New lower bounds on the cutwidth of graphs

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

New lower bounds on the cutwidth of graphs*

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

Université Côte d'Azur, Inria, CNRS, I3S, France

November 8, 2024

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. In particular, we give bounds 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 bounds based on grooming can be computed quickly and so are of interest to design faster branch-and-bound algorithms.

Keywords: Cutwidth; lower bounds; traffic grooming; density

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 [19] 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 $\mathrm{cw}(G) = \min\{w(G, \sigma) \mid \sigma \text{ layout of } G\}$.

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, 21]. 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 [20]. More recent applications of this problem include network reliability [18], automatic graph drawing [22], information retrieval [5] and as a subroutine for the cutting plane algorithm to solve the traveling salesman problem [17]. 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

*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-01. A preliminary version of this paper appears in [8].

corresponding graph is expected to be small. However, computing the cutwidth of graphs is NP-hard [19] in general. 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 [26, 41] 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 [35] 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. The quality of these bounds have 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 [24], 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 [40] 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 [16], the branch-and-bound algorithm proposed in [35] or the methods based on semidefinite programming proposed in [24]. However, these methods usually require hours of computations for graphs with 50 vertices.

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 maximum, minimum or maximum average degree of G (Section 3). Note that all these parameters can be computed in polynomial time. Moreover, the bounds are mostly incomparable and hence are complementary. 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 time $\mathcal{O}(\log n)$. Futhermore, 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 behaviors on Erdős-Rényi random graphs and on random (proper) interval graphs.

Notations. All the graphs considered in this paper are finite, undirected, unweighted and simple (without loops nor multiple edges). The graph $G = (V, E)$ has $n = |V|$ vertices and $m = |E|$ edges. Let $N(v)$ denotes the set of neighbors of $v \in V$, that is $N(v) = \{u \mid uv \in E\}$. Let $d(v) = |N(v)|$ denotes the degree of vertex $v \in V$, and let $\delta(G)$ and $\Delta(G)$ denote respectively the minimum and maximum degree of G.

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 [19, 31, 33, 35].

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 [19] for more details). So, we get:

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

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 [28] of the graph, which uses the algorithm proposed in [14] 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 [31]:

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

The time complexity of this bound is due to the computation of λ_2, which requires $\mathcal{O}(n^\omega)$ time [39], with $\omega \leq 2.371552$ [43], but fast numerical methods can be used to compute eigenvalues in practice [34].

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., $\lambda_2 \leq \nu(G) \leq \kappa(G)$ [23]. Furthermore, it is lower bounded by $\frac{1}{n\,\mathrm{diam}(G)}$ [37], 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,

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

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 [33].

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

The degeneracy of a graph can be computed in $\mathcal{O}(n + m)$ time [36] and so does this bound.

Other lower bounds can be obtained using linear programming [4] and SDP relaxation [24]. 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. 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 $\mathrm{ec}(X_k, \overline{X_k}) \geq k\lceil \delta(G)/2 \rceil = (\lfloor \delta(G)/2 \rfloor + 1)\lceil \delta(G)/2 \rceil$. Thus

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

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 [33]).

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. 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).

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

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 [19], 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 $\mathrm{Ad}(G) = 2m/n$.

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:

$$\mathrm{cw}(G) \geq \mathrm{LB}_{\mathrm{Ad}}(G) = \frac{1}{2}\frac{m}{n-1}\left(\frac{m}{n-1} + 1\right) = \frac{1}{8}\left(1 + \frac{1}{n-1}\right)^2 \mathrm{Ad}(G)\left(\mathrm{Ad}(G) + 2 - \frac{2}{n}\right) \tag{6}$$

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

The bounds given by Ineq. (4) and (6) are incomparable. 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 $k = 50$.

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 $n = 2s+1$.

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 [11,12,15,25,27,32]. 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

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

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. This subgraph maximizes $m_{H_0}/(n_{H_0}-1)$ among all the subgraphs of G with average degree $\mathrm{Mad}(G)$. Then, from Ineq. (6) and 7, we get

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

The following tight inequalities are well-known (see e.g. Proposition 3.1 of [38]): $\delta^*(G) \leq \mathrm{Mad}(G) < 2\delta^*(G)$. Thus the bounds given by Ineq. (4) and (8) are incomparable. 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 [25, 27] 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 $\tilde{\mathcal{O}}(m/\epsilon)$ [11,13] (the $\tilde{\mathcal{O}}$ notation hides logarithmic factors in n).

4 A lower bound 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].

Consider the path $P_\sigma = v_1v_2\cdots v_n$ associated to the layout $\sigma = (v_1, v_2, \cdots, v_n)$. 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_iv_{i+1} in P_σ under $\mathcal{R}_\sigma(G)$ denoted $l(v_iv_{i+1})$ is the number of edges v_jv_ℓ 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 number of requests that can be satisfied (or groomed) together on P such that the load of any edge is at most C is denoted by $T(C,n)$.

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 2. *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. 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 $T(\mathrm{cw}(G), n) \geq m$, and so

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

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 4) 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.

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 (resp. 1, 2 and 3).

Based on these results, it was conjectured the "intuitive fact" that the optimum was obtained by taking the greedy solution consisting of all requests of smallest length (this was presented as a result in [19]). 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 3) 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 $0 \leq r < s$. We have*

$$T(C,n) \leq sn - C_s - dq + (5 - 2\sqrt{6})C + \min\{d, s - r\}$$

Proposition 3. *Let $n \geq 2$ and $m \geq 0$ be positive integers. We have $C^*(m,n) \geq LB_{\mathrm{Ad}}(m,n)$.*

Proof. If $m \leq n-1$, the proposition is obviously true. In fact, we have $\mathrm{LB}_{\mathrm{Ad}}(m,n) \leq 1$ while $C^*(1,n) = C^*(2,n) = \ldots = C^*(n-1,n) = 1$ and $\mathrm{LB}_{\mathrm{Ad}}(0,n) = 0 = C^*(0,n)$. So in what follows we suppose $m > n-1$

We first consider the case when $m/(n-1)$ is an integer $s \geq 2$. Then, we have $\mathrm{LB}_{\mathrm{Ad}}(m,n) = \frac{1}{2}\frac{m}{n-1}\left(\frac{m}{n-1}+1\right) = s(s+1)/2$. According to the definition of $C^*(m,n)$, it suffices to prove that, for $C_b = \mathrm{LB}_{\mathrm{Ad}}(m,n) = s(s+1)/2 = C_s$ we have $T(C_s,n) \leq m$.

By Theorem 2, we get, as $d=0$, that $T(C_s,n) \leq sn - C_s + (5-2\sqrt{6})C_s = sn - (2\sqrt{6}-4)C_s$. As $m = sn - s$ and $(2\sqrt{6}-4)C_s > s$ for $s \geq 2$, we conclude that $T(C_s,n) < m$ and so $C^*(m,n) > C_s$.

Otherwise, let $s = \lceil m/(n-1) \rceil$, and so $m = (s-\tau)(n-1)$ with $0 \leq \tau < 1$. Let $C_s = s(s+1)/2$ and $C_b = \mathrm{LB}_{\mathrm{Ad}}(m,n) = \frac{1}{2}(s-\tau)(s+1-\tau) = C_s - \tau s - \tau/2 + \tau^2/2$. In general C_b is not an integer. Consider the function $f(C_b,n) = sn - C_s - dq + (5-2\sqrt{6})C_b + \min\{d, s-r\}$ where $d = C_s - C_b = \tau s + \tau/2(1-\tau/2)$. We have $T(\lfloor C_b \rfloor, n) \leq f(C_b,n)$ and so according to the definition of $C^*(m,n)$, the proposition is true if $T(\lfloor C_b \rfloor, n) < m$. Therefore, it suffices to prove that $f(C_b,n) < m$.

We note that, $d \geq \tau s$ as $0 \leq \tau < 1$; $\min\{d, s-r\} \leq d$ and $C_b \leq C_s$. Therefore, it suffices to verify that $sn - (2\sqrt{6}-4)C_s - \tau s(q-1) \leq m$. Using $m = (s-\tau)(n-1)$, that corresponds to verify that $(2\sqrt{6}-4)C_s + \tau s(q-1) \geq s + \tau(n-1)$, or, as $n = qs + r \leq qs + s - 1$, that $(2\sqrt{6}-4)C_s - \tau s \geq s + \tau(s-2)$, or equivalently that

$$(\sqrt{6}-2)(s+1) \geq 1 + 2\tau - 2\tau/s$$

For $s \geq 6$, the inequality is true as $7(\sqrt{6}-2) \geq 3 > 1 + 2\tau$.

For $s = 5$, it is true as $6(\sqrt{6}-2) \geq 13/5 \geq 1 + 8\tau/5$.

For $s = 4$, it is true if $5(\sqrt{6}-2) \geq 1 + 3\tau/2$, which is the case if $\tau \leq (10\sqrt{6}-22)/3$ ($\simeq 0.81$). If $s = 4$ and $\tau \geq (10\sqrt{6}-22)/3$ then $3 < m/(n-1) \leq (34-10\sqrt{6})/3$ and so $\mathrm{LB}_{\mathrm{Ad}}(m,n) < 7$. On the other side, for $C = 6$ we have $T(6,n) = 3n - 6 < m$ and so $C^*(m,n) \geq 7$, proving the proposition.

For $s = 3$, it is true if $4(\sqrt{6}-2) \geq 1 + 4\tau/3$, which is the case if $\tau \leq (12\sqrt{6}-27)/4$) ($\simeq 0.6$). If $s = 3$ and $\tau \geq (12\sqrt{6}-27)/4$, then $2 < m/(n-1) \leq (39-12\sqrt{6})/4$ and so $\mathrm{LB}_{\mathrm{Ad}}(m,n) < 5$. On the other side, for $C = 4$ we have $T(4,n) = \lfloor (7n-10)/3 \rfloor$. Then, either $m > T(4,n)$ and so $C^*(m,n) \geq 5$ and in this case the proposition is true. Otherwise, $m \leq T(4,n) < 7(n-1)/3$ and so $\mathrm{LB}_{\mathrm{Ad}}(m,n) < 70/18 < 4$ while $T(3,n) = 2n - 3 < m$, and so $C^*(m,n) \geq 4$, and therefore the proposition is also true in this case.

For $s = 2$, it is true if $3(\sqrt{6}-2) \geq 1 + \tau$ which is the case if $\tau \leq 3\sqrt{6} - 7$ ($\simeq 0.35$). If $s = 2$ then $1 < m/(n-1) < 2$ and so $\mathrm{LB}_{\mathrm{Ad}}(m,n) < 3$. For $C = 2$, $T(2,n) = \lfloor (3n-3)/2 \rfloor$. Then either $m > T(2,n)$ and so $C^*(m,n) \geq 3$ and in this case the proposition is true. Otherwise $m \leq T(2,n) < 3(n-1)/2$ and so $\mathrm{LB}_{\mathrm{Ad}}(m,n) < 15/8 < 2$ while $T(1,n) = n - 1 < m$ and so $C^*(m,n) \geq 2$ and therefore the proposition is also true in this case. □

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.

In summary, the bounds given in this section are better than the bounds based on degrees. Furthermore, since the value of $T(C,n)$ can be computed in time $\mathcal{O}(1)$, the bound of Ineq. (9) can be computed efficiently.

Proposition 4. *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 $\left[C_1 = 1, C_2 = \lfloor n^2/4 \rfloor\right]$. Indeed, we can compute $T(C_p,n)$, with $C_p = \lfloor (C_l + C_2)/2 \rfloor$, 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 $\left[C_{\lfloor s_2 \rfloor}, C_{\lceil s_1 \rceil}\right]$. □

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:

$$\mathrm{cw}(G) \geq C^*_{\mathrm{Mad}}(G) = C^*(m_{H_0}, n_{H_0}) \tag{10}$$

The design of a fast algorithm for computing the bounds of Ineq. (10) 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., [11, 12, 42]). For instance, we can use the method proposed in [12] for finding an approximation of the maximum average degree of a graph (see the bounds C^*_g and C^*_{g++} defined 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 [24] 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 [40], Cython [6], SciPy [44] (for computing the eigenvalue λ_2) and Cplex [29] (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 (resp. 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. (10)).

Furthermore, we introduce two lower bounds obtained by using the method proposed in [12] 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:

$$\mathrm{cw}(G) \geq C^*_g(m,n) = \max\left\{C^*(|E_i|,|V_i|) \mid G[\overline{X_i}] = (V_i, E_i), 1 \leq i < n\right\} \tag{11}$$

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 [11] to obtain a 2-approximation of the maximum average degree. Approximately, 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 neighbors 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 Two families of graphs

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 $i = 0 \ldots 6$. We note that $\mathrm{cw}(H^i_{50}) = 625$ for all i.

A first observation is that $\delta(H^i_k) = k$ for $i = 0$ and 1 when $i \geq 1$. LB_{λ_2} decreases rapidly to 1 when i increases. Indeed, $\lambda_2(G)$ is upper bounded by the vertex- / edge-connectivity of the graph, and so by its minimum degree $\delta(G)$. Hence, we have $\lambda_2(H^i_k) \leq 1$ 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 $\mathrm{LB}_{\mathrm{Mad}}(H^i_k) = \mathrm{LB}_{\mathrm{Mad}}(K_k) = \mathrm{LB}_{\mathrm{Ad}}(K_k) = k(k+2)/8$ and $C^*_{\mathrm{Mad}}(H^i_k) = C^*(k(k-1)/2, k) = (k^2 - \varepsilon)/4$, where $\varepsilon = 1$ when k is even and 0 otherwise.

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

Table 1: Evolution of the lower bounds for the graphs H^i_{50} with $\mathrm{cw}(H^i_{50}) = 625$.

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)) is better 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 $n \geq 2s + 1$.

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

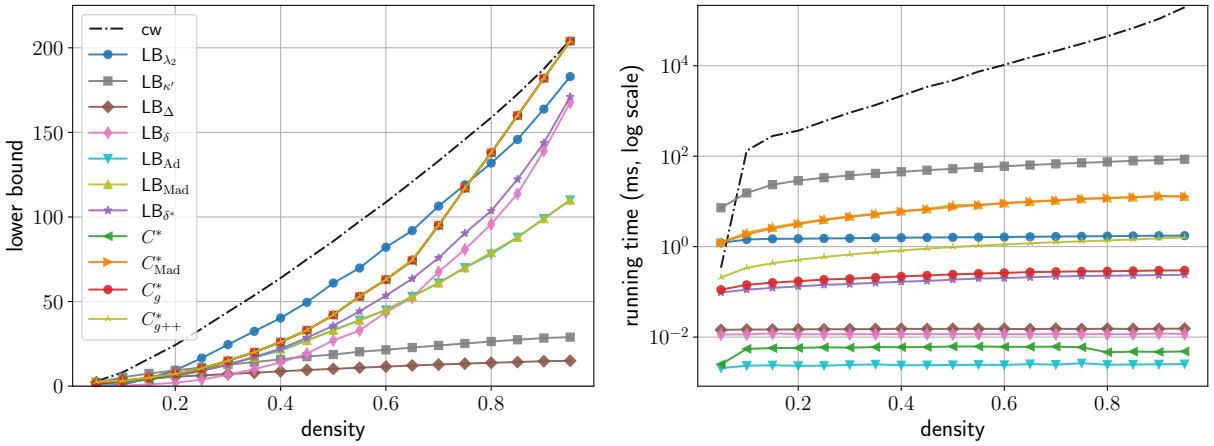
Table 2: Evolution of the lower bounds for the graphs $G(s, n)$ for $s = 1, 5, 10, 15, 20$ and $n = 2s + 1$.

5.3 Erdős-Rényi random graphs

In this section, we compare the different bounds presented in this paper on Erdős-Rényi random graphs. 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 1 and 2 the average of the computed bounds and running times.



(a) Lower bounds.

(b) Average running times (log scale).

Figure 1: Lower bounds and running times (log scale) on Erdős-Rényi graphs ($n = 30$).

In Figure 1a, 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 $\frac{\log n}{n}$. That is, if $p = p_0 \frac{\log n}{n}$, 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 $\frac{\log 30}{30} \simeq 0.11$, 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 [30] 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 1a.

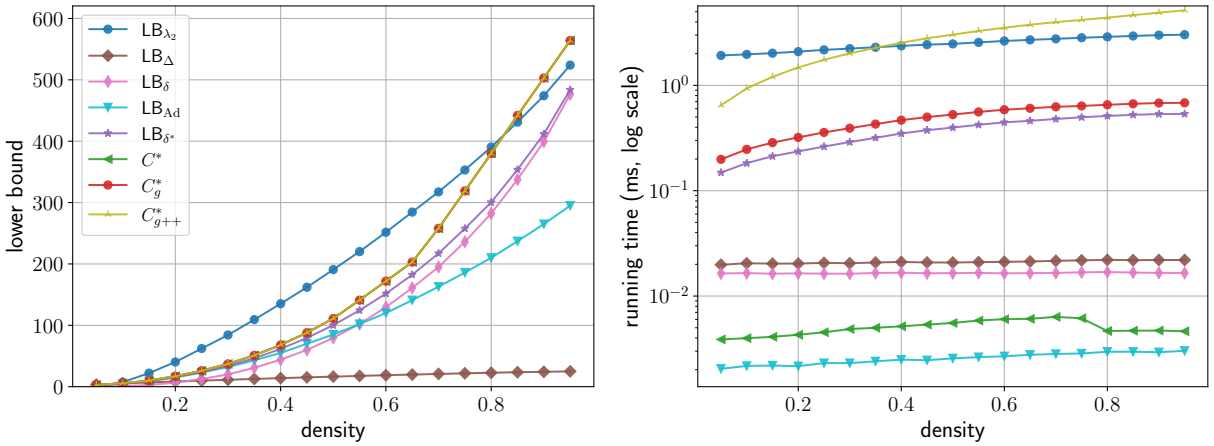
We then observe that all the bounds based on grooming (C^*, C^*_{Mad}, C^*_g, C^*_{g++}) dominate all other bounds when $p \geq 0.8$ and finally reach the optimal value of the cutwidth. Moreover, we observe that methods based on grooming provide the same bounds for these graphs. This is due to the properties 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 3, that is that $C^*(m, n) \geq \mathrm{LB}_{\mathrm{Ad}}(m, n)$.

The running times reported in Figure 1b reflect the time complexity of 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 $w + 1$. Fur-

thermore, 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 behavior for low densities since the graphs are not connected.



(a) Lower bounds.

(b) Average running times (log scale).

Figure 2: Lower bounds and running times (log scale) on Erdős-Rényi graphs ($n = 50$).

In Figure 2, we have reported for $n = 50$ the bounds on running times for the fastest methods only. We observe the same behaviors than in Figure 1 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

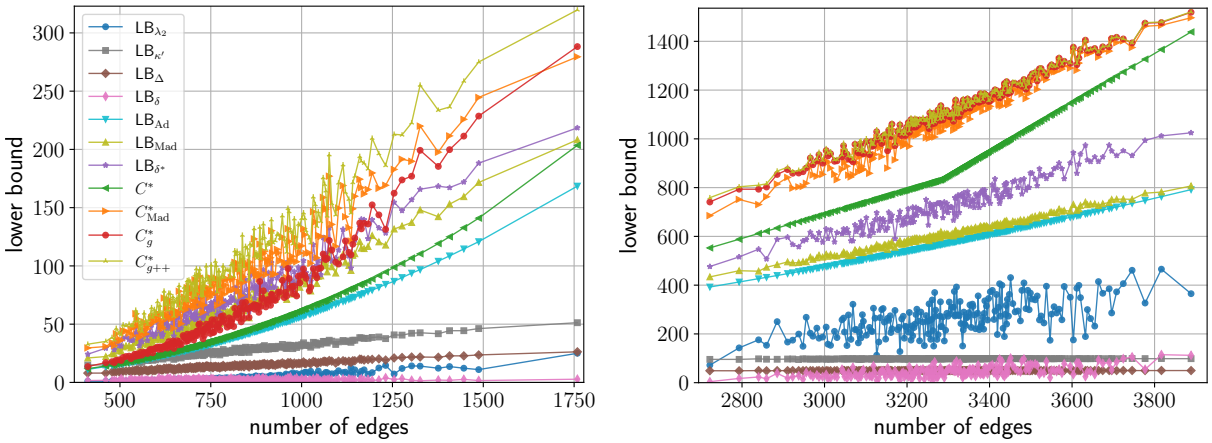
We now evaluate our bounds on random (proper) interval graphs. Consider a family $\mathcal{S} = \{S_i \mid i = 1, 2, \ldots, n\}$ of intervals on the real line. An interval graph is obtained from $\mathcal{S}$ by creating a vertex v_i for each interval S_i and connecting two vertices v_i and v_j by an 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 3.

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 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 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 3a), 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 3b), 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.



(a) Random proper interval graphs.

(b) Random interval graphs.

Figure 3: Lower bounds for random (proper) interval graphs with $n = 100$ nodes.

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 [24]

We now compare the bounds presented in this paper with the lower bounds obtained in [24] using SDP relaxation. To do so, we have computed our bounds on the graphs used in [24]. 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 [24] 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 [24], while the running times of the methods presented in this paper are several orders of magnitude smaller than those of [24].

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 [24] and obtained up to 6 orders of magnitude faster.

Let us mention that an objective of [24] 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.

			Results from [24]			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 [24] with the bounds obtained using the methods presented in this paper on Erdős-Rényi random graphs. We report the bounds and the running time (in seconds) from [24]. 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.

			Results from [24]			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 [24] 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 [24]. 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 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 grooming on a path which appear to be in many cases 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 the methods proposed in [24], using semidefinite programming, if given as input the bounds presented in this paper.

References

[1] Amir Abboud, Robert Krauthgamer, Jason Li, Debmalya Panigrahi, Thatchaphol Saranurak, and Ohad Trabelsi. 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)*, pages 884–895. IEEE, 2022.

[2] Udo Adamy, Christoph Ambühl, R. Sai Anand, and Thomas Erlebach. Call control in rings. *Algorithmica*, 47(3):217–238, 2007.

[3] Donald L. Adolphson and Te Chiang Hu. Optimal linear ordering. *SIAM Journal on Applied Mathematics*, 25(3):403–423, 1973.

[4] Nikhil Bansal, Dor Katzelnick, and Roy Schwartz. On approximatiing cutwidth and pathwidth. *CoRR*, abs/2311.15639, 2023.

[5] Dominique Barth, François Pellegrini, André Raspaud, and Jean Roman. On bandwidth, cutwidth, and quotient graphs. *RAIRO-Theoretical Informatics and Applications*, 29(6):487–508, 1995.

[6] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. Cython: The best of both worlds. *Computing in Science Engineering*, 13(2):31 –39, march-april 2011.

[7] Jean-Claude Bermond, Laurent Braud, and David Coudert. Traffic Grooming on the Path. *Theoretical Computer Science*, 384(2-3):139–151, 2007.

[8] Jean-Claude Bermond, Michel Cosnard, David Coudert, and Frédéric Havet. Groupage sur le chemin pour borner la largeur de coupe. In *AlgoTel 2023 - 25èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, page 4, Cargese, France, May 2023.

[9] Jean-Claude Bermond, Michel Cosnard, David Coudert, and Stephane Perennes. Maximum number of requests on a path with a given grooming factor. Technical report, hal-04736088, October 2024. A preliminary version was presented at the Advanced International Conference on Telecommunications (AICT), Le Gosier, Guadeloupe, France, 2006, IEEE.

[10] Hans L. Bodlaender, Fedor V. Fomin, Arie M. C. A. Koster, Dieter Kratsch, and Dimitrios M. Thilikos. A note on exact algorithms for vertex ordering problems on graphs. *Theory of Computing Systems*, 50(3):420–432, 2012.

[11] Digvijay Boob, Yu Gao, Richard Peng, Saurabh Sawlani, Charalampos E. Tsourakakis, Di Wang, and Junxing Wang. Flowless: Extracting densest subgraphs without flow computations. In *The Web Conference (WWW)*, pages 573–583. ACM / IW3C2, 2020.

[12] Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *3rd International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, volume 1913 of *Lecture Notes in Computer Science*, pages 84–95. Springer, 2000.

[13] Chandra Chekuri, Kent Quanrud, and Manuel R. Torres. Densest subgraph: Supermodularity, iterative peeling, and flow. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1531–1555. SIAM, 2022.

[14] Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *63rd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–623. IEEE, 2022.

[15] Nathann Cohen. *Three years of graphs and music : some results in graph theory and its applications*. PhD thesis, University of Nice Sophia Antipolis, France, 2011.

[16] David Coudert. A note on Integer Linear Programming formulations for linear ordering problems on graphs. Research Report hal-01271838, Inria ; I3S ; Universite Nice Sophia Antipolis ; CNRS, February 2016.

[17] Elizabeth Cuthill and James McKee. Reducing the bandwidth of sparse symmetric matrices. In *ACM'69: Proceedings of the 1969 24th national conference*, pages 157–172, 1969.

[18] Alexander Keewatin Dewdney. The bandwidth of a graph: Some recent results. In *Proceedings of the Seventh Southeastern Conference on Combinatorics, Graph Theory, and Computing*, number 17, pages 273–288, 1976.

[19] Josep Díaz, Jordi Petit, and Maria Serna. A survey on graph layout problems. *ACM Computing Surveys*, 34(3):313–356, 2002.

[20] Josep Díaz, Maria Serna, Paul Spirakis, and Jacobo Torán. *Paradigms for fast parallel approximability*. Cambridge University Press, 1997.

[21] Krzystof Diks, Hristo N. Djidjev, Ondrej Sykora, and Imrich Vrto. Edge separators for planar graphs and their applications. *Journal of Algorithms*, 14:258–279, 1993.

[22] Ulrich Elsner. *Graph Partitioning: A Survey*. Preprintreihe des Chemnitzer SFB 393/97-27. Technishe Universität Chemnitz, 1997.

[23] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[24] Elisabeth Gaar, Diane Puges, and Angelika Wiegele. Strong SDP based bounds on the cutwidth of a graph. *Computers & Operations Research*, 161:106449, 2024.

[25] Giorgio Gallo, Michael D. Grigoriadis, and Robert E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18(1):30–55, 1989.

[26] Archontia C. Giannopoulou, Michał Pilipczuk, Jean-Florent Raymond, Dimitrios M. Thilikos, and Marcin Wrochna. Cutwidth: Obstructions and algorithmic aspects. *Algorithmica*, 81:557–588, 2019.

[27] Andrew Vladislav Goldberg. Finding a maximum density subgraph. *University of California Berkeley*, 1984.

[28] Ralph Edward Gomory and Te Chiang Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.

[29] IBM ILOG. CPLEX Optimization Studio 22.1.1. `https://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`.

[30] Ferenc Juhász. The asymptotic behaviour of Fiedler's algebraic connectivity for random graphs. *Discrete Mathematics*, 96(1):59–63, 1991.

[31] Martin Juvan and Bojan Mohar. Optimal linear labelings and eigenvalues of graphs. *Discrete Applied Mathematics*, 36(2):153 – 168, 1992.

[32] Samir Khuller and Barna Saha. On finding dense subgraphs. In *36th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5555 of *Lecture Notes in Computer Science*, pages 597–608. Springer, 2009.

[33] Benoît R. Kloeckner. Cutwidth and degeneracy of graphs. arXiv 0907.5138, 2010.

[34] Richard B. Lehoucq, Danny C. Sorensen, and Chao Yang. *ARPACK users' guide - solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods.* Software, environments, tools. SIAM, 1998.

[35] Rafael Martí, Juan J. Pantrigo, Abraham Duarte, and Eduardo G. Pardo. Branch and bound for the cutwidth minimization problem. *Computers & Operations Research*, 40(1):137–149, 2013.

[36] David W. Matula and Leland L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the Association for Computing Machinery*, 30(3):417–427, jul 1983.

[37] Bojan Mohar. 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*, pages 871–898. Wiley, 1991.

[38] Jaroslav Nešetřil and Patrice Ossona De Mendez. *Sparsity: graphs, structures, and algorithms*, volume 28. Springer Science & Business Media, 2012.

[39] Victor Y. Pan and Zhao Q. Chen. The complexity of the matrix eigenproblem. In *Annual ACM Symposium on Theory of Computing (STOC)*, pages 507–516. ACM, 1999.

[40] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 10.4)*, 2024. `https://www.sagemath.org`.

[41] Dimitrios M. Thilikos, Maria J. Serna, and Hans L. Bodlaender. Cutwidth I: A linear time fixed parameter algorithm. *Journal of Algorithms*, 56(1):1–24, 2005.

[42] Charalampos E. Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria A. Tsiarli. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery (KDD)*, pages 104–112. ACM, 2013.

[43] Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3792–3835. SIAM, 2024.

[44] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.