Polynomial Time Approximation Schemes for MAX-BISECTION on Planar and Geometric Graphs

Klaus Jansen [*]   Marek Karpinski [†]   Andrzej Lingas [‡]   Eike Seidel [§]

Abstract

We present a unified framework for constructing polynomial time approximation schemes (PTASs) for the problems of Max-Bisection on planar and geometric intersection graphs.

[*]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.

[§]Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität zu Kiel, 24098 Kiel. Email: ese@informatik.uni-kiel.de.

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 [16] 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. Contrary to the status of planar max-cut, planar max-bisection has been proven recently to be NP-hard in exact setting by Jerrum [17]. Karpinski et al. observed in [18] that the max-bisection problem for planar does not fall directly into the Khanna-Motwani's syntactic framework for planar optimization problems [19]. 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.

Note that the NP-hardness of exact planar max-bisection makes our PTAS result best possible under usual assumptions.

Interestingly, our PTAS for planar max-bisection can be easily modified to a PTAS for the problem of min-bisection on planar graphs in the very special case where the min-bisection is relatively large, i.e., cuts $\Omega(n \log\log n/\log n)$ edges.

Unit disk graphs are another important class of graphs defined by the geometric conditions on a plane. An undirected graph is a unit disk graph if its vertices can be put in one to one correspondence with disks of equal radius in the plane in such a way that two vertices are joined by an edge if and only if the corresponding disks intersect. Tangent disks are considered to intersect.

Our third main result is the first polynomial time approximation scheme for the max-bisection problem on unit disk graphs. The scheme can be easily generalized to include other geometric intersection graphs. It is obtained by combining (again via tree-typed dynamic programming) the idea of Hunt et al. of dividing the input graph defined by plane conditions into families of subgraphs [15] with the aforementioned known methods of finding maximum partitions of dense graphs [2].

The structure of our paper is as follows. The next section complements the introduction with basic definitions and facts. In Section 3, the algorithms for optimal partitions of bounded treewidth graphs are given. Section 4 presents the PTAS for planar max-bisections. In Section 5, we make several observations on the approximability of planar min-bisection. Finally, Section 6 describes the PTAS for max-bisection on unit disk graphs. In conclusion we notice that same technique can be applied also for other geometric intersection graphs.

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 analogously.*

The notion of treewidth of a graph was originally introduced by Robertson and Seymour [21]. It has turned out to be equivalent to several other interesting graph theoretic notions, e.g., the notion of partial k-trees [1, 5].

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[6]: *For a bounded treewidth graph, a tree decomposition of minimum treewidth can be found in linear time.*

To state our approximation results on max-bisection 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).*

An approximation ratio and a PTAS for a minimization problem are defined analogously.

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 [7], 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 [18] 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 [19] 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, for convention extend the levels by k empty ones numbered $-k+1$, $-k+2$, ..., 0;

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, $0 \leq i \leq k$, set G_i to the union of the subgraphs H_j where $j \bmod k+1 = i$;

6. For each i, $0 \leq i \leq k$, 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, $1 \leq l < n$, output the largest among the maximum $(l, n-l)$-partitions of G_i, $0 \leq i \leq k$.

Lemma 3.1 *For each l, $1 \leq l < n$, 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, $0 \leq i \leq k$, such that e is not an edge of G_i. Consequently, there is i', $0 \leq i' \leq k$, 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 $G_{i'}$. □

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 $3k-1$ [7]. Hence, for a given i, $0 \leq i \leq k$, 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 $j \leq n$. 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 $l \leq n_i$ and the number of nodes in the dynamic programming tree is $O(n)$, the whole step 6 takes $O(kn^3)$ time. □

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 Observations on min-bisection for planar graphs

We can easily obtain an analogous PTAS for min-bisection of planar graphs in the very special case when the size of min-bisection is $\Omega(n)$. Simply, at least one of the subgraphs G_i of G misses at most $|E|/(k+1)$ edges of G. Therefore, the number of edges cut by a min-bisection of such a G_i can increase at most by $|E|/(k+1)$ in G. By picking k sufficiently large we can guarantee an arbitrarily close approximation of min-bisection in G.

In fact, we can obtain even a slightly stronger result on min-bisection for planar graphs by observing that our method runs in polynomial time even for non-constant k (up to $O(\log n)$) provided that a tree-decomposition of graphs with treewidth equal to such a k can be determined in polynomial time. At present, the best tree-decomposition algorithms have the leading term k^k [8] so we can set k to $O(\log n/\log\log n)$ keeping the polynomial time performance of our method. In this way, we obtain the following theorem.

Theorem 4.1 *The min-bisection problem on planar graphs in which the size of min-bisection is $\Omega(n\log\log n/\log n)$ admits a PTAS.*

Observe that the presence of large degree vertices in a planar graph can cause the large size of min-bisection, e.g., in a star graph. For bounded-degree planar graphs the size of min-bisection is $O(\sqrt{n})$ by the following argument.
For a planar graph of maximum degree d construct a separator tree by applying the planar separator theorem [20] recursively. Next, find a path in the tree from the root down to the median leaf. By deleting the edges incident to the vertex separators along the path and additionally $O(1)$ edges, we can easily halve the set of vertices of the graph such that none of the remaining edges connects a pair of vertices from the opposite halves. The number of deleted edges is clearly $O(d\sqrt{n})$. In fact, we do not have to construct the whole separator tree, but just the path, and this can be easily done in time $O(n\log n)$ [20].

Theorem 4.2 *For a planar graph on n vertices and maximum degree d, a bisection of size $O(d\sqrt{n})$ can be found in time $O(n\log n)$.*

Clearly, if a graph has an $O(1)$-size bisection, it can be found by exhaustive search in polynomial time. We conclude that at present we have efficient methods for at least $O(1)$-approximation of min-bisection in planar graphs if it is size is either $\Omega(n \log\log n / \log n)$ or $O(1)$, or $O(\sqrt{n})$ and the maximum degree is constantly bounded. These observations suggest that a substantial improvement of the logarithmic approximation factor for min-bisection on planar graphs given in [11] might be possible.

5 PTAS for max-bisection of a unit disk graph

In this section we design a PTAS for max-bisection of unit disk graphs, another important class of graphs defined by the geometric conditions on a plane.

Recall that an undirected graph G is a unit disk graph if its vertices can be put in one to one correspondence with disks of equal radius in the plane in such a way that two vertices are joined by an edge if and only if the corresponding disks intersect. Tangent disks are considered to intersect. We may assume w.l.o.g that the radius of each disk is one. Since the recognition problem for unit disk graph is NP-hard, we shall also assume that a geometric representation of the graph is given as input.

Our technique works in a similar way as in the case for planar graphs. The input graph G is divided into families of subgraphs $H_{i,j}$ using the ideas of Hunt et al. given in [15]. Next, approximative solution to all $(l, n_{i,j} - l)$-partitions of every subgraph $H_{i,j}$, where $n_{i,j}$ denotes the number of vertices in $H_{i,j}$, are computed by the methods given in [2]. Via a tree-type dynamic programming these solutions are used to obtain an overall solution for G.

In order to divide the graph G, we impose a grid of horizontal and vertical lines on the plane, that are 2 apart of each other. The v-th vertical line, $-\infty < v < \infty$, is at $x = 2v$. The h-th horizontal line, $-\infty < v < \infty$, is at $y = 2h$. We say, that the v-th vertical line has index v and that the h horizontal line has index h. Further we denote the vertical strip between the v-th and the $(v+1)$-th vertical line as the strip with index v and analogue for the horizontal strip between the h-th and the $(h+1)$-th horizontal line.

Each vertical strip is left closed and right open, each horizontal strip is closed at the top and open at the bottom. A disk is said to lie in a given strip if its center lies in that strip. Note that every disk lies in exactly one horizontal and vertical strip.

For a fixed k consider the subgraph $H_{i,j}$ of G, $-\infty < i, j < \infty$, induced by the disks that lie in the intersection of the horizontal strips $i, i+1, \ldots, i+k$ and the vertical strips $j, j+1, \ldots, j+k$. Let $n_{i,j}$ be the number of vertices of $H_{i,j}$. By a packing argument it can be shown that for fixed $k > 0$, the size of a maximum independent set of such a subgraph is at most $2(k+3)^2\pi$.

Lemma 5.1 *There is a positive constant c such that if $n_{i,j} > c \log n$ then the subgraph $H_{i,j}$*

of G is dense.

Proof: Partition the vertex-set of $H_{i,j}$ successively into maximal independent sets by determining a maximal independent set I_1, remove its vertices and again determine a maximal independent set I_2 and so on. As described above the number of independent sets is at least $n_{i,j}/2(k+3)^2\pi$. Since each I_j is maximal there is at least one edge from a vertex of I_j to every $I_{j'}$, $j < j'$. If we understand the set of independent sets as a complete graph on $n_{i,j}/2(k+3)^2\pi$ vertices it follows that $H_{i,j}$ has $\Omega(n_{i,j}^2)$ edges and hence $H_{i,j}$ is dense. □

Corollary 5.1 *If $n_{i,j} > c \log n$ then the size of a maximum bisection of $H_{i,j}$ is $\Omega(n_{i,j}^2)$.*

Proof: Partition the vertex-set of $H_{i,j}$ as before and use the maximum independent sets to build up the sets of the bisection. Since all independent sets are maximal there are $\Omega(n_{i,j}^2)$ edges between the sets of bisection. □

Consequently the techniques given in [2] are applicable to the subgraph $H_{i,j}$.

Algorithm 2

input: a unit disk graph $G = (V, E)$ specified by a set V of disks in the plane and the coordinates of their centers and a positive integer k;

output: $(1 - \frac{1}{k+1}^2)(1 - \delta)$-approximations of maximum bisection of G

1. Divide the plane by imposing a grid of width two;
2. Construct the subgraphs $H_{i,j}$ of G as described above;
3. For each i and each j set $n'_{i,j}$ to the number of vertices in $H_{i,j}$ and compute all $(l, n'_{i,j} - l)$-partitions of $H_{i,j}$ either approximatively or optimal if $n'_{i,j} = O(\log n)$;
4. For each r and s, $0 \leq r, s \leq k$, set $G_{r,s}$ to the union of the subgraphs $H_{i,j}$ where $i \pmod{k+1} = r$ and $j \pmod{k+1} = s$;
5. For each r and s, $0 \leq r, s \leq k$, set $n_{r,s}$ to the number of vertices in $G_{r,s}$ and compute a bisection of $G_{r,s}$ within $(1 - \delta)$ of its maximum by dynamic programming in a tree fashion. Therefore enumerate the subgraphs in increasing order of the sum $i + j$ and compute all partitions of pairs of "consecutive" $H_{i,j}$ respectively to this ordering on the basis of the computed partitions, then for quadruples of such $H_{i,j}$ etc.;
6. Output the largest bisection of $G_{r,s}$, $0 \leq r, s \leq k$.

If $n_{i,j} \leq c \log n$ we can solve the maximum bisection problem for the subgraph $H_{i,j}$ optimal by enumerating all possibilities. Otherwise the problem is solvable approximatively by solving the following polynomial integer program:

$$\begin{array}{lllr} \textit{maximize} & \sum_{\{i,j\} \in E(H_{i,j})} x_i(1-x_j) + x_j(1-x_i) & & (1) \\ \textit{subject to} & \sum x_i = l, & & (2) \\ & x_i \in \{0,1\} & i = (1, \ldots, n_{i,j}) & (3) \end{array}$$

This program can be solved by the use of Theorem 1.10 given in [2] within an error of at most $\epsilon n_{i,j}^2$, which also satisfies the linear constraint (2) of the program within an additive error of $O(\epsilon \sqrt{n_{i,j} \log n_{i,j}})$. In order to get a subset of size l we move at most $\epsilon \sqrt{n_{i,j} \log n_{i,j}}$ in or out. This affects the number of edges included in the partition by at most $\epsilon \sqrt{n_{i,j} \log n_{i,j}} \leq \epsilon n_{i,j}^2$. Hence we can compute a maximum $(l, n_{i,j} - l)$-partition of a subgraph $H_{i,j}$ that has more than $c \log n$ vertices within an additive error of $2\epsilon n_{i,j}^2$ of the maximum.

Lemma 5.2 *Algorithm 2 outputs an bisection of G within $(1 - \frac{1}{k+1})^2(1-\delta)$ of the maximum.*

Proof: Let P be a maximum bisection of G. For each edge $e \in P$ and a fixed r, $0 \leq r \leq k$, there is at most one s, $0 \leq s \leq k$, such that e crosses a vertical line whose index modulo $k+1$ is s. Analoguosly, there is for each $e \in P$ and a fixed s, $0 \leq s \leq k$, at most one r, $0 \leq r \leq k$, such that e crosses a horizontal line whose index modulo $k+1$ is r. Consequently there is a pair (r, s), $0 \leq r, s \leq k$, such that a maximum $(l, n-l)$-partition of $G_{r,s}$ cuts at least $(1 - \frac{1}{k+1})^2 |P|$ edges.

By Corollary 5.1, the size of maximum bisection of the subgraph $G'_{r,s}$ of $G_{r,s}$ that consists of all $H_{i,j}$ with more than $c \log n$ vertices is $\sum_{n_{i,j} > c \log n} \Omega(n_{i,j}^2)$. Consequently, the error caused by the solutions of the polynomial integer programs for the subgraphs $H_{i,j}$ of $G'_{r,s}$ are at most a $\delta = 2\epsilon$ fraction of an optimum solution of maximum bisection for $G'_{r,s}$. Since the partitions for each $H_{i,j}$ with at most $c \log n$ vertices are computed optimally, we obtain a bisection of $G_{r,s}$ within $(1 - \delta)$ of the maximum.
Thus algorithm 2 outputs a bisection of G within $(1 - \frac{1}{k+1})^2(1-\delta)$ of the maximum. □

Theorem 5.1 *The problem of max-bisection on unit disk graphs admits a PTAS.*

The same approach can be used to obtain a PTAS for the maximum bisection problem in geometric intersection graphs both of other regular polygons and also of regular geometric objects in higher dimensions.

6 Acknowledgments

We thank Hans Bodlaender, Uri Feige, Mark Jerrum, Miroslaw 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] H.L. Bodlaender, A tourist guide through treewidth. *Acta Cybernetica*, 11 (1993), pp. 1 – 23.

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

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

[8] H.L. Bodlaender, Personal communication, August, 2000.

[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] H.B. Hunt, M.V. Marathe, V. Radhakrishnan, S.S. Ravi, D.S. Rosenkrantz, R.E. Stearns. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. Proc. 2nd Annual European Symposium on Algorithms, (ESA), LNCS 855, pp. 468-477, Springer Verlag, June, 1994

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

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

[18] 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).

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

[20] R.J. Lipton and R.E. Tarjan. A separator theorem for planar graphs. SIAM J. Appl. Math. Vol. 36 (1979), pp. 177-189.

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

[22] 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