha, B., 2009. On finding dense subgraphs, in: 36th International Colloquium on Automata, Languages and Programming (ICALP), Springer. pp. 597–608. doi:10.1007/978-3-642-02927-1_50.

[37] Kloeckner, B.R., 2010. Cutwidth and degeneracy of graphs. arXiv 0907.5138. doi:10.48550/ARXIV.0907.5138.

[38] Lehoucq, R.B., Sorensen, D.C., Yang, C., 1998. ARPACK users' guide - solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods. Software, environments, tools, SIAM. doi:10.1137/1.9780898719628.

[39] Martí, R., Pantrigo, J.J., Duarte, A., Pardo, E.G., 2013. Branch and bound for the cutwidth minimization problem. Computers & Operations Research 40, 137–149. doi:10.1016/j.cor.2012.05.016.

[40] Matula, D.W., Beck, L.L., 1983. Smallest-last ordering and clustering and graph coloring algorithms. Journal of the Association for Computing Machinery 30, 417–427. doi:10.1145/2402.322385.

[41] Mohar, B., 1991. The laplacian spectrum of graphs, in: Graph Theory, Combinatorics, and Applications, Vol. 2. Proceedings of the sixth quadrennial international conference on the theory and applications of graphs held at Western Michigan University, Kalamazoo, MI, USA, May 30-June 3, 1988, Wiley. pp. 871–898. URL: https://users.fmf.uni-lj.si/mohar/Papers/Spec.pdf.

[42] Nešetřil, J., De Mendez, P.O., 2012. Sparsity: graphs, structures, and algorithms. volume 28. Springer Science & Business Media. doi:10.1007/978-3-642-27875-4.

[43] Palubeckis, G., Rubliauskas, D., 2012. A branch-and-bound algorithm for the minimum cut linear arrangement problem. Journal of Combinatorial Optimization 24, 540–563. doi:10.1007/S10878-011-9406-2.

[44] Pan, V.Y., Chen, Z.Q., 1999. The complexity of the matrix eigenproblem, in: Annual ACM Symposium on Theory of Computing (STOC), ACM. pp. 507–516. doi:10.1145/301250.301389.

[45] Scheinerman, E.R., 1988. Random interval graphs. Combinatorica 8, 357–371. URL: http://dx.doi.org/10.1007/BF02189092, doi:10.1007/bf02189092.

[46] The Sage Developers, 2024. SageMath, the Sage Mathematics Software System (Version 10.5). doi:10.5281/zenodo.8042260. https://www.sagemath.org.

[47] Thilikos, D.M., Serna, M.J., Bodlaender, H.L., 2005. Cutwidth I: A linear time fixed parameter algorithm. Journal of Algorithms 56, 1–24. doi:10.1016/J.JALGOR.2004.12.001.

[48] Tsourakakis, C.E., Bonchi, F., Gionis, A., Gullo, F., Tsiarli, M.A., 2013. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees, in: The 19th ACM SIGKDD International Conference on Knowledge Discovery (KDD), ACM. pp. 104–112. doi:10.1145/2487575.2487645.

[49] Vassilevska Williams, V., Xu, Y., Xu, Z., Zhou, R., 2024. New bounds for matrix multiplication: from alpha to omega, in: ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM. pp. 3792–3835. doi:10.1137/1.9781611977912.134.

[50] Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors, 2020. SciPy 1.0: Fundamental Algorithms

for Scientific Computing in Python. Nature Methods 17, 261–272. doi:10.1038/s41592-019-0686-2.

[51] Wu, Y., Austrin, P., Pitassi, T., Liu, D., 2014. Inapproximability of treewidth, one-shot pebbling, and related layout problems. Journal of Artificial Intelligence Research 49, 569–600. doi:10.1613/jair.4030. MR3195329.