2

Crossings and Planarization

Christoph Buchheim
TU Dortmund

Markus Chimani
Friedrich-Schiller-Universität Jena

Carsten Gutwenger
TU Dortmund

Michael Jünger
University of Cologne

Petra Mutzel
TU Dortmund

2.1 Introduction

In many respects, crossing minimization is an exceptional problem in the wide range of optimization problems arising in automatic graph drawing. First of all, it is one of the most basic and natural problems among these, and, at the same time, very easy to formulate: given a graph, draw it in the plane with a minimum number of crossings between its edges. In fact, this problem is much older than automatic graph drawing. Crossing number problems were first examined by Turán when he worked in a brick factory during the Second World War. This work motivated him to search for crossing minimal drawings of the complete bipartite graph $K_{n,m}$, without success. Later, Zarankiewicz gave a rule for creating a drawing of $K_{n,m}$ with $\lfloor \frac{m}{2} \rfloor \lfloor \frac{m-1}{2} \rfloor \lfloor \frac{n}{2} \rfloor \lfloor \frac{n-1}{2} \rfloor$ crossings, but his proof of optimality was shown to be incorrect. Still today, this is an open question. The same is true for the crossing number of K_n.

Besides its theoretical relevance as a topological problem, crossing minimization has many practical applications. In automatic graph drawing, it is well known that the readability of a two-dimensional graph layout strongly depends on the number of edge crossings. This was verified by empirical studies of Purchase [Pur97]. In fact, the main information given by an abstract graph is whether two vertices are connected by an edge. This information should be easily recognizable. In particular, it should be easily possible to trace the edges in the drawing. This task is complicated by the presence of crossing edges, as they distract the concentration of the human viewer. See Figure 2.1 for a comparison.

Another important application for the crossing minimization problem is VLSI (very large scale integration) design. In this context, the problem was first discussed in depth. The

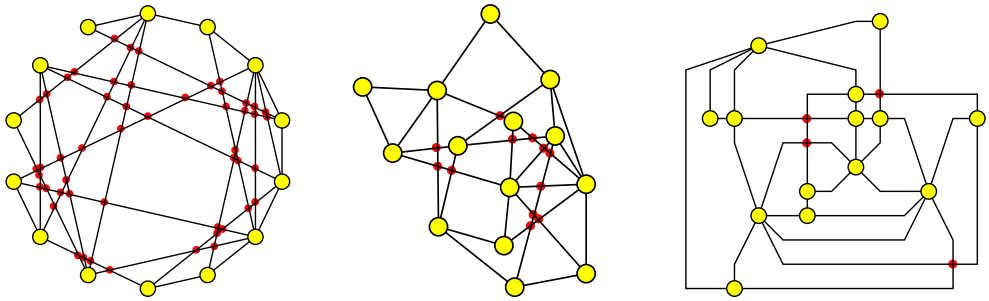


Figure 2.1 Different drawings of the same abstract graph with different numbers of edge crossings (51, 12, and 4, respectively). Most aesthetic criteria like few edge bends, uniform edge lengths, or a small drawing area favor the first two drawings. However, with respect to the number of edge crossings the last drawing is preferable.

aim of VLSI design is to arrange transistors on two-dimensional chips. Certain transistors need to be connected by wires, which have to be routed on the chip. Every crossing of two wires causes additional costs for realizing the chip, so that a small number of such crossings is desired to reduce these costs as far as possible.

An outstanding property of the crossing minimization problem is its hardness. It was shown by Garey and Johnson [GJ83] that this problem is NP-hard; see Section 2.3. However, several optimization problems arising in the area of automatic graph drawing are NP-hard and have nevertheless been solved in practice. In contrast, crossing minimization is extremely hard also practically. So far, exact approaches can only solve relatively sparse, medium sized instances within a reasonable running time; see Section 2.4. This is drastically shown by the fact that even the crossing numbers of the complete graphs K_n are unknown for $n \geq 13$.

Given the NP-hardness of the general problem, many restricted versions of crossing minimization have been considered, in the hope of finding polynomial time algorithms in these cases. However, in most cases, the problem remains NP-hard. Examples are bipartite drawings or linear embeddings; see Section 2.3. In practice, however, some of the resulting problems become easier as the degrees of freedom are reduced.

Besides considering special cases, it is natural to ask for approximation algorithms. However, up to now, only for graphs with bounded degree it was possible to find algorithms yielding provably near-optimal solutions; see Section 2.6. On the other hand, no negative results about approximability are known.

Currently used approaches to the general crossing minimization problem are of heuristic nature. The state-of-the-art approach for general crossing minimization is the planarization method, which is described in detail in Section 2.5. The main idea is to split up the problem into two steps: in the first step, a planar subgraph is computed. The aim in this step is to find a subgraph with as many edges as possible. In the second step, all edges not contained in this subgraph are reinserted into the drawing. Whenever an edge is inserted, the produced crossings are replaced by dummy vertices, so that the result is a planar graph again. Having added all edges in this way, a planar drawing algorithm can be used to compute a layout of the graph; see Chapters 6 and 7. After this, the dummy vertices are removed. For both steps of the planarization approach, a variety of possible algorithmic realizations has been discussed; see Section 2.5. This approach is also particularly interesting with respect to approximation algorithms, as it can be shown that certain insertion algorithms in fact approximate the crossing number in case of special graph classes; again see Section 2.6.

The second step of the planarization approach can also be realized in many different ways; see Section 2.5.3. Usually, it is again solved heuristically; edges are reinserted one after another, each with a minimal number of new edge crossings. It was shown recently that one can add a single edge optimally over all possible embeddings of the planar graph constructed so far.

After 60 years of research in different areas of mathematics and computer science, the crossing minimization problem is still far from being fully explored, both theoretically and practically. On the theoretical side, the most interesting open problems in our opinion are the crossing numbers of the complete graphs, including Turán's brick factory problem, as well as the approximability of crossing minimization. Practically, one can hope for new and better heuristic methods or faster exact approaches. At this point, we can only report on the status quo. We hope that parts of this chapter will become obsolete sooner or later.

2.2 Crossing Numbers

A drawing of a graph $G = (V, E)$ in the plane is a mapping of each vertex $v \in V$ to a distinct point and each edge $e = (v, w) \in E$ to a curve connecting the incident vertices v and w without passing through any other vertex. A common point of two edges in a drawing that is not an incident vertex is called a *crossing*. The *crossing number* $\mathrm{cr}(G)$ is defined to be the minimum number of crossings in any drawing of G.

In their paper "Which Crossing Number Is It, Anyway?", Pach and Tóth define two further possibilities on how to count the number of crossings in a graph (see [PT00]).

DEFINITION 2.1 Let $G = (V, E)$ be a simple graph.

1. The *pairwise crossing number* of G, denoted with $\mathrm{pcr}(G)$, is the minimum number of pairs of edges $(e_1, e_2) \in E \times E,\ e_1 \neq e_2$ such that e_1 and e_2 determine at least one crossing, over all drawings of G.
2. The *odd-crossing number* of G, denoted with $\mathrm{ocr}(G)$, is the minimum number of pairs of edges $(e_1, e_2) \in E \times E,\ e_1 \neq e_2$ such that e_1 and e_2 cross an odd number of times, over all drawings of G.

It is clear that $\mathrm{ocr}(G) \leq \mathrm{pcr}(G) \leq \mathrm{cr}(G)$, and we know that $\mathrm{cr}(G)$ cannot be arbitrarily large if $\mathrm{ocr}(G)$ is bounded. More precisely we have that $\mathrm{cr}(G) \leq 2(\mathrm{ocr}(G))^2$. Only after some years, an example was conceived by Pelsmajer et al. [PSŠ06], showing that there in fact exist graphs with $\mathrm{ocr}(G) \neq \mathrm{cr}(G)$. Yet, it is still unknown whether $\mathrm{pcr}(G) = \mathrm{cr}(G)$.

Another well-studied variant of the crossing minimization problem is the *rectilinear crossing number* $\mathrm{cr}_1(G)$, which is defined to be the minimum number of crossings in any drawing of a graph G where all edges are drawn as straight lines. Bienstock and Dean proved in [BD93] that for graphs with crossing number at most three, the rectilinear crossing number and the usual crossing number coincide. They could further show that there are graphs G_k such that $\mathrm{cr}_1(G_k)$ is arbitrarily large, even if $\mathrm{cr}(G_k)$ is only four.

As a generalization of the rectilinear crossing number, Bienstock introduced in [Bie91] the concept of the *t-polygonal crossing number*.

DEFINITION 2.2 Let $G = (V, E)$ be a graph. A t-polygonal drawing of G, for $t \geq 1$, is a good drawing where every edge is drawn as a t-polygonal line, i.e., a polygonal line with

at most t segments. The *t-polygonal crossing number* $\mathrm{cr}_t(G)$ is defined as the minimum number of crossings in any t-polygonal drawing of G.

A *good drawing* is a drawing that satisfies the following conditions:

1. no edge crosses itself
2. adjacent edges do not cross one another
3. non-adjacent edges cross each other at most once

Bienstock also showed that there cannot be a polynomial time algorithm for producing optimal t-polygonal drawings of G unless P = NP and that there is no fixed t such that $\mathrm{cr}(G) = \mathrm{cr}_t(G)$ for any graph G.

An even more restricted version of the crossing number problem is the *linear crossing number*: We call a drawing of a graph a *linear drawing* if all vertices lie on a straight line and edges are drawn as semicircles above and below this line. It is easy to see that the crossing number resulting from this drawing style is an upper bound for $\mathrm{cr}(G)$. Surprisingly, there is a further connection to the general crossing number problem, as was shown by Nicholson [Nic68]. He proved that any drawing in the plane with a minimum number of crossings can be converted into a linear drawing with an equivalent crossing structure such that all vertices are placed on a horizontal line and edges are drawn as a series of semicircles while successive semicircles lie on different sides of the horizontal line.

It is interesting to see that the complexity of the linear crossing number problem stays the same, even if we fix the ordering of the vertices of V (this is the so-called *fixed linear crossing minimization problem*). Masuda et al. proved in [MNKF90] that even this variant is NP-complete.

2.2.1 Known Bounds

No matter which definition or variant of the crossing number problem is used, its solution seems to be a difficult task. Even though crossing numbers have been investigated extensively in the past, useful theoretical results are rather limited. One of the first major results has been claimed in 1953 by Zarankiewicz (and, independently, by Urbaník) as a solution to Turán's brick factory problem, which in fact asks for the crossing number of the complete bipartite graph $K_{m,n}$.

$$\mathrm{cr}(K_{m,n}) = \lfloor \frac{m}{2} \rfloor \lfloor \frac{m-1}{2} \rfloor \lfloor \frac{n}{2} \rfloor \lfloor \frac{n-1}{2} \rfloor \qquad \text{(conjecture)} \tag{2.1}$$

Over ten years later, an error in the induction argument of Zarankiewicz's proof was unveiled, which is still unremedied. Hence, the correctness of equation (2.1) is still unknown. The conjecture is derived from the following drawing rule for complete bipartite graphs $K_{m,n} = (A \cup B, E)$: place the vertices in vertex set A at coordinates $(i(-1)^i, 0)$ for all $i = 1, \ldots, m$ and the vertices of vertex set B at coordinates $(0, j(-1)^j)$ for all $j = 1, \ldots, n$. All edges are drawn as straight lines. Figure 2.2 shows a sample drawing of $K_{6,6}$ with 36 crossings. Even though the correctness of equation (2.1) could never be verified, the provided drawing rule gives us an upper bound $Z(m,n)$ for $\mathrm{cr}(K_{m,n})$. Recently, de Klerk et al. [KMP+06, KPS07] devised a method for computing asymptotic lower bounds for $\mathrm{cr}(K_{m,n})$ based on semidefinite programming. They show that

$$\lim_{n\to\infty} \frac{\mathrm{cr}(K_{m,n})}{Z(m,n)} \geq 0.8594 \frac{m}{m-1} \; .$$

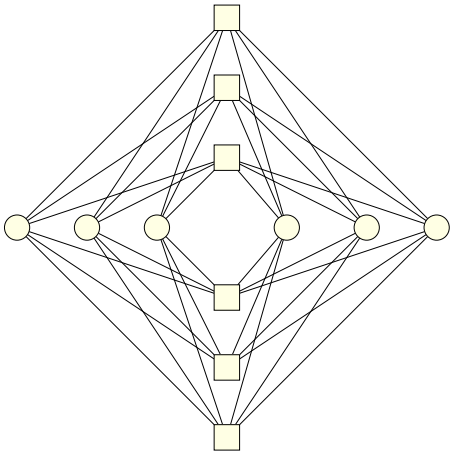


Figure 2.2 A drawing of $K_{6,6}$ with 36 crossings using Zarankiewicz's rule.

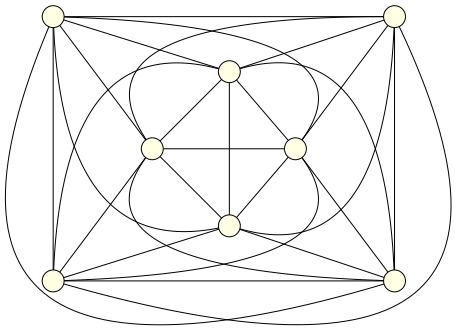


Figure 2.3 A drawing of K_8 with a minimum number of 18 crossings.

As for complete bipartite graphs, there is also a conjecture for the number of crossings of the complete graph K_n with n vertices.

$$\mathrm{cr}(K_n) = \frac{1}{4}\lfloor\frac{n}{2}\rfloor\lfloor\frac{n-1}{2}\rfloor\lfloor\frac{n-2}{2}\rfloor\lfloor\frac{n-3}{2}\rfloor \qquad \text{(conjecture)} \tag{2.2}$$

Constructions of corresponding drawings [GJJ68] show that also this conjecture yields an upper bound $Z(n)$ on $\mathrm{cr}(K_n)$. For complete graphs on up to ten vertices, its correctness has been verified by Guy [Guy72]. Pan and Richter [PR07] have extended this verification to K_{11} and K_{12}. We show a sample drawing of K_8 with a minimum number of 18 crossings in Figure 2.3. For K_n, the best-known asymptotic lower bound is again due to de Klerk et al. [KMP$^+$06]:

$$\lim_{n\to\infty} \frac{\mathrm{cr}(K_n)}{Z(n)} \geq 0.83.$$

The crossing number of a graph G with n vertices cannot exceed the crossing number of the complete graph K_n, hence $Z(n)$ also marks an upper bound for general graphs. Unfortunately, it is the only known upper bound. A simple *lower bound* can be obtained from Euler's formula. Since any planar simple connected graph $G = (V, E)$ cannot have more than $3|V| - 6$ edges, clearly $\mathrm{cr}(G) \geq |E| - 3|V| + 6$. If, in addition, G contains no triangle, then $\mathrm{cr}(G) \geq |E| - 2|V| + 4$.

In 1983, Leighton used induction on the number of vertices to show the following theorem; see [Lei83].

Theorem 2.1 *Let $G = (V, E)$ be a simple graph. If $|E| \geq 4|V|$, we have*

$$cr(G) \geq \frac{1}{100} \frac{|E|^3}{|V|^2} . \tag{2.3}$$

Ajtai et al. obtained the same result independently with a smaller constant of $\frac{1}{375}$ in [ACNS82]. One of the best-known results has been derived by Pach and Tóth [PT97]. For any simple graph $G = (V, E)$, $\mathrm{cr}(G)$ satisfies

$$\mathrm{cr}(G) \geq \frac{1}{33.75} \frac{|E|^3}{|V|^2} - 0.9|V| . \tag{2.4}$$

Apart from bounds with respect to the number of vertices and edges, several approaches to obtain tight lower bounds based on different graph properties can be found in the literature.

A simple example is the *skewness* $\mathrm{sk}(G)$ of a graph G. It is defined as the minimum number of edges that must be removed from G in order to obtain a planar subgraph. Clearly, the crossing number of a graph cannot be smaller than its skewness. Hence, we have that

$$\mathrm{cr}(G) \geq \mathrm{sk}(G) . \tag{2.5}$$

Cimikowski showed in [Cim92] that there is a family of graphs with skewness one, but an arbitrarily high crossing number. An example is shown in Figure 2.4. Computing the skewness is equivalent to the *maximum planar subgraph problem*, which was shown to be NP-hard by Liu and Geldmacher in [LG77] in general. For certain classes of graphs, i.e., complete and complete bipartite graphs, the skewness is known. We can derive it for K_n from Euler's formula and the observation that every maximal planar graph is also a maximum planar subgraph of K_n. Hence, the skewness for complete graphs K_n is given by

$$\mathrm{sk}(K_n) = \frac{n(n-1)}{2} - 3n + 6, \tag{2.6}$$

and we can use similar arguments to derive the skewness for complete bipartite graphs as

$$\mathrm{sk}(K_{m,n}) = mn - 2(m+n) + 4 . \tag{2.7}$$

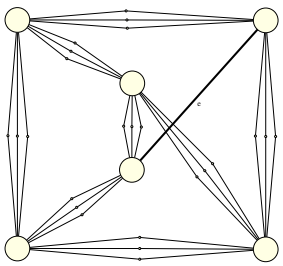


Figure 2.4 Construction of graphs with skewness one and arbitrarily high crossing number.

Another bound can be obtained from the *bisection width* $\mathrm{bw}(G)$. For any disjoint partition of the vertex set V into sets V_1 and V_2, we denote the edges (v_1, v_2) with $v_1 \in V_1$ and $v_2 \in V_2$ by $E(V_1, V_2)$. The bisection width $\mathrm{bw}(G)$ is defined as follows:

$$\mathrm{bw}(G) = \min_{|V_1|,|V_2| \geq \frac{|V|}{3}} \{|E(V_1, V_2)|\}.$$

More intuitively, the bisection width is the minimum number of edges that must be removed from G in order to partition the graph into two separate components with nearly equal size. The first known bound for the crossing number based on the bisection width goes back to Leighton. He proved the following theorem [Lei84].

Theorem 2.2 *For any graph $G = (V, E)$ of bounded degree, we have*

$$\mathrm{cr}(G) + |V| = \Omega(\mathrm{bw}(G)^2) \; .$$

Pach, Shahrokhi, and Szegedy [PSS96] use the bisection width to show the following, more general, result, which can be used to derive a lower bound for $\mathrm{cr}(G)$.

Theorem 2.3 *Let $G = (V, E)$ be a simple graph with $|V| \geq 2$ vertices, and let $k \geq 1$ be an integer. If G has a drawing with at most k crossings, then*

$$|E| \leq 3|V|(10 \log_2 |V|)^{2k-2} \; .$$

A very similar parameter is the *cutwidth* $\mathrm{cw}(G)$. Let $\phi : V \to \{1, 2, \ldots, |V|\}$ be an injection. We define $\mathrm{cw}(G)$ as follows:

$$\mathrm{cw}(G) = \min_{\phi} \max_{i} |\{(u, v) \in E : \phi(u) \leq i \leq \phi(v)\}|.$$

As a graphical interpretation, consider an injection of the vertices to the horizontal line and draw edges on one side of this line using semicircles. For each injection, we can "cut" the horizontal line between a pair of consecutive vertices such that the number of edges between each of the segments is maximized. The minimum value over all possible injections is the cutwidth. So far, the following relations are known (see [DV02], [PSS96], [SV94]; here $\delta(v)$ denotes the set of neighbors of vertex v):

$$\mathrm{cr}(G) + \frac{1}{16} \sum_{v \in V} |\delta(v)|^2 \geq \frac{1}{40} \mathrm{bw}^2(G), \tag{2.8}$$

$$\mathrm{cr}(G) + \frac{1}{16} \sum_{v \in V} |\delta(v)|^2 \geq \frac{1}{1176} \mathrm{cw}^2(G). \tag{2.9}$$

Unfortunately, the computation of both parameters $\mathrm{bw}(G)$ and $\mathrm{cw}(G)$ is NP-hard.

Both the bisection width and the cutwidth can be seen as a measure for the "non-planarity" of a graph. This applies also to the *thickness* $\Theta(G)$, which is defined as the minimum number of planar graphs whose union forms G. The only families of graphs whose thickness is known are complete graphs, complete bipartite graphs, and hypercubes. Mansfield proved in [Man83] that the determination of $\Theta(G)$ is NP-hard in general. There is a simple connection between thickness and crossing number:

$$\Theta(G) \leq \mathrm{cr}(G) + 1$$

So far, all those bounds are only of limited use. Either their quality is poor or their computation often exceeds practical limits. The investigation of tighter bounds could help to improve practical applications and lead to more insight into the crossing minimization problem.

2.3 Complexity of Crossing Minimization

Crossing minimization is not only one of the most important problems arising in automatic graph drawing, it is also one of the hardest. This is true both in practice and in theory: until recently, not a single exact algorithm being able to solve instances of nontrivial size had been devised. In fact, even for a graph as small and regular as K_{13}, the minimal number of crossings is still unknown. For a discussion of exact crossing minimization approaches, see Section 2.4.

2.3.1 NP-hardness

On the theoretical side, it is a well-known fact that the general crossing minimization problem is NP-hard. More precisely, consider the following *crossing number problem*:

> Given a graph G and a nonnegative integer K, decide whether there is a drawing of G with at most K edge crossings.

In 1983, Garey and Johnson proved that this problem is NP-complete [GJ83]. In the following, we reproduce their proof. It is based on a transformation of the NP-complete *optimal linear arrangement problem*:

> Given a graph $G = (V, E)$ and a nonnegative integer K, decide whether there is a one-to-one function $f: V \to \{1, \ldots, |V|\}$ with $\sum_{(v,w)\in E} |f(v) - f(w)| \leq K$.

The corresponding optimization problem is thus to order the vertices of G such that the total length of edges is minimal, where the length of an edge is defined as the distance of the two adjacent vertices in this ordering.

As an intermediate step in the proof, Garey and Johnson show the NP-completeness of the bipartite version of the crossing number problem for multigraphs, the *bipartite crossing number problem*:

> Given a bipartite multigraph $G = (V_1, V_2, E)$ and a nonnegative integer K, decide whether there is a drawing of G inside the unit square such that all vertices of V_1 lie on the northern boundary, all vertices of V_2 lie on the southern boundary, and the number of edge crossings is at most K.

In the following, we will call such drawings *bipartite drawings* for short. It is interesting that, contrary to widespread belief, the NP-completeness of the bipartite crossing number problem for *simple* graphs was long open—it was shown only very recently [Sch12].

Theorem 2.4 *The crossing number problem is NP-complete.*

It is easy to see that this problem is in NP: for every edge of G, one can guess all crossings involving this edge, and their order along the edge. To answer the question whether such a guessed crossing configuration is feasible, one can place dummy vertices on all chosen crossings and test the resulting graph for planarity. Clearly, the result is positive if and only if the given crossing configuration can be realized by some drawing of G.

The proof of completeness consists of several reduction steps and is split up into three separate lemmas in the following.

LEMMA 2.1 The optimal linear arrangement problem can be reduced to the bipartite crossing number problem in polynomial time.

Proof: The rough idea of the reduction is as follows: every vertex is doubled and the linear ordering is modeled on two parallel layers (the northern and southern boundary of the unit square) at the same time, with edges leading from one layer to the other. By a large number of artificial edges connecting corresponding pairs of vertices, the ordering is forced to be the same on both layers. The distance between two adjacent vertices in the linear ordering problem is then essentially proportional to the number of artificial edges crossed.

More formally, the transformation is defined as follows: Let an instance for the optimal linear arrangement problem be given, consisting of a graph G and an integer K, and assume $V = \{v_1, \ldots, v_n\}$. We then construct an instance $G' = (V_1, V_2, E_1 \cup E_2)$ and K' of the bipartite crossing number problem as follows:

$$\begin{aligned} V_1 &= \{u_i \mid i = 1, \ldots, n\} \\ V_2 &= \{w_i \mid i = 1, \ldots, n\} \\ E_1 &= \{|E|^2 \text{ copies of } (u_i, w_i) \mid i = 1, \ldots, n\} \\ E_2 &= \{(u_i, w_j) \mid (v_i, v_j) \in E \text{ with } i < j\} \\ K' &= |E|^2(K - |E|) + |E|^2 - 1 \end{aligned}$$

This construction is obviously polynomial. We have to show that the graph G admits a linear ordering with total edge length at most K if and only if the bipartite multigraph G' admits a bipartite drawing with at most K' crossings.

If a linear ordering f of G with total edge length at most K exists, we construct a bipartite drawing as follows. We place vertex u_i on position $(f(v_i)/(n+1), 1)$ and vertex w_i on position $(f(v_i)/(n+1), 0)$. Furthermore, we draw all edges as straight lines; bundles of parallel edges are drawn as nearly straight lines without mutual crossings; see Figure 2.5 for an example.

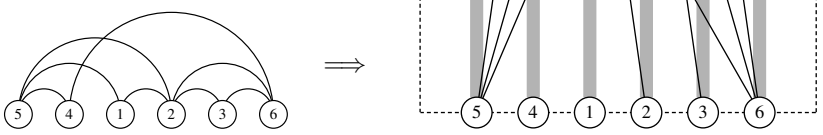


Figure 2.5 Reducing the optimal linear arrangement problem to the bipartite crossing number problem. Bold grey lines represent bundles of $|E|^2$ edges each.

In the constructed drawing, no artificial edge from E_1 will cross any other edge. Moreover, an edge $(u_i, w_j) \in E_2$ crosses exactly $|f(v_i) - f(v_j)| - 1$ bundles of $|E|^2$ edges each.

Consequently, the total number of such crossings is

$$\sum_{(u_i,w_j)\in E_2} |E|^2\Big(|f(v_i)-f(v_j)|-1\Big) = |E|^2 \sum_{(v,w)\in E}\Big(|f(v)-f(w)|-1\Big) \le |E|^2(K-|E|)\ .$$

The remaining crossings in the constructed drawing can only occur between pairs of edges in E_2, so that their total number is at most $|E|^2-1$. Summing up, the total number of edge crossings in our drawing is at most $|E|^2(K-|E|)+|E|^2-1=K'$.

For showing the other direction, assume that a bipartite drawing of G' with at most K' crossings is given. Then define $f(v_i)$ as the position of vertex u_i in the order of vertices on the northern boundary of the unit square. We claim that the linear ordering f leads to a total edge length of at most K. To see this, first observe that the order of vertices on both boundaries must be the same, as otherwise two bundles of $|E|^2$ edges each would cross each other, leading to $|E|^4 > K'$ crossings. Because of that, for each edge (v_i,v_j) with $i<j$, the distance $|f(v_i)-f(v_j)|$ is at most one more than the number of crossings of (u_i,w_j) with any edge bundle, so that

$$\sum_{(v,w)\in E} |f(v)-f(w)| \le |E| + K'/|E|^2 = |E| + (K-|E|) + 1 - 1/|E|^2 < K+1\ .$$

As the left-hand side of this inequality is integer, it is at most K. □

LEMMA 2.2 The bipartite crossing number problem can be reduced to the general crossing number problem for multigraphs in polynomial time.

Proof: Let $G=(V_1,V_2,E)$ and K be an instance of the bipartite crossing number problem. We construct a multigraph G' as follows: we add two vertices u and w to G. Moreover, we connect u with all vertices of V_1 by $K+1$ edges each. Analogously, we connect w with all vertices of V_2 by $K+1$ edges each. Finally, we add $K+1$ edges connecting u and w. Now we claim that G has a bipartite drawing with at most K crossings if and only if G' has a general drawing with at most K crossings.

The basic idea of this construction is that w.l.o.g. no bundle of $K+1$ edges will be crossed by any other edge, and that by this the crossing minimal bipartite drawings of G correspond to the crossing minimal general drawings of G'. In particular, it is clear that a bipartite drawing of G with at most K crossings yields a drawing of G' with at most K crossings by placing the vertices u and w outside the unit square; see Figure 2.6.

For showing the other direction, a drawing of G' with at most K crossings has to be converted into a bipartite drawing of G with at most K crossings. For this, a sequence of so-called normalization steps is applied in order to transform the original drawing of G' into one of the type of Figure 2.6 without increasing the number of edge crossings; deleting the vertices u and w then yields the desired drawing of G. This part of the proof was the most technical one in the original presentation; we give a simplified version here.

In the first normalization step, multiple crossings between one pair of edges and crossings between edges incident to a common vertex are removed in the obvious way. In particular, every bundle of $K+1$ edges connecting the same pair of vertices now defines a sequence of K regions in the drawing.

In a second step, one can obtain a drawing such that none of these bundle regions contains any vertex of G' or is crossed by any edge of G'. Indeed, for a fixed $v\in V_1$, consider the edge e connecting u and v that in the current drawing has the minimum number of crossings

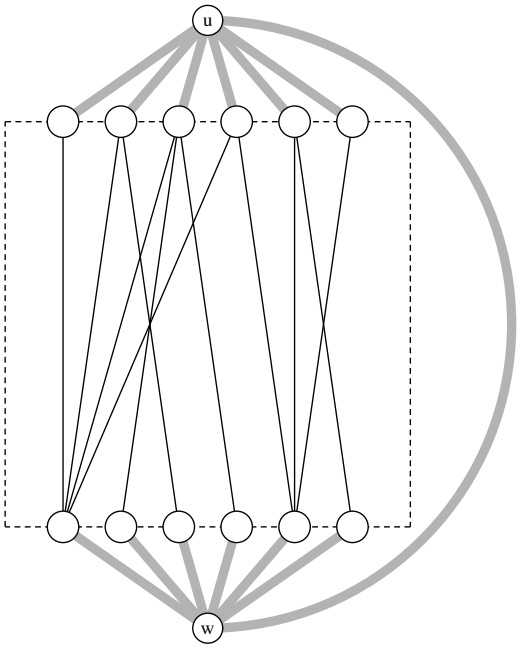


Figure 2.6 Reducing the bipartite crossing number problem to the general crossing number problem for multigraphs. Bold grey lines represent bundles of $K + 1$ edges each.

with other edges. Then one can reroute all edges (u, v), i.e., all edges parallel to e, along the same route as e. This yields a new drawing of G' with at most as many edge crossings as before. Repeating this for every $v \in V_1$ and analogously for w and every $v \in V_2$, we get a drawing without vertices in the bundle regions. Now it follows that no edge can cross any of these regions. The reason is that such an edge would have to cross all $K + 1$ edges of a bundle, as no vertices are contained in the bundle regions and multiple crossings were eliminated in the first normalization step.

Clearly, the drawing resulting from these two normalization steps is topologically equivalent to one of the type displayed in Figure 2.6. □

LEMMA 2.3 The crossing number problem for multigraphs can be reduced to the crossing number problem for simple graphs.

Proof: For the given multigraph, place an artificial vertex in the middle of every edge. The result is a simple graph with the same crossing number as the original multigraph. □

In the above scheme, we observe that the graph for which deciding on the crossing number is NP-hard requires two distinct vertices u and v of very high degree. One may think that they are central to the construction. Yet, using a different reduction strategy from optimal linear arrangement, Hliněný showed in [Hli06]:

Theorem 2.5 *The crossing number problem remains NP-complete even when restricted to cubic graphs, i.e., graphs where every vertex has degree 3.*

Theorem 2.4 shows that the crossing number problem is NP-complete. In particular, the crossing minimization problem is NP-hard, i.e., the problem of constructing a drawing of

a given graph with a minimal number of edge crossings. Nevertheless, one might hope for polynomial time algorithms at least for special classes of graphs, or for situations where the class of allowed drawings is restricted.

However, no interesting special class of graphs is known for which crossing minimization can be done in polynomial time. Exceptions are the classes of graphs for which a constant bound c on the number of crossings is given a priori, see Section 2.3.2, but this is a purely theoretical result in that this bound is not at hand in general and the running time increases heavily with the constant c.

The results also remain mostly negative if we restrict the set of feasible drawings by additional conditions. For instance, the problem is still NP-hard (even for simple graphs) if we require that

- the drawing is bipartite and the vertex order on one of the layers is fixed [EW94].
- all vertices have the same vertical coordinate and edges are drawn as semicircles. This is the so-called *linear crossing minimization problem* [MKNF86]. This problem remains NP-hard even if the horizontal order of vertices is fixed [MNKF90].
- the vertices lie on the unit circle and edges are drawn as straight lines. This is the *circular crossing minimization problem* [MKNF87].

However, we would like to point out that practically the problem might become considerably easier with the degrees of freedom for the drawing decreasing. To give an example, the bipartite crossing minimization problem with one layer fixed is NP-hard but can be solved quickly in practice [JM97]. By now, also reasonably sized general multi-layer crossing minimization instances can be tackled effectively with integer linear and semidefinite programs; see [CHJM11] for an overview.

2.3.2 Fixed Parameter Tractability

In the last section, we reproduced a proof by Garey and Johnson showing that it is NP-hard to decide whether a given graph G can be drawn with at most K edge crossings. We can also consider the situation where K is not given as part of the input but as a fixed parameter. It is then easy to see that one can decide in polynomial time whether a drawing of G with at most K crossings exists: broadly speaking, one could check all possible configurations of the up to K crossings, replace the chosen crossings by dummy vertices, and check the resulting graph for planarity. We can answer the original question affirmatively if and only if we find any planar graph in this way.

Even if the above algorithm runs in polynomial time for fixed K, the obvious drawback is the strong increase in running time for increasing K: if implemented in the straightforward way, the runtime is $O(|V| \cdot |E|^{2K})$. For a long time, it was an open question whether the problem is *fixed-parameter tractable*, i.e., whether the problem can be solved in $O(f(K) \cdot |V|^c)$ running time for some function $f(K)$ that is independent from the instance and some constant c that is independent from K. This question was answered by Grohe in 2001 with $c = 2$ [Gro01]. However, the running time of Grohe's algorithm is $O(2^{2^{p(K)}}|V|^2)$, where p is a polynomial, and hence grows strongly with K. Thus, the relevance of this algorithm is rather theoretical than practical. Kawarabayashi and Reed [KR07] improved on this result by giving a linear algorithm, i.e., $c = 1$; yet $f(K)$ remains too large for any practical application.

2.4 Exact Crossing Minimization

Exact methods to solve the crossing minimization problem constitute the youngest research field we are discussing in this chapter. The development showcases various algorithm engineering aspects of algorithm development, as its iterative improvements were always based on the analysis of the bottlenecks of the earlier approaches. The first approach [BEJ+05] already lay the setting used in the subsequent developments: it relies on mathematical programming in combination with branch-and-cut. Yet, its applicability was limited to very small graphs. By introducing column generation schemes into the branch-and-cut framework, its central ILP model, which we will describe in detail below, was later brought into the realm of applicability [CGM09, BCE+08] to some real-world graphs. The currently best exact approach replaces a key concept (the so-called *simple* crossing number) of the first formulation by integrating multiple linear-ordering problems instead [CMB08]. This leads to a mathematically more complex model but offers the advantage of fewer variables on the one hand, and the possibility for even stronger column generation strategies, on the other hand. Together with other developments like strong upper bounds (cf. Section 2.5), preprocessing strategies like the *non-planar core reduction* [CG09], and efficient extraction of multiple Kuratowski subdivisions (see below) at once [CMS08], we are now in the position to compute the exact crossing number of sparse graphs with up to 100 vertices. Figure 2.4 gives an overview of the algorithmic progress over the last years, comparing the various algorithms on the way to the currently most successful one. For a more detailed description of all exact algorithms discussed in the following, see [Chi08].

A *linear program* (LP) is an optimization problem consisting of continuous variables, a linear objective function, and linear constraints. The "father" of linear programming, George B. Dantzig, proposed the following standard model:

$$\begin{gathered} \text{maximize } c^\top x \\ \text{subject to } Ax \leq b \\ x \geq 0 \end{gathered}$$

where $c \in \mathbf{R}^n$, $A \in \mathbf{R}^{m \times n}$, and $b \in \mathbf{R}^m$. The linear function $c^\top x : \mathbf{R}^n \to \mathbf{R}$ is called the *objective function* and the inequalities in the system $Ax \leq b$ are called *constraints*. A vector $\hat{x}$ that satisfies the system of inequalities $Ax \leq b$ is called a *feasible* solution of the LP. Moreover, $\hat{x}$ is called an *optimal solution* if $c^\top \hat{x} \geq c^\top x'$ for all feasible solutions x'.

Linear programs proved to provide a powerful tool for various optimization problems in the past and extensive research led to efficient algorithms able to solve them in polynomial time, *e.g.*, the simplex [Chv83], the ellipsoid [GLS88], and the interior point method [RT97]. However, additional constraints that require some or all of the variables to be integer, render the problem NP-complete in general [GJ79].

Anyway, *(mixed) integer linear programs* are widely used to solve NP-hard combinatorial optimization problems in conjunction with polyhedral combinatorics, which aims at describing combinatorial optimization problems as linear programs and solving these with special-purpose methods. A key feature therefore is the possibility to alternatively describe the convex hull of the feasible points and extreme rays of a problem by a system of linear inequalities. For an introduction into this field, the interested reader is referred to [Pul89].

Before introducing the ideas of the ILP formulation presented in this section, we have to mention *Kuratowski's theorem*, which is one of the most important results in the field of planarity testing providing a full characterization of planar graphs based on the complete graph K_5 and the complete bipartite graph $K_{3,3}$.

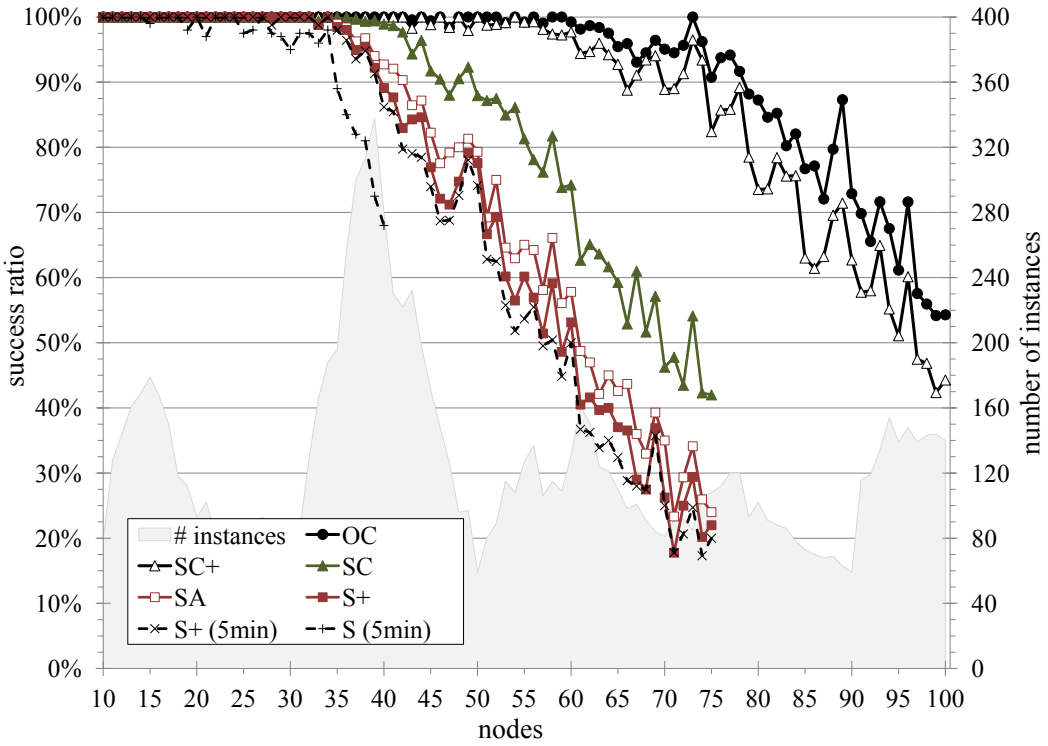


Figure 2.7 Success ratio (i.e., percentage of instances solved to provable optimality) in dependency to the graphs' size (number of vertices). Benchmark set: *Rome library* [DGL+97], see also Section 2.5. Different lines give different development steps of the algorithms; each instance was given 30 minutes of computation time unless specified otherwise. **S** is the first implementation of [BEJ+05], **S+** a more efficient reimplementation of the same algorithm. **SA** and **SC** denote the subdivision-based algorithms as considered in [CGM09], with algebraic pricing and the combinatorial column generation scheme, respectively. Finally, **SC+** and **OC** denote the latest implementations of the subdivision-based and the ordering-based ILPs, respectively, with combinatorial column generation and all further described improvements, as presented in [CMB08].

Theorem 2.6 *A finite graph is planar if and only if it contains no subgraph that is a subdivision of K_5 or $K_{3,3}$.*

We can obtain a subdivision S of a graph G by repeatedly replacing edges e by a path of length two. As a consequence of Theorem 2.6, at least two edges belonging to each Kuratowski subdivision have to cross. Based on this observation, we can try to address the crossing minimization problem using mathematical programming in the following way: we introduce a zero-one decision variable $x_{e,f}$ for each pair of edges $(e, f) \in E \times E$ that encode the crossings in an associated drawing: edges e and f cross each other if and only if $x_{e,f} = 1$. For each subdivision of K_5 or $K_{3,3}$, we can add constraints that force at least one of the involved variables to one.

Mutzel and Jünger [MJ01] pointed out the problems with this formulation. To our knowledge, there is no known polynomial time separation algorithm to identify the constraints of this type that are violated by a given fractional solution. Moreover, those constraints are not strong enough since it is not guaranteed that there is a realizing drawing if at least one of the involved crossing variables is one in every Kuratowski subdivision. Another severe problem of this formulation is the NP-hardness of the *realizability problem* [Kra91]:

Given a vector $x \in \{0,1\}^{\binom{E}{2}}$, does there exist a drawing consistent with x?

In order to efficiently answer this question, we also need to know the *order* of the edge crossings for a particular edge e. This additional information can be exploited by the introduction of a dummy vertex for each crossing and the application of a linear-time planarity testing algorithm to test the existence of a realizing drawing in polynomial time. Despite all these drawbacks, it is interesting that under certain conditions the above-described constraints, as well as similar ones, in fact constitute *facets* of the polytope defined by the convex hull of the feasible solutions [Chi11].

2.4.1 Subdivision-Based Formulation

One way to work around the realizability problem is the reduction to *simple drawings*. We call a drawing simple if each edge crosses at most one other edge. As for planar graphs, we can find a bound for the maximum number of edges of graphs that admit a simple drawing. More precisely, Pach and Tóth show the following theorem [PT97]:

Theorem 2.7 *Let $G = (V, E)$ be a simple graph drawn in the plane so that every edge is crossed by at most k others. If $0 \leq k \leq 4$, then we have*

$$|E| \leq (k+3)(|V|-2)\,. \tag{2.10}$$

They could further prove that this bound cannot be improved for $0 \leq k \leq 2$ and that for *any* $k \geq 1$ the following inequality holds:

$$|E| \leq \sqrt{16.875k}|V| \approx 4.108\sqrt{k}|V|\,. \tag{2.11}$$

Furthermore, Bodlaender and Grigoriev prove in [BG04] that it is NP-complete to determine whether there is a simple drawing for a given graph G. If there is such a drawing, we denote the minimum number of crossings among all simple drawings of G by $\mathrm{crs}(G)$.

It is easy to see that $\mathrm{cr}(G) \leq \mathrm{crs}(G)$. We cannot state equality because there are graphs G such that $\mathrm{crs}(G) > \mathrm{cr}(G)$. Consider the sample graph in Figure 2.8. The left drawing shows an optimal drawing with two crossings while the right drawing shows an optimal drawing among all simple drawings with three crossings.

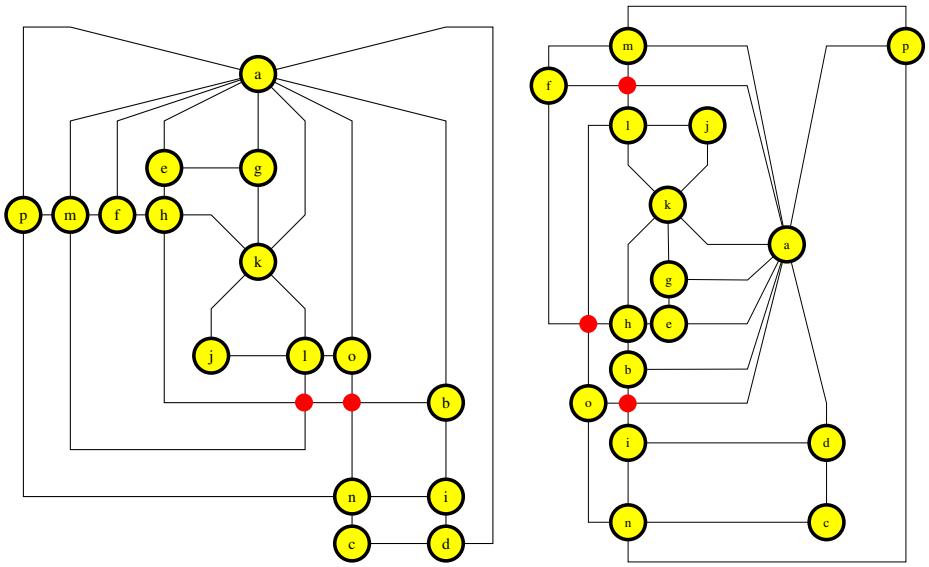


Figure 2.8 An optimal drawing of a graph with two crossings (left) and an optimal simple drawing of the same graph with three crossings (right). Both drawings were produced with the exact algorithm presented in this section.

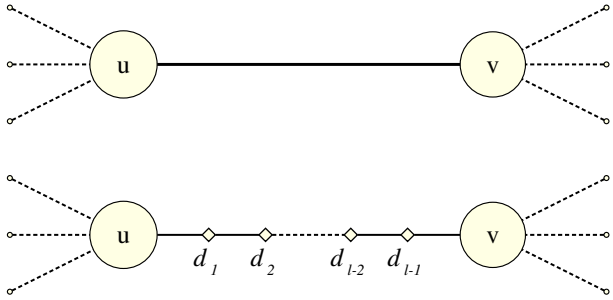


Figure 2.9 Edges are replaced with a path of length ℓ by inserting $\ell - 1$ dummy vertices.

Given an integer ℓ and a graph $G = (V, E)$ such that $\ell \geq |E|$, we can create a graph $G^* = (V^*, E^*)$ by replacing every edge $e \in E$ with a path of length ℓ. Figure 2.9 shows an example that illustrates this transformation. The graph G^* contains a total number of $|V| + (\ell - 1)|E|$ vertices and $\ell|E|$ edges.

It is easy to show that G can be drawn with n crossings if and only if there is a *simple drawing* of G^* with n crossings. Therefore, it is "sufficient" to solve the crossing minimization problem restricted to simple drawings in order to solve the "general" crossing minimization problem. Since the transformation obviously can be done in polynomial time, the NP-completeness of the corresponding decision problem for simple drawings follows immediately from the proof by Garey and Johnson; see Section 2.3. Since an edge $e = (u, v)$ never crosses itself or an adjacent edge in an optimal drawing it is sufficient to replace e with a path of length $|E| - |\delta(u)| - |\delta(v)| - 1$.

Let $G = (V, E)$ be a graph and let $D \subseteq E \times E$ be a set of unordered pairs of edges. We call D *realizable* if there is a drawing of G such that there is a crossing between edges e and f if and only if $(e, f) \in D$. Furthermore, D is called *simple* if for every $e \in E$ there is at most one $f \in E$ such that $(e, f) \in D$.

For every graph G and every simple D, G_D denotes the graph that is obtained by introducing a dummy vertex $d_{e,f}$ for each pair of edges $(e, f) \in D$: in other words, $d_{e,f}$ is the unique vertex when identifying the two vertices arising from subdividing both e and f. Note that G_D is only well defined if D is simple. For both edges e_1 and e_2 resulting from splitting e, we set $\hat{e}_1 = \hat{e}_2 = e$, analogously for f. Given a subgraph $H = (V', E') \subseteq G_D$, we denote with $\hat{H} \subseteq E$ the subset of original edges of G involved in the subgraph H of G_D, i.e., $\hat{H} = \{\hat{e} \mid e \in E'\} \subseteq E$. In the following, H will usually be a Kuratowski subdivision. We call the path corresponding to a single edge of the underlying K_5 or $K_{3,3}$ a *Kuratowski path*. By $\hat{H}^{[2]} \subset \hat{H}^2$ we will then denote the edge pairs (e, f) where e and f belong to different, nonadjacent Kuratowski paths. Hence, these are the only edge pairs that may actually form a crossing meaningful to the Kuratowski subdivision H.

COROLLARY 2.1 Let D be simple. Then D is realizable if and only if G_D is planar.

Using a linear-time planarity testing algorithm, we can test in time $O(|V| + |D|)$ whether D is realizable, and compute a realizing drawing if so.

DEFINITION 2.3 For a set of pairs of edges $D \subseteq E \times E$, we define

$$x^D_{e,f} = \begin{cases} 1 & \text{if } (e, f) \in D \\ 0 & \text{otherwise.} \end{cases}$$

PROPOSITION 2.1 Let D be simple and realizable. For an arbitrary set of pairs of edges $D' \subseteq E \times E$ of $G = (V, E)$ and any subdivision H of K_5 or $K_{3,3}$ in $G_{D'}$ the following inequality holds:

$$C_{D',H} : \sum_{(e,f) \in \hat{H}^{[2]} \setminus D'} x^D_{ef} \geq 1 - \sum_{(e,f) \in \hat{H}^{[2]} \cap D'} (1 - x^D_{ef}) \,. \tag{2.12}$$

Proof: Suppose inequality (2.12) is violated. Since every $x^D_{e,f} \in \{0, 1\}$, the left-hand side of the inequality must be zero and the right-hand side must be one, which means that

$$\begin{aligned} x^D_{e,f} &= 0 \quad \text{for all } (e, f) \in \hat{H}^{[2]} \setminus D', \text{ and} \\ x^D_{e,f} &= 1 \quad \text{for all } (e, f) \in \hat{H}^{[2]} \cap D' \,. \end{aligned}$$

It follows from the definition of x^D that $\hat{H}^{[2]} \cap D' = \hat{H}^{[2]} \cap D$, in other words, G_D agrees to $G_{D'}$ on the subgraph induced by $\hat{H}$, so that H is also a forbidden subgraph in G_D, i.e., a subdivision of K_5 or $K_{3,3}$. It follows from Kuratowski's Theorem that G_D is not planar. This contradicts the realizability of D by Corollary 2.1. □

Theorem 2.8 *Let G=(V,E) be a simple graph. A set of pairs of edges $D \subseteq E \times E$ is simple and realizable if and only if the following set of conditions holds:*

$$\begin{array}{ll} x^D_{e,f} \in \{0, 1\} & \forall\, e, f \in E,\ e \neq f \\ \sum_{f \in E} x^D_{e,f} \leq 1 & \forall\, e \in E \\ C_{D',H} & \textit{for every simple } D' \subseteq E \times E \\ & \textit{and every forbidden subgraph } H \textit{ in } G_{D'} \end{array}$$

Proof: It is easy to see that the constraints from the second row are satisfied if and only if D is simple. It remains to show that a simple D is realizable if and only if the conditions $C_{D',H}$ from the last row hold. For a realizable D, every $C_{D',H}$ is satisfied according to Proposition 2.1.

We have to show that any set of pairs of edges D that is not realizable violates at least one of the constraints $C_{D',H}$. It follows from Corollary 2.1 that G_D is not planar if D is not realizable and we know from Theorem 2.6 that there exists a subdivision H of K_5 or $K_{3,3}$ in G_D. Let $D' = D$ and consider the constraint $C_{D,H}$:

$$C_{D,H}: \sum_{(e,f)\in\hat{H}^{[2]}\setminus D} x^D_{ef} \;\geq\; 1 - \sum_{(e,f)\in\hat{H}^{[2]}\cap D} (1 - x^D_{ef}) \tag{2.13}$$

It follows from the definition of x^D that every $x^D_{e,f} \in \hat{H}^{[2]} \setminus D$ is zero, hence the left-hand side of inequality (2.13) is also zero. Since $\hat{H}^{[2]} \cap D \subseteq D$ we also know that $\sum_{(e,f)\in\hat{H}^{[2]}\cap D}(1 - x^D_{ef})$ is zero and the right-hand side of $C_{D,H}$ is one. Thus, $C_{D,H}$ is violated. □

Since we can compute a corresponding drawing for a simple and realizable D in polynomial time, we can reformulate the crossing minimization problem for simple drawings as

> Given a graph $G = (V, E)$, find a simple realizable subset $D \subseteq E \times E$ of minimum cardinality.

This leads to the following ILP-Formulation. We use $x(F)$ as an abbreviation for the term $\sum_{(e,f)\in F} x_{e,f}$.

> minimize $x(E \times E)$
>
> subject to
>
> $\sum_{f\in E} x_{e,f} \leq 1$ $\qquad \forall\ e \in E$
>
> $x(\hat{H}^{[2]} \setminus D') - x(\hat{H}^{[2]} \cap D') \geq 1 - |\hat{H}^{[2]} \cap D'|$ for every simple D' and every forbidden subgraph H in $G_{D'}$
>
> $x_{e,f} \in \{0, 1\}$ $\qquad \forall\ e, f \in E$

Given a simple set of crossings D we can easily check if D is realizable by applying a planarity testing algorithm to G_D. If the answer is "no" we also get a forbidden subdivision H of G_D and we can separate an additional constraint $C_{D,H}$ according to the proof of Theorem 2.8 that excludes D.

2.4.2 Ordering-Based Formulation

The above subdivision-based formulation requires up to $\Omega(|E|^4)$ variables, as every edge may have to be subdivided into $\Omega(|E|)$ segments. The currently best-performing ILP model avoids this subdivision and instead considers *linear ordering problems* on each edge. Recall that the reason for the graph extension was to be able to model the order of the crossings, in order to obtain a tractable realizability problem. The *ordering-based* ILP formulation achieves this by explicitly computing an ordering of the edge crossings.

Consider an arbitrary, fixed *orientation* of the given graph G, i.e., for each undirected edge we decide on one of the two possible directions. As in the original (problematic)

approach, we introduce $\Omega(|E|^2)$ many binary variables $x_{e,f}$, one for each edge pair e, f, which should be 1 if the two indexed edges cross. The objective function is simply the sum over all these variables. We then introduce binary variables $y_{e,f,g} \in \{0,1\}$ for all edge triples e, f, g. This results in only $\Omega(|E|^3)$ additional variables. A variable $y_{e,f,g}$ should be 1 if and only if the edge e is crossed by both edges f and g, and the crossing with f occurs prior to the crossing with g, w.r.t. the fixed edge orientation. Conceptually, the variables $y_{e,\cdot,\cdot}$, when properly bound to their corresponding x variables, then form the variables of a linear-ordering problem with the additional property that some elements need not to be ordered at all. We can achieve this via

$$\begin{aligned} x_{e,f} &\geq y_{e,f,g} \\ x_{e,g} &\geq y_{e,f,g} \\ 1 + y_{e,f,g} + y_{e,g,f} &\geq x_{e,f} + x_{e,g} \\ y_{e,f,g} + y_{e,g,f} &\leq 1 \\ y_{e,f,g} + y_{e,g,h} + y_{e,h,f} &\leq 2 \end{aligned}$$

over the suitable edge indices. The first two constraints guarantee that the x variables (counting the crossing in the objective function) are set whenever a corresponding y variable is set. The third constraint ensures that whenever there are two crossings occurring on the same edge (here: on e), their relative order has to be specified. Then, we have to ensure in the fourth constraint that this order is unique. The last constraint, known as a *3-cycle constraint*, ensures that the order given by the y variables is in fact a linear, i.e., acyclic, order.

Using this setup it remains to introduce Kuratowski constraints much like the ones described for the subdivision-based formulation. Recall that D' in the subdivision-based formulation described a simple set of edge crossings. Similarly, we now consider a (technically more involved) *crossing shadow* $(\mathcal{X}, \mathcal{Y})$ instead. It can be thought of as a minimal description of a not-necessarily realizable crossing situation. I.e., $\mathcal{X}$ ($\mathcal{Y}$) lists x variables (y variables, respectively) that should be set. The "minimality" of this description is achieved by avoiding to list an x variable if a corresponding y variable in $\mathcal{Y}$ already induces that it has to be set. Similarly, we use the transitivity property of a linear ordering, and, e.g., do not include the variable $y_{e,f,h}$ if $y_{e,f,g}$ and $y_{e,g,h}$ are already in $\mathcal{Y}$. For a concise definition, we refer the reader to [CMB08]. Considering all possible crossing shadows $(\mathcal{X}, \mathcal{Y})$ and all thereby induced Kuratowski subdivisions H, we can require:

$$x(\hat{H}^{[2]}) \geq 1 - \sum_{x' \in \mathcal{X}} (1 - x') - \sum_{y' \in \mathcal{Y}} (1 - y')$$

2.4.3 Branch-and-Cut-and-Prize

For a practical implementation, we can omit variables in some cases. The graph G can be split up into its blocks first, which can be solved separately. The crossing number of G is equal to the sum of the crossing numbers of its blocks. Furthermore, it is easy to show that adjacent edges do not cross in an optimal drawing and no edge crosses itself, i.e., we can restrict ourselves to good drawings. Furthermore, we may apply more sophisticated preprocessing strategies like the *non-planar core reduction* [CG09], which further shrinks the graph based on its triconnectivity structure. Thereby, it may be necessary to introduce integer weights $w : E \to \mathbb{N}$ on the edges. A crossing between the edges e and f should then be counted as $w(e) \cdot w(f)$, which is easily achievable in both above ILP formulations by using these products as the coefficient for the respective x variables.

```
L := {initial problem}                    {L denotes the list of unsolved problems}
repeat
  Choose a subproblem Π ∈ L and set L := L \ {Π}
  repeat
    Let x̂ be an optimal solution for the linear relaxation of Π
    if x̂ is not feasible for Π then
      Separate violated inequalities and add them to the LP
    end if
  until no more violated inequalities can be found
  if no feasible solution for Π could be found then
    Split Π into subproblems and add them to L
  end if
until L = ∅
Print the best found feasible solution
```

Figure 2.10 An overview of the branch-and-cut approach.

Because of the exponential number of constraints, we cannot create them in advance and solve the ILP in a single step. A well-suited method for this class of ILPs is the *branch-and-cut approach*. The basic structure of a branch-and-cut based algorithm is outlined in Figure 2.10. The referred linear relaxation of Π can be easily obtained by dropping the integrality constraints, i.e., variables are allowed to be fractional.

In the case of zero-one integer linear programs, the set of unsolved subproblems L is organized as a binary tree, called the *branch-and-bound tree*. Each subproblem corresponds to a node in the tree and the list of unsolved problems L is represented by its leaves. If we need to split a problem Π into subproblems, we choose a fractional *branching variable* and create two new subproblems by setting the branching variable to zero and one, respectively.

Whenever we split a problem into two subproblems by setting the branching variable to zero and one, respectively, we can compute a local lower bound. This is the best value for the objective function that can be obtained subject to the assignments of values for the branching variables up to the root node. If this value is greater than the global upper bound, we can discard all descendants of the current subproblem since they can never improve the current feasible solution.

A severe problem of this approach is the *separation problem*: "Given a class of valid inequalities and a vector $z \in \mathbf{R}^n$, either prove that z satisfies all inequalities in the class, or find an inequality which is violated by z." Although we can easily separate violated inequalities for integral solution vectors according to the proof of Theorem 2.8, the problem becomes severe within the branch-and-cut framework since we have to deal with fractional values.

This problem can be solved heuristically by rounding variables to either zero or one, but one cannot guarantee that there is no violated inequality if the graph realizing the crossing D or $(\mathcal{X}, \mathcal{Y})$ is planar. In this case, we have to select a branching variable and split the current problem into two subproblems by setting the branching variable to 0 and 1, respectively.

The major bottleneck when following this approach then remains the large number of variables, rendering both approaches useless as such. Yet the concept of *column generation* turns out to allow drastic speed-ups of the algorithms. Conceptually, and somewhat similar to the separation approach, we start with a small subset of the variables. After solving the LP relaxation, we not only have to solve the separation problem ("is our solution too good

because some constraints are missing?"), but also the pricing problem: "Is our solution too bad because some variables are missing?" Observe the difference in the obtained bounds when constraints or variables are missing, which, in general, leads to weaker bounding strategies than applicable to pure branch-and-cut approaches.

Following the traditional approach based on the Dantzig-Wolfe decomposition [DW60], we can solve the pricing problem in a purely algebraic way by computing the reduced costs of the variables not already in the model and adding them based on their sign. It turns out that this approach, denoted by *algebraic pricing*, already speeds up the computation, but we can do much better.

In a *combinatorial column generation scheme*, we refrain from computing reduced costs, but try to incorporate our problem-specific knowledge to obtain more efficient strategies. In particular, our special-purpose generation schemes allow us to overcome the aforementioned bounding problem, and retain the fact that the LP-relaxation always gives a lower bound to the problem, even when constraints and variables are missing.

We start with the observation that in most practical applications, most of the edges will not be crossed at all or are only involved in one crossing. On these edges, we do not have any ambiguity with the order of crossings, and the realizability problem is easy. The central idea for the combinatorial column generation scheme for both formulations can be roughly described as such: we start without any special constructions to avoid crossing-order ambiguities, i.e., we do not subdivide the edges for the subdivision-based formulation and do not introduce any y variables for the ordering-based formulation. Recall that the crossing order only becomes crucial when considering Kuratowski constraints; these are only generated via separation on a rounded solution. So, we use the branch-and-cut framework as outlined above. Whenever the separation routine considers a rounded solution where two or more edges cross the same edge, and their order is hence ambiguous, we introduce the necessary variables and constraints from the original model which are necessary to decide this order. Then, the LP relaxation is recomputed. We refrain from discussing the relatively technical details of which variables or subdivisions are necessary, and refer to [CGM09] and [CMB08] for the two formulations instead. The interesting part is that adding variables in such a way will never decrease the objective function.

Overall, the currently most efficient approach from the practical point of view is the ordering-based formulation, together with its combinatorial column generation scheme, the aforementioned preprocessing strategies and upper bounds obtained via the strong planarization heuristic that we will discuss in the following section. Furthermore, the separation routine is improved by not looking for single Kuratowski subdivisions in the rounded solution, but by applying an algorithm that obtains several such subdivisions in one pass requiring only linear time in input and output size [CMS08, CMS07]. This allows to solve sparse real-world graphs with up to 100 vertices to provable optimality within reasonable time bounds on average hardware; cf. Figure 2.4. Yet the subdivision-based formulation allows extensions to other crossing number concepts where a pair of edges crosses multiple times, e.g., the simultaneous crossing number [CJS08].

2.5 The Planarization Method

2.5.1 Overview

The most prominent and practically successful method for solving the crossing minimization problem heuristically is the *planarization approach*. This approach was introduced by Batini, Talamo, and Tamassia in [BTT84] and can be viewed as a general framework that addresses the problem with a two-step strategy. Each step aims at solving a particular

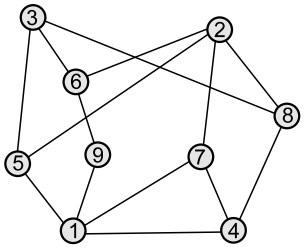
optimization problem for which various solution methods are possible. Let $G = (V, E)$ be the graph for which we want to find a crossing minimal drawing. Then, the two steps to be executed are:

1. Compute a planar subgraph $P = (V, E_p)$ of G. The objective is to have as many edges in P as possible.
2. Reinsert the edges not contained in the planar subgraph, i.e., insert the edges in $E \setminus E_p$ into P. During this edge insertion process, edge crossings that occur when inserting an edge are replaced by dummy vertices with degree four, so that the graph remains planar. The objective is to keep the number of dummy vertices (and thus the number of crossings in the final drawing) as small as possible.

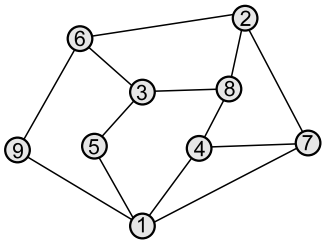
Figure 2.11 shows an example with the different stages of the approach. In this case, the planar subgraph contains all but one edge (edge (2,5) is missing) and the final drawing of G has only a single crossing.

The outcome of the planarization procedure is a planar graph $G_p = (V \cup V_d, E_p)$ such that every planar drawing of G_p implies a drawing of G with at most $|V_d|$ crossings. Hence, we also say that G_p is a *planarized representation* of G with (at most) $|V_d|$ crossings. We can obtain such a drawing of G as follows. First, we compute an embedding of G_p. Then, we have to distinguish two situations for each dummy vertex $v \in V_d$ (see Figure 2.12). If the corresponding edges of G, say, e and e', cross each other, then v in fact represents a crossing between e and e' in the drawing of G. Otherwise, e and e' just touch and we can save a crossing.

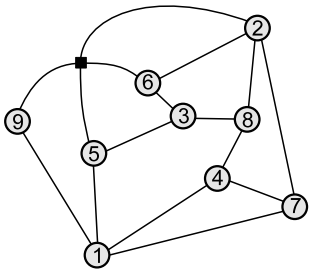
The two optimization problems we have to solve in the planarization approach are the *maximum planar subgraph problem* (MPSP) and the *edge insertion problem* (EIP). Both problems are NP-hard and are usually solved in practice by applying heuristic approaches. One reason for that is that even an optimal solution of MPSP in the first step and of EIP in the second step does not yield a crossing minimal solution in general. We show an example



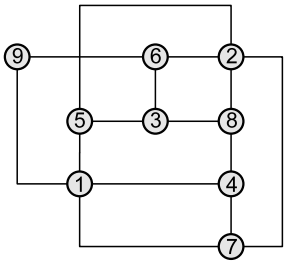
(a) Graph G.



(b) Planar subgraph P of G.



(c) Planarized representation G_p of G.



(d) Final drawing of G.

Figure 2.11 A sample application of the planarization method.

Figure 2.12 (a) The edges e and e' cross at dummy vertex v; (b) e and e' just touch at v.

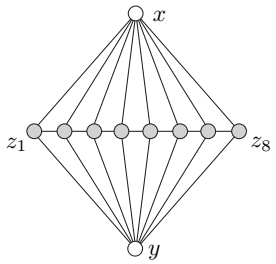


Figure 2.13 A wall with width 8.

(see [GMW05]) where the maximum planar subgraph contains all but one edge, but even an optimal solution of the edge insertion problem results in arbitrary many crossings, whereas a crossing minimal drawing has only two crossings.

We define a wall graph as follows. A *wall* with width k consists of the vertices $x, y, z_1, \ldots, z_k$, the edges (z_i, z_{i+1}) for $1 \leq i < k$, and the edges (x, z_i) and (y, z_i) for $1 \leq i \leq k$; see Figure 2.13 for an example of a wall with width 8. The vertices x and y are called the *poles* of the wall. A wall with width greater than 2 is a triconnected planar graph.

For an even number $m \geq 2$, the graph G_m is constructed in the following way; compare Figure 2.14(a). We start with a ring of walls $W_1, \ldots, W_6$ with width $m+1$, where the poles of adjacent walls in the ring are identified. We denote the pole vertices with $w_1, \ldots, w_6$ such that the poles of W_1 are w_1 and w_2, and so forth. For each wall W_j, the other two vertices on the boundary are denoted with u_j^i and u_j^e; see Figure 2.14(a). Moreover, the edges $e_1 = (u_1^e, w_3)$, $e_2 = (u_6^e, w_5)$, $e_3 = (u_2^i, u_3^i)$, and $e_4 = (u_5^i, u_4^i)$ are added, $m/2$ vertices are inserted by splitting edge (u_3^i, w_4) and $m/2$ vertices are inserted by splitting edge (w_4, u_4^i), and every created split vertex is connected with vertex w_1 by an edge h_j, $1 \leq j \leq m$. We want to insert edge (v_1, v_2) with $v_1 := u_1^i$ and $v_2 := u_6^i$, and we call the graph after addition of this edge G'_m.

By construction, G_m is triconnected and planar. In particular, G_m has only two embeddings which are mirror images of each other. It is easy to see that an optimal insertion of edge (v_1, v_2) crosses m edges, namely, $h_1, \ldots, h_m$, since passing through a wall would require at least $m + 1$ crossings. On the other hand, there is a drawing of G'_m with only 2 crossings as shown in Figure 2.14(b). Here, only the two crossings e_1 with e_3 and e_2 with e_4 occur, independent of the choice of m.

In summary, this construction shows that the planarization approach may yield arbitrarily bad solutions even if both steps are solved optimally. On the other hand, practical experience has shown that it leads to excellent results in many applications even if each step is only solved heuristically. In the sequel, we address the two optimization problems—finding a planar subgraph and reinserting a set of edges—in detail, and discuss various solution methods.

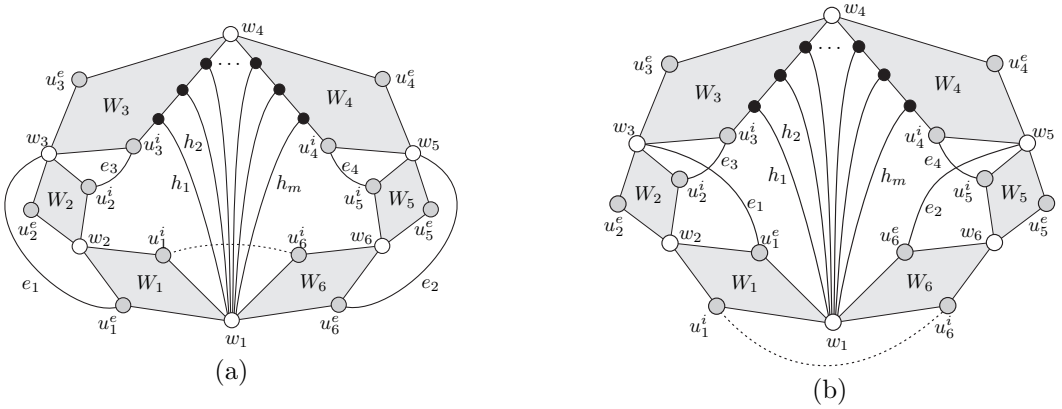


Figure 2.14 (a) The graph G_m; each shaded region represents a wall with width $m+1$. The dashed edge (u_1^i, u_6^i) is the edge to be inserted. (b) A drawing of the graph G'_m with only two crossings.

2.5.2 Planar Subgraphs

In many practical applications, we expect that a graph $G = (V, E)$ can be made planar by removing only a few edges. Therefore, it is reasonable to use a planar subgraph with as many edges as possible as a starting point for crossing minimization. A *maximum planar subgraph* of G is a planar subgraph with the maximum number of edges among all planar subgraphs of G. If, in addition, a weight w_e is given for each edge of G, a *maximum weight planar subgraph* is a planar subgraph $P = (V, E')$ of G such that the sum of all edge weights $\sum_{e \in E'} w_e$ of P is maximum. Hence, a maximum planar subgraph is a special case of the weighted version with $w_e = 1$ for every edge $e \in G$. In both the weighted and the unweighted case, the problem of finding such a subgraph is NP-complete as shown in [LG77, GJ79].

Jünger and Mutzel [JM96] presented a branch-and-cut algorithm for finding a maximum weight planar subgraph. An overview of the branch-and-cut approach can be found in Section 2.4.

Let $\mathcal{P}_G$ be the set of all planar edge-induced subgraphs of G. For each planar subgraph $P = (V, F) \in \mathcal{P}_G$, we define its incidence vector $\chi^P \in \mathbf{R}^E$ by setting $\chi_e^P = 1$ if $e \in F$ and $\chi_e^P = 0$ if $e \notin F$. This yields a 1-1-correspondence of the planar subgraphs with certain $\{0,1\}$-vectors in $\mathbf{R}^E$. The planar subgraph polytope $\mathcal{PLS}(G)$ of G is defined as the convex hull over all incidence vectors of planar subgraphs of G:

$$\mathcal{PLS}(G) := \text{conv}\{\chi^P \in \mathbf{R}^E \mid P \in \mathcal{P}_G\} .$$

Let $w \in \mathbf{R}^E$ be a vector assigning a weight to each edge. The problem of finding a maximum weight planar subgraph can thus be written as the linear program

$$\max\{w^{\mathrm{T}} x \mid x \in \mathcal{PLS}(G)\},$$

since the vertices of the polytope $\mathcal{PLS}(G)$ are exactly the incidence vectors of the planar subgraphs of G. In order to apply linear programming techniques, $\mathcal{PLS}(G)$ has to be represented as the solution of an inequality system. Because of the NP-hardness of the problem, we cannot expect to find a full description of $\mathcal{PLS}(G)$. Jünger and Mutzel show several facet-defining inequalities of the polytope, including Kuratowski inequalities, which are based on the fact that a planar graph contains no subdivision of K_5 and $K_{3,3}$, and Euler inequalities, which are based on the maximal number of edges in a planar graph given by Euler's formula. Further facet-defining inequalities can be found in [JM96].

```
Require: graph G = (V, E)
Ensure: maximal planar subgraph P of G
  P := a spanning tree of G
  F := E \ E(P)
  for all e ∈ F do
    if P ∪ e is planar then
      P := P ∪ e
    end if
  end for
```

Figure 2.15 A simple algorithm for computing a maximal planar subgraph.

Using these inequalities, a branch-and-cut algorithm can be derived that adopts the planarity testing algorithm by Hopcroft and Tarjan [HT74] for cutting plane generation and as lower-bound heuristic. Computational results show that the algorithm is able to provide a provably optimal solution quite fast if the number of edges to be deleted is small. However, the method is quite complicated to understand and to implement. Moreover, if the number of deleted edges exceeds 10, the algorithm usually needs far too long to be acceptable for practical computation.

Since finding a maximum planar subgraph is hard, the problem of finding just a maximal planar subgraph has received much attention. A *maximal planar subgraph* of $G = (V, E)$ is a planar subgraph $P = (V, E \setminus F)$ of G such that adding any edge of F to P destroys the planarity, i.e., $P \cup e$ is not planar for every $e \in F$.

A widely used standard heuristic for finding a maximal planar subgraph is to start with a spanning tree of G, and to iteratively try to add the remaining edges one by one; see Figure 2.15. In every step, a planarity testing algorithm is called for the obtained graph. If the addition of an edge would lead to a nonplanar graph, then the edge is disregarded; otherwise, the edge is added permanently to the planar graph obtained so far. After $|F|$ planarity tests, we obtain a maximal planar subgraph P of G. Planarity can be tested in linear time; see Chapter 1, or [HT74, BL76, BM04]. Hence, the running time of the procedure is $\mathcal{O}((1 + |F|)(|V| + |E|))$.

This incremental approach can be made more efficient by using incremental planarity testing algorithms. Di Battista and Tamassia [DT96] presented an algorithm that tests in $\mathcal{O}(\log |V|)$ time if an edge can be added while preserving planarity, and that performs the required updates of the data structure when adding an edge in $\mathcal{O}(\log |V|)$ amortized time. The algorithm uses the data structures BC-tree and SPQR-tree equipped with efficient, dynamic update operations. A *BC-tree* represents the *block-cutvertex tree* of a connected graph G which consists of the interrelation of the blocks (B-nodes) and cutvertices (C-nodes) of G. It has an edge (c, B) if c is a cutvertex of G contained in block B. *SPQR-trees* have been introduced by Di Battista and Tamassia in [DT89]. They represent the decomposition of a biconnected graph into its triconnected components, which essentially consists of serial (expressed by S-nodes), parallel (P-nodes), and simple, triconnected structures (R-nodes). Additionally, Q-nodes represent the original edges of G. The specific structures of tree nodes are given by skeleton graphs that are associated with each node. Using these data structures, a maximal planar subgraph can be found in $\mathcal{O}(|E| \log |V|)$ time. SPQR-trees are also useful in a static environment for the representation of all planar embeddings of a graph. The static data structure can be built in linear time [HT73, GM01] using an algorithm for dividing a graph into its triconnected components.

The running time for incremental planarity testing has been improved by La Poutré [La 94] to $\mathcal{O}(\alpha(|E|,|V|))$ amortized time per query and update operation. This yields an almost linear time algorithm for the maximal planar subgraph problem that runs in $\mathcal{O}(|V|+|E|\cdot\alpha(|E|,|V|))$ time. Here, $\alpha(x,y)$ denotes the inverse Ackermann function, which means that $\alpha(x,y)$ is a function that grows extremely slowly. A linear time algorithm for finding a maximal planar subgraph is given by Djidjev [Dji95]. This algorithm uses BC- and SPQR-trees and applies a fast data structure for online planarity testing in triconnected graphs.

Jayakumar et al. [JTS89, JLM98] proposed a method for computing a planar subgraph that is based on PQ-trees. The PQ-tree data structure has been developed by Booth and Lueker [BL76] for solving the problem of finding permissible permutations of a set U. The permissible permutations are those in which certain subsets $S \subseteq U$ occur as consecutive subsequences. Drawbacks of this planar subgraph algorithm are that it cannot guarantee to find a maximal planar subgraph, and that the theoretical worst-case running time is $\mathcal{O}(|V|^2)$. However, in practice it is usually very fast and the quality of the results can be improved by introducing random events and calling the algorithm several times. The algorithm starts by computing an st-numbering of G, which determines the order in which the vertices are processed. A simple but useful randomization is to choose a random edge (s,t) for each run.

The trivial approach for finding a planar subgraph consists of computing a spanning tree. If G is a graph with n vertices and c components, then this approach has an approximation factor of $\frac{n-c}{3n-6c} > \frac{1}{3}$ for MPSP, since a spanning tree of G contains $n-c$ edges, and a planar graph with c components has at most $3n-6c$ edges by Euler's formula. Surprisingly, we cannot guarantee a better approximation factor than that of the spanning tree approach if we demand that the computed subgraph must be maximal planar; see [DFF85]. Călinescu et al. [CFFK98] present an algorithm with approximation factor 4/9 that runs in $\mathcal{O}(m^{3/2}n\log^6 n)$ time, where m is the number of edges of G. For the maximum weight planar subgraph problem, the simple approach is to compute a maximum weight spanning tree, which gives an approximation factor of 1/3, and the best algorithm [CFKZ03] achieves an approximation factor of $1/3 + 1/72$.

2.5.3 Edge Insertion

The planar subgraph P computed in the first step of the planarization approach is a good starting point for finding a planarized representation G_p of G with few crossings. In practice, we expect that only a small number of edges has to be inserted into P in order to obtain G_p. However, the edge insertion step fixes the crossings in the final drawing, and the choice of the edge insertion technique may have a significant impact on the quality of the final solution. Ziegler and Mutzel [MZ99, Zie00] have shown that even a restricted variant of the edge insertion problem is NP-hard: The *constrained crossing minimization problem* (*CCMP*) asks for the minimum number of crossings required for inserting a set of edges into a fixed embedding. They also present a branch-and-cut algorithm to solve CCMP. However, experiments show that it can only solve instances to provable optimality if there are less than 10 edges to be inserted.

Gutwenger [GM04, Gut10] has conducted an extensive study on crossing minimization heuristics, including different methods for edge insertion. Figure 2.16 shows the general framework for edge insertion used in this study. It contains three essential parts leaving room for enhancement:

Single edge insertion: The edges are inserted into the planarized representation individually one after the other. The simple approach for inserting a single edge e

Require: planar subgraph $P = (V, E_P)$ of $G = (V, E)$
Ensure: planarized representation G_p^* of G

```
Let E \ E_P = {e_1, ..., e_k}
best := ∞
for i := 1 to nPermutations do
  Let σ be a randomly chosen permutation of {1, ..., k}
  G_p := P
  for j := 1 to k do
    Insert edge e_σ(j) into G_p
  end for
  Determine a set R ⊆ E of edges for which postprocessing shall be applied
  repeat
    for all e ∈ R do
      Remove edge e from G_p
      Insert edge e into G_p
    end for
  until number of crossings in G_p has not decreased
  current := number of crossings in G_p
  if current < best then
    G_p^* := G_p; best := current
  end if
end for
```

Figure 2.16 Edge insertion with postprocessing and permutation.

is to fix an embedding Π of G_p and to insert e into Π. However, the choice of Π may have a considerable influence on the number of edges that e has to cross. A more sophisticated algorithm introduced by Gutwenger et al. [GMW05] is able to insert e with the minimum number of crossings among all embeddings of G_p.

Postprocessing: After all edges have been inserted, a simple postprocessing technique tries to improve the current solution. It determines a set of edges R which have one or more crossings and repeatedly tries to find a better insertion path for each of them by removing an edge from G_p and inserting it again. Variants for the choice of R include all edges, only the edges $e_1, \ldots, e_k$, or some portion of the edges with the most crossings (see [Gut10] for more details).
An alternative approach combines the edge insertion with the postprocessing. Instead of performing the remove-reinsert strategy after all edges have been inserted, we can perform this strategy after each edge insertion. The idea behind this variation is to keep the number of crossings low as early as possible. We call this strategy incremental postprocessing.

Permutation: The order in which the edges $e_1, \ldots, e_k$ are processed also affects the final number of crossings. Calling the complete edge insertion process several times with different, randomly chosen permutations of the edge list $e_1, \ldots, e_k$ may significantly improve the solution. The parameter *nPermutations* in the algorithm determines the number of permutation rounds.

Apart from the choice of some parameters like the number of permutation rounds or the selection of the edges for postprocessing, the challenging part of the algorithm is the insertion of a single edge. We consider the two variants—insertion with *fixed* and with *variable* embedding—in more detail.

Fixed Embedding. Suppose, we want to insert edge $e = (v, w)$ into the planar graph G_p. Let Π be a fixed embedding of G_p. We construct the *extended dual graph* G^* of Π with respect to e as follows. The vertices of G^* are the faces of Π plus two new vertices v^* and w^* representing v and w. For each edge e' in G_p, we have an edge in G^* connecting the two faces separated by e' (if e' is a bridge, we have a self-loop in G_p). Additionally, we have an edge (v^*, f_v) for each face f_v adjacent to v, and (w^*, f_w) for each face f_w adjacent to w.

We observe that inserting e into Π corresponds to finding an (undirected) path from v^* to w^* in G^*. If such a path has length ℓ, then we can insert e with $\ell - 2$ crossings, since the first and the last edge on this path do not produce a crossing. Therefore, in order to insert e into Π with the minimum number of crossings, we have to find a shortest path from v^* to w^* in G^*. This is possible in linear time using a simple breadth-first search traversal starting at v^*. Figure 2.17 shows a nontrivial example. Here, we want to connect the vertices 1 and 2. The dashed vertices and edges belong to the extended dual graph. The optimal solution highlighted in bold crosses four edges.

Though we can easily find a crossing minimal solution if the embedding of G_p is fixed, the drawback of this method is that fixing an unfavorable embedding may result in an arbitrarily bad solution. Figure 2.18(a) gives an example of such a family of graphs G_k with embeddings Γ_k. The black fat lines in this figure denote bundles of $k + 1$ parallel

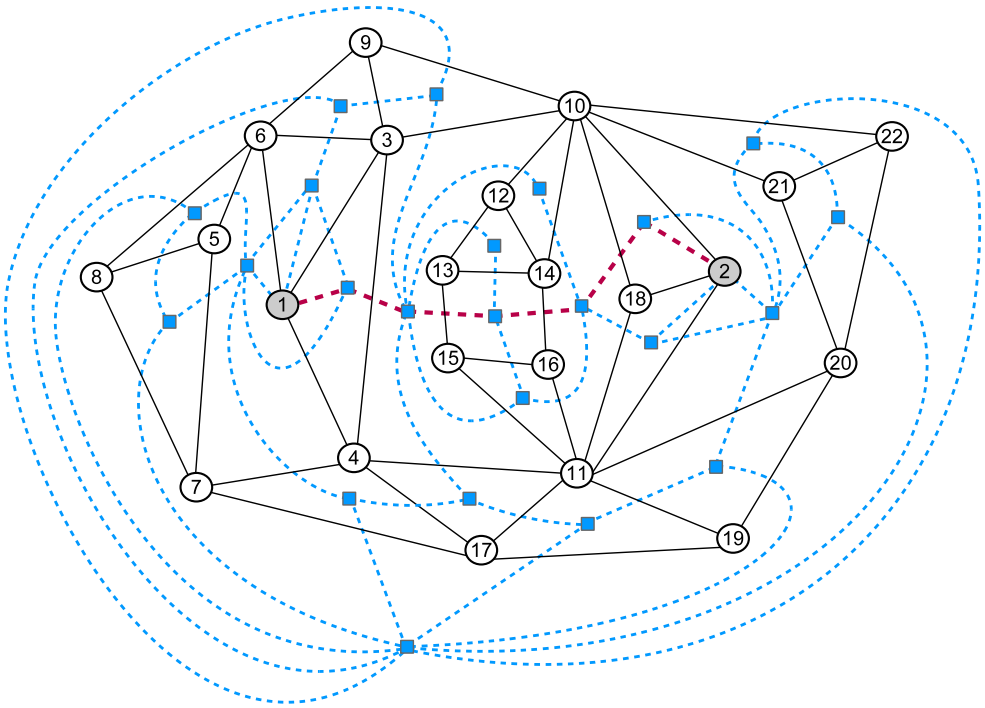
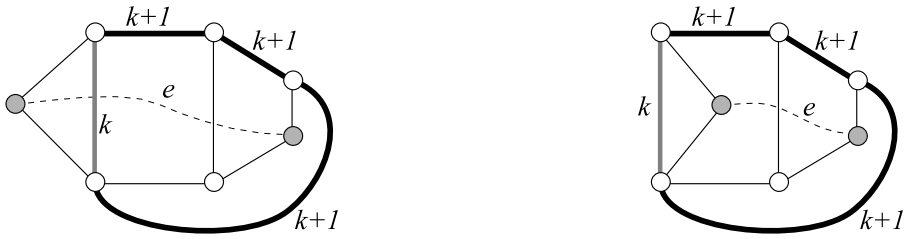


Figure 2.17 Edge insertion with fixed embedding by finding a shortest path in the extended dual graph.



(a) Fixed embedding Γ_k. (b) Optimal Embedding.

Figure 2.18 A family of graphs G_k and embeddings Γ_k for which the insertion of an edge e requires k crossings more than the optimal solution.

edges, and the gray fat line a bundle of k parallel edges. Hence, inserting edge e into the given embedding requires at least $k+1$ crossings. On the other hand, it is possible to insert e with only one crossing by changing the embedding; see Figure 2.18(b). It is easy to see that this example can also be adapted to the case of simple graphs by splitting all the edges in each bundle.

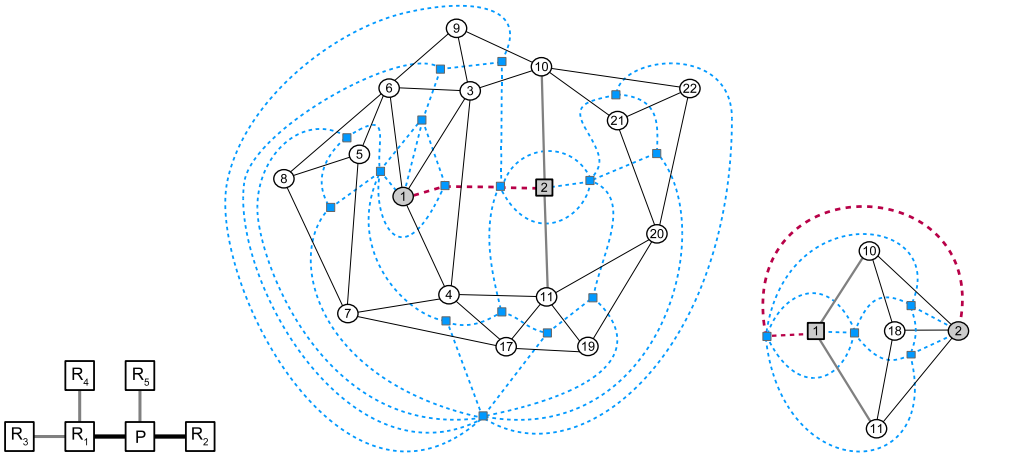
Variable Embedding. Surprisingly, there exists also a linear time algorithm for finding an optimal embedding of G_p which allows to insert e with the minimum number of crossings. The algorithm by Gutwenger et al. [GMW05] uses the data structures BC-tree and SPQR-tree for the representation of all planar embeddings of a connected graph. These decomposition trees allow to enumerate all possible embeddings of a connected graph. Basically, we can

- put any subgraph which is only attached to the rest of the graph at a cutvertex into any face containing this cutvertex;
- arbitrarily permute parallel structures joined at a separation pair; and
- mirror any subgraph that is only attached to the rest of the graph at a separation pair.

Let G be a graph and $\mathcal{T}$ its SPQR-tree. We denote the skeleton of a node μ in $\mathcal{T}$ with *skeleton*(μ). Each edge e in a skeleton represents a subgraph of G called the *expansion graph* of e. Replacing each edge in a skeleton by its expansion graph yields G again.

In order to find the optimal insertion path for (v, w) in G, it is essentially sufficient to consider only the R-nodes in the SPQR-trees of the blocks of G. Assume first that G is already biconnected. Let $\mathcal{T}$ be its SPQR-tree and let $\mu_1, \ldots, \mu_k$ be the shortest path in $\mathcal{T}$ between a node μ_1 with $v \in$ *skeleton*(μ_1) and a node μ_k with $w \in$ *skeleton*(μ_k). For each R-node on this path, we expand its skeleton S in the following way. First, we make sure that we have a representative for both v and w. If one of these vertices, say, v, is not yet contained in S, then there is an edge whose expansion graph contains v and we split this edge introducing a representative for v. Then, we replace every edge that was not split with its expansion graph and compute an arbitrary embedding Π of the resulting graph. For this fixed embedding, we determine the ordered list of edges we have to cross when inserting an edge from the representative of v to the representative of w as described above for the fixed embedding scenario. If we do this for every R-node in $\mu_1, \ldots, \mu_k$, and if we join the resulting edge lists in the order they appear on the path from μ_1 to μ_k, then we obtain an optimal edge insertion path for inserting the edge (v, w).

If G is not biconnected, we determine the shortest path $B_1, c_1, \ldots, c_{k-1}, B_k$ in the BC-tree of G such that $v \in B_1$ and $w \in B_k$. Then, we find an optimal edge insertion path



(a) SPQR-tree. (b) Expanded skeleton S_1 (c) Expanded skeleton S_2

Figure 2.19 Edge insertion with variable embedding.

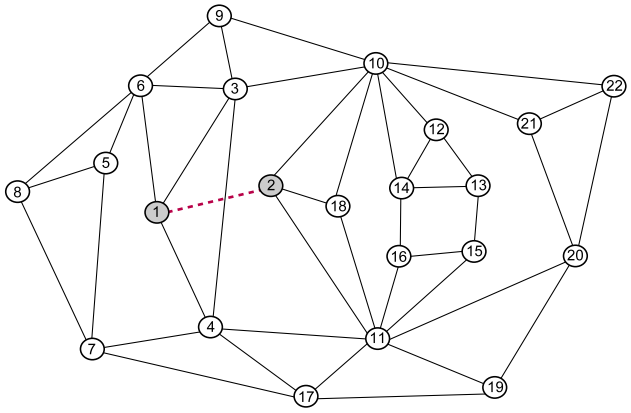


Figure 2.20 An embedding of the example graph that allows to embed edge (1,2) with the minimum number of crossings.

for (c_{i-1}, c_i) in B_i for every $i = 1, \ldots, k$, where $c_0 := v$ and $c_k := w$. We can simply join the resulting insertion paths p_i to obtain an optimal insertion path $p_1, \ldots, p_k$ for inserting (v, w).

The corresponding embedding that allows to insert (v, w) with the minimum number of crossings is easy to find. We in fact insert the edge (v, w) into the planarized representation G_p by creating dummy vertices for edge crossings. The construction above guarantees that the resulting graph is planar. Then, we compute an embedding of this planar graph and remove the inserted edge(s) again.

Figure 2.19 continues our example for the insertion strategy with variable embedding. In this case, the graph is biconnected and the corresponding SPQR-tree has the structure depicted in Figure 2.19(a). The relevant path in the SPQR-tree is R_1, P, R_2, which contains two R-nodes. The expanded skeleton graphs S_1 for R_1 and S_2 for R_2 are shown in Figure 2.19(b) and (c). We need only a single crossing in S_1 and no crossing at all in S_2. Hence, an optimal solution will only cross a single edge, which is edge (3,4) in our solution. Figure 2.20 shows an embedding that allows to insert (1,2) with only one crossing.

2.5.4 Experimental Results

Recently, Gutwenger [Gut10] presented an extensive experimental study on the planarization approach for crossing minimization and analyzed the effect of pre- and postprocessing strategies, edge insertion, and permutations, including the nonplanar core reduction (NPC) as preprocessing, edge insertion with fixed (FIX) and variable (VAR) embedding, and various postprocessing strategies (all edges (ALL), only the inserted edges (INS), x% of the edges with the most crossings (MOSTx), and incremental postprocessing (INC)). The planar subgraph was computed using the PQ-tree-based algorithm with 100 random iterations. Two benchmark sets of graphs have been used in this study:

- The *Rome graphs* [DGL+97] are a collection of more than 11.000 graphs ranging from 10 to 100 vertices, which have been generated from a core set of 112 graphs used in real-life software engineering and database applications.
- The *Artificial graphs*[1] are a collection of nonplanar graphs with known crossing numbers. It contains 1946 graphs with up to 250 vertices and consists of cross products of cycles ($C_m \times C_n$), 5-vertex graphs with paths ($G_i \times P_n$), 5-vertex graphs with cycles ($G_i \times C_n$), and generalized Petersen graphs ($P(m, 2)$ and $P(m, 3)$).

Table 2.1 shows a ranking of some selected strategies, sorted by average number of crossings for graphs with 100 vertices.

rank	crossings	time [s]	EI	PRE	POST	PERM
1	26.71	9.387	VAR	NPC	INC	20
2	27.14	4.681	VAR	NPC	INC	10
3	28.49	1.857	VAR	NPC	ALL	20
4	28.69	0.727	FIX		INC	20
5	30.43	0.490	VAR	NPC	INC	1
6	30.52	0.221	FIX		ALL	20
7	32.66	0.105	VAR	NPC	ALL	1
8	33.33	0.098	FIX	NPC	ALL	1
9	33.96	0.067	FIX		INC	1
10	35.09	0.041	FIX		ALL	1
11	35.79	0.040	FIX		MOST25	1
12	38.38	0.037	FIX		MOST10	1
13	41.61	0.036	FIX		INS	1
14	45.47	0.034	FIX		NONE	1

Table 2.1 The ranking list of crossing minimization heuristics; the table shows average number of crossings and running times for graphs with 100 vertices.

Figure 2.21 compares the two edge insertion variants and some postprocessing strategies. It shows that VAR clearly dominates FIX and that postprocessing helps a lot. Although the INC strategy is rather time consuming, it justifies this by achieving excellent improvements. Using permutations also gives significant improvements, but not as much as postprocess-

[1] `available at http://ls11-www.cs.uni-dortmund.de/people/gutweng/artificial-graphs.zip`

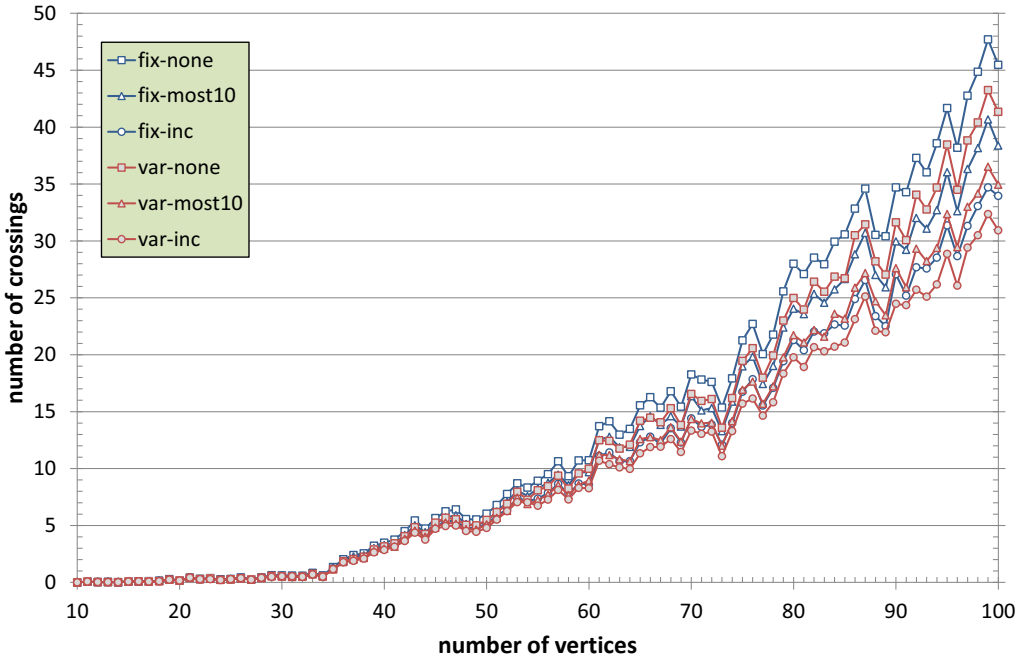


Figure 2.21 Average number of crossings for Rome graphs with various postprocessing strategies.

ing. Figure 2.22 demonstrates the effect of up to 500 permutations on graphs with 100 vertices. Performing only a few permutations achieves already good improvements; further permutations can still reduce the number of crossings, but the effect becomes smaller and smaller. The best result obtained for graphs with 100 vertices was 25.51 crossings with 500 permutations (NPC-VAR-INC).

We can judge the quality of the results better if we can compare them with the actual crossing numbers. Figure 2.23 shows the results for the artificial graphs, grouped by graph types. We observe that using edge insertion with variable embedding and incremental postprocessing already comes very close to the exact crossing numbers (OPT), whereas using fixed embedding without postprocessing achieves very bad results.

2.5.5 Beyond Edge Insertion

The central ingredient of the above discussed heuristic clearly is the efficient optimal edge insertion procedure that considers all possible embeddings. Starting from there, there are multiple extensions and generalizations.

Instead of considering only edges, we may also consider a vertex of the graph, together with all its incident edges. This problem is known as the *vertex-* or *star-insertion* problem. Reusing the ideas of the fixed-embedding edge insertion, we can easily find a BFS-based algorithm to insert a vertex into a fixed embedding of a planar graph. Yet, when considering the variable embedding setting, we cannot straightforwardly reuse many of the edge-insertion algorithm's methods, since vertex insertion does not offer the same degree of problem locality within each separate SPQR-skeleton. Chimani et al. [CGMW09] showed how to combine the SPQR-tree-based approach with a sophisticated dynamic programming scheme, to solve the vertex insertion problem in $O(\theta^2 \cdot |V|^3 \cdot |W|^2)$ time, where θ gives the

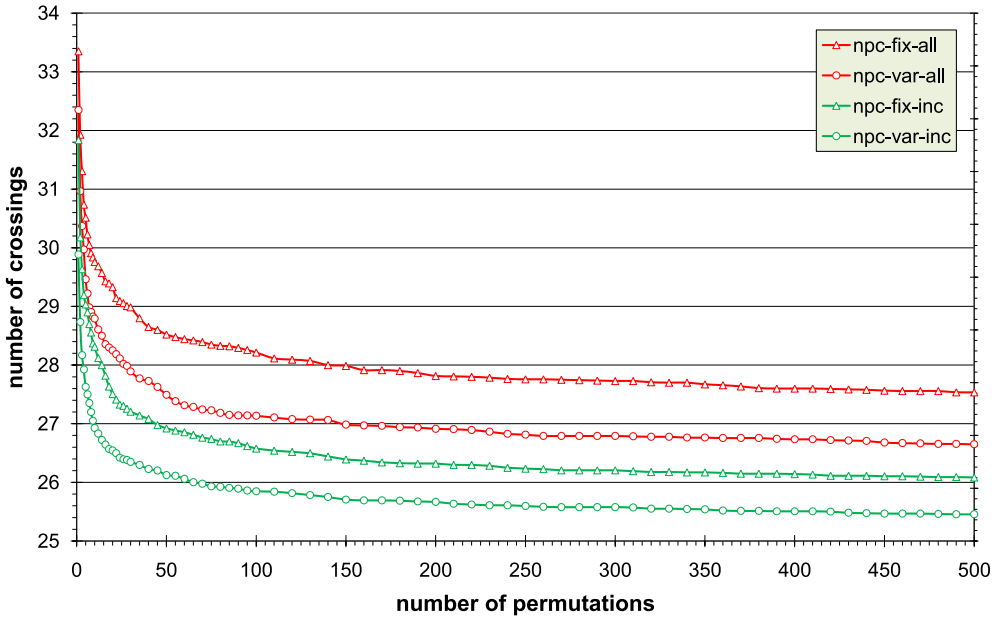


Figure 2.22 The effect of up to 500 permutations for Rome graphs with 100 vertices.

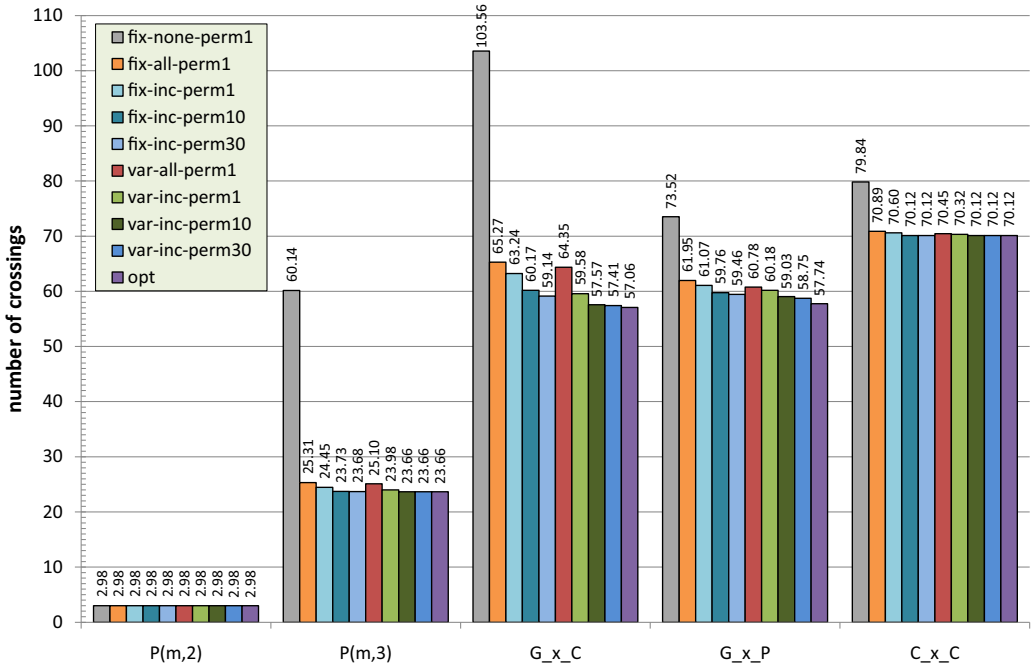


Figure 2.23 Average number of crossings for graphs with known crossing numbers.

thickness (number of edges) of the thickest P-node skeleton and W are the vertices adjacent to the inserted vertex. For a graph without P-nodes, this time reduces to $O(|V|^2 \cdot |W|^2)$.

We will revisit both the edge and the vertex insertion problem in the next section when discussing approximation algorithms. It remains to state that these two graph structures are currently the only structures for which we know that the insertion problem is efficiently solvable. It is therefore an open challenge to identify more complicated (planar) subgraphs that allow efficient insertion algorithms.

Based on the success in the traditional crossing number setting, one may consider the *minor crossing number*, or its set-restricted version that arises when considering an electrical network; see [CG07]. Such a network consists of several components (logic gates, chips, resistors,...) that are connected via wires. But such wires are usually not simple edges with one source and one target vertex, but *hyperedges*: they connect several components on the same electric potential, e.g., the output signal of one logic gate may serve as an input signal for several other gates. Graphs with hyperedges are usually called *hypergraphs*, and it is natural to try to adopt the planarization strategy to them. We may represent a hypergraph via a traditional graph by replacing each hyperedge ψ (adjacent to the vertices N) by a star, i.e., we introduce a new *hypervertex* v_ψ and add edges (v, v_ψ) for all vertices $v \in N$. For a final drawing, we are allowed to modify each such star into a tree T_ψ with N as its leaves. Such a modification is captured by the notion of the *minor crossing number* of a graph G: the smallest crossing number achievable by any graph H which has G as its minor. We may briefly describe the minor operations as removing edges and merging two adjacent vertices. It is easy to see that the expansion from a star to a tree can be obtained exactly by the inverse of the last operation. Therefore, the hypergraph crossing number is equivalent to the so-called *W-restricted minor crossing number*, where W is the set of hypervertices. By *W-restricted* we describe the constraint that the inverse minor operations may only be applied to the vertices W.

Chimani and Gutwenger [CG07] showed that inserting hyperedges (or, equivalently, inserting a vertex in the minor crossing number setting) is NP-hard, already when considering a fixed embedding of the (hyper)graph into which to insert. On the other hand, they show how to efficiently and optimally insert edges into a graph w.r.t. the minor crossing number, over all possible embeddings. This is equivalent to optimally inserting a simple edge into a hypergraph (note that during the insertion, other hyperedges become expanded to more general trees). This latter algorithm can then be applied iteratively, to (heuristically) insert a hyperedge by successive insertions of its star's edges. This in turn leads to a crossing number heuristic for electrical networks which generates drawings with astonishingly fewer crossings than the other known approaches, which are based on Sugiyama's framework.

The planarization strategy has also shown great potential when applied to the related issue of *upward drawings*, i.e., we want to draw a directed graph such that all edges point upward. Traditionally, this was solved via Sugiyama-style algorithms, but in the last years, Eiglsperger et al. [EKE03] and Chimani et al. [CGMW08] introduced algorithms reusing ideas of the planarization approach to find drawings of real-world graphs with drastically less crossings.

2.6 Approximation Algorithms

Finally, the last approach to crossing minimization that we will discuss is the search for approximation algorithms. By the end of the last century, this search has been mostly fruitless despite many attempts. It is still the case that on the one hand, no approximation algorithm for crossing minimization of general graphs with any type of guarantee could be

found; on the other hand, the theoretical complexity of approximation is unknown. The only relevant case in which provably near-optimal solutions can be generated is the case of bounded degrees.

Recall the bisection width $\mathrm{bw}(G)$ of G as defined in Section 2.2.1. Bhatt and Leighton [BL84], later improved by Even et al. [EGS00], used a bisection approach for devising a polynomial-time algorithm with a quality guarantee for the number of crossings *plus the number of vertices*; the quality, however, depends on the quality of the incorporated approximation algorithm for the bisection width. As shown later [CY94], the latter problem can be approximated within a constant factor in polynomial time. Using this result, Bhatt and Leighton's algorithm yields a drawing with $O(\log^2 |V|(\mathrm{cr}(G)+|V|))$ edge crossings in polynomial time. In other words, the number of edge crossings *plus vertices* in the constructed drawing of G is at most a factor of $O(\log^2 |V|)$ away from the optimum. In fact, for bounded degree graphs satisfying $|E| \geq 4|V|$, this yields an $O(\log^2 |V|)$-approximation algorithm for the crossing minimization problem, as in this case the number of vertices is at most linear in the minimal number of crossings.

After laying semidormant for some time, the topic of approximation algorithms for crossing numbers received a lot of attention in recent years. The first decade of this millennium saw the first constant factor approximation algorithms in this area, although only for special graph classes, and only when assuming bounded degrees. Let Δ be the maximum degree in the following.

The first class of approximation algorithms are *insertion based*. They use the insertion algorithms presented in the previous section.

An *almost planar* or *near-planar* graph G is a graph that has an edge e such that $G' := G-e$, the graph obtained from removing e, is planar. In other words, G is a planar graph plus one additional edge. For such a graph, Hliněný and Salazar [HS06] showed that inserting e optimally into a planarly embedded G' [GMW05] (considering all possible embeddings, as described for the planarization heuristic) approximates the crossing number of G. The provably tight approximation factor of $\Delta/2$ was established by Cabello and Mohar [CM10] using a different proof strategy: the lower bound is obtained by analyzing its relation to the *facial distance*, i.e., a shortest insertion path with respect to the *minor-monotone crossing number* model, unknowingly using an algorithm first outlined in [CG07]. In this setting, the inserted edge not only crosses edges but may also cross through vertices, resembling a crossing solution in a graph that has G' as its minor.

Shortly after the aforementioned algorithm to optimally insert a vertex with its incident edges into a planar graph was presented in [CGMW09], Chimani et al. [CHM12] showed that this solution in fact approximates the crossing number of *apex graphs*. Similar to above, such a graph becomes planar when removing a specific vertex, together with its incident edges. The proof argues over different flip structures in the graph's SPQR-tree and thereby reuses some strategic elements of [HS06], as the stronger bounding techniques of [CM10] seem not applicable to the apex case. Consequently, the proven approximation guarantee of factor $|W| \cdot \Delta/2$ might not be tight (thereby, W are the vertices incident to the inserted vertex, and Δ is the maximum degree of the graph into which we insert). In the worst known example, the obtained crossing number is only $|W| \cdot \Delta/4$ times the optimal solution.

The second known class of approximation algorithms are *topology based*. Thereby, we assume that the given graph is embeddable—i.e., drawable without edge crossings—on some specific surface, more complex than the traditional plane. This class of graphs is then a superset of the class of planar graphs. By clever simplification strategies, usually based on cutting the surface with its embedded graph, the surface is simplified until a planar drawing is reached. Therein, the cut edges and vertices have to be reconnected cheaply to obtain a drawing of the original graph. Interestingly, the algorithms themselves, as well as

estimating the number of the produced crossings, are relatively simply. The hard part is to show a matching lower bound in order to deduce the approximation factor.

In 2007, Gitler et al. showed in [GHLS08] that $\mathrm{cr}(G)$ is approximable within a factor of $4.5\Delta^2$ when considering *projective graphs*, i.e., graphs that are embeddable in the *projective plane*. One may think of such a projective plane as a large circular area A, on which to draw the graph without any crossings. Any line leaving A re-enters A exactly at the opposite position. Consequently, any vertex drawn on the border of A is mirrored on the opposite side as well. The key idea of the approximation algorithm now is to take such a drawing, paste A on a regular plane, and connect the "jump points" cheaply outside of A. In order to prove that this strategy yields an approximation algorithm, one has to show a matching lower bound. This is established by proving that any nonplanar projective graph contains a diamond grid of certain size, which in turn induces a lower bound on the crossing number.

Hliněný and Salazar showed in [HS07] that $\mathrm{cr}(G)$ is approximable within a factor of $12\Delta^2$ when considering (dense enough) *toroidal graphs*, i.e., graphs that are embeddable on the *torus*. Assume a graph is already drawn on a torus without any crossings (such an embedding can be found in linear time if it exists [Moh99]). We search for a shortest two-sided, non-separating loop around the torus (think, e.g., of a circular line "around" the thinner part of a torus) which only crosses through vertices of the graph, but not through edges. We then cut along this loop, effectively cutting the crossed vertices in two. The remaining surface can be thought of as a cylinder; when we cap its ends, it becomes topologically equivalent to a sphere and hence, for the purpose of drawings, to the plane. We denote this operation as *cut-and-cap*. For each pair of cut vertices, we remove the one with lower degree and route its incident edges to its twin, along the shortest path in the then-fixed embedding. In order to prove that this strategy yields an approximation algorithm, one has to show a matching lower bound for the number of crossings. This is established by proving that any toroidal graph of sufficient density contains a toroidal grid of certain minimal size as a minor. For toroidal grids of dimension $p \times q$ ($p \geq q \geq 3$), it is known that they require at least $(q-2)p/2$ crossings.

A torus is an (in fact, topologically, the unique) orientable surface of genus 1. Using the toroidal case as an inspiration, it is natural to try to generalize it to graphs embedded on any orientable surfaces of some fixed genus. Note that for every graph there is some g such that it is embeddable on a genus-g surface. The necessary basic tool of iteratively performing cut-and-cap operations on the surface's handles until we reach a sphere has already been investigated in [BPT06, DV06] in order to obtain upper bounds for the crossing number. Yet, there was no straightforward way to generalize the lower bound proof to higher genus. Only recently, Hliněný and Chimani [HC10] showed how to carefully choose the cycles to cut (both for the upper and the lower bound), such that the largest grid minor is retained within a factor depending only on the surface's genus and the graph's degree. They showed the following theorem:

Theorem 2.9 *Let G be a graph with maximum degree Δ and (densely enough) embeddable on an orientable surface of genus $g \geq 1$. There is an $O(n \log n)$ algorithm which generates a drawing of G in the plane with at most $3 \cdot 2^{3g+2} \cdot \Delta^2 \cdot cr(G)$ crossings. This is a constant factor approximation algorithm for bounded degree Δ and bounded genus g.*

These bounds are not known to be tight—in fact, they are likely not to be. Yet, some kind of density requirement (we refrain from defining the quite technical concise constraint here) will always be necessary in algorithms only performing surface cuts. Otherwise, the considered graph could even be a planar graph, awkwardly embedded on a higher genus surface.

Apart from considering restricted graph classes, one may also consider restricted crossing minimization problems, in order to obtain approximation results. For instance, for bipartite drawings with one layer fixed, Eades and Wormald [EW94] showed that there is a polynomial-time algorithm that produces drawings with at most three times as many edge crossings as necessary, for any graph G.

Acknowledgment

Markus Chimani was funded via a junior professorship by the Carl-Zeiss-Foundation.

References

[ACNS82] M. Ajtai, V. Chvátal, M.M. Newborn, and E. Szemerédi. Crossing-free subgraphs. *Annals of Discrete Mathematics*, 12:9–12, 1982.

[BCE+08] C. Buchheim, M. Chimani, D. Ebner, C. Gutwenger, M. Jünger, G. W. Klau, P. Mutzel, and R. Weiskircher. A branch-and-cut approach to the crossing number problem. *Discrete Optimization, Special Issue in Memory of George B. Dantzig*, 5(2):373–388, 2008.

[BD93] D. Bienstock and N. Dean. Bounds for rectilinear crossing numbers. *J. Graph Theory*, 17(3):333–348, 1993.

[BEJ+05] C. Buchheim, D. Ebner, M. Jünger, P. Mutzel, and R. Weiskircher. Exact crossing minimization. In P. Eades and P. Healy, editors, *Graph Drawing (Proc. GD '05)*, volume 3843 of *Lecture Notes in Computer Science*, pages 37–48. Springer-Verlag, 2005.

[BG04] H. Bodlaender and A. Grigoriev. Algorithms for graphs embeddable with few crossings per edge. Research Memoranda 036, Maastricht : METEOR, Maastricht Research School of Economics of Technology and Organization, 2004. available at `http://ideas.repec.org/p/dgr/umamet/2004036.html`.

[Bie91] D. Bienstock. Some provably hard crossing number problems. *Discrete Comput. Geom.*, 6(5):443–459, 1991.

[BL76] K. Booth and G. Lueker. Testing for the consecutive ones property interval graphs and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, 13:335–379, 1976.

[BL84] S. N. Bhatt and F. T. Leighton. A framework for solving VLSI graph layout problems. *J. Comput. Syst. Sci.*, 28:300–343, 1984.

[BM04] J. M. Boyer and W. Myrvold. On the cutting edge: simplified $o(n)$ planarity by edge addition. *J. Graph Algorithms Appl.*, 8(3):241–273, 2004.

[BPT06] K. Böröczky, J. Pach, and G. Tóth. Planar crossing numbers of graphs embeddable in another surface. *Internat. J. Found. Comput. Sci.*, 17:1005–1015, 2006.

[BTT84] C. Batini, M. Talamo, and R. Tamassia. Computer aided layout of entity-relationship diagrams. *Journal of Systems and Software*, 4:163–173, 1984.

[CFFK98] G. Călinescu, C. G. Fernandes, U. Finkler, and H. Karloff. A better approximation algorithm for finding planar subgraphs. *Journal of Algorithms*, 27(2):269–302, May 1998.

[CFKZ03] G. Călinescu, C. G. Fernandes, H. Karloff, and A. Zelikovsky. A new approximation algorithm for finding heavy planar subgraphs. *Algorithmica*, 36(2):179–205, 2003.

[CG07] M. Chimani and C. Gutwenger. Algorithms for the hypergraph and the minor crossing number problems. In *Proc. ISAAC '07*, volume 4835 of *LNCS*, pages 184–195. Springer, 2007.

[CG09] M. Chimani and C. Gutwenger. Non-planar core reduction of graphs. *Discrete Mathematics*, 309(7):1838–1855, 2009.

[CGM09] M. Chimani, C. Gutwenger, and P. Mutzel. Experiments on exact crossing minimization using column generation. *ACM Journal of Experimental Algorithmics*, 14(3):4.1–4.18, 2009.

[CGMW08] M. Chimani, C. Gutwenger, P. Mutzel, and H.-M. Wong. Layer-free upward crossing minimization. In *Proc. WEA '08*, volume 5038 of *LNCS*, pages 55–68. Springer, 2008.

[CGMW09] M. Chimani, C. Gutwenger, P. Mutzel, and C. Wolf. Inserting a vertex into a planar graph. In *Proc. SODA '09*, pages 375–383. ACM-SIAM, 2009.

[Chi08] M. Chimani. *Computing crossing numbers*. PhD thesis, TU Dortmund, 2008. `http://hdl.handle.net/2003/25955`.

[Chi11] M. Chimani. Facets in the crossing number polytope. *SIAM Journal on Discrete Mathematics*, 25(1):95–111, 2011.

[CHJM11] M. Chimani, P. Hungerländer, M. Jünger, and P. Mutzel. An SDP approach to multi-level crossing minimization. In *Proc. ALENEX'11*. SIAM, 2011.

[CHM12] M. Chimani, P. Hliněný, and P. Mutzel. Vertex insertion approximates the crossing number of apex graphs. *European Journal of Combinatorics*, 33(3):326–335, 2012.

[Chv83] V. Chvátal. *Linear Programming*. W. H. Freeman and Company, New York, 1983.

[Cim92] R. J. Cimikowski. Graph planarization and skewness. *Congressus Numerantium*, 88:21–32, 1992.

[CJS08] M. Chimani, M. Jünger, and M. Schulz. Crossing minimization meets simultaneous drawing. In *Proc. PacificVis '08*, pages 33–40, 2008.

[CM10] S. Cabello and B. Mohar. Crossing and weighted crossing number of near-planar graphs. *Algorithmica*, 2010. in print.

[CMB08] M. Chimani, P. Mutzel, and I. Bomze. A new approach to exact crossing minimization. In *Proc. ESA '08*, volume 5193 of *LNCS*, pages 284–296. Springer, 2008.

[CMS07] M. Chimani, P. Mutzel, and J. M. Schmidt. Efficient extraction of multiple Kuratowski subdivisions (TR). Technical Report TR07-1-002, June 2007, TU Dortmund, June 2007.

[CMS08] M. Chimani, P. Mutzel, and J. M. Schmidt. Efficient extraction of multiple Kuratowski subdivisions. In *Proc. GD '07*, volume 4875 of *LNCS*, pages 159–170. Springer, 2008.

[CY94] F. R. K. Chung and S.-T. Yau. A near optimal algorithm for edge separators. In *Proceedings of STOC'94*, pages 1–8, 1994.

[DFF85] M. E. Dyer, L. R. Foulds, and A. M. Frieze. Analysis of heuristics for finding a maximum weight planar subgraph. *European Journal of Operational Research*, 20(1):102–114, 1985.

[DGL+97] G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. *Comput. Geom. Theory Appl.*, 7:303–325, 1997.

[Dji95] H. N. Djidjev. A linear algorithm for the maximal planar subgraph problem. In *Proc. 4th Workshop Algorithms Data Struct.*, Lecture Notes Comput. Sci., pages 369–380. Springer-Verlag, 1995.

[DT89] G. Di Battista and R. Tamassia. Incremental planarity testing. In *Proc. 30th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 436–441, 1989.

[DT96] G. Di Battista and R. Tamassia. On-line planarity testing. *SIAM J. Comput.*, 25:956–997, 1996.

[DV02] H. Djidjev and I. Vrt'o. An improved lower bound for crossing numbers. In *GD '01: Revised Papers from the 9th International Symposium on Graph Drawing*, pages 96–101, London, UK, 2002. Springer-Verlag.

[DV06] H. Djidjev and I. Vrt'o. Planar crossing numbers of genus g graphs. In *Proc. ICALP '06*, volume 4051 of *LNCS*, pages 419–430. Springer, 2006.

[DW60] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[EGS00] G. Even, S. Guha, and B. Schieber. Improved approximations of crossings in graph drawing. In *Proc. STOC '00*, pages 296–305, 2000.

[EKE03] M. Eiglsperger, M. Kaufmann, and F. Eppinger. An approach for mixed upward planarization. *J. Graph Algorithms Appl.*, 7(2):203–220, 2003.

[EW94] P. Eades and N. C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994.

[GHLS08] I. Gitler, P. Hliněný, J. Leanos, and G. Salazar. The crossing number of a projective graph is quadratic in the face-width. *Electr. Journal of Combinatorics*, 15, 2008.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.

[GJ83] M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM J. Algebraic Discrete Methods*, 4(3):312–316, 1983.

[GJJ68] R. K. Guy, T. A. Jenkyns, and J.Schaer. The toroidal crossing number of the complete graph. *Journal of Combinatorial Theory*, 4:376–390, 1968.

[GLS88] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 1988.

[GM01] C. Gutwenger and P. Mutzel. A linear time implementation of SPQR trees. In J. Marks, editor, *Graph Drawing (Proc. GD 2000)*, volume 1984 of *LNCS*, pages 77–90. Springer-Verlag, 2001.

[GM04] C. Gutwenger and P. Mutzel. An experimental study of crossing minimization heuristics. In G. Liotta, editor, *11th Symposium on Graph Drawing 2003*, volume 2912 of *LNCS*, pages 13–24. Springer-Verlag, 2004.

[GMW05] C. Gutwenger, P. Mutzel, and R. Weiskircher. Inserting an edge into a planar graph. *Algorithmica*, 41(4):289–308, 2005.

[Gro01] M. Grohe. Computing crossing numbers in quadratic time. In *Proceedings of STOC'01*, 2001.

[Gut10] C. Gutwenger. *Application of SPQR-Trees in the Planarization Approach for Drawing Graphs*. PhD thesis, TU Dortmund, 2010. `http://hdl.handle.net/2003/27430`.

[Guy72] R. K. Guy. Crossing numbers of graphs. In *Graph Theory and Applications (Proceedings*, Lecture Notes in Mathematics, pages 111–124. Springer-Verlag, 1972.

[HC10] P. Hliněný and M. Chimani. Approximating the crossing number of graphs embeddable in any orientable surface. In *Proc. SODA'10*, pages 918–927. SIAM, 2010. Proc. SODA '10.

[Hli06] P. Hliněný. Crossing number is hard for cubic graphs. *Journal of Combinatorial Theory, Series B*, 96:455–471, 2006.

[HS06] P. Hliněný and G. Salazar. On the crossing number of almost planar graphs. In *Proc. GD '05*, volume 4372 of *LNCS*, pages 162–173. Springer, 2006.

[HS07] P. Hliněný and G. Salazar. Approximating the crossing number of toroidal graphs. In *Proc. ISAAC '07*, volume 4835 of *LNCS*, pages 148–159. Springer, 2007.

[HT73] J. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3):135–158, 1973.

[HT74] J. Hopcroft and R. E. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.

[JLM98] M. Jünger, S. Leipert, and P. Mutzel. A note on computing a maximal planar subgraph using PQ-trees. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(7):609–612, 1998.

[JM96] M. Jünger and P. Mutzel. Maximum planar subgraphs and nice embeddings: Practical layout tools. *Algorithmica*, 16(1):33–59, 1996. (special issue on Graph Drawing, edited by G. Di Battista and R. Tamassia).

[JM97] Michael Jünger and Petra Mutzel. 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. *J. Graph Algorithms Appl.*, 1(1):1–25, 1997.

[JTS89] R. Jayakumar, K. Thulasiraman, and M. N. S. Swamy. $O(n^2)$ algorithms for graph planarization. *IEEE Trans. Comp.-Aided Design*, 8:257–267, 1989.

[KMP+06] E. de Klerk, J. Maharry, D. V. Pasechnik, R. B. Richter, and G. Salazar. Improved bounds for the crossing numbers of $K_{m,n}$ and K_n. *SIAM Journal on Discrete Mathematics*, 20(1):189–202, 2006.

[KPS07] E. de Klerk, D. V. Pasechnik, and A. Schrijver. Reduction of symmetric semidefinite programs using the regular $*$-representation. *Mathematical Programming*, 109(2):613–624, 2007.

[KR07] K. Kawarabayashi and B. Reed. Computing crossing number in linear time. In *Proc. STOC '07*, pages 382–380, 2007.

[Kra91] J. Kratochvíl. String graphs. II: Recognizing string graphs is NP-hard. *J. Comb. Theory Ser. B*, 52(1):67–78, 1991.

[La 94] J. A. La Poutré. Alpha-algorithms for incremental planarity testing. In *Proc. 26th Annu. ACM Sympos. Theory Comput.*, pages 706–715, 1994.

[Lei83] F. T. Leighton. *Complexity issues in VLSI: optimal layouts for the shuffle-exchange graph and other networks.* MIT Press, 1983.

[Lei84] F. T. Leighton. New lower bound techniques for VLSI. *Mathematical Systems Theory*, 17:47–70, 1984.

[LG77] P. Liu and R. C. Geldmacher. On the deletion of nonplanar edges of a graph. In *Proc. 10th S.-E. Conf. on Combinatorics, Graph Theory and Computing*, pages 727–738, Boca Raton, FL, 1977.

[Man83] A. Mansfield. Determining the thickness of a graph is np-hard. In *Mathematical Proceedings of the Cambridge Philosophical Society*, pages 9–23, 1983.

[MJ01] Petra Mutzel and Michael Jünger. Graph drawing: Exact optimization helps! In M. Grötschel, editor, *The Sharpest Cut*, Series on Optimization. MPS - SIAM, 2001. Festschrift zum 60. Geburtstag von Manfred Padberg.

[MKNF86] S. Masuda, T. Kashiwabara, K. Nakajima, and T. Fujisawa. An NP-hard crossing minimization problem for computer network layout. Technical Report SRC TR 86-80, Electrical Engineering Department and Systems Research Center, University of Maryland, 1986.

[MKNF87] S. Masuda, T. Kashiwabara, K. Nakajima, and T. Fujisawa. On the NP-completeness of a computer network layout problem. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 292–295, 1987.

[MNKF90] S. Masuda, K. Nakajima, T. Kashiwabara, and T. Fujisawa. Crossing minimization in linear embeddings of graphs. *IEEE Trans. Comput.*, 39(1):124–127, 1990.

[Moh99] B. Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM J. Discrete Math.*, 12:6–26, 1999.

[MZ99] P. Mutzel and T. Ziegler. The constrained crossing minimization problem. In J. Kratochvil, editor, *Graph Drawing (Proc. GD '99)*, volume 1731 of *Lecture Notes in Computer Science*, pages 175–185. Springer-Verlag, 1999.

[Nic68] T. A. J. Nicholson. Permutation procedure for minimising the number of crossings in a network. *IEE Proceedings*, 115:21–26, 1968.

[PR07] S. Pan and R. B. Richter. The crossing number of K_{11} is 100. *Journal of Graph Theory*, 56:128–134, 2007.

[PSS96] J. Pach, F. Shahrokhi, and M. Szegedy. Applications of the crossing number. *Algorithmica*, 16:111–117, 1996.

[PSŠ06] M. Pelsmajer, M. Schaefer, and D. Štefankovič. Odd crossing number is not crossing number. In *Proc. GD '05*, volume 3843 of *LNCS*, pages 386–396. Springer, 2006.

[PT97] J. Pach and G. Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17(3):427–439, 1997.

[PT00] J. Pach and G. Tóth. Which crossing number is it, anyway? *J. Comb. Theory Ser. B*, 80(2):225–246, 2000.

[Pul89] W. R. Pulleyblank. Polyhedral combinatorics. In G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, editors, *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science*, pages 371–446. North-Holland, 1989.

[Pur97] Helen Purchase. Which aesthetic has the greatest effect on human understanding? In G. Di Battista, editor, *Graph Drawing (Proc. GD '97)*, volume 1353 of *Lecture Notes Comput. Sci.*, pages 248–261. Springer-Verlag, 1997.

[RT97] C. Roos and T. Terlaky. Advances in linear optimization, 1997.

[Sch12] M. Schaefer, 2012. personal communication, joint work with D. Štefankovič. Also mentioned in the unpublished manuscript *The Graph Crossing Number and Its Variants: A Survey*.

[SV94] O. Sýkora and I. Vrt'o. On VLSI layout of the star graph and related networks. *The VLSI Journal*, 17:83–93, 1994.

[Zie00] T. Ziegler. *Crossing Minimization in Automatic Graph Drawing.* PhD thesis, Max-Planck-Institut für Informatik, Saarbrücken, 2000.