Maximum Cut Parameterized by Crossing Number

Markus Chimani\textsuperscript{1}, Christine Dahn\textsuperscript{2}, Martina Juhnke-Kubitzke\textsuperscript{1}, Nils M. Kriege\textsuperscript{2}, Petra Mutzel\textsuperscript{2}, and Alexander Nover\textsuperscript{1}

\textsuperscript{1} School of Mathematics/Computer Science, University Osnabrück, Germany
\{markus.chimani|juhnke-kubitzke|anover\}@uni-osnabrueck.de
\textsuperscript{2} Department of Computer Science, TU Dortmund University, Germany
\{christine.dahn|nils.kriege|petra.mutzel\}@cs.tu-dortmund.de

Abstract. Given an edge-weighted graph $G$ on $n$ nodes, the NP-hard Max-Cut problem asks for a node bipartition such that the sum of edge weights joining the different partitions is maximized. We propose a fixed-parameter tractable algorithm parameterized by the number $k$ of crossings in a given drawing of $G$. Our algorithm achieves a running time of $O(2^k \cdot p(n + k))$, where $p$ is the polynomial running time for planar Max-Cut. The only previously known similar algorithm [8] is restricted to 1-planar graphs (i.e., at most one crossing per edge) and its dependency on $k$ is of order $3^k$. A direct consequence of our result is that Max-Cut is fixed-parameter tractable w.r.t. the crossing number, even without a given drawing.

Keywords: maximum cut, fixed-parameter tractable, crossing number

1 Introduction

Cut problems in graphs are a well-established class of problems attracting interest since the beginning of modern algorithmic research. Given an edge-weighted undirected graph, the Max-Cut problem asks for a node partition into two sets, such that the sum of the weights of the edges between the partition sets is maximized. The problem is getting increasing attention in the literature due to its applicability to various scenarios: these range from $\ell^1$-embeddibility [11], to the layout of electronic circuits [4, 9], to solving Ising spin glass models, which are of high interest in physics [1]. Besides the theoretical merits, such models need to be solved in adiabatic quantum computation [29]. Furthermore, quadratic unconstrained binary optimization (QUBO) problems can be solved via Max-Cut. Many combinatorial optimization problems can be stated in the form of QUBO such as multicommodity-flow problems, maximum clique, vertex cover, scheduling, and many others. Also see [10, 11] for a more in-depth overview on applications.

The Max-Cut problem has been shown to be NP-hard for general graphs [23]. Papadimitriou and Yannakakis [33] have shown that the Max-Cut problem is even APX-hard, i.e., there does not exist a polynomial-time approximation
scheme unless \( P=NP \). Goemans and Williamson proposed a randomized constant-
factor approximation algorithm [16], which has been derandomized by Mahajan
and Ramesh [28], achieving a ratio of 0.87856. Several special cases of the prob-
lem allow polynomial algorithms: If the weights of all edges are negative we
obtain a Min-Cut problem, which can be solved, e.g., via network flow. Other
special cases are, e.g., graphs without long odd cycles [19] or weakly bipartite
graphs [18]. The case of planar input graphs is of particular interest. Orlova and
Dorfman [31] and Hadlock [20] have shown how to solve Max-Cut in polynomial
time for unweighted planar graphs. Those algorithms can be extended to
weighted planar graphs; the currently fastest algorithms for the weighted case
have been suggested by Shih et al. [34] and by Liers and Pardella [27], and
achieve a running time of \( p(n) = O(n^{3/2} \log n) \) on planar graphs with \( n \) nodes.
Barahona has shown that the planarity condition can be relaxed to graphs not
contractible to \( K_5 \) [3] and to toroidal graphs (i.e., graphs that can be embed-
ded on a genus-1 surface) with edge weights \( \pm 1 \) [2]. Along these lines, it was
also shown that Max-Cut can be solved in polynomial time if the graph can
be embedded on a surface of constant genus and at the same time has integral
dge-weights whose absolute values are bounded by a constant; however, this
approach does not work for general graph sizes, as it depends on the existance
of sufficiently many prime numbers within a given interval, which cannot be
guaranteed in general [14].

A graph is 1-planar if it allows a drawing where each edge is involved in at
most one crossing. A 1-plane graph is such a graph, together with an embedding
realizing this property. The Max-Cut problem on 1-plane graphs with \( k \) edge
crossings has recently been shown to be fixed-parameter tractable (FPT) with
parameter \( k \) [7, 8]. More precisely, it was shown that such instances can be solved
in \( O(3^k \cdot p(n)) \) time. There are no restrictions on the edge weights.

Our contribution. In this paper, we improve on the latter result in several ways:
Firstly, we drop the requirement of 1-planarity, i.e., we consider graphs that can
be drawn with at most \( k \) crossings (even if multiple such crossings reside on the
same edge). We therefore handle the true case of the well-established notion of
the graph’s crossing number. Secondly, we reduce the runtime dependency on \( k \)
from \( 3^k \) to \( 2^k \). Finally, unlike the previous result, our approach can be extended
to an FPT algorithm that does not even require a crossing-realizing drawing as
an input; however, this increases the running time and requires a deep algorithm
from literature as a black box. Interestingly, we achieve these results not by a
more complicated, but an arguably simpler approach.

The paper is organized as follows. Section 2 recapitulates the basic definitions
for cuts and crossings. In Section 3, we present our new algorithm and prove its
correctness and running time. We end with a conclusion and open problems in
Section 4.
2 Preliminaries

Throughout this paper we consider undirected edge-weighted graphs. The input for our MAX-CUT problem is a graph $G = (V, E, c)$ where $c$ are arbitrary (negative and positive) edge weights. A partition of the nodes $V$ into two sets $S \subseteq V$ and $\overline{S} = V \setminus S$ defines the cut $\delta(S) = \{uv \in E \mid u \in S \iff v \in \overline{S}\}$. The value $c(\delta(S)) = \sum_{e \in \delta(S)} c_e$ of a cut is the sum of all edge weights of the edges in the cut. Given $G$, the MAX-CUT problem asks for a cut with highest value. Since a graph can have multiple cuts of equal value, only the value of the maximum cut is unique.

A non-degenerated drawing of a graph in the plane is a map of its nodes to distinct points in $\mathbb{R}^2$, and a map of its edges to curves connecting the respective endpoints, but not including the points of any other nodes. Any point mapped to the plane either corresponds to a graph node, or is contained in at most two edge curves. A shared non-endpoint between two curves is called a crossing. A graph is planar if it admits a drawing without any crossings. It is well known that planarity can be tested in linear time [22]. For non-planar graphs it is natural to ask for a drawing with as few edge crossings as possible. The smallest such number is the crossing number $\text{cr}(G)$ of $G$. Not only is it NP-hard to compute $\text{cr}(G)$ [15], but even the so called realizability problem turns out to be NP-hard [26]: Given a graph $G$ and a set $X$ of edge pairs, is there a drawing $D$ of $G$ such that $X$ contains an edge pair if and only if the pair’s two edge curves cross in $D$? The key problem in testing realizability is that it is hard to figure out whether there exist orderings of the crossings along the respective edges that allow the above properties.

Therefore, sometimes more restricted crossing variants are considered. For example, 1-planar graphs admit drawings where every edge is involved in at most one crossing. Not all graphs can be drawn in such a way, since 1-planar graphs can have at most $4|V| - 8$ edges; also, the 1-planar number of crossings is in general larger than $\text{cr}(G)$ [32].

For a general drawing (not necessarily 1-plane), we typically encode its crossings as a crossing configuration $\mathcal{X}$. Therein, we not only store the pairs of edges that cross, but for each edge also the order of the crossings as they occur along its curve. Adapting the planarity testing algorithm, the feasibility of a crossing configuration can be tested in linear time (we replace crossings with dummy vertices of degree 4, and test planarity). Although we will not require this fact in the following, this also allows us to efficiently deduce a drawing that respects $\mathcal{X}$. It is well understood that we can restrict ourselves to good drawings when considering the (traditional) crossing number of graphs: adjacent edges never cross and no edge pair crosses twice.
3 Algorithm

Our main idea for computing the maximum cut in an embedded weighted graph is to eliminate its crossings one by one, in order to use a Max-Cut algorithm for planar graphs in the end. We first introduce a slight variant of Max-Cut, denoted by PF-Max-Cut:

**Definition 1 (Partially-Fixed Maximum Cut).** Given an edge weighted graph $G = (V,E,c)$ and a set of fixed edges $F \subseteq E$, find a cut of maximum value that contains all elements of $F$.

A cut is feasible if it contains $F$. A PF-Max-Cut instance is infeasible if it does not allow a feasible cut. It is easy to see that an instance is infeasible if and only if $F$ contains a cycle of odd length. We denote a maximum objective value by $\text{MaxCut}_{pf}(G,F)$, and let $\text{MaxCut}_{pf}(G,F) = -\infty$ for infeasible instances.

Observe that (as for Max-Cut) we do not need to consider a given crossing configuration as part of the problem description; however, we will always attach such a configuration to the considered graph.

Given any edge $vw$ with weight $c_{vw}$ in a PF-Max-Cut instance, we define the operation to bisubdivide $vw$ at $v$ as follows: Subdivide $vw$ twice, i.e., replace $vw$ by a path of length 3 with two new degree-2 nodes. We denote the new node incident to $v$ or $w$ by $\tilde{v}$ or $\tilde{w}$, respectively (note that we consider this notation an operand). The edges $\tilde{v}v$ and $\tilde{w}w$ have weight 0, $ww$ retains the weight $c_{vw}$. Furthermore, we add $\tilde{v}v, \tilde{w}w$ to $F$, and if $vw \in F$, we replace it in $F$ by $\tilde{w}w$. Clearly, both $\tilde{v}v, \tilde{w}w$ will be in any feasible cut; node $\tilde{w}$ will always lie in the same partition set as $v$, and $\tilde{v}$ in the other (cf. Figure 1b). Most importantly this gives:

**Observation 2.** The feasible cuts in the original PF-Max-Cut instance are in 1-to-1 correspondence to feasible cuts of equal value in the bisubdivided instance.

When we identify two nodes $a,b$ in a graph with one another, they become a common entity that is incident to all of their former neighbors. We will only identify nodes that are neither adjacent nor share neighbors. When identifying nodes in $G$ of some PF-Max-Cut instance $(G,F)$, the set $F$ is retained, subject to replacing the edges formerly incident to $a$ or $b$ with their new counterparts.

We are now ready to describe our algorithm. We are given a Max-Cut instance $G = (V,E,c)$, together with some crossing configuration $\mathcal{X}$ attaining $k$ crossings. Let $F = \emptyset$ and consider $(G,F)$ as a PF-Max-Cut instance. From $(G,F,\mathcal{X})$, we pick two special nodes $v,w$ and derive two new triplets $T_i = (G_i,F_i,\mathcal{X}_i)$, $i \in \{v,w\}$. Both derived crossing configurations $\mathcal{X}_i$ attain at most $k-1$ crossings and we can call our algorithm recursively on $T_v$ and $T_w$. As a base case, the derived graphs become planar and (after a preprocessing to deal with the fixed edges) we apply an efficient Max-Cut algorithm for planar graphs. The solutions of $(G_i,F_i)$, $i \in \{v,w\}$, yield a solution of $(G,F)$. Observe, however, that $(G_i,F_i)$ may become infeasible. Let us describe this recursion step formally (cf. also Figures 1 and 2):
Lemma 3. Let \((G = (V, E, c), F)\) be a PF-Max-Cut instance and \(X\) a crossing configuration of \(G\) with \(k\) crossings. Consider any crossing \(\chi \in X\) with edges \(vw\) and \(xy\). For \(j \in \{v, w, x, y\}\), let \(Y_j\) be the ordered sets of crossings in \(X\) between \(j\) and \(\chi\). Let \((G', F')\) be obtained from \((G, F)\) by bisubdividing \(vw\) at \(v\) and bisubdividing \(xy\) at \(x\). For \(i \in \{v, w\}\), let \((G_i, F_i)\) be the PF-Max-Cut instance obtained from \((G', F')\) by identifying \(x\) with \(i\); we obtain \(X_i\) from \(X\) by removing \(\chi\) and placing the crossings \(Y_j\) (retaining their order) on the edge \(jj\), for all \(j \in \{v, w, x, y\}\). Then we have:

1. for \(i \in \{v, w\}\), \(X_i\) is a feasible crossing configuration for \(G_i\) and has at most 
   \(k - 1\) crossings; and
2. \(\text{MaxCut}_{\text{pf}}(G, F) = \max_{i \in \{v, w\}} \{ \text{MaxCut}_{\text{pf}}(G_i, F_i) \} \).

Proof. Consider any drawing \(D\) of \(G\) realizing \(X\). We obtain a drawing \(D'\) of \(G'\) from \(D\) by routing the new paths along the curves of their original edges. Thereby, for \(j \in \{v, w, x, y\}\), we place the new nodes \(j\) in a close neighborhood of \(\chi\) on the curve segment between \(j\) and \(\chi\). The number of crossings in \(D'\) is equal to that of \(D\). The original crossing \(\chi\) has now a counterpart \(\chi'\) between the edges \(vw\) and \(xy\). Most importantly, we can now locally perturb the drawing such that \(\chi'\) vanishes when we identify \(x\) with \(v\), see Figure 2c. The same holds true when identifying \(x\) with \(w\), see Figure 2d. This establishes claim (1).

By Observation 2, the maximum cut in \((G', F')\) induces a maximum cut in \((G, F)\) of the same value. A maximum cut in \((G, F)\) (or \((G', F')\), for that matter) is attained by either placing \(v\) and \(x\) in the same partition set, or in opposing ones. In \(G_v\) (where we identify \(x\) with \(v\)), we have the path of two edges \(v \alpha\) and \(\alpha x\), both of which are in \(F_v\); thus, \(v\) and \(x\) have to be in the same partition set. Conversely, in \(G_w\) (where we identify \(x\) with \(w\)), we have the path of three edges \(v \alpha\), \(\alpha w\), and \(wx\), all of which are in \(F_w\); thus \(v\) and \(x\) end up in different partition sets. Finally, we can see that the respective constructions do not induce any
Fig. 2: An illustration of the two cases where \( v \) and \( x \) are either on the same side of the partition (a/c) or on different sides of the partition (b/d). In the two graphs \( G_v \) and \( G_w \), the crossing was removed while retaining the partition property. The node coloring gives a partition of the nodes that is induced by the newly added edges in \( F' \), resp. \( F_v \) or \( F_w \). (Dashed and dotted edges show examples of other edges in \( G' \), resp. \( G_v \) or \( G_w \).)

If we are in a base case – the considered graph is planar – we can use an efficient Max-Cut algorithm for planar graphs:

**Lemma 4.** Consider a PF-Max-Cut instance \( \langle G = (V, E, c), F \rangle \) with a planar graph \( G \). Let \( p(|V|) \) be a polynomial upper bound on the running time of a Max-Cut algorithm on the planar graph \( G \). We can compute an optimal solution to \( \langle G, F \rangle \) – or decide that the instance is infeasible – in \( O(p(|V|)) \).

**Proof.** We transform the (planar) PF-Max-Cut instance into a traditional (planar) Max-Cut instance by attaching a large weight to the edges in \( F \). Namely, we add \( M \) to the weight of each edge \( f \in F \), where \( M = 2\sum_{e \in E} |c_e| \). The omission of a single edge of \( F \) from the solution cut (even if picking all other edges of positive weight) will already result in a worse objective value than picking all further restrictions on the set of cuts; in particular, both derived instances still allow any partition choice between \( w \) and \( x \), between \( w \) and \( y \), and between \( x \) and \( y \). Overall, every feasible cut in \( \langle G', F' \rangle \) can be realized either in \( \langle G_v, F_v \rangle \) or in \( \langle G_w, F_w \rangle \). Claim (2) follows. \( \square \)
Maximum Cut Parameterized by Crossing Number

of $F$ and all edges of negative weight. The instance is infeasible if and only if the computed cut does not contain all of $F$; this can also be deduced purely by checking whether the objective value is at least $M|F| + \sum_{e \in E : c_e < 0} c_e$.

We prefer our lemma in the above form, but in fact only require a slightly weaker version: in our algorithm, $c_e = 0$ for all $e \in F$. Thus it suffices to set $c_e = M$ instead of adding $M$. Using any of the currently fastest MAX-CUT algorithms for planar graphs [27,34] leads to $O(|V|^3/2 \log |V|)$ time in the above lemma. We could speed-up infeasibility detection by checking whether $F$ contains a cycle of odd length prior to the transformation; while this only requires $O(|V|)$ time via depth-first search, the overall asymptotic runtime for the lemma’s claim does of course not improve.

**Theorem 5.** Let $G = (V,E,c)$ be an edge-weighted graph and $\mathcal{X}$ a crossing configuration of $G$ with $k$ crossings. Let $p(n)$ be a polynomial upper bound on the running time of a MAX-CUT algorithm on planar graphs with $n$ nodes. We can compute a maximum cut in $G$ in $O(2^k \cdot p(|V| + k))$ time.

**Proof.** As described above, we solve the instance by considering the PF-MAX-CUT instance $\langle G,F = \emptyset \rangle$ together with $\mathcal{X}$. Thus the triplet $\langle G,F,\mathcal{X} \rangle$ forms the initial input of our recursive algorithm $\mathcal{R}$.

Algorithm $\mathcal{R}$ proceeds as follows on a given triplet: If the triplet’s graph is planar, we solve $\langle G,F \rangle$ via Lemma 4. Otherwise, we use Lemma 3 to obtain two new input triples $T_v, T_w$, for each of which we call $\mathcal{R}$ recursively. Their returned solutions (i.e., their solution values) induce the optimum solution for the current input triplet. However, while the number of crossings decreases by (at least) one per recursion step, the graph’s size increases by three nodes.

The runtime complexity follows from the fact that we consider two choices per crossing in given $\mathcal{X}$, and thus construct $2^k$ graphs. For each such graph, which has $|V| + 3k$ nodes, we run the planar MAX-CUT algorithm. Therefore, we trivially have $k \in O(|V|^4)$ and thus $|V| + k \in O(poly(|V|))$.

**Corollary 6.** The above algorithm is an FPT algorithm with parameter $k = cr(G)$, provided $\mathcal{X}$ is part of the input. Moreover, the attained running time is polynomial for any $k \in O(\log |V|)$. Using the currently fastest MAX-CUT algorithm for planar graphs, our algorithm yields a running time of $O(2^k \cdot (|V| + k)^{3/2} \log(|V| + k))$.

Quite sophisticated results by Grohe [17] and Karabayashi and Reed [25] show that the crossing number is FPT (even in linear time) w.r.t. its natural parameterization: Given a graph $G$ and considering a number $k \in \mathbb{N}$, we can answer the question “$cr(G) \leq k$?” in time $O(f(k) \cdot n)$. In case of a yes-instance, we obtain a corresponding crossing configuration $\mathcal{X}$ as a witness. Thereby, the computable function $f(k)$ is purely dependent on $k$. However, this dependency is doubly exponential, and the algorithm far from being practical. Still, these results formally allow us to get rid of the requirement that $\mathcal{X}$ is part of the input:

**Corollary 7.** Given an edge-weighted undirected graph $G$. Computing a maximum cut in $G$ is FPT with parameter $cr(G)$.
4 Conclusion and Open Problems

Given a graph together with a feasible crossing configuration with $k$ crossings, we previously only knew that MAX-CUT is polynomial time solvable if $k$ is constant and the graph is 1-planar, i.e., each edge is involved in at most one crossing. The runtime dependency on $k$ has been to the order of $3^k$ [8]. Herein, we improved on this in several ways: Firstly, we decreased the dependency on $k$ to the order of $2^k$. Secondly, we extended the applicability to any graph with (at most) $k$ crossings. That is, our parameter becomes the true crossing number of the graph, without any 1-planarity restriction. This shows that MAX-CUT is in FPT w.r.t. the graph’s crossing number. Moreover, we achieve these improvements by introducing simpler ideas than those proposed for the former result, yielding an overall surprisingly simple algorithm.

The skewness of a graph is the minimum number of edges to remove such that the graph becomes planar. The genus of a graph is the minimum oriented genus of a surface onto which the graph can be embedded without crossings. In FPT research, there are many algorithmic approaches that consider graphs with bounded genus $g$, see, e.g. [5,6,12,13]. However, the obtained FPT algorithms are typically parameterized by the objective value $z$, or by the combined parameter $(z,g)$. There are much fewer results that obtain FPT algorithms parameterized purely with $g$. Notable examples are the graph genus problem itself [30] (where $z$ and $g$ coincide by definition), and the graph isomorphism problem [24] (which generalizes the linear-time algorithm for the problem on planar graphs). There are even fewer parameterized results w.r.t. skewness; the probably best known example is that maximum flow can be solved in the running time of planar graphs, if the graph’s skewness is fixed [21]. Our above algorithm seems to be the first time that the crossing number has been proposed as an efficient non-trivial FPT parameter for any widely known problem.

Besides the special case of [14] (briefly described in the introduction), it is unclear whether MAX-CUT could be FPT w.r.t. either skewness or genus. We deem this an interesting question for further research.

References

1. Francisco Barahona. On the computational complexity of ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241, 1982.
2. Francisco Barahona. Balancing signed toroidal graphs in polynomial time. *Departamento de Matematicas, Universidad de Chile, Santiago, Chile*, 1983.
3. Francisco Barahona. The Max-Cut problem on graphs not contractible to $K_5$. *Operations Research Letters*, 2(3):107–111, 1983.
4. Francisco Barahona, Martin Grötschel, Michael Jünger, and Gerhard Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36(3):493–513, 1988.
5. Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) Kernelization. *Journal of the ACM (JACM)*, 63(5):44:1–44:69, 2016.
6. Jianer Chen, Iyad A. Kanj, Ljubomir Perkovic, Eric Sedgwick, and Ge Xia. Genus characterizes the complexity of certain graph problems: Some tight results. *Journal of Computer and System Sciences*, 73(6):892–907, 2007.

7. Christine Dahn, Nils M. Kriege, and Petra Mutzel. A fixed-parameter algorithm for the Max-Cut problem on embedded 1-planar graphs. In Costas S. Iliopoulos, Hon Wai Leong, and Wing-Kin Sung, editors, *Combinatorial Algorithms - 29th International Workshop, IWOCA 2018, Singapore, July 16-19, 2018, Proceedings*, volume 10979 of *Lecture Notes in Computer Science*, pages 141–152. Springer, 2018.

8. Christine Dahn, Nils M. Kriege, Petra Mutzel, and Julian Schilling. Fixed-parameter algorithms for the weighted Max-Cut problem on embedded 1-planar graphs. *CoRR*, abs/1812.03074, 2018.

9. Caterina De Simone, Martin Diehl, Michael Jünger, Petra Mutzel, Gerhard Reinelt, and Giovanni Rinaldi. Exact ground states of ising spin glasses: New experimental results with a branch-and-cut algorithm. *Journal of Statistical Physics*, 80(1-2):487–496, 1995.

10. Michel Deza and Monique Laurent. Applications of cut polyhedra I. *J. Comput. Appl. Math.*, 55(2):191–216, 1994.

11. Michel Deza and Monique Laurent. Applications of cut polyhedra II. *J. Comput. Appl. Math.*, 55(2):217–247, 1994.

12. John A. Ellis, Hongbing Fan, and Michael R. Fellows. The dominating set problem is fixed parameter tractable for graphs of bounded genus. *Journal of Algorithms*, 52(2):152–168, 2004.

13. Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Bidimensionality and EPTAS. In Dana Randall, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 2011, San Francisco, California, USA, January 23-25, 2011, pages 748–759. SIAM, 2011.

14. Anna Galluccio, Martin Loebl, and Jan Vondrak. Optimization via enumeration: a new algorithm for the max cut problem. *Mathematical Programming*, 90:273–290, 2001.

15. Michael R. Garey and David S. Johnson. Crossing number is np-complete. *SIAM Journal on Algebraic Discrete Methods*, 4(3):312–316, 1983.

16. Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

17. Martin Grötschel. Computing crossing numbers in quadratic time. *Journal of Computer and System Sciences*, 68(2):285–302, 2004.

18. Martin Grötschel and William R. Pulleyblank. Weakly bipartite graphs and the Max-Cut problem. *Operations Research Letters*, 1(1):23–27, 1981.

19. Martin Grötschel and George L. Nemhauser. A polynomial algorithm for the Max-Cut problem on graphs without long odd cycles. *Mathematical Programming*, 29(1):28–40, 1984.

20. F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM Journal on Computing*, 4(3):221–225, 1975.

21. Jan M. Hochstein and Karsten Weihe. Maximum s-t-flow with k crossings in $O(k^3 n \log n)$ time. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007, pages 843–847. SIAM, 2007.

22. John E. Hopcroft and Robert E. Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158, 1973.
23. Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations*, New York, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

24. Ken-ichi Kawarabayashi. Graph isomorphism for bounded genus graphs in linear time. *arXiv.org*, abs/1511.02460, 2015.

25. Ken-ichi Kawarabayashi and Bruce A. Reed. Computing crossing number in linear time. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 382–390. ACM, 2007.

26. Jan Kratochvíl. String graphs. ii. recognizing string graphs is np-hard. *Journal of Combinatorial Theory, Series B*, 52(1):67 – 78, 1991.

27. Frauke Liers and Gregor Pardella. Partitioning planar graphs: a fast combinatorial approach for Max-Cut. *Computational Optimization and Applications*, 51(1):323–344, 2012.

28. Sanjeev Mahajan and Hariharan Ramesh. Derandomizing semidefinite programming based approximation algorithms. *SIAM Journal on Computing*, 28(5):1641–1663, 1995.

29. Catherine C. McGeoch. *Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice*. Synthesis Lectures on Quantum Computing. Morgan & Claypool Publishers, 2014.

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

31. G. I. Orlova and Y. G. Dorfman. Finding maximum cut in a graph. *Engineering Cybernetics*, 10(3):502–506, 1972.

32. János Pach and Géza Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17:427–439.

33. Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425 – 440, 1991.

34. Wei-Kuan Shih, Sun Wu, and Yue-Sun Kuo. Unifying maximum cut and minimum cut of a planar graph. *IEEE Transactions on Computers*, 39(5):694–697, 1990.