White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of Leeds.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/221122/

Version: Accepted Version

Proceedings Paper:
Lokshtanov, D., Panolan, F., Saurabh, S. et al. (2025) Crossing Number in Slightly Superexponential Time (Extended Abstract). In: Azar, Y. and Panigrahi, D., (eds.) Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). ACM-SIAM Symposium on Discrete Algorithms (SODA25), 12-15 Jan 2025, New Orleans, Louisiana, U.S.A.. SIAM, pp. 1412-1424. ISBN: 978-1-61197-832-2.

https://doi.org/10.1137/1.9781611978322.44

This is an author produced version of a proceedings paper published in ACM-SIAM Symposium on Discrete Algorithms (SODA25), made available under the terms of the Creative Commons Attribution License (CC-BY), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

Reuse
This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here: https://creativecommons.org/licenses/

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.



Crossing Number in Slightly Superexponential Time (Extended Abstract)

Daniel Lokshtanov* Fahad Panolan† Saket Saurabh‡ Roohani Sharma§ Jie Xue¶
Meirav Zehavi‖

Abstract

A *drawing* of an (undirected) graph G is a mapping ϕ that assigns to each vertex a distinct point in the plane and to each edge uw a continuous curve $\phi(uv)$ in the plane from $\phi(u)$ to $\phi(v)$, not passing through the image of any other vertex. Two edges e and f *cross* in a point p if $p \in \phi(e) \cap \phi(f)$ and p is not the image of a vertex of G. In a drawing no three edges are allowed to cross in the same point. The *crossing number* of a drawing of G is the number of points p such that some two edges e and f cross in p. In the Crossing Number problem, the input consists of a graph G and integer k. The task is to determine whether there exists a drawing of G with crossing number at most k, and to output such a drawing if it exists.

Grohe [STOC 2001, JCSS 2004] gave an algorithm for Crossing Number with running time $f(k)n^2$ where $f(k) = 2^{2^{2^{2^{\Omega(k)}}}}$. He conjectured that there exists an algorithm with running time $2^{\mathcal{O}(k)}n$. Kawarabayashi and Reed [STOC 2007] outlined an algorithm with running time $f(k)n$ where $f(k) = 2^{2^{2^{\Omega(k)}}}$. Combining the main combinatorial lemma by Kawarabayashi and Reed with the recent algorithm for Crossing Number parameterized treewidth plus k by de Verdière and Magnard [ESA 2021] would yield a running time of $f(k)n$ where $f(k) = 2^{\mathcal{O}(k^4 \log k)}$. This still falls far away from the dependency on k in the conjecture by Grohe. Furthermore, critical details of the proof of the correctness of the algorithm of Kawarabayashi and Reed, and, in particular, of the aforementioned combinatorial lemma, have never been published.

In this work, we give an algorithm with running time $2^{\mathcal{O}(k \log k)}n$. Thus, our algorithm resolves Grohe's 23-year old conjecture up to a logarithmic factor in k in the exponent. .

1 Introduction

A *drawing* φ of a graph G is a drawing of G on the Euclidean plane where each vertex of G is mapped by φ to a point and each edge e of G is mapped by φ to a (continuous) curve between its endpoints, such that: *(i)* for every two distinct vertices $v, v' \in V(G)$, $\varphi(v) \neq \varphi(v')$; *(ii)* for every two distinct edges $e, e' \in E(G)$, $\varphi(e)$ and $\varphi(e')$ intersect in only a finite number of points, and for every point p in which $\varphi(e)$ and $\varphi(e')$ intersect, there does not exist an edge $e'' \in E(G) \setminus \{e, e'\}$ such that $\varphi(e'')$ intersects p; *(iii)* for every vertex $v \in V(G)$ and edge $e \in E(G)$, $\varphi(v)$ is not an internal point of $\varphi(e)$. With respect to a given drawing φ of a graph G, a *crossing* is a point p such that there exist two distinct edges $e, e' \in E(G)$ that satisfy the following condition: the point p is an internal points of both $\varphi(e)$ and $\varphi(e')$. The number of crossings in φ is denoted by $\mathsf{cr}(\varphi)$.

The *crossing number* of a graph G, denoted by $\mathsf{cr}(G)$, is the minimum number of crossings in a drawing of G. Sometimes drawings are further restricted so that each pair of edges can only have one crossing. Importantly, this restriction does not increase the crossing number of the graph because every crossing-minimal drawing satisfies it [38]. Computationally, the input of the Crossing Number problem is a simple[1] graph G on n vertices and a non-negative integer $k \in \mathbb{N}_0$, and the task is to decide whether $\mathsf{cr}(G) \leq k$, and, if the answer is positive, then output a drawing φ of G with $\mathsf{cr}(\varphi) = \mathsf{cr}(G)$. Sometimes the problem is formulated such that if the answer

*University of California, Santa Barbara, USA, daniello@ucsb.edu. Supported by NSF award CCF-2008838.

†University of Leeds, UK, F.panolan@leeds.ac.uk.

‡The Institute of Mathematical Sciences, HBNI, Chennai, India, saket@imsc.res.in. Supported by the European Research Council (ERC) grant no. 819416, and Swarnajayanti Fellowship (no. DST/SJF/MSA01/2017-18).

§University of Bergen, Norway, r.sharma@uib.no.

¶New York University Shanghai, China, jiexue@nyu.edu.

‖Ben-Gurion University, Beersheba, Israel, meiravze@bgu.ac.il. Supported by the European Research Council (ERC) grant no. 101039913.

[1]Later, after we bound $|E(G)|$ using a simple argument (only once, when the input is first given), we slightly abuse notation for the sake of readability and allow G to not be simple.

Copyright © 2025 by SIAM
Unauthorized reproduction of this article is prohibited

is positive, then one only needs to output a drawing φ of G with $\mathsf{cr}(\varphi) \leq k$. Up to a $\mathcal{O}(k)$-factor in the time complexity (or $\mathcal{O}(\log k)$ using binary search), these formulations are equivalent.[2]

The crossing number of a graph G is a natural measure of how close G is to being planar (indeed, G is planar if and only if $\mathsf{cr}(G) = 0$). The notion of the crossing number of a graph was first considered in 1940 by Turán [41] for bipartite graphs in the context of the minimization of the number of crossings between tracks connecting brick kilns to storage sites.

The problem has applications to VLSI circuit layouts [5, 35, 36] and is very well studied both from the algorithmic and purely combinatorial perspectives. At least one book [40] (also see [39, 37]) has been fully devoted to the study of crossing numbers, and yet the problem is very far from being well-understood. Even obtaining tight bounds on the crossing number of complete graphs and complete bipartite graphs remain a well-known open problems in mathematics [15].

1.1 Known Results. The Crossing Number problem was shown NP-complete by Garey and Johnson in 1983 [22], and it remains NP-complete on graphs of maximum degree 3 [26], on graphs which can be made planar by removing of at most one edge [9], and even on graphs which can be made planar by removing at most one edge and have at most constantly many vertices of degree greater than 3 [31, 27] (see also [21], Section 3.2). Very recently, Crossing Number was shown to be NP-hard also when restricted to graphs of pathwidth 12 (or treewidth 9), and therefore it is para-NP-hard parameterized by pathwidth (or treewidth) alone [29].

From the perspective of approximation algorithms, it is known that there exists a fixed constant $\epsilon > 0$ such that $(1+\epsilon)$-approximating the crossing number of a graph is NP-hard [7]. No non-trivial approximation algorithms are known on general graphs; the most general known results give approximation algorithms whose approximation ratio is bounded in terms of the number n of vertices and the maximum degree Δ of the graph [12]. The current best such approximation algorithm by Chuzhoy and Tan [13] achieves an approximation ratio of $2^{\mathcal{O}(\log^{\frac{7}{8}} n \log\log n)}\Delta^{\mathcal{O}(1)}$. Better approximation algorithms were known for some restricted graph classes, such as graphs embeddable on a fixed surface [23, 11] and graphs from which few edges or vertices can be removed to make them planar [3, 8, 10, 30].

In this paper, we focus on parameterized algorithms for the Crossing Number problem, specifically on algorithms whose running time is allowed to depend super-polynomially on the number k of crossings (see the textbooks [14, 19, 20] for an introduction to parameterized algorithms). It is easy to design an algorithm with running time $n^{\mathcal{O}(k)}$ by guessing which pairs of edges cross (and in which order) in the drawing of G. In their monograph, Downey and Fellows [18] posed whether the problem is *fixed parameter tractable* (FPT), that is, admits an algorithm with running time $f(k)n^{\mathcal{O}(1)}$. This question was resolved in the affirmative by Grohe [24] who gave an algorithm with running time $f(k)n^2$. The running time dependence $f(k)$ of the algorithm of Grohe is an exponential tower of height at least 4. Grohe conjectured that both the dependence on k and the dependence on n can be substantially improved simultaneously, all the way to $2^{\mathcal{O}(k)}n$. In 2007, Kawarabayashi and Reed [34] outlined an algorithm with running time $f(k)n$, that is, they claimed a linear time dependence on n with an unspecified (and also at least triple-exponential)[3] running time dependence on k. Unfortunately, the manuscript [34] only presents a sketch of the proof—the critical details have never been published, shedding doubt on its validity.

Recently, de Verdière and Magrand [16] proved the following theorem:[4]

Theorem 1.1. ([16]) *The* Crossing Number *problem is solvable in time* $2^{\mathcal{O}((k+w)\log(k+w))} \cdot n$ *where* w *is the treewidth of the input graph.*

Replacing the invocations of Courcelle's Theorem in [24] and [34] with the algorithm in Theorem 1.1 leads to algorithms with running time dependencies on k of $2^{2^{\mathcal{O}(k)}}$ and $k^{\mathcal{O}(k^4)}$, respectively. Yet, as we argued above, the second result cannot be substantiated due to its missing proof details.

[2]Indeed, to minimize the number of crossings of the output φ, simply iterate over every $k' = 0, 1, \ldots, k$, terminating with the first φ that is returned.

[3]The fact that the dependencies are exponential towers of heights 4 and 3, respectively, can be derived from the formula that is used in order to apply Courcelle's theorem in these manuscripts.

[4]This result is implicit in [16], who study a different problem. However, it follows from a reduction in [17]. We comment that we do not use this theorem in our work, but prove a generalization required for our purposes.

Copyright © 2025 by SIAM
Unauthorized reproduction of this article is prohibited

On the negative side, Hlinený and Dernár [28] showed that the problem does not admit a polynomial kernel unless coNP $\subseteq$ NP/poly. When parameterized by the vertex cover number, Crossing Number was shown to admit an FPT algorithm by Hliněný and Sankaran [32]. This algorithm was later modified to resolve the more general problem of completing a partially specified drawing of G with at most k crossings by Hamm and Hliněný [25].

1.2 Our Contribution.

We prove the following theorem.

Theorem 1.2. (**Main Theorem**) *The* Crossing Number *problem is solvable in time* $2^{\mathcal{O}(k \log k)} \cdot n$.

The algorithm of Theorem 1.2 comes quite close to completely resolving the 23-years old conjecture of Grohe; the running time of the algorithm is just off by a factor of $\log k$ in the exponent. Our algorithm is inspired by the same general template as the previous two parameterized algorithms for Crossing Number: first reduce the treewidth of G to a function of k and then apply an algorithm for Crossing Number parameterized by k and the treewidth w of the input graph. Most importantly, our algorithm critically differs from the previous ones in the implementation of the first step, that is, the treewidth reduction step. Having a radically new approach for treewidth reduction, here, is *essential* as the previous ones hit a provable barrier, as we describe below. We believe that this barrier is the reason why Grohe's conjecture has remained opened for more than two decades.

Notably, for us, we will not have a single application of the treewidth reduction step, followed by a single resolution of the case of a bounded-treewidth graph, but interleave the two steps for a (possibility large) number of times on various subgraphs of the input graph. Thus, for us, it will be meaningless to consider them as a first step and a second step. In fact, we will consider the "second" step before the "first" step (both in the Overview and in the manuscript itself), since it will make the paper much more readable—specifically, the treewidth reduction step will make repeated calls to the algorithm developed for the bounded-treewidth case, and will rely on concepts and statements that are more naturally introduced there. Next, for each of these two components, we briefly outline our contribution and compare our work with the literature on the subject.

Algorithm for Bounded Treewidth Graphs. Unfortunately, we cannot use Theorem 1.1 in a black box manner—we need to solve a more general and technical task (whose formal statement is omitted from this extended abstract). Specifically, we need to compute a small representative sample of (combinatorial representations of) partial drawings of *boundaried subgraphs* of G. More precisely, we will make repeated calls to an algorithm on a subgraph of the input graph that has some set of "special vertices" (termed a *boundary*), and we will need to "replace" this subgraph by a smaller subgraph in a way that, in particular, will not affect the answer (Yes or No) to the problem. We cannot do exactly that, but, instead, "replace" this subgraph by a set of smaller subgraphs, so that, at least one of them will be "good" in the aforementioned sense. It might be possible (though quite non-trivial) to generalize the ideas of de Verdière and Magrand [16] to replace at least some of the algorithmic steps that we perform towards our task. However, this would yield a convoluted solution. Instead, we present a direct algorithm to solve our more general task. In particular, throughout our proof, we make transparent use of the concept of representation.

The Treewidth Reduction Step: A Completely New Approach to Break the Barrier. For the treewidth reduction step, Grohe [24] proves that his upper bound on the treewidth w of the graph after the treewidth reduction is not better than $2^{\mathcal{O}(k)}$. Kawarabayashi and Reed [34] claim that they can reduce the treewidth of the input graph all the way down to $\mathcal{O}(k^4)$ (though the proof was left incomplete).

One might think that perhaps the authors of the previous works [24, 34] simply did not seriously optimize the dependence of the treewidth w on k in their treewidth reduction step, and that some modifications to their previous treewidth reduction steps could lead to a treewidth bound of $\mathcal{O}(k)$, thereby yielding a $2^{\mathcal{O}(k \log k)} n$ time algorithm for Crossing Number when combined with Theorem 1.1. However, there is an *impassable barrier* that prevents the treewidth reduction steps of Grohe [24] and Kawarabayashi and Reed [34] from producing graphs with treewidth $o(k^{3/2})$. Namely, both of the previous algorithms rely on repeatedly removing the middle vertex of a *flat* $\Omega(k)$-*wall* in G. To keep the treewidth reduction steps efficient, the previous algorithms do not remove *all* such vertices, but just enough to be able to prove their treewidth upper bounds. What if we pushed the previous treewidth reduction algorithms to the limit and were somehow able to remove all vertices that are in the middle of a $\Omega(k)$-wall? Even in that case, we would not be able to upper bound treewidth by $o(k^{3/2})$! To see this, consider the $k^{3/2} \times k^{3/2}$ grid split into $\sqrt{k} \times \sqrt{k}$ sub-grids such that each sub-grid is of size $k \times k$ and has a crossing. This

Copyright © 2025 by SIAM
Unauthorized reproduction of this article is prohibited

grid has crossing number k and treewidth $\Omega(k^{3/2})$, but no flat ck-wall for some constant c. Therefore, even with Theorem 1.1 at hand, it is impossible to attain the running time of Theorem 1.2 simply by a more exhaustive application of the main reduction rule of [24, 34], or a better analysis of the reduced graph.

Critically, *our treewidth reduction breaks the $k^{3/2}$ barrier.* We start by finding a set F of at most $2k$ edges such that $G-F$ is planar. Such a set can be found in time $2^{\mathcal{O}(k \log k)} \cdot n$ using the algorithm of Jansen, Lokshtanov and Saurabh [33].

Since $G - F$ is planar, we can apply the following (standard) trick. First, we computes an outer-planarity layering of $G - F$. We peel off the outer layer of $G - F$, giving all the vertices of the outer layer label 0. Then we peel off the next layer, giving those vertices label 1, and so on. We keep going until label $3(4k + 1)$; at this point, we re-start from label 0. Since at most $2k$ vertices are incident to edges that participate in crossings (with respect to a solution drawing of G, if one exists) and at most $2k$ vertices are incident to edges in F, there exist 3 consecutive labels $i, i + 1, i + 2$ such that no vertex with such a label is incident to a crossing edge or an edge in F. Let L be the set of all vertices labeled i, $i + 1$, or $i + 2$.

If $G - F$ is 3-connected, then we can prove a result that is, in a sense, a strengthening of the classic "cycle separator planarity criterion" (see Theorem 3.8 in [4]) to (essentially) show that (i) no edge of F connects two different components of $G-F-L$, and that (ii) each connected component of $G-L$ can be considered independently (substantial technical details swept under the rug here). Since each connected component of $G - L$ consists of at most $\mathcal{O}(k)$ consecutive outerplanarity layers (plus at most k edges from F), the treewidth of each component is at most $\mathcal{O}(k)$ based on a result by Bodlaender [6].

To handle the general case when $G - F$ is not 3-connected, we proceed by a two-level dynamic programming over the tree decomposition of $G-F$ into its 3-connected components. In the outer layer we consider a 3-connected component of $G-F$, assuming that all of its descendants in the tree have already been processed. We then reduce the treewidth of this component to $\mathcal{O}(k)$ by the method for 3-connected graphs, before we apply the bounded treewidth algorithm of Theorem 1.1 for the treewidth-reduced version of this 3-connected component to fill the entries of the DP table (of the outer DP layer) corresponding to this component. Actually, executing this plan requires us to handle a number of non-trivial technical difficulties, which we address formally in the technical sections.

Lastly, we note that our layering-step is inspired by Baker's technique [1]. However, generally speaking, Baker's technique is applied to problems closed under natural operations such as contractions. However, for Crossing Number, contraction can turn a yes-instance into a no-instance (and a no-instance into a yes-instance). This makes the application of Baker's technique challenging and quite different than usual.

1.3 Structure of the Extended Abstract. We defer the definition of well-known concepts and standard notations to the full version of this extended abstract. Also, for the sake of clarity of our overviews, we will (over-)simplify some of the technical details. First, in Section 2, we outline the proof of our generalization of Theorem 1.1. Then, in Section 3, we outline the proof of Theorem 1.2. Lastly, in Section 4, we conclude the extended abstract.

2 Overview for the Proof of Our Generalization of Theorem 1.1

Since the statement of our generalization of Theorem 1.1 required the introduction of several technical components, whose presentation will make the overview much less readable, we refrain from doing that. Instead, for intuition, suppose that our aim is to prove a generalization of Theorem 1.1 where, roughly speaking, instead of just solving the input graph, we aim to store a representation of it—that is, enough (but little) information on the input graph so that if it will be "extended" in the future, then the information that we have will suffice to solve the extension. This will be made more formal (and clearer) in the next two paragraphs.

The main concept used in our proof is that of the *representation* of a *boundaried subgraph* of an input graph G or boundaried graph (G, B_G). Here, a boundaried subgraph is simply a subgraph H of G with a *boundary* $B_H \subseteq V(H)$ satisfying $B_G \subseteq B_H$, which, intuitively, separates H from the rest of the graph: So, there do not exist edges between $V(H) \setminus B_H$ and $V(G) \setminus V(H)$, and $E(G[V(H) \setminus B_H]) \subseteq E(H)$ (i.e., all edges incident to vertices in $V(H) \setminus B_H$ in G are present in H). For context, given a tree decomposition $\mathcal{T} = (T, \beta)$ of G of width t and $x \in V(T)$, think of H as $\Gamma(x)$, the induced subgraph $G[\gamma(x)]$[5] from which we remove the following edges:

[5]The notation $\gamma(x)$ refers to the union of the bags of x and all of its descendants in T.

Copyright © 2025 by SIAM
Unauthorized reproduction of this article is prohibited

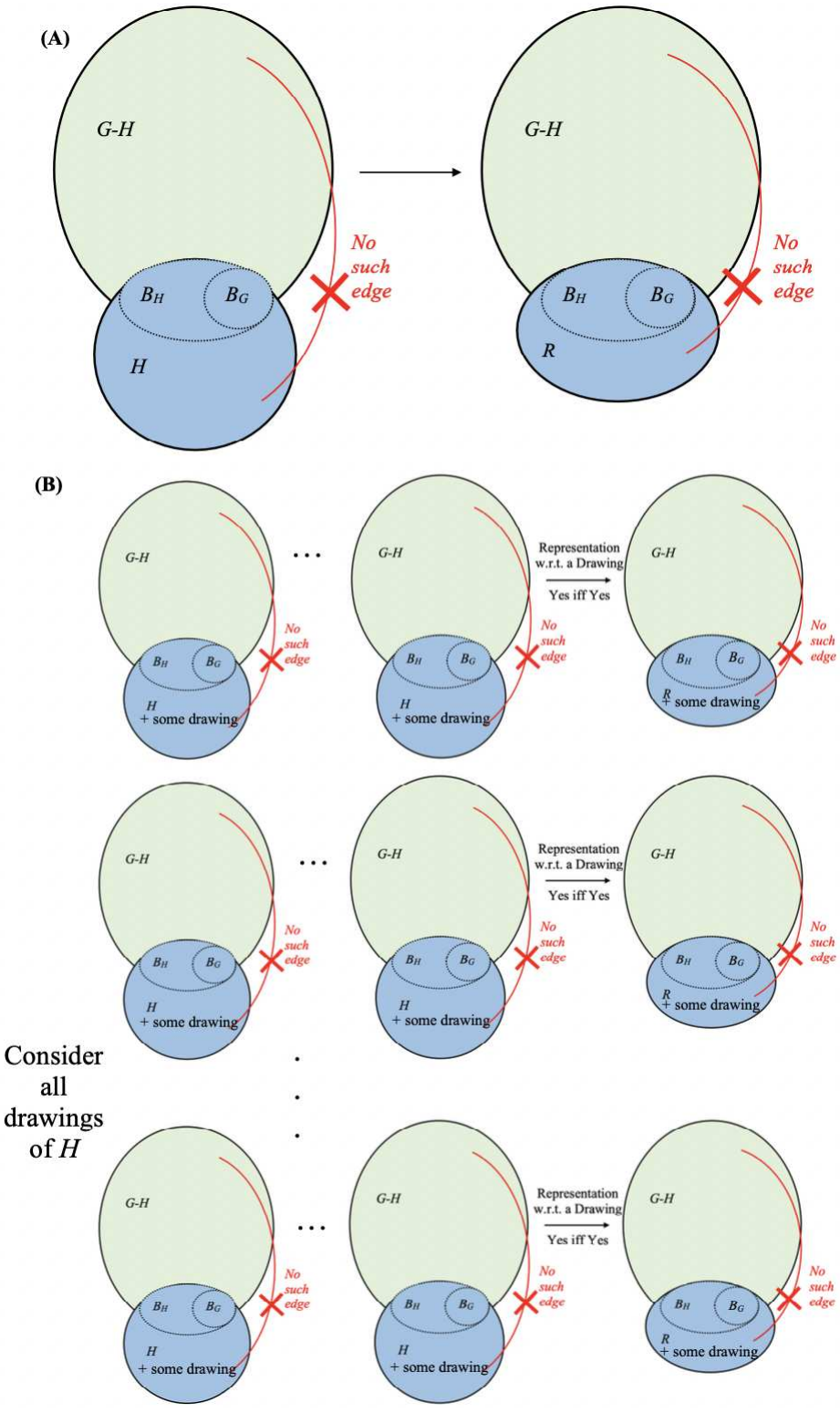

Figure 1: An abstraction of the notion of representation. (A) The "ideal" notion, which is not implemented. (B) The actual notion, which is implemented.

Copyright © 2025 by SIAM
Unauthorized reproduction of this article is prohibited

those with (i) both endpoints in $\beta(x)$, and (ii) at least one endpoint not in $\beta(p)$ for the parent p of x. So, $\Gamma(x)$ and $\Gamma(y)$ for siblings x and y do not share edges. Also, $B_H = (\beta(x) \cap \beta(p)) \cup B_G$ and denote $|B_H| = t$. Now, we would like to *represent* H in the following sense: We would like to obtain a new graph G' by "replacing" H in G with a "small" graph R (with $\mathcal{O}(k+t)$ vertices and edges) and k' by some $k' \leq k$ so that (G, k) is yes-instance if and only if (G', k') is a yes-instance. We refer to Fig. 1(A) for an illustration. Unfortunately, that is difficult—so, instead of producing a single pair (G', k') in this manner, we will produce a "small" (of size $2^{\mathcal{O}((k+t)\log(t+k))}$) collection of such pairs, so that (G, k) is a yes-instance if and only if at least one of them is.

Intuitively, for every possible drawing φ of H with crossing number at most k, we will have one representative pair (R, k') with a drawing φ' of R in our collection, so that φ can be extended to a drawing of G[6] with k crossings if and only if φ' can be extended to a drawing of G' with k' crossings. We refer to Fig. 1(B) for an illustration. Naturally, since our collection should be small, the mapping of drawings φ to triples (R, k', φ') will be far from injective. Next, to discuss this mapping, consider some specific drawing φ.

Observe that at most $2k$ edges (or, more precisely, $2k$ minus twice the crossing number of φ) drawn by φ should be crossed in the "future" (when drawing the edges in $E(G) \setminus E(H)$). For now, suppose that we know this set of edges, and call its complement F (the set of uncrossable edges). Practically, during the dynamic programming, for edges between vertices in the current bag of the tree decomposition, this information will be "guessed", while for edges between vertices in descendant bags, this information will be already encoded in the representatives for the (at most) two children bags. Then, we consider our four operations that reduce the size of H while preserving the property that if its drawing φ can be extended to a solution, then also the drawing of the reduced graph can be extended to a solution, and vice versa. For this overview, we only describe the modification of H, but we note that the drawing φ is also modified accordingly in the "expected" manner.

The four operation are, informally, defined as follows. We refer to Fig. 2 for an illustration.

1. Planarizing H by adding new vertices on crossing points, and decreasing k by the number of newly added vertices.
2. Contracting edges in F whose endpoints are not in the boundary B_H. Notice that this operation can create self-loops and parallel edges.
3. Deleting self-loops and parallel edges in F that do not correspond to closed curves separating B_H in a non-trivial way, such that no other "uncrossable closed curve" already does this job.
4. Deleting or dissolving vertices of degree at most 2 that do not belong to B_H and which are incident to neither edges not in F nor self-loops and parallel edges.

We prove that after exhaustive application of these operations, we end up with an "equivalent" graph with F and its own drawing φ', having only $\mathcal{O}(t+k)$ vertices and edges. This irreducible triple is said to represent (H, F, φ).

To illustrate how to use this reduction notion algorithmically, consider some boundaried graph (H, B_H), and suppose that we want to find a collection that represents it. Additionally, suppose that we already have such collections for boundaried subgraphs of H that are, in some sense, "non-conflicting". Naively, we can obtain the desired collection by iterating over every tuple corresponding to choosing one representative per boundaried subgraph (representing some possible drawings of it), and for each tuple, doing all the replacements (simultaneously). Afterwards, for the part of H that is still not drawn (as it did not belong to any boundaried subgraph that was replaced), we brute-force over every possibility to draw it with at most k crossings—it can be shown that, if this part is "small" (say, of size r), there are not "too many" such options (specifically, $(r+k)^{\mathcal{O}(r+k)}$). Lastly, for every possibility, we perform the reduction operations to get a single representative. The representing collection is then composed of all representatives obtained in this manner (removing duplicates). Practically, we substitute this naive approach by dynamic programming, since we might need to consider a large number of boundaried subgraphs of H, which will make the naive approach highly inefficient.[7]

[6]Or, more generally, we can consider here any graph that can be obtained by "gluing" (H, B_H) with another boundaried graph with boundary B_H.

[7]If we consider the context of a graph of bounded treewidth mentioned in the first paragraph in this overview, then we can ensure that only two boundaried subgraphs (corresponding to the two children of x, assuming a binary tree decomposition) are to be considered. However, to prove our main theorem, we will need to consider more general contexts, where this simplification cannot be applied.

Copyright © 2025 by SIAM
Unauthorized reproduction of this article is prohibited

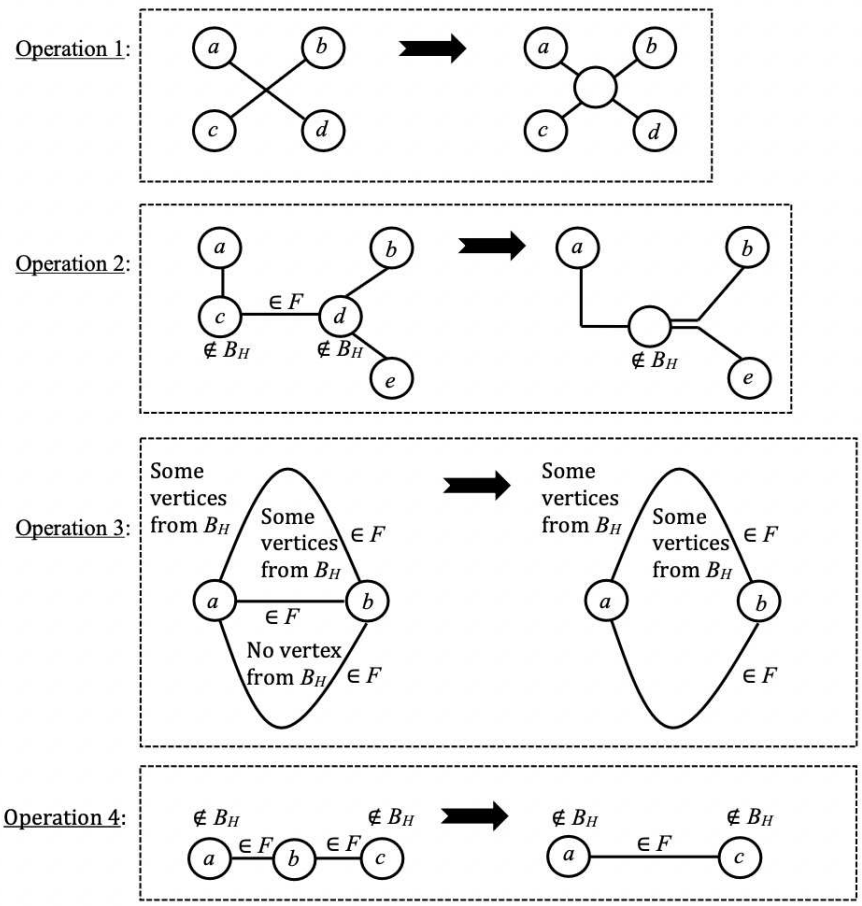

Figure 2: Four operation to reduce the size of a boundaried graph H.

With the above procedure of obtaining a representing collection in mind, the algorithm is quite simple. Suppose that we are given a boundaried graph (G, B_G) of bounded treewidth w. We execute bottom-up dynamic programming on T (the tree of the tree decomposition, obtained using a known algorithm). For every $x \in V(T)$, we store a representing collection for $\Gamma(x) \cup B_G$—this is computed by calling the above procedure. We remark that if the input graph is not boundaried, we can determine whether the entire graph G has a drawing with k crossings or not (i.e., obtain an alternative proof for Theorem 1.1) just by checking whether the table entry corresponding to the root of T stores a non-empty collection. The drawing itself can be obtained by backtracking the brute-force decisions and reduction operations.

3 Overview for the Proof of Theorem 1.2

We begin by finding a subset of edges $F \subseteq E(G)$ of size at most k whose removal from G makes it planar, using the algorithm for the Edge Planarization problem of Jansen et al. [33], where, if such a subset F is not found, we simply say No. Then, we compute a Tutte decomposition $\mathcal{T} = (T, \beta)$ of $G - F$, which, intuitively, is a tree decomposition that partitions the graph into its triconnected components—more precisely, for every $x \in V(T)$, we have the property that the *torso* of $G[\beta(x)]$ is triconnected. We refer to Fig. 3 for an illustration. We resolve the Crossing Number problem while performing a dynamic programming computation by traversing T in postorder.

Notice that among the vertices in $\Gamma(x)$, only vertices in $\beta(x) \cap \beta(p)$, where p is the parent of x, can be incident to edges in $E(G) \setminus F$ that go to vertices outside $\Gamma(x)$, and $|\beta(x) \cap \beta(p)| \leq 2$. However, we also need to remember that the Tutte decomposition is of $G - F$ and not G. Thus, accordingly, for every node $x \in V(T)$, our purpose is to compute a representation of $\Gamma(x)$ with the boundary $(\beta(x) \cap \beta(p)) \cup (V(F) \cap \gamma(x))$. Again, we refer to Fig. 3 for an illustration. Importantly, while the boundary $(\beta(x) \cap \beta(p)) \cup (V(F) \cap \gamma(x))$ is bounded by a linear function in

Copyright © 2025 by SIAM
Unauthorized reproduction of this article is prohibited

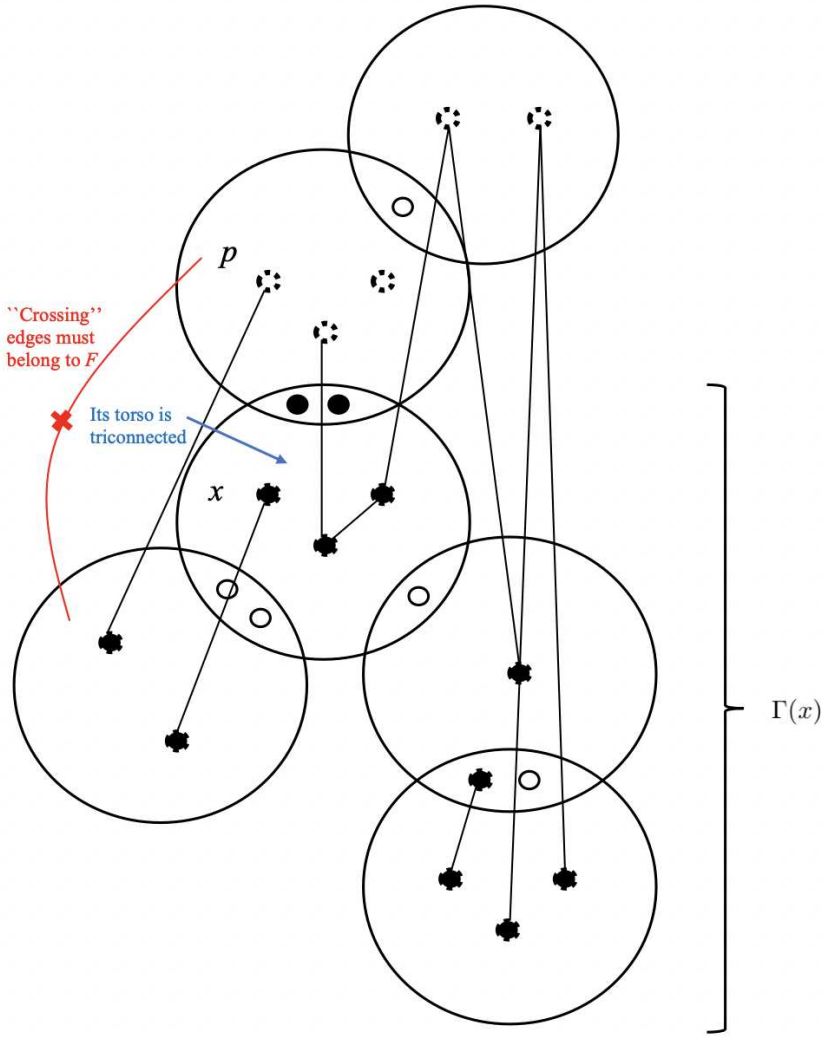

Figure 3: Abstraction of the notions of a Tutte decomposition of $G - F$ and the boundary of $\Gamma(x)$. For clarity, we show only edges in F, and vertices that belong to the intersection of two bags or are endpoints of edges in F (one vertex belongs to both categories). Vertices that are endpoints of edges in F are further highlighted by dashed lines. Vertices colored black belong to the boundary of $\Gamma(x)$.

k, $|\beta(x)|$ can be huge (indeed, $\beta(x)$ can even be the entire vertex set $V(G)$), and, more than that, the treewidth of $(G-F)[\beta(x)]$ can be huge as well, thus we cannot directly use the algorithm for the case of bounded treewidth in Theorem 1.1 on $G[\beta(x)]$.

Our main contribution in this context is to show how to reduce the treewidth of $\Gamma(x)$ (with the boundary $(\beta(x) \cap \beta(p)) \cup (V(F) \cap \gamma(x))$), which includes some edges in F, given that we have representations of $\Gamma(y)$ (with the boundary $(\beta(y) \cap \beta(x)) \cup (V(F) \cap \gamma(y))$) for all children y of x. Clearly, we cannot simply replace the graphs $\Gamma(y)$ by brute-forcing through their representations, since the number of such graphs $\Gamma(y)$ (being the number of children of x) can be huge. However, this can be handled by another layer of dynamic programming. For the sake of the proof of Theorem 1.2, this is done already in the proofs of other statements which are part of the proof of our generalization of Theorem 1.1. Thus, for the sake of the (over-)simplicity of this overview, we next suppose that the representation of each $\Gamma(y)$ contains just a single representative, and, so, it can be substituted.

For our treewidth reduction, we take an approach inspired by Baker's technique [2] as follows. See Fig. 4 for an illustration accompanying the labeling process described next. We consider some planar drawing φ of the torso

Copyright © 2025 by SIAM
Unauthorized reproduction of this article is prohibited

of $(G-F)[\beta(x)]$. Then, we perform a breadth-first search on the *radial completion* of this torso,[8] and starting the search from all vertices on the boundary of the outer-face. Thus, the 0-th layer L_0 can be defined as the set of vertices on the boundary of the outer-face, the 1-st layer L_1 can be defined as the set of vertices on the boundary of the outer-face once L_0 is removed, and so on. Now, we label the layers using $\alpha = \mathcal{O}(k)$ labels, making the first three get label 0, the next three get label 1, and so on, where when (and if) we reach the three that get label $\alpha-1$, then the next three (if exist) get label 0 again, and so on. It can be shown (using [6]) that the subgraph of the torso of $(G-F)[\beta(x)]$ induced on any t consecutive layers has treewidth $\mathcal{O}(t)$. We call the union of any $3(k+1)$ consecutive layers that start and end with three layers having the same label a, an *extended a-piece*. See Fig. 5 for an illustration of this notion. In particular, the treewidth of $G-F$ (and hence also of G, since $|V(F)| \leq 2k$) induced on such an extended a-piece is $\mathcal{O}(k)$.

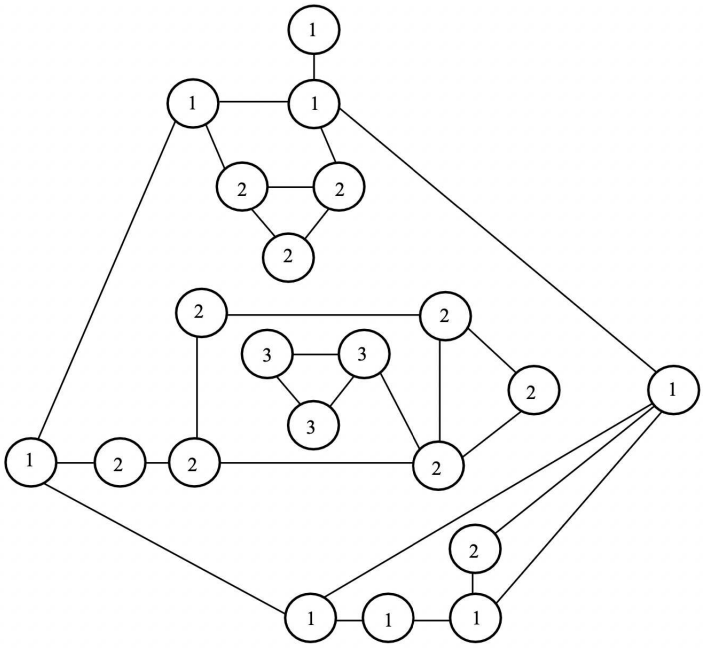

Figure 4: Breadth-first search on the radial completion of the depicted graph, starting from all vertices on the boundary of the outer-face. The layer to which each vertex belong is written inside of it.

The pigeon-hole principle implies that if we choose $\alpha \geq 4k+3$ and there exists a drawing ψ of G with at most k crossings, then there exists a label $a \in [\alpha-1]_0$ (called *excellent*) such that for all layers labeled a, neither these layers nor the graphs $\Gamma(y)$, for children y of x, that intersect these layers, contain any of the following vertices: *(i)* vertices in $V(F)$; *(ii)* endpoints of edges crossed by ψ; *(iii)* the (at most two) vertices in $\beta(x) \cap \beta(p)$. However, notice that different extended a-pieces (and the graphs $\Gamma[y]$ "associated" with them) can be, potentially, highly dependent on one another as the edges in F can go between them, having one endpoint in one extended a-piece, and another endpoint in another extended a-piece. Again, see Fig. 5 for an illustration. Our main insight—which is, in fact, the reason why we use a Tutte decomposition in the first place—is that if a is excellent, then such edges in F cannot exist.

The non-existence of such edges in F follows from a deeper result that we prove. Very informally, this result states the following. Consider some three layers labeled by an excellent label a, L_{i-1}, L_i and L_{i+1}, and a simple cycle C in the torso of $(G-F)[\beta(x)]$ further induced on L_i. Now, let Ext' (resp., Int') denote the set of all the vertices and edges of $(G-F)[\beta(x)]$ drawn by φ strictly outside (resp., inside) C. Also, informally speaking, let Ext (resp., Int) denote the set of all the vertices and "real edges" (those that belong to G) in Ext' (resp., Int') as well as all the vertices and edges in the graphs $\Gamma(y)$ associated with Ext' (resp., Int'). Lastly, let $C^\star$ be a cycle obtained from C by replacing each one of its "fake edges" (i.e., the edges in the torso and not in G) by a path in

[8]The radial completion is obtained by making, for each face, all vertices on the boundary of that face adjacent to one another.

Copyright © 2025 by SIAM
Unauthorized reproduction of this article is prohibited

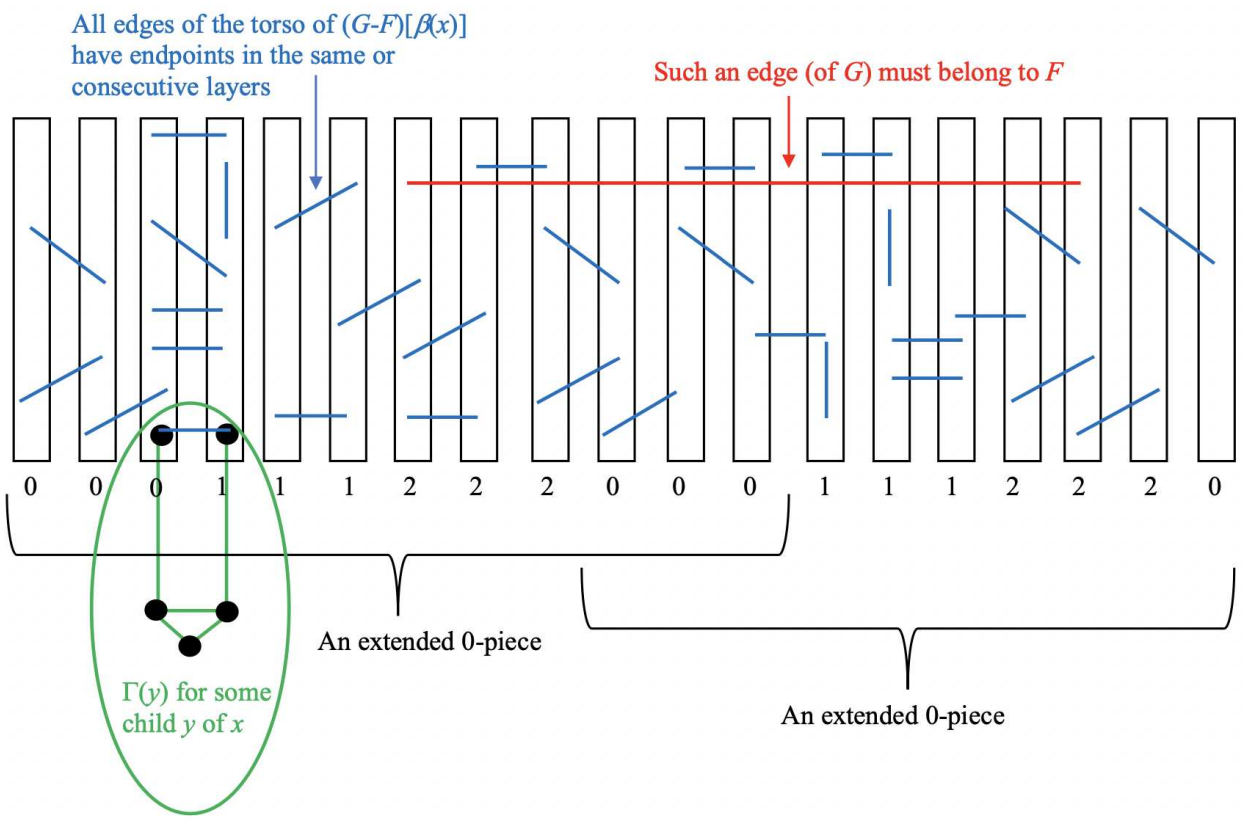

Figure 5: An abstraction of the notions of labeling, extended pieces, and their intersection with a graph $\Gamma(y)$ for some child y of x.

a $\Gamma(y)$ that yielded that edge. See Fig. 6 for an illustration. Then, we prove that, necessarily, ψ draws all vertices and edges in Ext outside $C^\star$, and all vertices and edges in Int inside $C^\star$. The reason why the non-existence of the aforementioned edges in F follows from this result is the following. Since ψ does not cross $C^\star$ (by the definition of an excellent label), the existence of an edge between the exterior and the interior of C (including the graphs $\Gamma(y)$ associated with them) with respect to φ implies that their endpoints should be drawn on the same side of $C^\star$, yielding a contradiction to the result.

With the above result at hand, we derive that, roughly speaking, the crossing number of G equals the sum of the crossings numbers of subgraphs associated with the different extended a-pieces,[9] when we demand that all edges incident to vertices in layers labeled a as well as in the subgraphs $\Gamma(y)$ associated with them are uncrossable. This is why, in the first place, we will consider a slight generalization of the Crossing Number problem, which we term Crossing Number with Uncrossable Edges.

Thus, we can iterate over every choice for $a \in [\alpha - 1]_0$, knowing that if G has a drawing ψ with at most k crossings, then there is a choice of a label that is excellent with respect to it. Given some choice of a, we proceed to solve each of its corresponding extended a-pieces independently: we compute a representation for each one of them independently by making use of our algorithm for the case of bounded treewidth in our generalization of Theorem 1.1. Then, we combine these representations to one that represents the entire graph $\Gamma(x)$ and can be used when we proceed with the dynamic programming on the Tutte decomposition $\mathcal{T}$ of G. We note that some technicalities here are "hidden under the rug". For example, we need to represent both $\Gamma(x)$ with no uncrossable edges and $\Gamma(x)$ with all of its edges being uncrossable, since the correctness of our arguments rely on making, for different choices of a, different sets of edges uncrossable.

[9] That is, the subgraphs of G induced by the vertices in these pieces and in the subgraphs $\Gamma(y)$ associated with them.

Copyright © 2025 by SIAM
Unauthorized reproduction of this article is prohibited

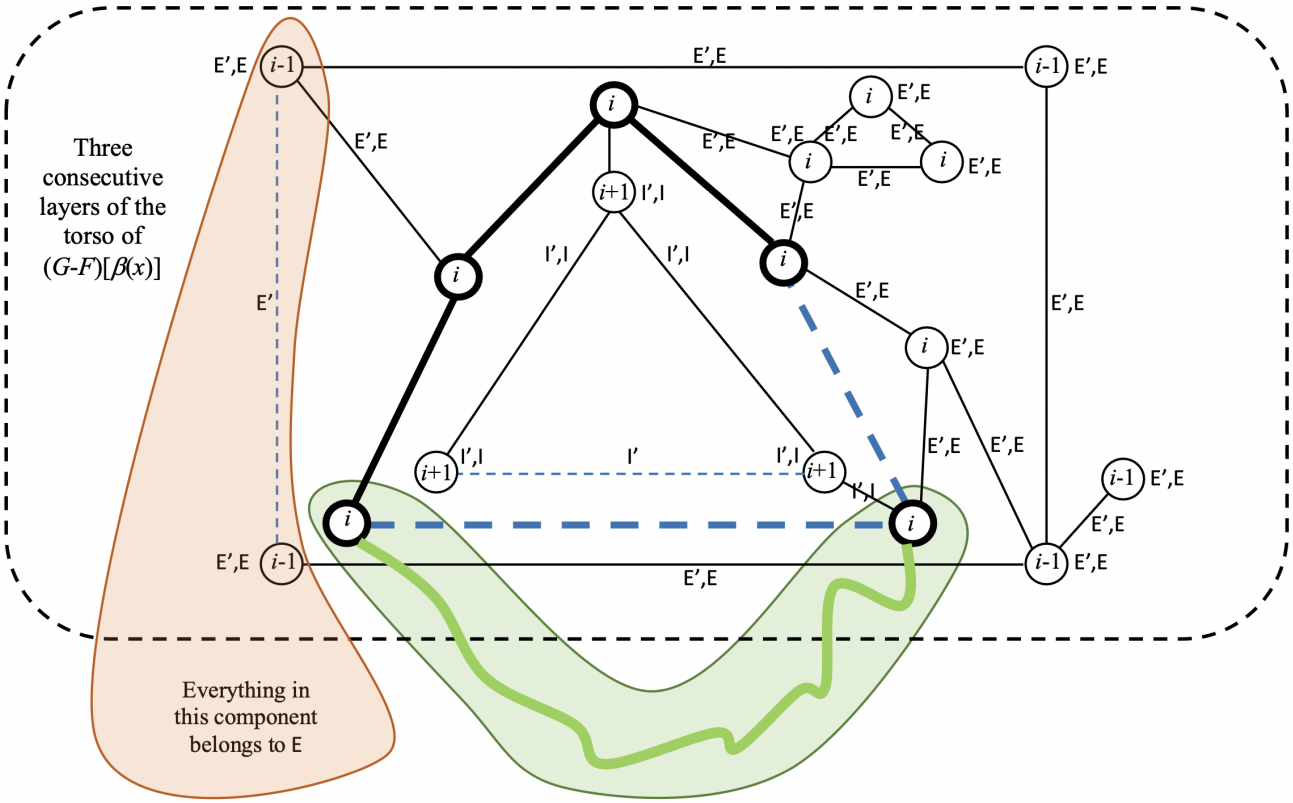

Figure 6: An abstraction of C, Ext', Ext, Int', Int and $C^\star$. The layer to which a vertex in $(G-F)[\beta(x)]$ belongs is written inside of it. Edges that belong to G are solid black, and edges that do not belong to G (but belong to $(G-F)[\beta(x)]$) are dashed blue. Whether a vertex or an edge belongs to Ext', Ext, Int' or Int is indicated by the initial letters (with or without ') next to the entity. The vertices and edges of the cycle C are shown in bold. Two examples of graphs $G[\gamma(y)]$ for children y of x are shown in green and orange (their elements are not shown). Then cycle $C^\star$ is derived from C by replacing each of its two dashed blue edges by a path in a corresponding components—this is illustrated for the bottom dashed blue edge, being replaced by the green path.

4 Conclusion

In this paper, we proved that the Crossing Number problem is solvable in time $2^{\mathcal{O}(k \log k)} \cdot n$. Up to one logarithmic factor, this resolved a 23-years old open question.

We remark that the main obstacle towards shaving off the $\log k$ factor in the running time based on our approach lies in the bound on the number of our different representatives. Specifically, we do not know how to define the set of representatives differently so that their number will be $2^{\mathcal{O}(k+t)}$ and yet they will store "enough information" as required for correctness.

Our algorithm also seems to be quite simple in comparison to previous works. Thus, besides being an advantage on its own, the ideas behind it might find other applications in the resolution of problems in Graph Drawing, particularly problems involving "almost planar" graphs as well as other notions of crossings.

References

[1] B. S. Baker, *Approximation algorithms for np-complete problems on planar graphs*, J. ACM, 41 (1994), pp. 153–180.

[2] B. S. Baker, *Approximation algorithms for np-complete problems on planar graphs*, Journal of the ACM (JACM), 41 (1994), pp. 153–180.

[3] M. J. Bannister, S. Cabello, and D. Eppstein, *Parameterized complexity of 1-planarity*, Journal of Graph Algorithms and Applications, 22 (2018), pp. 23–49.

Copyright © 2025 by SIAM
Unauthorized reproduction of this article is prohibited

[4] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph drawing: algorithms for the visualization of graphs*, Prentice Hall PTR, 1998.
[5] S. N. Bhatt and F. T. Leighton, *A framework for solving VLSI graph layout problems*, J. Comput. Syst. Sci., 28 (1984), pp. 300–343.
[6] H. L. Bodlaender, *A partial k-arboretum of graphs with bounded treewidth*, Theoretical computer science, 209 (1998), pp. 1–45.
[7] S. Cabello, *Hardness of approximation for crossing number*, Discrete and Computational Geometry, 49 (2013), pp. 348–358.
[8] S. Cabello and B. Mohar, *Crossing number and weighted crossing number of near-planar graphs*, Algorithmica, 60 (2011), pp. 484–504.
[9] S. Cabello and B. Mohar, *Adding one edge to planar graphs makes crossing number and 1-planarity hard*, SIAM Journal on Computing, 42 (2013), pp. 1803–1829.
[10] M. Chimani and P. Hliněný, *A tighter insertion-based approximation of the crossing number*, Journal of Combinatorial Optimtimization, 33 (2017), pp. 1183–1225.
[11] M. Chimani, P. Hliněný, and G. Salazar, *Toroidal grid minors and stretch in embedded graphs*, J. Comb. Theory, Ser. B, 140 (2020), pp. 323–371.
[12] J. Chuzhoy, S. Mahabadi, and Z. Tan, *Towards better approximation of graph crossing number*, in Proceedings of the 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS, 2020, pp. 73–84.
[13] J. Chuzhoy and Z. Tan, *A subpolynomial approximation algorithm for graph crossing number in low-degree graphs*, in Proceedings of the 54th Annual ACM Symposium on Theory of Computing, STOC, 2022, p. TBA.
[14] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*, Springer, 2015.
[15] E. de Klerk, J. Maharry, D. V. Pasechnik, R. B. Richter, and G. Salazar, *Improved bounds for the crossing numbers of* $\mathrm{K}_{\mathrm{m},\,\mathrm{n}}$ *and* K_{n}, SIAM J. Discret. Math., 20 (2006), pp. 189–202.
[16] É. C. de Verdière and T. Magnard, *An FPT algorithm for the embeddability of graphs into two-dimensional simplicial complexes*, in 29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference), 2021, pp. 32:1–32:17.
[17] É. C. de Verdière, T. Magnard, and B. Mohar, *Embedding graphs into two-dimensional simplicial complexes*, Computing in Geometry and Topology, 1 (2022), pp. 1–23.
[18] R. G. Downey and M. R. Fellows, *Parameterized Complexity*, Monographs in Computer Science, Springer, 1999.
[19] R. G. Downey and M. R. Fellows, *Fundamentals of parameterized complexity*, vol. 4, Springer, 2013.
[20] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Texts in Theoretical Computer Science. An EATCS Series, Springer, 2006.
[21] R. Ganian, F. Montecchiani, M. Nöllenburg, and M. Zehavi, *Parameterized complexity in graph drawing (dagstuhl seminar 21293)*, Dagstuhl Reports, 11 (2021), pp. 82–123.
[22] M. R. Garey and D. S. Johnson, *Crossing number is np-complete*, SIAM Journal on Algebraic Discrete Methods, 4 (1983), pp. 312–316.
[23] I. Gitler, P. Hliněný, J. Leaños, and G. Salazar, *The crossing number of a projective graph is quadratic in the face-width*, Electronic Notes in Discrete Mathematics, 29 (2007), pp. 219–223.
[24] M. Grohe, *Computing crossing numbers in quadratic time*, Journal of Computer and System Sciences, 68 (2004), pp. 285–302.
[25] T. Hamm and P. Hliněný, *Parameterised partially-predrawn crossing number*, in 38th International Symposium on Computational Geometry, SoCG 2022, June 7-10, 2022, Berlin, Germany, X. Goaoc and M. Kerber, eds., vol. 224 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, pp. 46:1–46:15.
[26] P. Hliněný, *Crossing number is hard for cubic graphs*, Journal of Combinatorial Theory, Series B, 96 (2006), pp. 455–471.
[27] P. Hliněný, *Complexity of anchored crossing number and crossing number of almost planar graphs*, CoRR, abs/2306.03490 (2023).
[28] P. Hliněný and M. Dernár, *Crossing number is hard for kernelization*, in Proceedings of the 32nd International Symposium on Computational Geometry, SoCG, 2016, pp. 42:1–42:10.
[29] P. Hliněný and L. Khazaliya, *Crossing number is NP-hard for constant path-width (and tree-width)*, arXiv preprint arXiv:2406.18933, (2024).
[30] P. Hliněný and G. Salazar, *On the crossing number of almost planar graphs*, in Proceedings of the 14th International Symposium on Graph Drawing and Network Visualization, GD, 2006, pp. 162–173.
[31] ———, *On hardness of the joint crossing number*, in Proceedings of the 26th International Symposium on Algorithms and Computation, ISAAC, 2015, pp. 603–613.
[32] P. Hliněný and A. Sankaran, *Exact crossing number parameterized by vertex cover*, in Proceedings of the 27th International Symposium on Graph Drawing and Network Visualization, GD, 2019, pp. 307–319.

Copyright © 2025 by SIAM
Unauthorized reproduction of this article is prohibited

[33] B. M. P. Jansen, D. Lokshtanov, and S. Saurabh, *A near-optimal planarization algorithm*, in Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014, 2014, pp. 1802–1811.
[34] K.-i. Kawarabayashi and B. Reed, *Computing crossing number in linear time*, in Proceedings of the 39th Annual ACM Symposium on Theory of Computing, STOC, 2007, pp. 382–390.
[35] F. T. Leighton, *Complexity issues in VLSI: optimal layouts for the shuffle-exchange graph and other networks*, MIT press, 1983.
[36] F. T. Leighton, *New lower bound techniques for VLSI*, Math. Syst. Theory, 17 (1984), pp. 47–70.
[37] J. Pach and G. Tóth, *Thirteen problems on crossing numbers*, Geombinatorics, 9 (2000), pp. 194–207.
[38] J. Pach and G. Tóth, *Which crossing number is it anyway?*, J. Comb. Theory, Ser. B, 80 (2000), pp. 225–246.
[39] M. Schaefer, *The graph crossing number and its variants: A survey*, The Electronic Journal of Combinatorics, (2012), pp. DS21–Sep.
[40] ———, *Crossing numbers of graphs*, CRC Press, 2018.
[41] P. Turán, *A note of welcome*, Journal of Graph Theory, 1 (1977), pp. 7–9.

Copyright © 2025 by SIAM
Unauthorized reproduction of this article is prohibited