A Polynomial Time Approximation Scheme for MAX-BISECTION on Planar Graphs

Klaus Jansen* Marek Karpinski† Andrzej Lingas‡

Abstract

The Max-Bisection and Min-Bisection are the problems of finding partitions of the vertices of a given graph into two equal size subsets so as to maximize or minimize, respectively, the number of edges with exactly one endpoint in each subset.

In this paper we design the first polynomial time approximation scheme for the Max-Bisection problem on arbitrary planar graphs. The method of solution involves designing exact polynomial time algorithms for computing optimal partitions of bounded treewidth graphs, in particular their Max- and Min-Bisection, which could be also of independent interest.

*Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität zu Kiel, 24098 Kiel. Email: kj@informatik.uni-kiel.de.

†Department of Computer Science, University of Bonn, 53117 Bonn. Email: marek@cs.uni-bonn.de.

‡Department of Computer Science, Lund University, 22100 Lund. Email: Andrzej.Lingas@cs.lth.se.

1 Introduction

The max-bisection and min-bisection problems, i.e., the problems of constructing a halving of the vertex set of a graph that respectively maximizes or minimizes the number of edges across the partition, belong to the basic combinatorial optimization problems.

The best known approximation algorithm for max-bisection yields a solution whose size is at least 0.701 times the optimum [15] whereas the best known approximation algorithm for min-bisection achieves "solely" a log-square approximation factor [11]. The former factor for max-bisection is considerably improved for regular graphs to 0.795 in [10] whereas the latter factor for min-bisection is improved for graphs excluding any fixed minor (e.g., planar graphs) to a logarithmic one in [11]. For dense graphs, Arora, Karger and Karpinski give polynomial time approximation schemes for max- and min-bisection in [2].

In this paper, we study the max-bisection and min-bisection problems on bounded treewidth graphs and on planar graphs. Both graph families are known to admit exact polynomial-time algorithms for max-cut, i.e., for finding a bi-partition that maximizes the number of edges with endpoints in both sets in the partition [9, 14].

Our first main result are exact polynomial-time algorithms for finding a partition of a bounded treewidth graph into two sets of a priori given cardinalities, respectively maximizing or minimizing the number of edges with endpoints in both sets. Thus, in particular, we obtain polynomial-time algorithms for max-bisection and min-bisection on bounded treewidth graphs.

The complexity and approximability status of max-bisection on planar graphs have been long-standing open problems. Karpinski et al. observed in [17] that the max-bisection problem for planar does not fall directly into the Khanna-Motwani's syntactic framework for planar optimization problems [18]. On the other hand, they provided a polynomial-time approximation scheme (PTAS) for max-bisection in planar graphs of sublinear maximum degree. (In fact, their method implies that the size of max-bisection is very close to that of max-cut in planar graphs of sublinear maximum degree.)

Our second main result is the first polynomial-time approximation scheme for the max-bisection problem for arbitrary planar graphs. It is obtained by combining (via tree-typed dynamic programming) the original Baker's method of dividing the input planar graph into families of k-outerplanar graphs [4] with our method of finding maximum partitions of bounded treewidth graphs.

It is interesting to note that our PTAS result for Max-Bisection on planar graphs is the best possible under usual assumptions. Very recently, Jerrum [16] established NP-hardness of exact Max-Bisection on planar graphs, in contrast to the status of Max-Cut problem on planar graphs ([14]). The technique used in his proof was similar to the method used by Barahona [5] to prove hardness of the planar spin glass problem within a magnetic field (P5).

2 Preliminaries

We start with formulating the underlying optimal graph partition problems.

Definition 2.1 *A partition of a set of vertices of an undirected graph G into two sets X, Y is called an* $(|X|, |Y|)$-partition *of G. The edges of G with one endpoint in X and the other in Y are said to be cut by the partition. The* size *of an (l,k)-partition is the number of edges which are cut by it. An (l,k)-partition of G is said to be a* maximum (l,k)-partition *of G if it has the largest size among all (l,k)-partitions of G. An (l,k)-partition of G is a* bisection *if $l = k$. A bisection of G is a* max bisection *or a* min bisection *of G if it respectively maximizes or minimizes the number of cut edges. An (l,k)-partition of G is a* max cut *of G if it has the largest size among all (l', k')-partitions of G. The* max-cut problem *is to find a max cut of a graph. Analogously, the* max-bisection problem *is to find a max bisection of a graph. The* min-cut problem *and the* min-bisection problem *are defined similarly.*

The notion of treewidth of a graph was originally introduced by Robertson and Seymour [19]. It has turned out to be equivalent to several other interesting graph theoretic notions, e.g., the notion of partial k-trees [1, 6].

Definition 2.2 *A* tree-decomposition *of a graph $G = (V, E)$ is a pair $(\{X_i \mid i \in I\}, T = (I, F))$, where $\{X_i \mid i \in I\}$ is a collection of subsets of V, and $T = (I, F)$ is a tree, such that the following conditions hold:*

1. *$\bigcup_{i \in I} X_i = V$.*
2. *For all edges $(v, w) \in E$, there exists a node $i \in I$, with $v, w \in X_i$.*
3. *For every vertex $v \in V$, the subgraph of T, induced by the nodes $\{i \in I \mid v \in X_i\}$ is connected.*

The treewidth of a tree-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ is $\max_{i \in I} |X_i| - 1$. The treewidth *of a graph is the minimum treewidth over all possible tree-decompositions of the graph. A graph which has a tree-decomposition of treewidth $O(1)$ is called a* bounded treewidth graph.

Fact 1[7]: *For a bounded treewidth graph, a tree decomposition of treewidth $O(1)$ can be found in linear time.*

To state our results on max-bisection for planar graphs we need the following definition.

Definition 2.3 *A real number α is said to be an approximation ratio for a maximization problem, or equivalently the problem is said to be* approximable within a ratio α, *if there* *is a polynomial time algorithm for the problem which always produces a solution of size at least α times the optimum. If a problem is approximable for arbitrary $\alpha < 1$ then it is said to admit a* polynomial time approximation scheme *(a PTAS for short). Similarly we define approximation ratios and PTASs for minimization problems.*

2.1 Optimal partitions for graphs of bounded treewidth

Let G be a graph admitting a tree-decomposition $T = (I, F)$ of treewidth at most k, for some constant k. By [9], one can easily modify T, without increasing its treewidth, such that one can see T as a rooted tree, with root $r \in I$, fullfiling the following conditions:

1. T is a binary tree.
2. If a node $i \in I$ has two children j_1 and j_2, then $X_i = X_{j_1} = X_{j_2}$.
3. If a node $i \in I$ has one child j, then either $X_j \subset X_i$ and $|X_i - X_j| = 1$, or $X_i \subset X_j$ and $|X_j - X_i| = 1$.

We will assume in the remainder that such a modified tree-decomposition T of G is given.

For each node $i \in I$, let Y_i denote the set of all vertices in a set X_j with $j = i$ or j is a descendant of i in the rooted tree T. Our algorithm is based upon computing for each node $i \in I$ a table $maxc_i$. For each subset S of X_i, there is an entry in the table $maxc_i$, fulfilling

$$maxc_i(S) = \max_{S' \subseteq Y_i,\ S' \cap X_i = S} |\{(v, w) \in E \mid v \in S',\ w \in Y_i - S'\}|.$$

In other words, for $S \subseteq X_i$, $maxc_i(S)$ denotes the maximum number of cut edges for a partition of Y_i, such that all vertices in S are in one set in the partition, and all vertices in $X_i \setminus S$ are in the other set in the partition.

Our algorithm computes for each $i \in I$, an array $maxp_i$ with $O(2^k |Y_i|)$ entries. For each $l \in \{0, 1, ..., |Y_i|\}$ and each subset S of X_i, the entry $maxp_i(l, S)$ is set to $\max_{S' \subseteq Y_i, |S'| = l, S' \cap X_i = S} |\{(v, w) \in E | v \in S' \ \&\ w \in Y_i \setminus S'\}|$. In other words, $maxp_i(l, S)$ is set to the maximum number of cut edges in an $(l, |Y_i| - l)$-partition of Y_i where S and $X_i \setminus S$ are in the different sets of the partition and the set including S is of cardinality l. For convention, if such a partition is impossible, $maxp_i(l, S)$ will be set to $-\infty$.

The entries of the array are computed following the levels of the tree-decomposition T in a bottom-up manner. The following lemma shows how the array can be determined efficiently.

Lemma 2.1

- *Let i be a leaf in T. Then for all $l \in \{0, 1, ..., |X_i|\}$ and $S \subseteq X_i$ where $|S| = l$, $maxp_i(l, S) = |\{(v, w) \in E | v \in S, w \in X_i \setminus S\}|$. The remaining entries of $maxp_i$ are set to $-\infty$.*
- *Let i be a node with one child j in T. If $X_i \subseteq X_j$ then for all $l \in \{0, 1, ..., |Y_i|\}$ and $S \subseteq X_i$, $maxp_i(l, S) = \max_{S' \subseteq X_j, S' \cap X_i = S} maxp_j(l, S')$.*
- *Let i be a node with one child j in T. If $X_j \cup \{v\} = X_i$ where $v \notin X_j$ then for all $l \in \{0, 1, ..., |Y_i|\}$ and $S \subseteq X_i$, if $v \in S$ then $maxp_i(l, S) = maxp_j(l - 1, S \setminus \{v\}) + |\{(v, s) | s \in X_i \setminus S\}|$ else $maxp_i(l, S) = maxp_j(l, S) + |\{(v, s) | s \in S\}|$.*
- *Let i be a node with two children j_1, j_2 in T, with $X_i = X_{j_1} = X_{j_2}$. For all $l \in \{0, 1, ..., |Y_i|\}$ and $S \subseteq X_i$, $maxp_i(l, S) = \max_{l_1 + l_2 - |S| = l \& l_1 \geq |S| \& l_2 \geq |S|} maxp_{j_1}(l_1, S) + maxp_{j_2}(l_2, S) - |\{(v, w) \in E | v \in S, w \in X_i \setminus S\}|$.*

It follows that computing an array $maxp_i$ on the basis of the arrays computed for the preceding level of T can be done in time $O(2^k |Y_i|^2)$. Consequently, one can compute the array $maxp_r$ for the root r of T in cubic time.

Theorem 2.1 *All maximum $(l, n - l)$-partitions of a graph on n nodes given with a tree-decomposition of treewidth k can be computed in time $O(2^k n^3)$.*

By substituting *min* for *max*, we can analogously compute all minimum $(l, n - l)$-partitions of a graph with constant treewidth.

Theorem 2.2 *All minimum $(l, n - l)$-partitions of a graph on n nodes given with a tree-decomposition of treewidth k can be computed in time $O(2^k n^3)$.*

By Fact 1 we obtain the following corollary.

Corollary 2.1 *All maximum and minimum $(l, n - l)$-partitions of a bounded treewidth graph on n vertices can be computed in time $O(n^3)$.*

Since a tree-decomposition of a planar graph on n vertices with treewidth $O(\sqrt{n})$ can be found in polynomial time by the planar separator theorem [8], we obtain also the following corollary.

Corollary 2.2 *All maximum and minimum $(l, n - l)$-partitions of a planar graph on n vertices can be computed in time $2^{O(\sqrt{n})}$.*

3 A PTAS for max-bisection of an arbitrary planar graph

The authors of [17] observed that the requirements of the equal size of the vertex subsets in a two partition yielding a max bisection makes the max-bisection problem hardly expressible as a maximum planar satisfiability formula. For this reason we cannot directly apply Khanna-Motwani's [18] syntactic framework yielding PTASs for several basic graph problems on planar graphs (e.g., max cut). Instead, we combine the original Baker's method [4] with our algorithm for optimal maximum partitions on graphs of bounded treewidth via tree-type dynamic programming in order to derive the first PTAS for max-bisection of an arbitrary planar graph.

Algorithm 1

input: a planar graph $G = (V, E)$ on n vertices and a positive integer k;

output: $(1 - \frac{1}{k+1})$-approximations of all maximum $(l, n-l)$-partitions of G

1. Construct a plane embedding of G;
2. Set the level of a vertex in the embedding as follows: the vertices on the outer boundary have level 1, the vertices on the outer boundary of the subgraph obtained by deleting the vertices of level $i-1$ have level i;
3. For each level j in the embedding construct the subgraph H_j of G induced by the vertices on levels $j, j+1, ..., j+k$;
4. For each level j in the embedding set n'_j to the number of vertices in H_j and compute all maximum $(l, n'_j - l)$-partitions of H_j;
5. For each i, $0 \le i \le k$, set G_i to the union of the subgraphs H_j where $j \bmod k+1 = i$;
6. For each i, $0 \le i \le k$, set n_i to the number of vertices in G_i and compute all maximum $(l, n_i - l)$-partitions of G_i by dynamic programming in a tree fashion, i.e., first compute all maximum partitions for pairs of "consecutive" H_j where $j \bmod k+1 = i$, then for quadruples of such H_j etc.;
7. For each l, $1 \le l < n$, output the largest among the maximum $(l, n-l)$-partitions of G_i, $0 \le i \le k$.

Lemma 3.1 *For each l, $1 \le l < n$, Algorithm 1 outputs an $(l, n-l)$-partition of G within $k/(k+1)$ of the maximum.*

Proof: Let P be a maximum $(l, n-l)$-partition of G. For each edge e in P, there is at most one i, $0 \le i \le k$, such that e is not an edge of G_i. Consequently, there is i', $0 \le i' \le k$, such that $G_{i'}$ does not include at most $|P|/(k+1)$ edges of P. It follows that a maximum $(l, n-l)$-partition of such a $G_{i'}$ cuts at least $k|P|/(k+1)$ edges. Algorithm 1 outputs an $(l, n-l)$-partition of G cutting at least so many edges as a maximum $(l, n-l)$-partition of $G_{i'}$. $\square$

Lemma 3.2 *Algorithm 1 runs in $O(k2^{3k-1}n^3)$ time.*

Proof: The time complexity of the algorithm is dominated by that of step 4 and 6.

The subgraphs H_j of G are so called k-outerplanar graphs and have bounded treewidth $3k-1$ [8]. Hence, for a given i, $0 \le i \le k$, all maximum $(l, n'_j - l)$-partitions of H_j where $j \bmod k+1 = i$ can be computed in time $O(2^{3k-1}n^3)$ by Lemma 2.1, the pairwise disjointness of the subgraphs and $j \le n$. It follows that the whole step 4 can be implemented in time $O(k2^{3k-1}n^3)$.

In step 6, a maximum $(l, n_i - l)$-partition of the union of 2^{q+1} "consecutive" H_j's satisfying $j \bmod k+1 = i$ can be determined on the basis of appropriate maximum partitions of its two halves, each being the union of 2^q of the H_j's, in time $O(n)$. Hence, since $l \le n_i$ and the number of nodes in the dynamic programming tree is $O(n)$, the whole step 6 takes $O(kn^3)$ time. $\square$

Theorem 3.1 *Algorithm 1 yields a PTAS for all maximum $(l, n-l)$-partitions of a planar graph.*

Corollary 3.1 *The problem of max-bisection on planar graphs admits a PTAS.*

4 Final remark

We can easily obtain an analogous PTAS for Min-Bisection on planar graphs in a special case when the size of Min-Bisection is $\Omega(n)$. If the size of Min-Bisection is $o(n)$ and many of the removed edges have endpoints in the different sides of the bisection, such a method may however fail to produce a good approximation.

5 Acknowledgments

We thank Uri Feige, Mark Jerrum, Mirek Kowaluk, Mike Langberg, and Monique Laurent for many stimulating remarks and discussions.

References

[1] S. Arnborg, Efficient algorithms for combinatorial problems on graphs with bounded decomposability — A survey, *BIT*, 25 (1985), pp. 2 – 23.

[2] S. Arora, D. Karger and M. Karpinski. Polynomial Time Approximation Schemes for Dense Instances of NP-hard Problems, Proceedings 27th ACM STOC, pp. 284-293, 1995.

[3] A.A. Ageev and M.I. Sviridenko. Approximation algorithms for Maximum Coverage and Max Cut with cardinality constraints. Proceedings of IPCO99, Lecture Notes in Computer Science 1610, pp. 17-30, 1999.

[4] B.S. Baker. Approximation algorithms for NP-complete problems on planar graphs. Proceedings of the 24th IEEE FOCS, 1983, pp. 265-273.

[5] F. Barahona. On the computational complexity of Ising spin glass models. J. Phys. A; Math. Gen. 15 (1982), pp.3241-3253.

[6] H.L. Bodlaender, A tourist guide through treewidth. *Acta Cybernetica*, 11 (1993), pp. 1 – 23.

[7] H.L. Bodlaender, A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Computing*, 25 (1996), pp. 1305 – 1317.

[8] H.L. Bodlaender, A partial k-arboretum of graphs with bounded treewidth. Available at http://www.cs.ruu.nl/ hansb/index.html .

[9] H.L. Bodlaender and K. Jansen. On the complexity of the Maximum Cut problem. Nordic Journal of Computing, vol. 7, pp. 14-31, 2000.

[10] U. Feige, M. Karpinski and M. Langberg. A Note on Approximating MAX-BISECTION on Regular Graphs. ECCC (http://www.eccc.uni-trier.de/eccc/), TR00-043 (2000).

[11] U. Feige and R. Krauthgamer. A polylogarithmic approximation of the minimum bisection. To appear in Proceedings FOCS'2000.

[12] A. Frieze and M. Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. Algorithmica 18, pp. 67-81, 1997.

[13] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of ACM, 42, pp. 1115-1145, 1995.

[14] F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. SIAM J. Comput. Vol. 4, No. 3, pp. 221-225, 1975.

[15] E. Halperin and U. Zwick, Improved approximation algorithms for maximum graph bisection problems, Manuscript, 2000.

[16] M. Jerrum, Personal communication, August, 2000.

[17] M. Karpinski, M. Kowaluk and A. Lingas. Approximation Algorithms for Max-Bisection on Low Degree Regular Graphs and Planar Graphs. ECCC (http://www.eccc.uni-trier.de/eccc/), TR00-051 (2000).

[18] S. Khanna and R. Motwani. Towards a Syntactic Characterization of PTAS. Proceedings of the 28th ACM STOC, 1996, pp. 329-337.

[19] N. Robertson and P.D. Seymour, Graph minors. II. Algorithmic aspects of tree-width, *Journal of Algorithms*, 7 (1986), pp. 309-322.

[20] Y. Ye, A O.699 - approximation algorithm for Max-Bisection, Submitted to Math Programming, available at URL http://dollar.biz.uiowa.edu/col/ye, 1999