Robust Algorithms for Noisy Minor-Free and Bounded Treewidth Graphs

Nikhil Bansal∗ Daniel Reichman† Seeun William Umboh‡

June 17, 2016

Abstract

We give a general approach to solve various optimization problems on noisy minor-free and bounded treewidth graphs, where some fraction of the edges have been corrupted adversarially. Our results are motivated by a previous work of Magen and Moharrami, who gave a \((1 + \epsilon)\)-approximate estimation algorithm for the independent set problem on noisy planar graphs, but left open the question of how to actually find a \((1 + \epsilon)\)-approximate independent set. While there are several approaches known for planar independent set, based on recursively using separators or decomposition into \(k\)-outerplanar or bounded treewidth graphs, they break down completely with noise. Our main contribution is to design robust variants of these algorithms using LP-based techniques that also work in the noisy setting.

1 Introduction

Several hard optimization problems often become substantially easier on special classes of graphs such as planar graphs and bounded treewidth graphs. For example, while the maximum independent set problem is notoriously hard on general graphs \[25\], it admits an efficient approximation scheme on planar graphs \[29, 6\] and can be solved exactly in polynomial time on bounded treewidth graphs \[8\]. In general, there has been extensive work done on designing better algorithms for special graph classes and several general techniques have been developed for this purpose. For example, for planar graphs many surprising approximation guarantees can be obtained based on decomposition approaches such as the planar-separator theorem, Baker’s decomposition into outerplanar graphs, bounded diameter partitions or using other structural properties of planar graphs \[29, 6, 27, 1\]. Similarly, for problems on bounded treewidth graphs, several techniques based on dynamic programming and deep results from algorithmic graph minor theory and logic have been developed \[13, 8, 15, 12\].

A Noisy Graph Model. In this paper, we consider a natural question that was first studied by Magen and Moharrami \[30\]: What happens to these special graph classes when these graphs are corrupted by adding some arbitrary edges adversarially? They call such edges noisy edges.

In particular, \[30\] considers the setting where an input graph \(G\) on \(n\) vertices is obtained from some (hidden) underlying planar graph \(G_0\) by adding \(\delta n\) arbitrary edges to it, for some small\[∗\] Department of Mathematics and Computer Science, Eindhoven University of Technology, Netherlands. Email: n.bansal@tue.nl. Supported by NWO Vidi grant 639.022.211 and an ERC consolidator grant 617951.

\[†\] Institute of Cognitive and Brain Sciences, University of California, Berkeley, CA. Email: daniel.reichman@gmail.com.

\[‡\] Department of Mathematics and Computer Science, Eindhoven University of Technology, Netherlands. Email: seeun.umboh@gmail.com. Supported by ERC consolidator grant 617951.
constant $\delta > 0$, and ask: how well can one approximate the maximum independent set (MIS) problem on $G$?

Let $\alpha(G)$ denote the size of a MIS in $G$. As an independent set of $G_0$ can lose at most one vertex per noisy edge, so $\alpha(G) \geq \alpha(G_0) - 3\delta n$. Since $\alpha(G_0) \geq n/4$ (planar graphs are 4-colorable), this gives $\alpha(G) \approx (1 \pm O(\delta))\alpha(G_0)$. As $\alpha(G_0)$ can be approximated to any $(1 + \epsilon)$ factor, it is plausible to expect that $\alpha(G)$ should also be approximable within any $(1 + \epsilon + O(\delta))$ factor.

**Estimation based on Lift-and-Project Methods.** Magen and Moharrami made a simple but very interesting observation that indeed $\alpha(G)$ can be approximated to within a $(1 + \epsilon)$ factor, for $\epsilon = \Omega(\delta)$, using $O(1/\epsilon)$ levels of the Sherali-Adams (SA) Hierarchy. This follows directly by observing that (i) SA solutions for $G_0$ and $G$ can differ by at most $3\delta n$, as each noisy edge added to $G$ can reduce the LP objective by at most $1$, and (ii) the $O(1/\epsilon)$-level SA linear program is exact for graphs with treewidth $1/\epsilon$ [7], and hence implies a $(1 + \epsilon)$ approximation for $\alpha(G_0)$ via Baker’s decomposition.

Interestingly, this only yields an efficient estimation algorithm for $\alpha(G)$ and does not give any way to actually find the corresponding independent set$^1$. In particular, the SA approach only uses the existence of $G_0$ to argue that $SA(G) \approx \alpha(G)$, without actually detecting the noisy edges. Designing an algorithm to actually find the independent set was left as the key open question in [30]. In general, there are relatively few problems with a separation between estimation and approximation (see e.g. [21]), and closing this gap often leads to novel algorithmic ideas.

**Limitations of current algorithms.** As discussed in detail in [30], the current techniques for solving MIS on planar graphs break down completely with noise. In particular, the known algorithms for finding planar separators or decomposition into $k$-outerplanar graphs inherently use the planar structure in various ways, and are easily tricked even with very few noisy edges. In fact, a key motivation of [30] for studying the noisy setting was to design more “robust” algorithms that are not specifically tailor-made for particular graph classes. Note that in our noise model the adversary is even allowed to add a bounded degree expander on some subset of $O(\delta n)$ vertices, completely destroying the planar structure and increasing the treewidth to $\Omega(n)$.

Another approach might be to recover some planar graph $\tilde{G}$ from $G$ without removing too many edges, and then apply the algorithms for planar graphs to $\tilde{G}$. However the best known guarantees for this Minimum Planarization problem are too weak for our purpose. In particular, even for bounded degree graphs these algorithms [11, 10] only achieve a $\text{poly}(n, \text{OPT})$ approximation, where OPT is the number of noisy edges, and thus only work in our setting when the noise parameter $\delta = n^{-\Omega(1)}$.

### 1.1 Our Results and Techniques:

We give a general approach for solving various problems on noisy versions of simple graph classes. As a simple application, we get the following result for the MIS problem on noisy planar graphs.

**Theorem 1.1.** Let $G$ be an $n$-vertex graph obtained by adding $\delta n$ arbitrary edges to some planar graph $G_0$, for some $\delta > 0$. Then there is an algorithm such that given any $\epsilon > 0$, finds an independent set of size within a $(1 + O(\epsilon + \delta))$ factor of $\alpha(G)$, and runs in time $n^{O(1/\epsilon^4)}$.

Note that the dependence on $\delta$ in the approximation guarantee cannot be avoided: If $G_0$ is empty and the noisy part is an arbitrary 3-regular graph on $2\delta n/3$ vertices, then as MIS is APX-

---

1. This says that by removing an $\epsilon$ fraction of vertices or edges, any planar graph $G_0$ can be decomposed into vertex disjoint $1/\epsilon$-outerplanar graphs, and noting that $k$-outerplanar graphs have treewidth $O(k)$.

2. Self-reducibility techniques such as those used for finding independent set in perfect graphs do not seem to work here as the estimation algorithm only gives an approximate answer.
General Framework. Formally, the noisy input graph $G$ is obtained by adding some arbitrary $\delta n$ noisy edges to some (hidden) underlying graph $G_0$. We will consider the setting where this underlying $G_0$ is either (i) planar, or more generally excludes some fixed graph $H$ as a minor, or (ii) has constant (bounded) treewidth.

The starting point for our results is the following simple observation: Most algorithms for planar graphs use the planar structure only to find a certain structured decomposition of the graph, and once this is found, apply some simple or brute-force algorithm.

For example, the planar separator theorems are used to argue that given any $\epsilon > 0$, a planar graph can be decomposed into disjoint components of size $O(1/\epsilon^2)$ by removing some subset $X$ of at most $\epsilon n$ vertices. Similarly in Baker’s decomposition, an $\epsilon$ fraction of edges $F$ (or vertices) can be removed from a planar graph to decompose it into $O(1/\epsilon)$-treewidth graphs.

Now consider a noisy planar graph $G$. We claim that such nice decompositions also exist for $G$ (although it is unclear how to find them). For example, consider the decomposition of the underlying planar graph $G_0$ into bounded size pieces and for every noisy edge in $G \setminus G_0$ put one of its endpoint in $X$. This subset $X$ has size $|X| \leq (\epsilon + \delta)n$ and removing $X$ splits $G$ into components of size $O(1/\epsilon^2)$. Similarly, let $F$ be the edges removed in Baker’s decomposition of $G_0$, plus the noisy edges in $G \setminus G_0$. Clearly, $|F| \leq (\epsilon + \delta)n$ and removing $F$ decomposes $G$ into $O(1/\epsilon)$-treewidth graphs.

So this leads to the natural question of whether we can directly find such good decompositions, without relying on the topological or other specific structure of the graph. Our main contribution is to show that this can indeed be done using general LP-based techniques. Specifically, we consider the following problems.

1. **Bounded Size Interdiction (weaker form):** Suppose $G$ can be decomposed into components of size at most $1/\epsilon^2$ by removing some (small) subset $X$ of vertices. Find such a small $X$.

2. **Bounded Treewidth Interdiction:** Suppose $G$ can be decomposed into graphs of treewidth at most $w$, by removing a subset $F$ of edges. Find such a small $F$.

Remark: To prove Theorem 1.1, we will actually need to consider a stronger and more technical version of the Bounded Size Interdiction problem above. We defer its statement to Section 3.

Our Results. We show the following results for these general interdiction problems.

**Theorem 1.2.** (weaker form) For the weaker form of the Bounded Size Interdiction problem stated above, given any $\beta \leq 1$, we can find in time $n^{O(1/\epsilon^2)}$ a subset of vertices $X'$ with $|X'| \leq O(|X|/\beta + \beta |E|)$ such that $G[V \setminus X']$ has no component larger than $1/\epsilon^2$. Here, $|E|$ is the number of edges in $G$.

For a noisy planar graph $G$, there exists $X$ of size $|X| \leq (\epsilon + \delta)n$ (as discussed above) and $|E| \leq (3 + \delta)n = O(1)n$. The latter follows as a planar graph has at most $3n - 6$ edges. So setting $\beta = (\epsilon + \delta)^{1/2}$ in Theorem 1.2 already gives $X'$ of size $O((\epsilon + \delta)^{1/2}n)$ in time $n^{O(1/\epsilon^2)}$. This can be used to design a $(1 + O((\epsilon + \delta)^{1/2}n))$-approximation algorithm for independent set in noisy planar
graphs. To get the better approximation factor of $1 + O(\epsilon + \delta)$ of Theorem 1.1 above, we will use a more refined result (Theorem 3.2) that decouples the dependence of $|X'|$ on $\epsilon$ and $\delta$.

To prove Theorem 3.2, we write a configuration LP based formulation and round it suitably. Next we show the following result for treewidth interdiction.

**Theorem 1.3.** Given a graph $G$ and an integer $w > 0$, let $F$ be some subset of edges such that removing them reduces the treewidth of $G$ to $w$. Then there is an algorithm that runs in time $n^{O(1)}$ and finds a subset of edges $F'$ such that $|F'| = O(\log n \log \log n) |F|$ and removing $F$ from $G$ reduces the treewidth to $O(w \log w)$.

Note that this gives a bicriteria $(\log n \log \log n, \log w)$-approximation to the Bounded Treewidth Interdiction problem. We also remark that the approximation factors and the running time of the above algorithm do not depend on $w$. This can be used to give a robust algorithm (with $\delta = o(1/\log n \log \log n)$) for many settings where Baker’s approach, or its extensions for minor-free graphs [17, 16] are used. The proof of Theorem 1.3 is much more challenging and requires several new ideas, and we give a broad overview of the algorithm and the proof below.

**Remark:** While the results above are only stated for noisy edges, they also extend rather directly (sometimes with a worse dependence on $\delta$) to the setting when noisy vertices are also added in addition to the edges. For clarity of presentation, we do not discuss this extension here.

**Overview of Techniques for Theorem 1.3** First, observe that if $G$ has treewidth at most $w$, then $F = \emptyset$, and the algorithm must return $F' = \emptyset$. Thus, the problem is at least as hard as determining the treewidth of $G$. This is well known to be NP-Hard, and in fact unlikely to admit a polynomial time $O(1)$ approximation under reasonable complexity assumptions [37]. This implies that the bicriteria guarantee is necessary, and that it is unlikely that the approximation with respect to $w$ can be made $O(1)$.

At a high level, our algorithm will try to construct a good tree decomposition of width $w$, while removing some problematic edges along the way. To this end, let us first see how the known algorithms for finding tree decompositions work. Treewidth is characterized up to $O(1)$ factor by the well-linkedness property of a graph, which allows one to construct a tree decomposition by computing small balanced vertex separators recursively. Either the algorithm succeeds at each step and eventually finds a tree decomposition, or returns a well-linked set as a certificate that $G$ has large treewidth. Finding balanced vertex separators is hard but one can use LP or SDP (resp.) formulations [9, 3, 20] based on spreading constraints [19, 5], and lose an $O(\log w)$ or $O(\sqrt{\log w})$ (resp.) factor in the quality of the treewidth.

In the noisy setting, our algorithmic task can thus be viewed as detecting which edges to remove so that the above recursive procedure works. To do this, we formulate an LP with variables for which edges to remove (let us call these the $x_{uv}$ variables) so that in the residual graph every subset $S$ of vertices has a small fractional balanced vertex separator of size at most $w$. However, as there are exponentially many such sets $S$, this gives a huge overall LP with (both) exponentially many variables and constraints, and it is unclear how to solve it. In particular, we have exponentially many different vertex separator LPs coupled together with the common $x_{uv}$ variables.

We describe the algorithm in two parts. First, we assume that we are given the $x_{uv}$ values from some feasible optimum LP solution. Using these $x$-values, for any given set $S$, we can now formulate a balanced edge-and-vertex separator LP, where the $x$-values give the fractional amount by which edges are removed, and in addition at most $w$ vertices are removed. Using standard region-growing techniques jointly on these edge and vertex values, we decide which edges to delete (this adds to $F'$), and which vertices lie in the separator for $S$ (these enter the bags in the tree.
decomposition). Doing this directly gives an \(O(\log^2 n)\) approximation (provided we ensure that the separator tree is balanced and has depth \(O(\log n)\)), due to the loss of an \(O(\log n)\) factor on each level of recursion. To reduce this to \(O(\log n \log \log n)\), we use “Seymour’s trick” of more careful region-growing\(^3\), together with some additional technical steps needed to make it work together with the tree decomposition procedure.

Second, we describe how to “solve” the LP. Perhaps surprisingly, this turns out to be quite challenging and requires some new ideas, which may be be useful in other contexts. We only sketch these here, and details can be found in Section 5. First, we bypass the need to completely solve this LP, by using the Round-or-Separate framework (as in \([1, 28]\)). In particular, the algorithm starts with some possibly infeasible solution \(x\), and tries to construct the tree decomposition. If it succeeds, we are done. Otherwise, it gets stuck at finding a small balanced vertex separator for some set \(S\). At that point, we try to add a violated inequality. However, a crucial point is that we need to find a violated inequality only involving the \(x\)-variables. So, a key step is to reformulate the LP to only have the \(x\)-variables. This crucially uses LP duality and the structure of the LP that the variables for different sets \(S\) are only loosely coupled via the \(x\)-variables. After this reformulation, it is still unclear how to find a violated inequality due to the exponential size of the LPs involved. We get around this issue by using some further properties of the Ellipsoid Method and the LP duality.

Other Related Work. The noise model considered here gives an interesting interpolation between easy and general worst-case instances. This is similar in spirit to approaches such as smoothed analysis\([36]\), planted models and semi-random models\([22, 31]\), although unlike these models our noise model is completely adversarial. Fomin et al.\([23]\) have considered the vertex deletion variant of our treewidth interdiction problem. They obtain a constant approximation factor for the problem, however their approximation factor depends at least exponentially on \(w\) which makes it inapplicable in our setting\(^5\). Also their algorithm is polynomial time only for \(w = O(1)\). A related model was considered by\([24]\) in the context of property-testing and sublinear time algorithms in the bounded degree model.

2 Notation and Preliminaries

We always use \(G_0\) for the underlying graph, and \(G\) for the noisy graph. The number of vertices of \(G\) is always \(n\). For a subset \(S \subseteq V\) and \(F \subseteq E\), we use the notation \(G[S] − F\) to denote the subgraph induced on the vertices \(S\), excluding the edges in \(F\). We use the notation \(E(S)\) to denote the subset of \(E\) with both endpoints in \(S\), and given another subset \(S' \subseteq V\), we use \(E(S, S')\) to denote the subset of \(E\) with one endpoint in \(S\) and another in \(S'\).

Planar and Minor-Free graphs. The classic planar separator theorem\([29]\) states that any planar graph has a \(2/3\)-balanced vertex separator of size \(O(\sqrt{n})\) and that it can be found efficiently. Applying this recursively gives the following.

**Lemma 2.1.** For any planar graph \(G\) and any \(\alpha > 0\), there is subset of vertices \(X \subset V\) with \(|X| = O(\alpha n)|\), such that every component \(C_i\) of \(G_0[V − X]\) has at most \(1/\alpha^2\) vertices.

\(^3\)We need a sublinear dependence on \(w\) for the following reason. Consider say the noisy MIS problem with \(G_0\) as a grid. If we wish to reduce the treewidth to \(w\), we would need to remove an \(O(1/w)\) fraction of the vertices, so if the interdiction algorithm is not an \(o(w)\) approximation, it might end up deleting all the vertices.
A more generally applicable technique (see e.g. [26, 18, 14]) is Baker’s decomposition \cite{Baker}, which states that for any integer \( k \), a planar graph can be decomposed into pieces of treewidth \( O(k) \) (specifically, \( k \)-outerplanar graphs) by removing \( O(1/k) \) fraction of edges or vertices.

A minor of \( G \) is a graph \( G' \) obtained by deleting and contracting edges. A graph \( G \) is \( H \)-free if \( G \) does not contain a subgraph \( H \) as a minor. Planar graphs are exactly the graphs excluding \( K_{3,3} \) and \( K_5 \) as minors. In fact, Robertson and Seymour proved that every graph family closed under taking minors is characterized by a set of excluded minors. Both the planar separator theorem and Baker’s decomposition approach extend more generally to \( H \)-free graphs \cite{RobertsonSeymour2, RobertsonSeymour3, RobertsonSeymour4}.

**Definition 2.2** (\( \alpha \)-separator of \( S \) in \( G \)). Given a graph \( G = (V, E) \) and a set \( S \subseteq V \), a vertex set \( X \subseteq V \) is an \( \alpha \)-separator of vertex set \( S \) in \( G \) if every component \( C \) of \( G[V - X] \) has \( |C \cap S| \leq \alpha |S| \).

**Definition 2.3** (Well-linked sets). A vertex set \( S \subseteq V \) is \( w \)-linked in \( G \) if it does not have a \( \frac{1}{2} \)-separator \( X \) with \( |X| < w \). The linkedness of \( G \) is defined to be the maximum integer \( w \) such that there exists a \( w \)-linked set in \( G \), and is denoted as \( \text{link}(G) \).

**Definition 2.4** (Tree decomposition). A tree decomposition of \( G \) is a tree \( T \) whose nodes \( t \) correspond to vertex subsets \( V_t \) of \( G \) (called bags) that satisfies the following properties: (i) for every edge \((u, v) \in E\), there exists a bag \( V_t \) containing both \( u \) and \( v \); (ii) for every vertex \( v \), the bags that contain \( v \) form a non-empty subtree of \( T \). The width of the decomposition is \( \text{width}(T) = \max_{s \in T} |V_s| - 1 \).

**Definition 2.5** (Treewidth). The treewidth of \( G \) is the minimum integer \( w \) such that it has a tree decomposition of width \( w \), and is denoted as \( \text{tw}(G) \).

The following well-known (see e.g. \cite{RobertsonSeymour1}) approximate characterization of treewidth in terms of linkedness will be useful for us.

**Lemma 2.6** (\cite{RobertsonSeymour1}). For any graph \( G \), \( \text{link}(G) < \text{tw}(G) < 4 \text{link}(G) \).

### 3 Bounded Size Interdiction and MIS in Noisy Planar Graphs

We consider the Bounded Size Interdiction problem and show how it implies Theorem 1.1.

Consider the noisy graph \( G \) obtained by adding \( \delta n \) edges to some planar \( G_0 \). Let us view \( G \) as obtained by superimposing the noisy edges on the recursive decomposition of \( G_0 \) given by Lemma 2.1. This directly implies the following (noisy) decomposition for \( G \).

**Lemma 3.1** (Noisy Decomposition). Given a \( \delta \)-noisy planar graph \( G \), for any \( \alpha > 0 \), there exists a partition \( X, C_1, \ldots, C_k \) of \( V \) with (i) \( |X| \leq cn \) for some universal constant \( c \), (ii) \( |C_i| \leq 1/\alpha^2 \) for all \( i \in [k] \), and (iii) at most \( \delta n \) edges whose endpoints lie in distinct (two different) \( C_i \)s.

Of course as we do not know \( G_0 \), it is not clear how to find such a decomposition. Theorem 3.2 shows that this can be done approximately.

**Theorem 3.2** (Noisy Bounded Size Interdiction). Let \( G = (V, E) \) be any graph that has a vertex partition \( X, C_1, \ldots, C_k \) with \( |C_i| \leq s \) for each \( i \in [k] \), and let \( b \) be the total number of edges whose endpoints lie in distinct \( C_i \)s.

Then for every \( \beta \leq 1 \), we can find in time \( n^{O(s)} \) a vertex partition \( X', C_1', \ldots, C_{k'}' \) such that (i) \( |X'| = O(|X|/\beta) \), (ii) \( |C_i'| \leq s \) for each \( i \in [k'] \), and (iii) at most \( O(b + \beta |E|) \) edges whose endpoints lie in distinct \( C_i \)s.
This implies the following proper decomposition (where the $C'_i$ are components of $G[V - X]$).

**Corollary 3.3.** There is a subset $X' \subset V$ of size $O(|X|/\beta + b + \beta|E|)$ such that the components in $G[V - X']$ have size at most $s$.

**Proof.** For each edge $(u, v)$ with $u \in C'_i$ and $v \in C'_j$ for $i \neq j \in [k']$, put an endpoint (say $u$) in $X'$ and remove $u$ from $C'_i$. As a result, $X'$ has size $O(|X|/\beta + b + \beta|E|)\cdot n$ and there are no edges between different $C'_i$ and $C'_j$.

Before we prove Theorem 3.2, let us first see how it implies Theorem 1.1. In particular, it says that if $G$ does not lie in some $X$ that contains at most $4/3$ vertices. The second set of constraints ensure that each vertex lies in some $C_i$ of size $s$. For each subset $S \subset V$ with $|S| \leq s$, the variable $z_S$ indicates if $S$ is one of the pieces $C_i$. Let $S$ be the collection of all such subsets $S$. For each $(u, v) \in E$, the variable $x_{uv}$ indicates whether the edge $(u, v)$ is such that $u \in C_i$ and $v \in C_j$ for some $i \neq j$.

Consider the following program.

\[
\begin{align*}
\text{minimize} & \quad \sum_{(u, v) \in E} x_{uv} \\
\text{subject to} & \quad \sum_v y_v \leq a \quad \forall v \in V \\
& \quad \sum_{S: v \in S} z_S = 1 - y_v \quad \forall v \in V \\
& \quad \sum_{S: u \in S \land v \notin S} z_S \leq x_{uv} + y_v \quad \forall (u, v) \in E \\
& \quad x_{uv}, y_v, z_S \in \{0, 1\} \quad \forall (u, v) \in E, v \in V, S \in S
\end{align*}
\]

This is easily seen to be a valid formulation for the problem. The first set of constraints ensure that $X$ contains at most $a$ vertices. The second set of constraints ensure that each vertex lies in either $X$ or some $C_i$. The third set of constraints are more involved and force $x_{uv}$ to be 1 if some edge has endpoints in distinct $C_i$ and $C_j$. In particular, it says that if $u$ lies in some $C_i$ and $v$ does not lie in that $C_i$, then either $v$ lies in $X$ (i.e. $y_v = 1$) or $x_{uv} = 1$. Note that the third set of

**Remark.** Theorem 1.1 extends directly to the minor-free case using the separator theorem for minor-free graphs [2], and the fact that these graphs have bounded average degree and thus $\alpha(G) = \Omega(n)$.  

### 3.1 Proof of Theorem 3.2

**LP formulation.** Given $\beta > 0$ and $G$ as input, we first write an integer program to find $X$ and the $C_i$s. Let $a = |X|$ (we can assume that $a$ is known as the algorithm can try every value). For each vertex $v$, the variable $y_v$ indicates if $v \in X$. For each subset $S \subset V$ with $|S| \leq s$, the variable $z_S$ indicates if $S$ is one of the pieces $C_i$. Let $S$ be the collection of all such subsets $S$. For each $(u, v) \in E$, the variable $x_{uv}$ indicates whether the edge $(u, v)$ is such that $u \in C_i$ and $v \in C_j$ for some $i \neq j$.
constraints are asymmetric in \( u \) and \( v \), and we will put two such constraints (with \( u \) and \( v \) swapped) for each edge \((u, v)\).

As the objective function exactly measures the number of edges with endpoints in two different \( C_i \)'s, it follows that the IP above has a feasible solution with value at most \( \delta n \).

Consider the LP relaxation of this program. It has \( O(n^4) \) variables, and \( O(n^2) \) non-trivial constraints. So in time \( n^{O(s)} \), we can find some basic feasible solution with support size at most \( O(n^2) \). We reuse \( x_{uv}, y_v \) and \( z_S \) to denote some fixed optimum solution to the LP.

The Rounding Algorithm. The algorithm will construct the required \( X' \) and the collection \( C \) of sets \( C_i' \) from the LP solution by the following preprocessing and sampling procedure.

1. Initialization. We set \( X', C = \emptyset \) and \( U = V \), where \( U \) denotes the set of vertices not covered by \( C \).
2. Preprocessing. Add every vertex \( v \) with \( y_v \geq \beta \) to \( X' \). Set \( U = U \setminus X' \)
3. Sampling to create \( C \). Arbitrarily order the sets \( S_1, \ldots, S_k \) in the support of the LP solution. Repeat the following (phase) until \( U \) is empty:
   Phase. For \( i = 1, \ldots, k \), sample the set \( S_i \) randomly with probability \( z_{S_i} \). If \( S_i \) is picked, add \( C_i' = S_i \cap U \) to the collection \( C \), and update \( U = U \setminus S_i \).

Analysis. Clearly the sets \( C_i' \) produced by the algorithm have size at most \( s \), and they are disjoint.

Lemma 3.4. \( |X'| \leq a/\beta \).

Proof. As \( X' \) is the set of vertices \( v \) with \( y_v \geq \beta \), there can be at most \( a/\beta \) such vertices by the LP constraint \( \sum_v y_v \leq a \).

Henceforth, we also assume that \( \beta \leq 1/2 \), otherwise choosing \( X' = \emptyset \) and partitioning \( V \) arbitrarily into sets \( C_1, \ldots, C_k \) of size at most \( s \) trivially suffices for Theorem 3.2.

We now show that the algorithm runs in expected polynomial time and does not generate a vertex partition with too many edges between distinct \( C_i' \).

Lemma 3.5. Let \( F \) be the set of edges with endpoints in two distinct \( C_i' \)'s. Then \( \mathbb{E}[|F|] \leq O(b + \beta|E|) \). Moreover, the algorithm terminates after at most \( O(n^2 \log n) \) sampling steps with high probability.

Proof. We claim that \( U \) is empty after \( O(\log n) \) phases, with high probability. After the preprocessing step, each uncovered vertex in \( U \) has \( y_v < \beta \leq 1/2 \). Thus, by the second LP constraint, \( p_v := \sum_{S \ni v} z_S = 1 - y_v \geq 1/2 \). So the probability that a vertex \( v \) is not covered after \( j \) phases is

\[
\left( \prod_{S \ni v} (1 - z_S) \right)^j \leq \exp (-jp_v) \leq \exp (-j/2) \leq (2/3)^j
\]

The claim now follows from a union bound over the \( n \) vertices.

We now bound the size of \( F \). Let us focus on an edge \( e = (u, v) \) and bound the probability that it is cut, that is, added to \( F \) during the Sampling step. Let \( U_j \) denote the vertices in \( U \) at the end of phase \( j \). The edge is cut in phase \( j \) if and only if both \( u \) and \( v \) remain in \( U \) at the
end of phase $j - 1$ (i.e. $u, v \in U_{j-1}$) and a set $S$ with $|S \cap \{u, v\}| = 1$ is chosen in phase $j$. As $\Pr[u, v \in U_{j-1}] \leq \Pr[v \in U_{j-1}] \leq (2/3)^{j-1}$, this implies that
\[
\Pr[(u, v) \text{ cut in phase } j] \leq (2/3)^{j-1} \sum_{S: |S \cap \{u, v\}| = 1} z_S. \tag{2}
\]
Moreover, by the third set of constraints in LP \([1]\),
\[
\sum_{S: |S \cap \{u, v\}| = 1} z_S = \sum_{S: u \in S \land v \notin S} z_S + \sum_{S: v \in S \land u \notin S} z_S \leq 2x_{uv} + y_u + y_v. \tag{3}
\]
Summing (2) over all the phases and using (3), we get
\[
\Pr[(u, v) \text{ cut}] \leq 3(2x_{uv} + y_u + y_v) \leq 6x_{uv} + 6\beta,
\]
where the second inequality follows as both $u$ and $v$ were not chosen in $X$ during the preprocessing step and hence $y_u, y_v \leq \beta$. By linearity of expectation, this implies that
\[
\mathbb{E}[|F|] = \sum_{(u, v) \in E} (6x_{uv} + 6\beta) = O(b + \beta|E|).
\]

## 4 Bounded Treewidth Interdiction

Recall that in the Bounded Treewidth Interdiction problem, we are given a graph $G = (V, E)$, a target treewidth $w$, and we want to find the minimum set $F$ of edges to delete from $G$ such that $\text{tw}(G - F) < w$. In this section, we begin the proof of Theorem 1.3 by describing the exponential-size LP and the rounding algorithm. In the following, when $X$ is a vertex set and $F$ is an edge set, we use the shorthand $G - X$ to mean $G[V - X]$, and $G - X - F$ to mean $G[V - X] - F$.

### 4.1 An Exponential-Sized LP

Lemma 2.6 gives us a convenient characterization of feasible solutions $F$ which we can use to write an LP. In particular, it says that if $\text{tw}(G - F) < w$, then every vertex set $S \subseteq V$ has a $\frac{1}{2}$-separator $X^S$ in $G - F$ of size less than $w$.

Consider the following LP. It has a variable $x_{uv}$ indicating if edge $(u, v) \in E$ belongs to $F$. For every subset $S \subseteq V$ and vertex $v \in V$, variable $y^S_v$ indicates if $v$ belongs to the minimum-size $\frac{1}{2}$-separator $X^S$ of $S$ in $G - F$. For $u, v \in V$, let $\mathcal{P}(u, v)$ denote the set of paths between $u$ and $v$. For a path $P$, define $E(P)$ to be the set of edges in $P$ and $V(P)$ to be the set of vertices on $P$, including the endpoints.

\[
\begin{align*}
\text{minimize} & \quad \sum_{(u, v) \in E} x_{uv} \\
\text{subject to} & \quad \sum_{v \in V} y^S_v \leq w \quad \forall S \subseteq V \\
& \quad d^S_{uv} \leq \sum_{e \in E(P)} x_e + \sum_{t \in V(P)} y^S_t \quad \forall S \subseteq V, u, v \in V, P \in \mathcal{P}(u, v) \\
& \quad \sum_{e \in U} d^S_{uv} \geq |U| - \frac{|S|}{2} \quad \forall U \subseteq S \subseteq V, u \in U
\end{align*}
\]

\[9\]
We interpret the solution as follows: the LP assigns a length \(x_{uv}\) to each edge \((u, v)\) \(\in E\) and a weight \(y_v^S\) for each vertex set \(S\) and vertex \(v\). Consider a fixed set \(S\). Without loss of generality, we can assume that the variable \(d_{uv}^S\), denotes the distance between \(u\) and \(v\) induced by the edge lengths \(x_e\) and vertex weights \(y_v^S\). In particular, if we define the length of a path \(P \in \mathcal{P}(u, v)\) to be the sum of edge lengths and vertex weights on the path, including the weights on \(u\) and \(v\), then \(d_{uv}^S\) is the length of the shortest path between \(u\) and \(v\). The variables \(d_{uv}^S\) and the last set of constraints are often called spreading metrics and spreading constraints, respectively. It is also easy to see that without loss of generality any feasible solution satisfies \(d_{uv}^S \leq 1\). Note that there is a potentially different metric \(d^S\) for each set \(S\), and that the LP has exponentially many constraints and exponentially many variables.

**Lemma 4.1.** LP (4) is a relaxation of the treewidth interdiction problem.

**Proof.** We show that for every edge set \(F\) such that \(\text{tw}(G - F) \leq w\), there exists a feasible solution \((x, y, d)\) to LP (4) with \(\sum_{(u,v) \in E} x_{uv} \leq |F|\). Let \(x\) be the indicator vector for \(F\).

As \(\text{tw}(G - F) \leq w\), by Lemma 2.6 for each vertex set \(S\), there exists a set \(X_S\) of at most \(w\) vertices such that no component of \(G - X_S - F\) contains more than half of \(S\). Define \(y^S\) to be the indicator vector for \(X_S\) and \(d_{uv}^S = 1\) if either \(u\) or \(v\) lies in \(X_S\), or if \(u\) and \(v\) lie in separate components of \(G - X_S - F\).

The solution \((x, y, d)\) has \(\sum_{(u,v) \in E} x_{uv} = |F|\) and satisfies the first two sets of constraints. It remains to show that it satisfies the spreading constraints. Fix some \(S\) and consider \(U \subseteq S\) with \(|U| \geq |S|/2\). Let \(u\) be a vertex of \(U\). There are two cases to consider: (i) If \(u \in X_S\), we have \(d_{uv}^S = 1\) for all \(v \in U \setminus \{u\}\) and so \(\sum_{v \in U} d_{uv}^S \geq |U| - 1 \geq |U| - |S|/2\); (ii) otherwise if \(u \notin X_S\), let \(C\) be the component of \(G - X_S - F\) that contains \(u\). We have \(|C \cap S| \leq |S|/2\) since \(X_S\) is a \(1/2\)-separator of \(S\) in \(G - F\). We also have \(d_{uv}^S = 1\) for every \(v \in U - (C \cap S)\) since \(v\) is either in a different component of \(G - X_S - F\) or in \(X_S\). Thus,

\[
\sum_{v \in U} d_{uv}^S \geq \sum_{v \in U - (C \cap S)} d_{uv}^S \geq |U| - |C \cap S| \geq |U| - |S|/2.
\]

Therefore, the solution \((x, y, d)\) is a feasible solution with \(\sum_{(u,v) \in E} x_{uv} = |F|\). \(\square\)

In the rest of this section, we focus on developing a rounding algorithm that only needs the variables \(y^S\) and \(d^S\) for polynomially many sets \(S\). In particular, we prove the following lemma.

**Lemma 4.2.** Given oracle access to a feasible solution \((x, y, d)\) of LP (4), we can find in time \(\text{poly}(n)\) an edge set \(F\) such that \(\text{tw}(G - F) \leq O(w \log w)\) and \(|F| \leq O(\log n \log \log n) \sum_{(u,v) \in E} x_{uv}\).

### 4.2 Sketch of the Rounding Algorithm

Let \((x, y, d)\) be a feasible solution to LP (4). As mentioned in the Introduction, our rounding algorithm is based on a recursive algorithm for constructing tree decompositions with Seymour’s recursive graph decomposition trick layered on top. Before going into the details, let us first see at a high level how these ideas are combined together and highlight the key issues that arise.

**Classic Tree Decomposition.** We begin by outlining the relevant parts of the classic tree decomposition algorithm [34] (which we call DECOMPOSITION). It is a recursive algorithm: Given a subgraph \(H\) of \(G\) and an integer \(w\), it either constructs a tree decomposition of width \(O(w)\) or it finds a \(w\)-linked set \(S\) which certifies\(^4\) that \(\text{tw}(G) \geq w\). There are two key steps that are important to us:

\(^4\)It certifies \(\text{tw}(H) \geq w\), and the treewidth of a graph is at least the treewidth of any subgraph.
Figure 1: DECOMPOSITION finds a separator $X$, and for each component $C_i$ of $H - X$, it recurses on the subgraph $H_i$ consisting of edges of $H$ with at least one endpoint in $C_i$.

- Separate: find a minimum-size $\frac{2}{3}$-separator $X$ of a vertex set $S$ in $H$ (the particular choice of $S$ depends on previous recursive steps).

- Recurse: for each component $C_i$ of $H - X$, recurse on the subgraph $H_i$ of $H$ which consists of the edges of $H$ induced by $C_i$ and those between $C_i$ and $X$.

See Figure 1. We say that DECOMPOSITION “succeeds” if it does not encounter a $w$-linked vertex set $S$. In particular, when DECOMPOSITION succeeds, the separators found during its execution can be used to construct a tree decomposition of width $O(w)$; when it fails, it has found a $w$-linked set $S$ during its recursion.

**Our Algorithm.** Our algorithm (which we call INTERDICT) largely follows along the lines of DECOMPOSITION. The main difference is that we also want to delete edges to ensure that size of the separators found in the recursion are small enough so that we succeed in constructing a tree decomposition of width $O(w \log w)$. We still make the same choices about which set $S$ to separate and how to recurse; this allows us to reuse the analysis of DECOMPOSITION to prove that we have deleted enough edges to reduce the treewidth down to $O(w \log w)$. In particular, instead of the Separate step, we want to perform a “Delete and Separate” step instead.

- Delete and Separate: delete a subset $D$ of edges and find a vertex set $X$ of size $O(w \log w)$ such that $X$ is a $\frac{2}{3}$-separator of $S$ in $H - D$.

Here is where LP (4) is useful. It is similar to the spreading metric relaxation for finding minimum balanced vertex separators, except that it gives edge-and-vertex separators. As mentioned in the Introduction, one can apply standard region growing techniques in an almost black box fashion along with other tricks to obtain a $(\log^2 n, \log w)$-approximation to treewidth interdiction.

Obtaining a $(\log n \log \log n, \log w)$-approximation needs some care. At a high level, we want to apply region growing recursively using Seymour’s recursion. The basic idea is to ensure that not only is $|D|$ bounded by the cost of the LP solution projected on $H$, but it is also bounded in some nice way by the cost of the LP solution on the subgraphs $H_i$ we recurse on. In particular, we want to use region growing to partition the spreading metric $d^S$ into “regions” (see Figure 2) $B_1, \ldots, B_k$ with the following properties:

- (bounded charge) the set of edges $\delta_x(B_i)$ (vertices $\delta_y(B_i)$ resp.) cut by $B_i$ can be charged to the $x_{uv}$ variables ($y^S_{uv}$ variables, resp.) “contained” in the region,

- (containment) each subgraph $H_i$ we recurse on is contained inside a region.
Figure 2: **Interdict** uses the spreading metric given by LP solution (left), partitions the metric into regions with cut vertices and edges shown in red (middle), and then recurses on the subgraph contained in each region (right).

We can then choose $D = \delta_x(B_1) \cup \cdots \cup \delta_x(B_k)$ and $X = \delta_y(B_1) \cup \cdots \cup \delta_y(B_k)$. Due to the structure of the subproblems in the Recurse step, ensuring the second property requires some care with how we implement region growing. The problem is that the separators involve both edges and vertices, and they play different roles, i.e. edges are deleted globally, while vertices are deleted locally for each $S$. Normally, the region growing technique proceeds by finding a region $B_i$ in the current graph that satisfies the bounded charge property, removes $B_i \cup \delta_y(B_i)$—the vertices contained in or cut by $B_i$ from the graph—and repeats on the residual graph. However, this would also remove any edge $(u,v)$ with $u \in \delta_y(B_i)$, even though $v$ is still remaining in the residual graph. In this case, no future region can contain or cut the edge $(u,v)$. This is a problem if $v$ ends up being in some component $C_i$ later, as we would have a subgraph $H_i$ that is not contained in any region.

Thus, we need to somehow preserve edges between $\delta_y(B_i)$ and the residual vertices. Later, in Section 4.3, we will do this by introducing copies of these edges that we need to preserve, called “zombie edges”.

We now describe the Delete and Separate step in detail.

### 4.3 The Delete and Separate Step

The main ingredient of the Delete and Separate step is the region growing technique.

**Region Growing.** We begin with some standard definitions. We define the ball $B(s,r)$ centered at vertex $s$ with radius $r$ as $B(s,r) = \{v : d_{sv} \leq r\}$. We say that a vertex $v$ is cut by $B(s,r)$ if $d^S_{sv} - y_v^S < r < d^S_{sv}$, i.e. $v$ is only “partially inside” the ball, and use $\delta_y(s,r)$ to denote the set of vertices cut by $B(s,r)$; likewise, an edge $(u,v)$ is cut by $B(s,r)$ if $d^S_{su} \leq r < d^S_{su} + x_{uv}$ and we use $\delta_x(s,r)$ to denote the set of edges cut by $B(s,r)$. We define two quantities for the $x$-cost and $y$-cost. The cost $\text{LP-cost}(s,r)$ of $B(s,r)$ is defined to be the total cost of the LP solution “contained” in $B(s,r)$:

$$\text{LP-cost}(s,r) = \sum_{u \in B(s,r), v \in B(s,r) \cup \delta_y(s,r)} x_{uv} + \sum_{(u,v) \in \delta_x(s,r) : u \in B(s,r), v \notin B(s,r)} r - d^S_{su}$$

Similarly, its weight is defined to be the total fractional weight that is “contained” in $B(s,r)$:

$$\text{wt}(s,r) = \sum_{v \in B(s,r)} y_v^S + \sum_{v \in \delta_y(s,r)} r - (d^S_{sv} - y_v^S).$$

See Figure 3 for an illustration. The $x$-volume and $y$-volume is then defined to be $\text{vol}_x(s,r) = \frac{\text{LP-cost}(G)}{n} + \text{LP-cost}(s,r)$ and $\text{vol}_y(s,r) = \frac{1}{w} + \text{wt}(s,r)$.

---

\(^5\)See Figure 4 for an example.
The spreading constraints imply that Lemma 4.3 (Region Growing Lemma). B by B of edges cut by s, r.

Proof. We prove the contrapositive. Suppose, that $|U \cap S| > 2|S|/3$. Let $u$ be a vertex in $U \cap S$. The spreading constraints imply that

$$\sum_{v \in U \cap S} d_{uw}^S \geq |U \cap S| - |S| > 2|U \cap S| - 3|U \cap S| = |U \cap S|.$$  

Thus, by averaging, there exists $v \in U \cap S$ such that $d_{uw}^S > 1/4 > 1/6$.  

We are now ready to describe the PARTITION algorithm, which executes the Delete and Separate component of our algorithm. Recall that in the Delete and Separate step, the input is a subgraph $H = (V', E')$ and a vertex set $S \subseteq V'$, and our goal is find a set $D$ of edges to delete and a vertex set $X$ of size $O(w \log w)$ such that $X$ is a $\frac{d}{r}$-separator of $S$ in $H - D$. 

Figure 3: In this figure, the edge lengths represents the $x$ variables, and vertices are represented as circles whose diameters correspond to their $y^S$ weights. The dashed arc is centered at $v_1$ and has radius 0.8. We have $B(v_1, 0.8) = \{v_1, v_2, v_3\}$. Its boundary edges and vertices are $\delta_x(v_1, 0.8) = \{(v_2, v_3), (v_1, v_4)\}$ and $\delta_y(v_1, 0.8) = \{v_3\}$, respectively. Towards $\text{vol}_x(v_1, 0.8)$, edge $(v_1, v_2)$ contributes 0.1, $(v_1, v_3)$ contributes 0.3, $(v_1, v_4)$ contributes 0.7, and $(v_2, v_4)$ contributes 0.1. Towards $\text{vol}_y(v_1, 0.8)$, $v_1$ contributes 0.1, $v_2$ contributes 0.5, and $v_3$ contributes 0.4.

These notions also extend to subgraphs. Given a subgraph $H'$, we define $\text{LP-cost}(H') = \sum_{u,v \in H'} x_{uv}$ and $\text{wt}(H') = \sum_{v \in H'} y_v^S$. Similarly, $\text{vol}_x(H') = \frac{\text{LP-cost}(G)}{n^2} + \text{LP-cost}(H')$ and $\text{vol}_y(H') = \frac{1}{w} + \text{wt}(H')$.

The following lemma shows that there exists a good radius $r \in [0, 1/12]$ such that the number of edges cut by $B(s, r)$ can be charged to $\text{vol}_x(s, r)$ and simultaneously, the number of vertices cut by $B(s, r)$ can be charged to $\text{vol}_y(s, r)$. We defer the proof of this lemma to Appendix A.

Lemma 4.4. Let $U$ be a set of vertices. If $\max_{u,v \in U} d_{uv}^S \leq 1/6$, then $|U \cap S| \leq 2|S|/3$.

Proof. We prove the contrapositive. Suppose, that $|U \cap S| > 2|S|/3$. Let $u$ be a vertex in $U \cap S$. The spreading constraints imply that

$$\sum_{v \in U \cap S} d_{uw}^S \geq |U \cap S| - |S|/2 > |U \cap S| - 3|U \cap S|/4 = |U \cap S|/4.$$  

Thus, by averaging, there exists $v \in U \cap S$ such that $d_{uw}^S > 1/4 > 1/6$.  

We are now ready to describe the PARTITION algorithm, which executes the Delete and Separate component of our algorithm. Recall that in the Delete and Separate step, the input is a subgraph $H = (V', E')$ and a vertex set $S \subseteq V'$, and our goal is find a set $D$ of edges to delete and a vertex set $X$ of size $O(w \log w)$ such that $X$ is a $\frac{d}{r}$-separator of $S$ in $H - D$. 

13
The Partition algorithm.

1. Initialization. Let $\hat{H} = H$

2. Region growing. While $\hat{H}$ contains more than two-thirds of $S$,
   (a) Find region. Choose an arbitrary vertex $v \in S$ that is contained in $\hat{H}$ and find a good radius $r$ such that $B(v, r)$ satisfies the conditions of Lemma 4.3. Note that distances, balls and boundaries are defined with respect to $\hat{H}$.
   (b) Removal. Remove all vertices in $B(v, r)$ and their incident edges from $\hat{H}$.
   (c) Add zombies. For each edge $(s, u)$ that was removed, if $s \in \delta_y(v, r)$ and $u$ is still in $\hat{H}$, add zombie vertex $z_u(s)$ with weight $y^{z_u(s)}_z = 0$ to $\hat{H}$ and zombie edge $(z_u(s), u)$ with length $x_{z_u(s), u} = x_{su}$.

3. Let $B(v_i, r_i)$ be the ball found in the $i$-th iteration.

4. Return the vertex set $X = \bigcup_i \delta_y(v_i, r_i)$ and the edge set $D = \bigcup_i \delta_x(v_i, r_i)$, replacing each zombie vertex $z_u(s)$ in $X$ with its original vertex $s$ and each zombie edge $(z_u(s), u)$ in $D$ with its original edge $(s, u)$.

Lemma 4.5. Suppose Partition took $\ell$ iterations and let $B(v_1, r_1), \ldots, B(v_\ell, r_\ell)$ be the regions it found. Let $C_1, \ldots, C_k$ be the components of $H - X - D$. For every $j \in [k]$, define the subgraph $H_j = (V_j, E_j)$ where $E_j$ is the subset of $E' - D$ with at least one endpoint in $C_j$, and $V_j$ is the set of endpoints of $E_j$. We have the following:

1. $|D| \leq O(\log \log n) \cdot \sum_{i=1}^\ell \ln \left( \frac{\text{vol}_x(H)}{\text{vol}_x(v_i, r_i)} \right) \text{vol}_x(v_i, r_i)$,
2. $|X| \leq \gamma \cdot (w + |S|/w) \log w$, for some constant $\gamma$,
3. $|C_j \cap S| \leq 2|S|/3$, and
4. The edge set $E_j$ of each subgraph $H_j$ is either contained in a region $B(v_i, r_i)$ (and so $\text{vol}_x(H_j) \leq \text{vol}_x(v_i, r_i)$) or it is contained in the residual graph $\hat{H}$ at the end of the execution of Partition.

Proof. The first statement follows from the fact that we chose a good radius $r_i$ for each region $B(v_i, r_i)$ and that the $x$-volume of any region can be at most $\text{vol}_x(H)$. Let us now consider the second statement. The fact that we chose a good radius for each region implies that

$$|X| = \sum_{i=1}^\ell |\delta_y(v_i, r_i)| \leq \kappa \cdot \log w \sum_{i=1}^\ell \text{vol}_y(v_i, r_i)$$
for some constant $\kappa$. Recall that $\text{vol}_y(v_i, r_i) = \text{wt}(v_i, r_i) + 1/w$. Since each vertex can only contribute towards the weight of at most one region and zombie vertices have zero weight, the total weight of any region is at most $\sum_{v \in V'} y_v^i \leq w$. So, $\sum_{i=1}^{\ell} \text{vol}_y(v_i, r_i) \leq (w + \ell/w)$. Moreover, since the center of each region is a vertex $v_i$ of $S$, we remove at least one vertex of $S$ in each iteration. Thus, the number of iterations $\ell$ can be at most $|S|$. So, $|X| \leq \kappa \cdot \log w \cdot (w + \ell/w) \leq \kappa \cdot (w + |S|/w) \log w$ and this proves the second statement of the lemma.

Next, we argue that $|C_i \cap S| \leq 2|S|/3$. If $C_i$ was a component that remained at the end of the execution of \textsc{Partition}, then by definition, $|C_i \cap S| \leq 2|S|/3$. Suppose $C_i$ was a component that is contained in $B(v_i, r_i)$. By Lemma 4.4, it suffices to check that the distance between any two remaining vertices $u$ and $v$ at the start of iteration $i$ is at least $d_{uv}^S$. Over the previous iterations, \textsc{Partition} modifies the graph in two ways: by removing edges and vertices, and by introducing zombie edges. Removing edges and vertices clearly cannot decrease the distance between $u$ and $v$. Zombie edges do not create a shortcut between $u$ and $v$ either. Thus, the distance between $u$ and $v$ in iteration $i$ is at least $d_{uv}^S$.

Finally, each subgraph $H_j$ is connected (since they are the set of edges with one endpoint in a connected component $C_j$), thus for each region $B(v_i, r_i)$, either all the vertices of $H_j$ are in $B(v_i, r_i)$ or all of them are not in $B(v_i, r_i)$.

\section{Putting it together}

We are now ready to describe the \textsc{Interdict} algorithm, which recursively rounds a feasible solution to LP \ref{LP}. It takes as input a subgraph $H = (V', E')$ and a vertex set $S$, and deletes a set of edges $F$ such that $\text{tw}(H - F) \leq O(w \log w)$. Let $\gamma$ be the constant in Lemma \ref{lem:tw-bound}.

\textbf{The Interdict Algorithm.}

1. Delete and Separate. Use algorithm \textsc{Partition} to find a set $D$ of edges to delete and a $2/3$-separator $X$ of $S$ in $H - D$. Let $C_1, \ldots, C_k$ be the components of $H - D - X$. Delete $D$.

2. Define subproblems. For $i \in [k]$, define the subgraph $H_i = (V_i, E_i)$ where $E_i$ is the subset of $E' - D$ with an endpoint in $C_i$, and $V_i$ is the set of endpoints of $E_i$.

3. Recurse. For $i \in [k]$, call \textsc{Interdict}($H_i, V_i \cap (X \cup S)$).

To round a feasible solution of LP \ref{LP}, we call \textsc{Interdict}($G, S_0$) where $S_0$ is an arbitrary set of at most $O(w \log w)$ vertices. Let $F$ be the set of edges deleted by \textsc{Interdict}($G, S_0$).

\textbf{Lemma 4.6.} $\text{tw}(G - F) \leq O(w \log w)$.

\textbf{Proof.} We can apply exactly the same analysis of the classic tree decomposition algorithm \ref{algo} to argue that the $O(w \log w)$-size separators found in the recursion can be used to construct a tree decomposition $T$ of $G - F$ such that the width of $T$ is at most $O(w \log w)$. \hfill \square

Recall that $\text{LP-cost}(G) = \sum_{(u,v) \in F} x_{uv}$, the cost of the LP solution $(x, y, d)$. We defer the proof of the following lemma to Appendix A.

\textbf{Lemma 4.7.} $|F| \leq O(\log n \log \log n) \text{LP-cost}(G)$.

Lemmas \ref{lem:tw-bound} and \ref{lem:sep-bound} imply Lemma \ref{lem:interdict-bound}.
5 Using the LP

We now come to the problem of how to handle the LP (4) in $n^{O(1)}$ time. As discussed in the Introduction, we bypass the need to completely solve this LP using the Round-or-Separate framework. A crucial ingredient here is that if the rounding step gets stuck (is unable to find a small separator), we need to find a violated inequality for the $x$-variables, and not just for the LP (4). In Section 5.1 we show how to reformulate LP (4) to only have the $x$ variables. In this reformulation, the coefficients of the inequalities come from feasible points in a polytope with exponentially many variables, so Section 5.2 deals with how to fix such a violated inequality.

5.1 Reformulating the LP

We first reformulate LP (4)—the original LP with variables ($xxx, yyy, ddd$)—to obtain another LP with only $x$-variables.

Call a vector $xxx$ feasible if there exist vectors $yyy$ and $ddd$ such that $(xxx, yyy, ddd)$ is a feasible solution to LP (4). Define $F$ to be the set of all feasible vectors $xxx$, and observe that $F$ is simply the feasible region of LP (4) with the $y$ and $d$ variables projected out.

Next, we show how to describe $F$ using linear inequalities. For every vertex set $S$, define the following LP parameterized by $xxx$. We call this sep-LP($xxx, S$).

\[(\text{sep-LP}(xxx, S)) \quad \text{minimize} \quad \sum_y y_v \quad \text{subject to} \quad d_{uv} \leq \sum_{e \in E(P)} x_e + \sum_{t \in V(P)} y_t \quad \forall u, v \in V, P \in \mathcal{P}(u, v) \]
\[\sum_{u \in T} d_{uv} \geq |T| - |S|/2 \quad \forall T \subseteq S \subseteq V, v \in T \]

We emphasize that in the LP above only $y$ and $d$ are variables. Recall that this LP is related to the problem of finding a small balanced vertex separator of $S$ in $G$, provided the edges are removed fractionally to extent $x_e$.

**Definition 5.1.** Let $\lambda(x, S)$ be the value of the optimum solution to the sep-LP($xxx, S$). We say that $xxx$ is $S$-feasible if $\lambda(x, S) \leq w$, and denote by $F(S)$ the set of $S$-feasible vectors.

**Lemma 5.2.** $F = \bigcap_{S \subseteq V} F(S)$.

**Proof.** Suppose $xxx \in F$. Then, for every vertex set $S$, the vectors $y^S$ and $d^S$ are a feasible solution to sep-LP($xxx, S$) and $\sum_y y^S \leq w$, so $\lambda(x, S) \leq w$.

On the other hand, suppose $\lambda(x, S) \leq w$ for all vertex sets $S$. Define $y$ and $d$ such that for each $S \in \mathcal{S}$, $(y^S, d^S)$ is the optimal solution to sep-LP($xxx, S$). As $\lambda(x, S) \leq w$ for all $S \in \mathcal{S}$, we have that $(xxx, y, d)$ is a feasible solution to LP (4).

Thus, to describe $F$ using linear inequalities, it suffices to describe $F(S)$ using linear inequalities. By linear programming duality, we have that $\lambda(x, S) \leq w$ if and only if for every feasible solution to the dual of sep-LP($xxx, S$) has objective value at most $w$.

So we consider the dual of the LP (6) below, and denote it by flow-LP($xxx, S$). (We call this flow-LP as it is dual to a “cut-type” LP.)

**Lemma 5.3.** The dual to sep-LP($xxx, S$) is given by LP (6) below.
Proof. Let us introduce the variables $f_{Puv}$ for the first set of constraints in LP (5), and the variables $g_{T,v}$ for the second set of constraints. Again, we emphasize that only $f$ and $g$ are variables here (and that $x$ is not a variable).

Let us check that the dual is exactly LP (6). Rewrite the first set of primal constraints as
\[
\sum_{t \in V(P)} (P) y_t - d_{uv} \geq - \sum_{e \in E(P)} x_e \quad \forall u, v \in V, P \in \mathcal{P}(u,v)
\]
In the objective, the coefficient of $f_{Puv}$ is $- \sum_{e \in E(P)} x_e$ and the coefficient of $g_{T,v}$ is $|T| - |S|/2$. The dual constraints correspond to primal variables $d_{uv}$ and $y_t$.

In the primal, the variable $d_{uv}$ has a coefficient of $-1$ for the constraints corresponding to the dual variables $f_{Puv}$ for $P \in \mathcal{P}(u,v)$, and a coefficient of 1 in the constraints corresponding to the dual variables $g_{T,t}$ for $t \in \{u,v\}$ and $T \ni t$. Similarly, the variable $y_t$ has a coefficient of 1 for the constraints corresponding to the dual variables $f_{Puv}$ for $u,v \in V$ and $P \in \mathcal{P}(u,v)$ such that $t \in V(P)$. Thus, LP (6) is the dual of LP (5).

\[\text{(flow-LP}(x, S)) \quad \text{maximize} \quad \sum_{T \subseteq S, v \in T} g_{T,v}(|T| - |S|/2) - \sum_{u,v \in V, P \in \mathcal{P}(u,v)} f_{Puv} \left( \sum_{e \in P} x_e \right) \]
subject to \[
\sum_{T \subseteq V: u,v \in T} g_{T,u} + g_{T,v} \leq \sum_{P \in \mathcal{P}(u,v)} f_{Puv} \quad \forall u, v \in V
\]
\[
\sum_{u,v \in V} \sum_{P \in \mathcal{P}(u,v): t \in V(P)} f_{Puv} \leq 1 \quad \forall t \in V
\]
(6)

**Separation oracle for $\mathcal{F}(S)$**. Given a solution $(f, g)$ to flow-LP$(x, S)$, we denote its objective value as flow-LP$(x, S, f, g)$. LP duality implies the following lemma.

**Lemma 5.4.** $x \in \mathcal{F}(S)$ if and only if flow-LP$(x, S, f, g) \leq w$ for all feasible dual solutions $(f, g)$.

Let us observe what we have achieved so far. The expression flow-LP$(x, S, f, g) \leq w$ is a linear constraint on $x$, whose coefficients $(f, g)$ are feasible points in the polytope given by (6). Now we crucially note that the coefficients of the constraints on $(f, g)$ depend only on the topology of the graph $G$ and not on $x$.

**Definition 5.5.** We say that $(f, g)$ is $S$-valid if it satisfies the constraints of flow-LP$(x, S)$.

Thus, Lemma 5.4 gives a description of $\mathcal{F}(S)$ in terms of linear inequalities. Together with Lemma 5.2, we get a description of $\mathcal{F}$ in terms of linear inequalities and so we can reformulate LP (4) as follows.

\[
\text{minimize} \quad \sum_{(u,v) \in E} x_{uv} \quad \text{subject to} \quad \text{flow-LP}(x, S, f, g) \leq w \quad \forall S \subseteq V, (f, g) \text{ $S$-valid}
\]
5.2 Round or Separate

We will work with the reformulated LP (7). One immediate obstacle is that it is unclear how to get an efficient separation oracle for $\mathcal{F}$. In fact, even for a fixed vertex set $S$, it is not immediately clear how to find a violated inequality for $\mathcal{F}(S)$. The problem is that for any constraint, the coefficients of the $x$-variables are based on $f, g$, which need to satisfy LP (8), and it is not clear how to generate them; e.g. there are exponentially many $f, g$ variables in flow-LP($x, S$). To get over these problems, we do the following.

We apply the Round-or-Separate framework with the Interdict algorithm in Section 4. Roughly speaking, we start with some candidate solution $x$ (possibly infeasible), and try to construct a tree decomposition of width $O(w \log w)$ using Interdict. If we succeed, we are done. Otherwise, Interdict could not find a small balanced separator for some vertex set $S$, and this can only happen if $x \notin \mathcal{F}(S)$. Later, in Lemma 5.6, we give an efficient procedure that given $x$ and $S$, determines whether $x \in \mathcal{F}(S)$ and if so, outputs a solution $(y^S, d^S)$ that is feasible to sep-LP($x, S$) and satisfies $\sum_v y^S_v \leq w$; otherwise, it outputs a separating hyperplane, i.e. an $S$-valid $(f, g)$ such that flow-LP($x, S, f, g$) $> w$. In the first case, Interdict can use $(y^S, d^S)$ to find a $O(w \log w)$-size separator of $S$ and make progress. In the second case, we can add the separating hyperplane to find a new candidate $x$ and repeat the whole tree decomposition procedure. By the standard Ellipsoid method-based, separation-versus-optimization framework, the number of such iterations is polynomially bounded.

It remains to show how to efficiently separate $\mathcal{F}(S)$ for any given subset $S$. We will do this indirectly by using the Ellipsoid Method and applying duality to sep-LP($x, S$).

**Lemma 5.6.** There exists a polynomial-time algorithm that given $x$ and $S$, determines whether $x \in \mathcal{F}(S)$, and if so, outputs a solution $(y^*, d^*)$ that is feasible to sep-LP($x, S$) and satisfies $\sum_v y^*_v \leq w$; otherwise, it finds a violated inequality for LP (7).

**Proof.** We apply the Ellipsoid method to find an optimal solution $(y^*, d^*)$ to sep-LP($x, S$). We can do this efficiently by using a separation oracle that uses Dijkstra’s algorithm to determine the distances $d$ given $x$ and $y$. If $\sum_v y^*_v \leq w$, then $x \in \mathcal{F}(S)$ by definition.

On the other hand, if $\sum_v y^*_v > w + \epsilon$ (where $\epsilon$ can be made exponentially small), then we can find a violated inequality for (7) as follows. As the Ellipsoid method added only polynomially many constraints we can re-solve this LP only on these added constraints and assume that $y^*$ is a solution to this smaller LP on only polynomially many constraints. By complementary slackness, there exists an optimal dual solution $(f^*, g^*)$ where the only non-zero dual variables are those that correspond to these polynomially many primal constraints. Thus, it suffices to solve flow-LP($x, S$) restricted to these dual variables which makes flow-LP($x, S$) polynomial in size. The objective function in flow-LP then gives the violated inequality for LP (7).

Acknowledgements

We would like to thank Marcin Pilipczuk and Fedor Fomin for many useful discussions and pointers. Part of this work was done when all the authors were visiting the Simons Institute for the Theory of Computing.
A Deferred Proofs

Lemma A.1 (Lemma 4.3 Restated). For every vertex $s$, we can efficiently find a radius $r \in [0, 1/12]$ such that the boundaries of the region $C(s, r)$ satisfies the following:

$$|\delta_x(s, r)| \leq 48 \cdot \ln \frac{e \cdot \text{vol}_x(s, 1/12)}{\text{vol}_x(s, r)} \cdot \ln [e(n + 1)] \cdot \text{vol}_x(s, r),$$

$$|\delta_y(s, r)| \leq 48 \cdot \ln (w^2 + 1) \cdot \text{vol}_y(s, r).$$

Proof. Say that $r$ is $x$-good if it satisfies the first inequality and $y$-good if it satisfies the second inequality. Define the following sets of radii:

$$A_x = \left\{ r \in [0, 1/12] : \frac{|\delta_x(s, r)|}{\text{vol}_x(s, r)} > 48 \cdot \ln \frac{e \cdot \text{vol}_x(s, 1/12)}{\text{vol}_x(s, r)} \cdot \ln [e(n + 1)] \right\},$$

$$A_y = \left\{ r \in [0, 1/12] : \frac{|\delta_y(s, r)|}{\text{vol}_y(s, r)} > 48 \cdot \ln (w^2 + 1) \right\}.$$

In other words, $A_x$ and $A_y$ are the sets of radii that are $x$-bad and $y$-bad, respectively. We claim that the measure of both $A_x$ and $A_y$ are small: $\mu(A_x), \mu(A_y) \leq 1/48$. The claim then implies that there exists $r \in [0, 1/12]$ that is simultaneously $x$-good and $y$-good.

Observe that $\frac{\partial \text{vol}_x(s, r)}{\partial r} = |\delta_x(s, r)|$ and $\frac{\partial \text{vol}_y(s, r)}{\partial r} = |\delta_y(s, r)|$. Suppose, towards a contradiction, that $\mu(A_x) > 1/48$. We have

$$\frac{\partial}{\partial r} \left(- \ln \frac{e \cdot \text{vol}_x(s, 1/12)}{\text{vol}_x(s, r)}\right) = \frac{|\delta_x(s, r)|}{\text{vol}_x(s, r)} \cdot \ln \frac{e \cdot \text{vol}_x(s, 1/12)}{\text{vol}_x(s, r)}$$

so

$$\ln \frac{e \cdot \text{vol}_x(s, 1/12)}{\text{vol}_x(s, 0)} = -\ln \frac{e \cdot \text{vol}_x(s, 1/12)}{\text{vol}_x(s, r)} \bigg|_{0}^{1/12} = \int_{0}^{1/12} \frac{|\delta_x(s, r)|}{\text{vol}_x(s, r)} \ln \frac{e \cdot \text{vol}_x(s, 1/12)}{\text{vol}_x(s, r)} \partial r$$

$$\geq \mu(A_x) \cdot 48 \ln [e(n + 1)] \geq \frac{1}{48} \cdot 48 \ln [e(n + 1)] = \ln [e(n + 1)].$$

But

$$\ln \frac{e \cdot \text{vol}_x(s, 1/12)}{\text{vol}_x(s, 0)} \leq \ln \frac{e \cdot \text{vol}_x(s, 1/12)}{\text{vol}_x(s, 0)} = \ln [e(n + 1)].$$

Thus, we have a contradiction and so $\mu(A_x) \leq 1/48$.

We now turn to bounding $\mu(A_y)$. Suppose, towards a contradiction, that $\mu(A_y) > 1/48$.

$$\ln \frac{\text{vol}_y(s, 1/48)}{\text{vol}_y(s, 0)} = \int_{0}^{1/12} \frac{1}{\text{vol}_y(s, r)} \frac{\partial \text{vol}_y(s, r)}{\partial r} \partial r = \int_{0}^{1/12} \frac{|\delta_y(s, r)|}{\text{vol}_y(s, r)} \partial r$$

$$\geq \mu(A_y) \cdot 48 \ln (w^2 + 1) \geq \frac{1}{48} \cdot 48 \ln (w^2 + 1) = \ln (w^2 + 1).$$

But

$$\ln \frac{\text{vol}_y(s, 1/48)}{\text{vol}_y(s, 0)} \leq \ln \frac{w + 1/w}{1/w} = \ln (w^2 + 1).$$

Therefore, both $\mu(A_x), \mu(A_y) \leq 1/48$ and so there exists $r \in [0, 1/12]$ that is simultaneously $x$-good and $y$-good.
To find such an \( r \) efficiently, note that as \( r \) grows, \( \text{vol}_x(s, r) \) and \( \text{vol}_y(s, r) \) are non-decreasing. Thus, we only need to check the condition at points \( r \) when either \( \delta_x(s, r) \) and \( \delta_y(s, r) \) changes, which happens at most \( 2|V'| \) and \( 2|E'| \) times, respectively. This is because as \( r \) grows, once a vertex leaves \( \delta_y(s, r) \), it is inside \( B(s, r) \) and will never reappear in \( \delta_y(s, r) \); and once an edge leaves \( \delta_x(s, r) \), both of its endpoints are inside \( B(s, r) \) and the edge will never reappear in \( \delta_y(s, r) \).

### Lemma A.2 (Lemma 4.7 Restated)

Let \( F \) be the set of edges deleted by \( \text{INTERDICT}(G, S_0) \). Then, \( |F| \leq O(\log n \log \log n) \text{LP-cost}(G) \).

**Proof.** Let \( k \) be the recursion depth of \( \text{INTERDICT}(G, S_0) \). For each depth \( j \), let \( H_j \) be the collection of subgraphs that \( \text{INTERDICT} \) recursed on at depth \( j \), and for each \( H \in H_j \), let \( R(H) \) be the set of regions found by \( \text{PARTITION} \) when \( \text{INTERDICT} \) recursed on \( H \). (Note that at depth 0, we have \( H_0 = \{ G \} \).) By Lemma 4.5, we have the following bound on \( |F| \):

\[
|F| \leq O(\log \log n) \sum_j \sum_{H \in H_j} \sum_{B(s, r) \in R(H)} \ln \left( \frac{\text{vol}_x(H)}{\text{vol}_x(s, r)} \right) \text{vol}_x(s, r). \tag{8}
\]

For each edge \( (u, v) \in E \), define \( g(u, v) = x_{uv} + \frac{\text{LP-cost}(G)}{n^2} \). We have that

\[
\sum_{(u, v) \in E} g(u, v) \leq 2 \text{LP-cost}(G). \tag{9}
\]

We now show that \( \sum_j \sum_{H \in H_j} \sum_{B(s, r) \in R(H)} \ln \left( \frac{\text{vol}_x(H)}{\text{vol}_x(s, r)} \right) \text{vol}_x(s, r) \leq O(\log n) \sum_{(u, v) \in E} g(u, v) \).

Let us first see why the total amount charged per region \( B(s, r) \in R(H) \) is at least \( \ln \left( \frac{\text{vol}_x(H)}{\text{vol}_x(s, r)} \right) \text{vol}_x(s, r) \). Let \( E_H(s, r) \) be the edges with at least one endpoint in \( B(s, r) \). This is exactly the set of edges that contribute to the region’s \( x \)-volume \( \text{vol}_x(s, r) \). Therefore,

\[
\sum_{(u, v) \in E_H(s, r)} g(u, v) \geq \sum_{(u, v) \in E_H(s, r)} x_{uv} + \frac{\text{LP-cost}(G)}{n^2} \geq \text{vol}_x(s, r),
\]

and so the total amount charged is at least \( \ln \left( \frac{\text{vol}_x(H)}{\text{vol}_x(s, r)} \right) \text{vol}_x(s, r) \). Overall, we have

\[
\sum_j \sum_{H \in H_j} \sum_{B(s, r) \in R(H)} \ln \left( \frac{\text{vol}_x(H)}{\text{vol}_x(s, r)} \right) \text{vol}_x(s, r) \leq \sum_{(u, v) \in E} \phi(u, v), \tag{10}
\]

where \( \phi(u, v) \) is the total charge received by \( (u, v) \).

Fix an edge \( (u, v) \in E \). Let us now upper bound \( \phi(u, v) \). Since \( \text{INTERDICT} \) recurses on edge-disjoint subgraphs, \( (u, v) \) is contained in at most one subgraph of \( H_j \) for each depth \( j \), and if it does not belong to a subgraph of \( H_j \), then it does not belong to any subgraph of \( H_j' \) at lower depths \( j' > j \). In the worst case, \( (u, v) \) is contained in some subgraph \( H_j \in H_j \) for every depth \( j \). Let us assume this is so and define \( B(s_j, r_j) \in R(H_j) \) to be the region containing \( (u, v) \). Lemma 4.5 tells...
us that all of $H_j$ is contained in the region $B(s_{j-1}, r_{j-1})$ for each $j$ and so $\text{vol}_x(s_j, r_j) \geq \text{vol}_x(H_{j+1})$. Thus,

$$\phi(u, v) = \sum_{j=0}^{k} \ln \left( \frac{\text{vol}_x(H_j)}{\text{vol}_x(s_j, r_j)} \right) g(u, v) \leq \sum_{j=0}^{k} \ln \left( \frac{\text{vol}_x(H_j)}{\text{vol}_x(H_{j+1})} \right) g(u, v)$$

$$
\leq \ln \left( \frac{\text{vol}_x(G)}{\text{vol}_x(H_{k+1})} \right) g(u, v) \leq \ln(n^2 + 1) g(u, v),
$$

where the last inequality follows from the fact that $x$-volume is always at least $\frac{\text{LP-cost}(G)}{n^2}$.

Combining this with Inequality (10), we get

$$\sum_j \sum_{H \in \mathcal{H}_j} \sum_{B(s, r) \in R(H)} \ln \left( \frac{\text{vol}_x(H)}{\text{vol}_x(s, r)} \right) \text{vol}_x(s, r) \leq \sum_{(u, v) \in E} \ln(n^2 + 1) g(u, v) \leq 2 \ln(n^2 + 1) \text{LP-cost}(G),$$

where the last inequality follows from Inequality (9). Finally, plugging this into the right hand side of (8) gives us $|F| \leq O(\log n \log \log n) \text{LP-cost}(G)$, as desired.

\vspace{1em}

References

[1] Ittai Abraham, Cyril Gavoille, Anupam Gupta, Ofer Neiman, and Kunal Talwar. Cops, robbers, and threatening skeletons: padded decomposition for minor-free graphs. In *Symposium on Theory of Computing, STOC*, pages 79–88, 2014.

[2] Noga Alon, Paul Seymour, and Robin Thomas. A separator theorem for nonplanar graphs. *Journal of the American Mathematical Society*, 3(4):801–808, 1990.

[3] Eyal Amir. Approximation algorithms for treewidth. *Algorithmica*, 56(4):448–479, 2010.

[4] Hyung-Chan An, Mohit Singh, and Ola Svensson. Lp-based algorithms for capacitated facility location. In *Foundations of Computer Science, FOCS*, pages 256–265, 2014.

[5] Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2), 2009.

[6] Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM (JACM)*, 41(1):153–180, 1994.

[7] Daniel Bienstock and Nuri Özbay. Tree-width and the sherali-adams operator. *Discrete Optimization*, 1(1):13–21, 2004.

[8] Hans L. Bodlaender. Dynamic programming on graphs with bounded treewidth. In *ICALP*, pages 105–118, 1988.

[9] Hans L. Bodlaender, John R. Gilbert, Hjálmtýr Hafsteinsson, and Ton Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *J. Algorithms*, 18(2):238–255, 1995.

[10] Chandra Chekuri and Anastasios Sidiropoulos. Approximation algorithms for euler genus and related problems. In *Foundations of Computer Science, FOCS*, pages 167–176, 2013.
[11] Julia Chuzhoy, Yury Makarychev, and Anastasios Sidiropoulos. On graph crossing number and edge planarization. In SODA 2011, pages 1050–1069, 2011.

[12] Bruno Courcelle and Joost Engelfriet. Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach, volume 138 of Encyclopedia of mathematics and its applications. Cambridge University Press, 2012.

[13] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Parameterized Algorithms. Springer, 2015.

[14] D. Erik Demaine and MohammadTaghi Hajiaghayi. Encyclopedia of Algorithms, chapter Approximation Schemes for Planar Graph Problems, pages 59–62. Springer US, 2008.

[15] Erik D. Demaine and MohammadTaghi Hajiaghayi. The bidimensionality theory and its algorithmic applications. Comput. J., 51(3):292–302, 2008.

[16] Erik D. Demaine, MohammadTaghi Hajiaghayi, and Ken-ichi Kawarabayashi. Algorithmic graph minor theory: Decomposition, approximation, and coloring. In FOCS 2005, pages 637–646, 2005.

[17] Matt DeVos, Guoli Ding, Bogdan Oporowski, Daniel P. Sanders, Bruce Reed, Paul Seymour, and Dirk Vertigan. Excluding any graph as a minor allows a low tree-width 2-coloring. Journal of Combinatorial Theory, Series B, 91(1):25–41, 2004.

[18] D. Eppstein. Graph families. Algorithmica, 27(3):275–291, 2000.

[19] Guy Even, Joseph (Seffi) Naor, Satish Rao, and Baruch Schieber. Divide-and-conquer approximation algorithms via spreading metrics. Journal of the ACM (JACM), 47(4):585–616, 2000.

[20] Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee. Improved approximation algorithms for minimum weight vertex separators. SIAM J. Comput., 38(2):629–657, 2008.

[21] Uriel Feige and Shlomo Jozeph. Separation between estimation and approximation. In ITCS 2015, pages 271–276, 2015.

[22] Uriel Feige and Joe Kilian. Heuristics for finding large independent sets, with applications to coloring semi-random graphs. In Foundations of Computer Science, FOCS, pages 674–683, 1998.

[23] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar f-deletion: Approximation, kernelization and optimal FPT algorithms. In Foundations of Computer Science, FOCS, pages 470–479, 2012.

[24] Avinatan Hassidim, Jonathan A. Kelner, Huy N. Nguyen, and Krzysztof Onak. Local graph partitions for approximation and testing. In Foundations of Computer Science, FOCS, pages 22–31, 2009.

[25] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In Foundations of Computer Science, FOCS, pages 627–636, 1996.

[26] Sanjeev Khanna and Rajeev Motwani. Towards a syntactic characterization of ptas. In Symposium on Theory of Computing, STOC '96, pages 329–337, 1996.
[27] Philip N. Klein, Serge A. Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. In *Symposium on Theory of Computing, STOC*, pages 682–690, 1993.

[28] Retsef Levi, Andrea Lodi, and Maxim Sviridenko. Approximation algorithms for the capacitated multi-item lot-sizing problem via flow-cover inequalities. *Math. Oper. Res.*, 33(2):461–474, 2008.

[29] Richard J. Lipton and Robert E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

[30] Avner Magen and Mohammad Moharrami. Robust algorithms for on minor-free graphs based on the sherali-adams hierarchy. In *APPROX*, pages 258–271, 2009.

[31] Konstantin Makarychev, Yury Makarychev, and Aravindan Vijayaraghavan. Approximation algorithms for semi-random partitioning problems. In *Symposium on Theory of Computing, STOC*, pages 367–384, 2012.

[32] Dániel Marx. On the optimality of planar and geometric approximation schemes. In *Foundations of Computer Science, FOCS*, pages 338–348, 2007.

[33] B A Reed. Tree width and tangles: A new connectivity measure and some applications. In R. A. Bailey, editor, *Surveys in Combinatorics, 1997*, pages 87–162. Cambridge University Press, 1997. Cambridge Books Online.

[34] Neil Robertson and Paul D. Seymour. Graph minors .xiii. the disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995.

[35] Paul D. Seymour. Packing directed circuits fractionally. *Combinatorica*, 15(2):281–288, 1995.

[36] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, 2004.

[37] Yu Wu, Per Austrin, Toniann Pitassi, and David Liu. Inapproximability of treewidth and related problems. *J. Artif. Intell. Res. (JAIR)*, 49:569–600, 2014.