Improved Approximations of Crossings in Graph Drawings and VLSI Layout Areas*

Guy Even† Sudipto Guha‡ Baruch Schieber§

Abstract

We give improved approximations for two classical embedding problems: (i) minimizing the number of crossings in a drawing of a bounded degree graph on the plane; and (ii) minimizing the VLSI layout area of a degree four graph. These improved algorithms can be applied to improve a variety of VLSI layout problems. Our results are as follows. (i) We compute a drawing on the plane of a bounded degree graph in which the sum of the numbers of vertices and crossings is $O(\log^3 n)$ times the optimal minimum sum. This is a logarithmic factor improvement relative to the best known result. (ii) We compute a VLSI layout of a degree four graph in a grid with constant aspect ratio the area of which is $O(\log^4 n)$ times the optimal minimum layout area. This is an $O(\log^2 n)$ improvement over the best known long standing result.

1 Introduction

In this paper we study two related problems: (1) drawing a bounded degree graph on the plane with the fewest number of crossings of edges and (2) minimization of the VLSI layout area of a degree 4 graph in a grid with constant aspect ratio. A considerable attention had been devoted to these problems in the past (see, e.g., [L80, V81, BL84, U84, SSSV97, LR98]).

A *drawing of a graph on the plane* is an injection of the vertices of the graph to points in the plane and a mapping of the edges to simple continuous curves between the vertices' images. A curve may not contain a drawing of a vertex as an internal point and three (or more) curves may intersect only at an image of a vertex. A *crossing* is an intersection of two curves at a point that is not an image of a vertex. The *size* of a drawing is the sum of the numbers of vertices and crossings in the drawing. The problem of determining the minimum size of a drawing of a graph on the plane is NP-Complete [GJ83].

Bhatt and Leighton, in [BL84], apply a $B(n)$-approximate bisection procedure recursively to decompose a bounded degree graph with n vertices. They prove that this recursive decomposition induces a drawing of size $O(B^2(n)\log^2 n)$ times the minimum size drawing. Shahrokhi *et al.* [SSSV97] considered straight line drawings induced by decomposition trees and presented a simpler construction of a drawing of the same size as the one achieved in [BL84]. Leighton and Rao [LR98] showed that the above result can be realized with a $(\frac{1}{3}, \frac{2}{3})$ separator. Leighton and Rao also showed how to find such a separator with size bounded by $\alpha(n) = O(\log n)$ times the optimal bisector. This implied an $O(\log^4 n)$ (or $O(\alpha^2(n)\log^2 n)$) approximation algorithm for the drawing size of a bounded degree graph. The bound on the approximation factor in this algorithm relies only on the fact that an optimal drawing induces a planar graph which admits small vertex separators. The reason that the approximation algorithm applies only to bounded degree graphs is that, in bounded degree graphs, the vertex separators and the edge separators have roughly the same size. No nontrivial approximation algorithms are known for the general case of arbitrary degrees. From now on we consider only bounded degree graphs for the drawing problem.

A *VLSI layout* of a degree 4 graph is an embedding of the graph in a grid. The vertices of the graph are injected into the grid vertices, and the edges of the graph are mapped into edge disjoint paths in the grid. In fact, we consider the more restrictive "Manhattan" model of routing (in which layer assignment is trivial) and not the "knock-knee" model. Namely, an intersection of two paths (i.e., images of edges) at an internal grid vertex is allowed provided that one path traverses the grid vertex horizontally and the other path traverses the grid vertex vertically. The objective in the VLSI layout area problem is to minimize the area of the grid. Note that to be able to embed a graph in a grid it must be of degree 4. From now on we consider only degree 4 graphs for the VLSI layout problem.

Leiserson [L80] and Valiant [V81] showed that a planar graph can be embedded in a grid with constant aspect ratio of size $O(n\log^2 n)$. Bhatt and Leighton suggested to embed an arbitrary (degree 4) graph in two steps. First, draw it on the plane, and then apply the approximation algorithm for

---

*Part of this work was done while the first two authors visited IBM T.J. Watson Research Center.

†Dept. of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel. E-mail: *guy@eng.tau.ac.il*.

‡Computer Science Department, Stanford University, Stanford, CA 94305. Supported by IBM cooperative fellowship, an ARO MURI Grant DAAH04-96-1-0007 and NSF Award CCR-9357849, with matching funds from IBM, Schlumberger Foundation, Shell Foundation, and Xerox Corporation. E-mail: *sudipto@cs.stanford.edu*.

§IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598. E-mail: *sbar@watson.ibm.com*.

VLSI layouts of planar graphs to the planar graph resulting from the drawing by adding vertices in crossings. The $O(\log^4 n)$ approximation algorithm for drawing graphs on the plane combined with the $O(\log^2 n)$ approximation algorithm for VLSI layouts of planar graphs yield an $O(\log^6 n)$ approximation algorithm for the VLSI layout problem.

A *decomposition tree* of a graph G is a tree together with a one-to-one correspondence between its leaves and the vertices of G. Each internal tree node is associated with the cut separating the vertex sets mapped to the sets of leaves of the subtrees rooted at its children. (A more rigorous definition of such trees is given later.) Bhatt and Leighton, in [BL84], proposed a special type of decomposition trees, called *bifurcators*. These decomposition trees are binary trees, and the cut sizes associated with their nodes decrease exponentially relative to the cut size of the root. Bhatt and Leighton [BL84] demonstrated that a $\sqrt{2}$-bifurcator can be used for a wide variety of problems in VLSI layouts: minimizing capacitive delay, producing fault tolerant layouts, layouts for graphs using prefabricated chips, regular layouts, and layouts minimizing wire crossing, to name a few. (The detailed description of these applications is provided in [BL84].) Using the approximation of drawing size, Bhatt and Leighton provided an $O(\log^{2.5} n)$ approximation for the optimal $\sqrt{2}$-bifurcator

Our Results. In this paper we provide improved approximation algorithms for the above problems. We first provide an $O(\log^3 n)$ approximation for the drawing size problem. As a consequence of this result, based on Bhatt and Leighton's results, we obtain an $O(\log^2 n)$ approximation for the optimal $\sqrt{2}$-bifurcator problem and a corresponding improvement for all its applications.

Our $O(\log^3 n)$ approximation for the drawing size problem immediately yields an $O(\log^5 n)$ approximation for the minimum VLSI layout area. Using further structural properties of the decomposition tree computed in our algorithm, we show that the VLSI layout area problem can be approximated to a factor of $O(\log^4 n)$.

In terms of the best approximation factor known for separators, our results can be stated as follows. Let $\alpha(n)$ denote the smallest known ratio of the separator size (that can be computed efficiently) to the optimal bisector size. We show an $O(\alpha^2(n)\log n)$-approximation of the minimum drawing size and an $O(\alpha^2(n)\log^2 n)$-approximation of the minimum VLSI layout area.

Our approximation algorithms construct a decomposition tree that can be viewed as an approximation of a decomposition tree $\widetilde{T}$ obtained by recursively bisecting the planar graph induced by an optimal drawing. Such a decomposition tree $\widetilde{T}$ induces a drawing of size $O(\log n)$ times the optimal drawing size. In addition, $\widetilde{T}$ can be used in Leiserson's embedding algorithm [L80] to compute a layout of area $O(\log^2 n)$ times the optimal layout area.

The decomposition tree that we compute mimics the useful properties of $\widetilde{T}$. In the problem of drawing a graph on the plane we attach an estimator $\phi(t)$ to every tree node that estimates the optimal drawing size of the corresponding subgraph. The estimator quality is one-sided; it may not surpass (twice) the drawing size, but might be much smaller than the drawing size. The estimators have the following two properties that follow the properties of the drawing sizes of the subgraphs in $\widetilde{T}$: (a) the cut sizes are bounded by the square root of the corresponding estimators upto an logarithmic error term; and (b) the estimators decrease exponentially as one goes down the tree.

The main difficulty in constructing such an "approximated" decomposition tree with estimators is that the only tool we have is approximate separators. A bottom-up approach fails because merging can take place only when the subgraphs have comparable drawing sizes. Estimates on drawing sizes are based on lower bounds derived from separators. Such a lower bound would need to be based on the subgraph with the largest separator; however, finding such a subgraph is computationally prohibitive. A naive top-down approach of recursive separators fails since the partitioning is not guaranteed to divide the drawing sizes in a balanced fashion, and hence, the estimators may not decay exponentially. We, therefore, apply a top-down approach with re-balancing to guarantee that the estimators decrease exponentially.

Our technique extends to a framework for computing decomposition trees with estimators of other super-additive functions. (Super-additive functions are functions in which the sum of the function values on disjoint subgraphs is no more than the value of the function on the whole graph). Guha [G00] applied this technique to the fill-in function, yielding better approximations for chordal completion, operation count and certain cases of elimination height and pathwidth.

For the grid embedding algorithm we follow the paradigm suggested by Leiserson [L80]. First, we present a "weighted" version of Leiserson's algorithm that can be implemented when the separation properties of the decomposition tree are given relative to *weights* of subgraphs rather than *sizes* of subgraphs. A tempting approach is to use the decomposition tree computed for approximating the drawing size together with the estimators associated with its nodes as weights for the implementation of Leiserson's algorithm. However, this approach fails since the estimators are not adequate as weights. This is because they do not satisfy a super-additivity property. Therefore, we compute new node weights. These weights require a re-balancing of the decomposition tree before the weighted version of Leiserson's can be applied.

Organization. In Section 2 we define the problems, and describe the basic tools: decomposition trees, separators, etc. In Section 3 we describe decomposition trees with estimators and show that they are useful for drawing graphs. In Section 4 we present an algorithm for computing decomposition trees with estimators. In Section 5 an improved approximation of $\sqrt{2}$-bifurcators is presented. In Section 6 we present an algorithm for approximating the minimum VLSI layout area.

2 Preliminaries

Minimum drawing size of a graph. A *drawing* D of a graph G is a one-to-one mapping of the vertices to points on the plane. We call the image of a vertex its *position*. Every edge is mapped to a simple (non-self intersecting) continuous curve connecting the positions of the end-points of the edge. A curve corresponding to an edge may not contain a position of a vertex as an interior point. A *crossing* is an intersection of the interiors of two curves corresponding to images of edges. (Note that a position of a vertex does not count as a crossing since a position of a vertex cannot be in the interior of a curve.) We do not allow more than two curves to intersect at every crossing. A *point* in a drawing is either a position of a vertex of the graph or a crossing. We denote the minimum number of points, over all drawings of G, by $\mathrm{MDS}(G)$. Note that $\mathrm{MDS}(G) = n + CR(G)$, where $CR(G)$

denotes the (minimum) crossing number of G, and n is the number of vertices in G.

The minimum drawing size of a graph satisfies the following *super-additivity* property.

Proposition 1 *If the vertex set of G is partitioned into disjoint sets and the respective graphs induced by them are denoted by $\{G_i\}$ then* $\mathrm{MDS}(G) \geq \sum_i \mathrm{MDS}(G_i)$.

Minimum layout area of a graph. A *VLSI layout* of a graph of degree 4 is an embedding of the graph in a grid. The embedding consists of an injection of the graph vertices to the grid vertices and a mapping of the graph edges into edge disjoint paths in the grid. The paths into which two edges are mapped may intersect at an internal grid vertex, provided that one edge is mapped to the two horizontal grid edges touching this vertex and the other edge is mapped to the vertical grid edges touching it. The *aspect ratio* of a (rectangular) grid is the ratio of its short dimension to its long one. Let $\mathrm{AREA}_\sigma(G)$ denote the minimum area required to embed a graph G in a host grid with aspect ratio σ. Let $\mathrm{AREA}(G) = \min_{\sigma>0} \mathrm{AREA}_\sigma(G)$. Using a "folding" argument, Leiserson[L80] showed that $\mathrm{AREA}_1(G) \leq 3 \cdot \mathrm{AREA}(G)$. It follows that considering only embeddings in squares has only a small effect on the minimum area.

Since every layout is also a drawing, and the area of the layout is an upper bound on the size of the drawing, the following property holds.

Proposition 2 *For every graph of degree* 4, $\mathrm{AREA}(G) \geq \mathrm{MDS}(G)$.

Decomposition trees. For two disjoint subsets $U, W \subset V$, let $cut(U, W)$ be the edges between vertices in U and vertices in W.

A (binary) decomposition tree[1] T of $G(V, E)$ is defined by a binary tree T and a one-to-one mapping from the leaves of T to the vertices of G. We extend this mapping to the internal nodes of T and map each such internal node $t \in T$ to the subset of vertices in G mapped to the leaves of the subtree rooted at t. (To unify the presentation we assume also that each leaf of T is mapped to a singleton set of vertices of G.) Let V_t be the set of vertices mapped to the tree node t, and let $n_t = |V_t|$. Let $G_t(V_t, E_t)$ denote the subgraph of G induced by V_t. To each internal tree node $t \in T$, we associate the cut between the vertex sets mapped to the two children of t. Formally, $cut_t = cut(V_{t_\ell}, V_{t_r})$, where t_ℓ and t_r are the children of t. For an edge $e = (u, v) \in E$, let $t(e)$ be the tree node for which $e \in cut_{t(e)}$. Note that $t(e)$ is the lowest tree node that is mapped to a set containing both u and v.

Drawings induced by decomposition trees. In [BL84], Bhatt and Leighton considered (recursive) drawings that are induced by decomposition trees. Shahrokhi *et al.* [SSSV97] described simple drawings that are induced by ordered decomposition trees.[2] An ordered decomposition tree induces a permutation of the vertices by considering the order of the leaves in a preorder traversal. (Recall that each such leaf is mapped to a vertex in G.) Let $v_1, \ldots, v_n$ denote the vertices of G in this order. Map the vertices to points along a line (also called the *spine*), unit distances apart, according to their order. All the edges are drawn on one side of the spine as half circles; each edge $e = (v_i, v_j)$ is drawn as half a circle with diameter $|i - j|$ and endpoints at v_i and v_j.

[1]Decomposition trees may not be binary. However, in this paper we consider only binary decomposition trees.

[2]An ordered tree is a tree where the children of each internal node are ordered.

Shahrokhi *et al.* count the number of crossings in such an induced drawing using the following observation.

Lemma 3 *If the curves corresponding to edges $e = (v_i, v_j)$ and $e' = (v_k, v_\ell)$ cross, then one of the following two conditions must hold:*

1. *The tree node $t(e)$ is a descendant of $t(e')$ and $t(e)$ is an ancestor of either v_k or v_ℓ. (A node is considered a descendant of itself.)*

2. *The tree node $t(e)$ is an ancestor of $t(e')$ and $t(e')$ is an ancestor of either v_i or v_j.*

Proof: Observe that, for each tree node t, the vertices in V_t are drawn contiguously along the spine. From the drawing it follows that whenever two edges $e = (v_i, v_j)$ and $e' = (v_k, v_\ell)$ cross, then either $k < i \leq \ell \leq j$ or $i \leq k \leq j < \ell$. The lemma follows. □

A crossing of the curves of e and e' is charged to edge e if $t(e)$ is an ancestor of $t(e')$, and to edge e' otherwise. From Lemma 3 it follows that a crossing of the curves of $e = (u, v)$ and e' is charged to e if $t(e')$ is either on the path connecting $t(e)$ to u in T, or on the path connecting $t(e)$ to v. Recall that for an internal tree node t the number of edges e with $t(e) = t$ is $|cut_t|$. We get the following bound on the number of crossings charged to an edge e.

Corollary 4 *Let $P(u, v)$ denote the set of nodes in T on the path from the leaf mapped to u to the leaf mapped to v. The number of crossings that are charged to the edge $e = (u, v)$ is bounded by* $\sum_{t \in P(u,v)} |cut_t|$.

Existence of special separators. In our algorithm we need to compute a *simultaneous* edge separator. Such an edge separator is a cut that partitions a graph in a balanced way according to *two* measures: the number of vertices and their weights. Suppose that we are given a graph G with weights $w(v)$ associated with each vertex of G, where the weights $w(v)$ are bounded by a constant. A simultaneous edge separator $(S, V - S)$ partitions the vertex set into two sets S and $V - S$ such that (i) $\min\{|S|, |V - S|\} \geq \frac{1}{4}|V|$; and (ii) $\min\{w(S), w(V - S)\} \geq \frac{1}{3}w(V)$. (For a set of vertices S, $w(S)$ denotes the sum of the weights of vertices in the set.) In this case we say that $(S, V - S)$ is a $(\frac{1}{3}, \frac{2}{3})$ separator with respect to the weights and a $(\frac{1}{4}, \frac{3}{4})$ separator with respect to the number of vertices.

Lemma 5 *There exists a cut of size $O(\sqrt{\mathrm{MDS}(G)})$ that is a simultaneous edge separator, i.e., it is a $(\frac{1}{3}, \frac{2}{3})$ separator with respect to the weights and a $(\frac{1}{4}, \frac{3}{4})$ separator with respect to the number of vertices.*

Proof: Consider a graph G with weights defined over its vertices such that the maximum vertex weight is at most $\frac{2}{3}$ of the total weight. Define the weight of a graph to be the sum of the weights of its vertices.

First, we show that there exists a cut of size $O(\sqrt{\mathrm{MDS}(G)})$ that separates G into subgraphs of weight between one third and two thirds of the total weight. Consider an optimal drawing of G, and let $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ denote the planar graph that is obtained by introducing vertices in the crossings of the optimal drawing. Note that $|\widetilde{V}| = \mathrm{MDS}(G)$. We iteratively apply the Planar Separator Theorem [LT79] to the planar graph $\widetilde{G}$. This theorem implies that $\widetilde{G}$ has a cut of size $c \cdot \sqrt{\mathrm{MDS}(G)}$, for some constant c, that separates it into subgraphs such that the number of vertices in each of them at most $\frac{2}{3}\mathrm{MDS}(G)$. In case the weight of these subgraphs is at most $\frac{2}{3}w(G)$, then we are done. Otherwise, we separate the subgraph with weight more than $\frac{2}{3}w(G)$ (and number of vertices at most $\frac{2}{3}\mathrm{MDS}(G)$) into two subgraphs such that the number of vertices in each of them is at most $\left(\frac{2}{3}\right)^2 \mathrm{MDS}(G)$. The cut size is guaranteed to be at most $c \cdot \sqrt{\frac{2}{3}\mathrm{MDS}(G)}$. Again, in case the weight of these newly created subgraphs is at most $\frac{2}{3}$ of the total weight, then we are done since the heavier subgraph defines a separator as required. Otherwise, we separate the higher weight subgraph. Since the number of vertices is decreasing exponentially in each iteration, we are guaranteed to stop eventually with a partitioning in which the weight of every subgraph is at most $\frac{2}{3}w(G)$. Moreover, when we stop, the weight of one of the subgraphs is at least $\frac{1}{3}w(G)$. Since the cut sizes decrease exponentially, the size of the cut that separates the subgraph of weight between $\frac{1}{3}w(G)$ and $\frac{2}{3}w(G)$ from the rest of the graph is $O(\sqrt{\mathrm{MDS}(G)})$.

A simultaneous separator can be shown to exist as follows. Let $(S, V-S)$ denote a $(\frac{1}{3}, \frac{2}{3})$ separator with respect to the weights $w(v)$. If $|S|, |V| - |S| \geq \frac{1}{4}|V|$, we are done. Otherwise, without loss of generality, $|S| > \frac{3}{4}|V|$. Let $(S_1, S - S_1)$ denote a $(\frac{1}{3}, \frac{2}{3})$ separator of S with respect to the number of vertices. Without loss of generality, assume that $w(S_1) \leq w(S - S_1)$. In case $w(S - S_1) \geq \frac{1}{3}w(V)$, we are done since the partition into $S - S_1$ and $V - (S - S_1)$ is a good partition and the size of the cut is $O(\sqrt{\mathrm{MDS}(G)})$.

If $w(S - S_1) < \frac{1}{3}w(V)$, let $(S_2, V - S - S_2)$ denote a $(\frac{1}{3}, \frac{2}{3})$ separator of $V - S$ with respect to the weights. Without loss of generality, assume that $w(S_2) \leq w(V - S - S_2)$. We claim that the partition into $(S - S_1) \cup S_2$ and $(V - S - S_2) \cup S_1$ is a good partition. First, note that the size of the cut is $O(\sqrt{\mathrm{MDS}(G)})$. Clearly, it is a $(\frac{1}{4}, \frac{3}{4})$ partition with respect to the number of vertices since $|S_1|, |S - S_1| \geq \frac{1}{4}|V|$.

We show that $\frac{1}{3}w(V) \leq w(V - S - S_2) + w(S_1) \leq \frac{2}{3}w(V)$, and thus it is a $(\frac{1}{3}, \frac{2}{3})$ partition with respect to the weights. The lower bound is proved as follows:

$$\begin{aligned} w(V-S-S_2) + w(S_1) &\geq \frac{1}{2}w(V-S) + w(S_1) \\ &= \frac{1}{2} \cdot [(w(V) - w(S-S_1)) + w(S_1)] \end{aligned}$$

Since $w(V) - w(S - S_1)$ is at least $\frac{2}{3}w(V)$ the lower bound follows. The upper bound is proved as follows:

$$\begin{aligned} w(V-S-S_2) + w(S_1) &\leq \frac{2}{3}w(V-S) + \frac{1}{2}w(S) \\ &\leq \frac{2}{3}w(V) - \frac{1}{6}w(S) \end{aligned}$$

□

Computing Simultaneous Separators. Simultaneous separators can be computed using the Leighton-Rao algorithm [LR98] for approximate unweighted as well as weighted separators. A cut $cut(U, V-U)$ is a b-balanced cut if the weights of U and $V - U$ are at most $1 - b$ times the total weight. The Leighton-Rao algorithm receives two balance parameters $b \leq \frac{1}{2}$ and $b' \leq \min\{b, \frac{1}{3}\}$ and returns a b'-balanced cut the size of which is $O(\frac{\log n}{b - b'} \cdot \mathrm{SEP}_b)$, where SEP_b denotes the size of an optimal b-balanced cut.

If the balance parameter b is at most $\frac{1}{3}$ (as in the proof of Lemma 5), then the Leighton-Rao algorithm can be used to compute a b-balanced cut the size of which is $O(\sqrt{\mathrm{MDS}(G)} \cdot \log n)$. This observation follows by setting $b' = 1 - \sqrt{1-b}$ and applying the Leighton-Rao algorithm at most twice. The first partitioning yields a b'-balanced cut $cut(U, V-U)$. If this cut happens to be b-balanced, then we are done. Otherwise, assume U is the heavier part (i.e., $w(U) \geq w(V-U)$). Since $cut(U, V-U)$ is not b-balanced, it follows that $w(U) > (1-b)w(V)$. Apply the Leighton-Rao algorithm to U to obtain a b'-balanced cut $(U_1, U - U_1)$. Assume U_1 is the heavier part. Then the cut $cut(U_1, V - U_1)$ is b-balanced. The upper bound follows since $w(U_1) \leq (1-b')w(U) \leq (1-b')^2 w(V) = (1-b)w(V)$. The lower bound follows since $w(U_1) \geq \frac{1}{2}w(U) \geq \frac{1}{2}(1-b) \cdot w(V) \geq b \cdot w(V)$. (The last inequality holds since $b \leq \frac{1}{3}$.) Note that the sizes of both cuts are bounded by $O(\sqrt{\mathrm{MDS}(G)} \cdot \log n)$, since the sizes of the optimal b-balanced cuts of V and U are bounded by $\sqrt{\mathrm{MDS}(G)}$. Therefore, the proof of Lemma 5 can be made constructive based on approximate b-balanced cuts.

Another way to compute simultaneous separators with "relaxed" balance parameters uses the spreading metric based algorithm [ENRS99] for simultaneous separators. One application of this algorithm computes a simultaneous separator $(S, V-S)$ of size $c_S\sqrt{\mathrm{MDS}(G)}\log n$ (for some constant c_S) with the following "relaxed" balance parameters:

1. $\min\{|S|, |V-S|\} \geq \frac{1}{8}|V|$; and
2. $\min\{w(S), w(V-S)\} \geq \frac{1}{6}w(V)$.

Weight functions induced by cuts. The weight functions used in our decomposition algorithm are defined by cuts. We consider two subsets of vertices $U, W \subseteq V$ and call U the *home* set and W the *outside* set. The weight of $v \in U$ with respect to its *home* set U and the *outside* set W, denoted $w_{U,W}(v)$, is defined to be the number of edges connecting v with vertices in $W - U$. Note that $\sum_{v \in U} w_{U,W}(v) = cut(U, W-U)$. Since the graphs are of bounded degree, the weight function is bounded as well.

3 A decomposition tree for minimizing the drawing size

3.1 Motivation

Following Bhatt and Leighton, the drawings which we obtain are induced by decomposition trees. The decomposition trees are constructed based on partitions computed by a separator procedure. To motivate our definition of decomposition trees that are used for approximating the minimum drawing size, consider a decomposition tree $\widetilde{T}$ that is constructed by recursively separating the graph using an "ideal" separator procedure. The ideal separator procedure is based on an optimal drawing D that induces a planar graph by considering crossings as vertices. The ideal separator procedure returns separators of the planar graph induced by the drawing. This implies that in the ideal decomposition tree $\widetilde{T}$ the *drawing size* of the subgraph associated with a tree node is roughly half the drawing size of the subgraph associated with its parent. (Note that such a decomposition tree can be computed only given the optimal drawing.)

Let us analyze the size of a drawing induced by such a ideal decomposition tree $\widetilde{T}$. For every tree node t, let $size(t)$ denote the size of the drawing obtained by restricting D to G_t. The Planar Separator Theorem implies that $|cut_t| =$

$O(\sqrt{size(t)})$. By Corollary 4, an edge $e = (u, v)$ is charged for at most $\sum_{t \in P(u,v)} |cut_t| = \sum_{t \in P(u,v)} O(\sqrt{size(t)})$ crossings. The recursive separation implies that $size(t)$ decreases exponentially as one goes down the tree, therefore, an edge $e = (u, v)$ is charged for $O(\sqrt{size(t(e))})$ crossings. To bound the total charges, charge a tree node t for all the edges e for which $t = t(e)$. Hence, the charging of a tree node t is bounded by $O(|cut_t| \cdot (\sqrt{size(t(e))}) = O(size(t))$. The tree nodes in the same layer of $\widetilde{T}$ induce a partition of G. By the Super-additivity Property, it follows that the sum of $size(t)$ over all tree nodes t in the same layer is bounded by $O(|D|) = O(\mathrm{MDS}(G))$. Since there are only a logarithmic number of layers in the tree, the size of the drawing induced by $\widetilde{T}$ is $O(\mathrm{MDS}(G) \cdot \log n)$.

Our goal is to define a decomposition tree that mimics the properties of a ideal decomposition tree. We attach to every tree node an estimator $\phi(t)$ which "behaves" like $size(t)$, so that we can adapt the analysis above for the drawing size that is actually computed. To be able to apply the same analysis $\phi(t)$ should have the following properties:

1. $|cut_t| \leq \sqrt{\phi(t)}$.
2. $\phi(t)$ decreases exponentially along every path that goes down the tree.
3. $\phi(t) = O(\mathrm{MDS}(G_t))$.

However, we do not have a ideal separator procedure, and hence, we relax Property 1 to $|cut_t| \leq \sqrt{\phi(t)} \cdot O(\log n)$. This degrades the approximation factor by an additional factor of $O(\log^2 n)$.

3.2 A decomposition tree with estimators

Definition 1 *A decomposition tree T of a graph G together with a function $\phi(t)$ defined over the tree nodes is called* a decomposition tree with estimators *if the following properties are satisfied for every internal tree node $t \in T$:*

P1: $|cut_t| \leq c\sqrt{\phi(t)} \cdot \log n_t$, *for some constant c.*

P2: *For every descendant t' of t: $\phi(t') \leq \phi(t) \cdot (\frac{3}{4})^{d(t,t')}$, where $d(t, t')$ is the number of hops along the path from t to t'.*

P3: $\phi(t) < 2 \cdot \mathrm{MDS}(G_t)$.

The following claim shows that drawing induced by a decomposition tree with estimators is within $O(\log^3 n)$ factor from optimal.

Claim 6 *The size of a drawing of G that is induced by a decomposition tree T with estimators $\phi(\cdot)$ is $O(\mathrm{MDS}(G) \cdot \log^3 n)$.*

Proof: Consider an edge $e = (u, v)$. From Corollary 4 it follows that the number of crossings that are charged to e is bounded by $\sum_{t \in P(u,v)} |cut_t|$. Define $P(e, u)$ and $P(e, v)$ to be the paths in T from $t(e)$ to the leaves that are mapped to u and v respectively. It follows that

$$\sum_{t \in P(u,v)} |cut_t| \leq \sum_{t \in P(e,u)} |cut_t| + \sum_{t \in p(e,v)} |cut_t|$$

Each summand evaluates to $O(\sqrt{\mathrm{MDS}(G_{t(e)})} \cdot \log n)$, using the fact that the sum involves geometrically decreasing terms.

By Properties P1 and P3, for each tree node t, $|cut_t| = O(\sqrt{\mathrm{MDS}(G_t)} \cdot \log n)$, hence we can bound the total number of crossings charged to the edges in cut_t by $O(\log^2 n \cdot \mathrm{MDS}(G_t))$.

Fix some height in T and consider all the tree nodes $t_1, \ldots, t_r$ that are of this height. Notice that the sets V_{t_i} are disjoint and thus by the Super-additivity Property (Proposition 1), $\mathrm{MDS}(G) \geq \sum_{i=1}^{r} \mathrm{MDS}(G_{t_i})$. It follows that the the total number of crossings charged to the edges in $\cup_{i=1}^{r} cut_{t_i}$ is bounded by

$$\sum_i O(\log^2 n \cdot \mathrm{MDS}(G_{t_i})) \leq O(\log^2 n \cdot \mathrm{MDS}(G)).$$

Observe that the tree T has height $O(\log n)$. This is because the estimator $\phi(\cdot)$ decreases exponentially along each path from the root to a leaf, and the estimator of the root of the tree is $O(m^2+n)$ since $\mathrm{MDS}(G) = O(m^2+n)$. It follows that the total number of crossing points is $O(\log^3 n \cdot \mathrm{MDS}(G))$. □

Remark: A better approximation algorithm for separators would yield a better approximation factor for the crossing number and the layout area. In particular, an $\alpha(n)$-approximation algorithm for separators would imply a tightened Property P1, i.e., $cut_t \leq c\sqrt{\phi(t)} \cdot \alpha(n_t)$, for every tree node t. This would imply an approximation factor of $O(\alpha^2(n) \cdot \log n)$ for the crossing number and $O(\alpha^2(n) \cdot \log^2 n)$ for the layout area (see Sec. 6). This improves the corresponding results of Bhatt and Leighton (Theorems 17 and 19 in [BL84]) by a logarithmic factor and a log-square factor, respectively.

4 Constructing a decomposition tree with estimators

Our goal is to construct a decomposition tree T of G with estimators $\phi(t)$. The challenge in computing the tree is due to the requirements that: (i) $\sqrt{\phi(t)}$ cannot be too small, since we must be able to find a separator of size $c\sqrt{\phi(t)} \log n_t$; (ii) $\phi(t)$ must decay exponentially along every "downward" path; and (iii) $\phi(t)$ cannot be greater than (twice) $\mathrm{MDS}(G_t)$. The algorithm avoids an exponential search by using "failures" for pruning backtracking and for re-balancing.

Throughout the description, let $G(V, E)$ denote the graph for which a decomposition tree is being computed. To simplify notation, we refer to induced subgraphs of G simply by their vertex sets.

Our algorithm is recursive. The input to the recursive procedure consists of three inputs:

1. a vertex subset V' (the goal is to compute a decomposition tree for the subgraph $G'(V', E')$ induced by V'),
2. a "guess" g on the estimate $\phi(r')$ of the root of this tree, and
3. a weight function $w_{V',U}$ on the vertices of G' induced by a home set V' and an outside set U. This weight function plays a role only in some of the cases.

Let $\mathsf{decompose}(V', g, w)$ denote the recursive procedure (in case the weight does not play a role, this parameter is omitted and the procedure is denoted by $\mathsf{decompose}(V', g)$). The procedure returns one of the following:

- "fail" - if the guess is too small (i.e., $g < \mathrm{MDS}(V')$); or
- "success" - in which case the procedure returns also a decomposition tree T' of G' with estimators $\phi(t)$ for which $\phi(r') \leq g$, where r' is the root of T'.

Since $n + \binom{m}{2}$ is an upper bound on $\mathrm{MDS}(G)$, the decomposition tree with estimators of the input graph $G(V, E)$ is computed by calling $\mathsf{decompose}(V, n + \binom{m}{2})$.

The procedure $\mathsf{decompose}(V', g, w)$ searches for an approximation of the smallest feasible guess in the range $[1, g]$. This search is performed by calling a second procedure $\mathsf{test\text{-}decompose}(V', g, w)$, the output of which is also either "fail" (when the guess is too small) or "success" (when the guess is large enough, but might be too large). In case of success, $\mathsf{test\text{-}decompose}(V', g, w)$ returns a decomposition tree with "lax" estimators. The estimate $\phi(r')$ assigned to the root by the lax estimators might not satisfy Property P3. That is, it outputs a success also in case the estimators $\phi(t)$ satisfy Properties P1-P3 for all tree nodes but the root node, and the estimator for the root $\phi(r')$ satisfies P1 and P2 but $\phi(r') \geq 2 \cdot \mathrm{MDS}(V')$. The two procedures could be merged into a single procedure in which a range of guess values is input (the range is $[1, g]$ for $\mathsf{decompose}$ and $[g, g]$ for $\mathsf{test\text{-}decompose}$), and a search takes place within the guess range for an approximation of the smallest feasible guess. However, the partitioning into two procedures seems to simplify the description.

The procedure $\mathsf{decompose}(V', g, w)$**.** This procedure calls successively $\mathsf{test\text{-}decompose}(V', g/2^i, w)$, for $i = 0, 1, 2 \ldots$ until $\mathsf{test\text{-}decompose}$ returns "fail". We distinguish between two cases.

CASE 1: The first call to $\mathsf{test\text{-}decompose}$ (with $i = 0$) returns "fail". In this case $\mathsf{decompose}(V', g, w)$ returns "fail" as well.

CASE 2: Some calls to $\mathsf{test\text{-}decompose}$ returned "success". Let s be the index of the last success. In this case $\mathsf{decompose}(V', g, w)$ returns "success" with the decomposition tree and estimators computed by the procedure $\mathsf{test\text{-}decompose}(V', g/2^s, w)$.

The failure with $s+1$ in Case 2 is used to guarantee Property P3 with respect to $\phi(r')$. In Lemma 8, we prove that if $\mathsf{test\text{-}decompose}(V', g', w)$ fails, then $\mathrm{MDS}(V') > g'$. Therefore, the failure with $s+1$ implies that $g/2^s < 2 \cdot \mathrm{MDS}(V')$, and Property P3 is satisfied.

The procedure $\mathsf{test\text{-}decompose}(V', g, w)$**.** This procedure has several steps.

STEP 0: If V' contains a single vertex v, then return "fail" if $g < 1$, "success" if $g \geq 1$. In case of success the decomposition tree is a single leaf t with $V_t = \{v\}$ and $\phi(t) = 1$. If G' contains more than one vertex, go to the next step.

STEP 1: Find a simultaneous separator of V'. (Recall that such a separator is a $(\frac{1}{3}, \frac{2}{3})$ separator with respect to the weight and $(\frac{1}{4}, \frac{3}{4})$ with respect to the cardinality.) Denote the two partitions by V_1' and $V_2' = V' - V_1'$. Note that when the weight function is not given as input, a non-weighted $(\frac{1}{4}, \frac{3}{4})$-separator suffices.

STEP 2: If the size of the separator is more than $c_S \cdot \sqrt{g} \cdot \log n'$, where $n' = |V'|$, then return "fail". Otherwise, go to the next step. The reason for the failure is that we are guaranteed to find a separator of size at most $c_S \cdot \sqrt{\mathrm{MDS}(V')} \log n'$, and thus we have a proof that $\mathrm{MDS}(V') > g$.

STEP 3: Call $\mathsf{decompose}(V_1', g/2)$ and $\mathsf{decompose}(V_2', g/2)$. Note that these calls are made with no weight function even if the original call was with a weight function.

STEP 4: Distinguish between three cases depending on the outcome of the Step 3.

CASE 1: Both recursive calls returned a success. In this case we have computed decomposition trees with estimators for V_1' and V_2' that satisfy Properties P1-P3. We construct a decomposition tree for V' by connecting both subtrees to a root r'. Set the estimator $\phi(r')$ to be g, and return "success" with the resulting tree and estimators. Note that in this case, $\phi(r')$ might be much larger than $\mathrm{MDS}(V')$.

CASE 2: Both recursive calls returned a failure. In this case we return "fail". The failures imply that $\mathrm{MDS}(V_i') > g/2$, for $i = 1, 2$, hence by Super-additivity $\mathrm{MDS}(V') > g$, which justifies the failure.

CASE 3: One recursive call returned a success and the other a failure. Assume that $\mathsf{decompose}(V_1', g/2)$ succeeded and $\mathsf{decompose}(V_2', g/2)$ failed. This is the most complicated case which may occur, even when g is a good guess, due to lack of balance between $\mathrm{MDS}(V_1')$ and $\mathrm{MDS}(V_2')$. Such an imbalance can hamper satisfying P2 (i.e., the exponential decay of the estimators) along the branch from V' to V_2'. The algorithm attempts to re-balance the partitioning by recursing on V_2' and searching for a balanced cut $cut(V', V' - U)$, where $U \subseteq V_2'$. To avoid a significant increase in the cut size (compared to $cut(V_1', V_2')$), simultaneous separators are employed as follows.

Define a weight function w_2 with the home set V_2'. If the procedure $\mathsf{test\text{-}decompose}(V', g)$ was called with no weight function the outside set is V'. Otherwise, that is, if the procedure $\mathsf{test\text{-}decompose}(V', g, w)$ was called with a weight function, the outside set is the same one used for w. The idea is that the outside set is set to be the outside set of the first vertex set that failed in the current sequence of failures.

In Case 3 continue with the following sub-steps.

STEP 4.1: Call $\mathsf{test\text{-}decompose}(V_2', g, w_2)$. If this call fails, then we have proof that $\mathrm{MDS}(V') \geq \mathrm{MDS}(V_2') > g$, therefore, return "fail".

Suppose that this call succeeded. Consider the recursion tree created by the call to $\mathsf{test\text{-}decompose}(V_2', g, w_2)$. Since this call succeeded, the two recursive calls of $\mathsf{decompose}$ with the partitions of V_2' and guess $g/2$ as inputs either both succeeded or at most one of them failed. In case one of the calls failed, we recursively call $\mathsf{test\text{-}decompose}$ on this "failed" partition with a guess g. Since the size of the graph is only smaller, and since g is sufficiently large, $\mathsf{test\text{-}decompose}$ must succeed on this "failed" partition with a guess g. Again, this implies that the two recursive calls of $\mathsf{decompose}$ with the partitions of this "failed" partition and $g/2$ either both succeeded or at most one failed. It follows that the recursion tree has a chain of "failure" nodes, where each such failure node denotes a failure in a call to $\mathsf{decompose}$ with the corresponding subgraph and guess $g/2$. We are guaranteed that the calls to $\mathsf{decompose}$ with the subgraphs that correspond to the siblings of these failed nodes succeeded. This sequence of "mixed" siblings must end with two siblings for which $\mathsf{decompose}$ succeeded on their corresponding subgraphs and $g/2$.

Figure 1 depicts the recursion tree until two succeseful siblings are encountered. We use the following notation for the vertex sets inducing the subgraphs corresponding to the siblings in this sequence. Let $F_0 = V_2'$ and $S_0 = V_1'$ be the vertex sets of the first pair of mixed siblings. Suppose that the sequence of mixed pairs is of length $x \geq 1$. Then, for $1 \leq i < x$, the sets F_i (representing failure) and S_i (representing

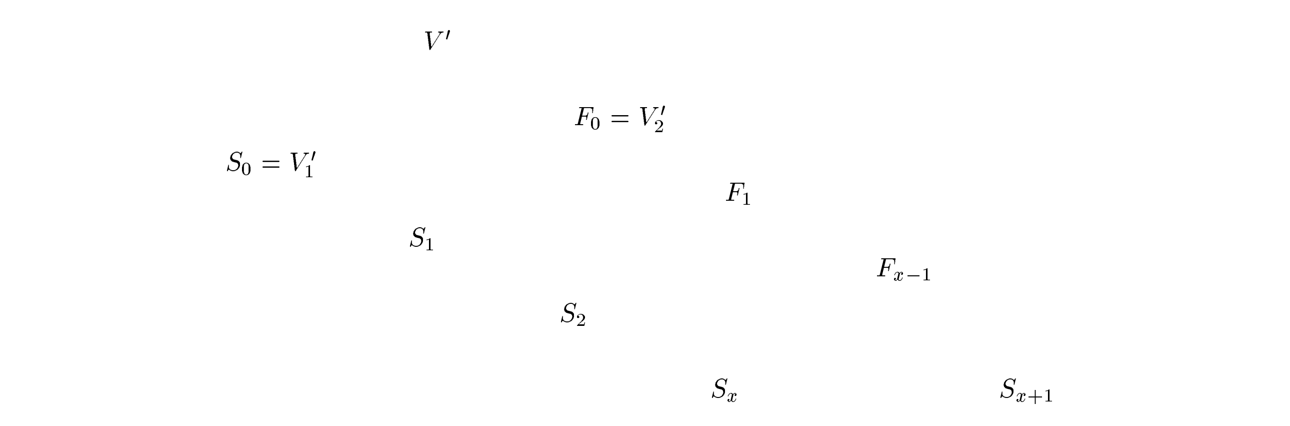

Figure 1: A recursion tree for which: (a) decompose$(F_i, g/2)$ failed for $i = 0, 1, \ldots, x-1$; and (b) decompose$(S_i, g/2)$ succeeded for $i = 0, 1, \ldots, x+1$.

success) denote the partition of F_{i-1} computed in Step 1 of test-decompose$(F_{i-1}, g, w_{F_{i-1},V'})$. The set F_i is the one for which the call decompose$(F_i, g/2)$ failed and the set S_i is the one for which the call decompose$(S_i, g/2)$ succeeded. This chain of failures ends with the vertex set F_{x-1} that is partitioned into two sets S_x and S_{x+1} for which both calls decompose$(S_x, g/2)$ and decompose$(S_{x+1}, g/2)$ succeeded. Let $S = S_0 \cup S_1 \cup \cdots \cup S_{x-1}$.

STEP 4.2: Call decompose$(S \cup S_x, \frac{3}{4}g)$ and decompose$(S \cup S_{x+1}, \frac{3}{4}g)$. If both fail, then return "fail". Lemma 8 proves that these two failures imply that $\mathrm{MDS}(V') > g$.

Suppose that decompose$(S \cup S_x, \frac{3}{4}g)$ succeeded. Recall that decompose$(S_{x+1}, g/2)$ succeeded as well. Claim 7 proves that $|cut(S \cup S_x, S_{x+1})| \leq c \cdot \sqrt{g} \cdot \log n'$. Return "success" with the decomposition tree obtained by connecting the decomposition trees computed for $S \cup S_x$ and S_{x+1} with a common root r' and the estimators returned by the two recursive calls together with $\phi(r') = g$.

Correctness. The following claim proves that the re-balancing performed in Step 4.2 does not increase the cut by much, so that Property P1 holds.

Claim 7 *If* test-decompose(V', g) *reaches Step 4.2, then there exists a constant c such that*

$$|cut(S \cup S_x, S_{x+1})| \leq c \cdot \sqrt{g} \cdot \log n'.$$

Proof: We use the same notations as in the algorithm. Notice that $S \cup S_x = V' - S_{x+1}$, and

$$cut(S_{x+1}, V' - S_{x+1}) = cut(S_{x+1}, S_x) \cup cut(S_{x+1}, V' - F_{x-1}).$$

The success of the test-decompose$(F_{x-1}, g, w_{F_{x-1},V'})$ implies that

$$|cut(S_{x+1}, S_x)| \leq c_S \cdot \sqrt{g} \cdot \log n'.$$

Since (S_{x+1}, S_x) is a $(\frac{1}{3}, \frac{2}{3})$-separator with respect to the weight function $w_{F_{x-1},V'}$, and since for any subset $U \subseteq F_{x-1}$, the total weight $w_{F_{x-1},V'}$ over the vertices of U is $cut(U, V' - F_{x-1})$ we get

$$|cut(S_{x+1}, V' - F_{x-1})| \leq \frac{2}{3}|cut(F_{x-1}, V' - F_{x-1})|.$$

Continuing in the same manner we get that for $i = x-1, \ldots, 1$,

$$|cut(F_i, V' - F_i)| \leq |cut(F_i, S_i)| + \frac{2}{3}|cut(F_{i-1}, V' - F_{i-1})|,$$

and for $i = 0, \ldots, x-1$, $|cut(F_i, S_i)| \leq c_S \cdot \sqrt{g} \cdot \log n'$. Hence, there exists a constant c that depends on c_S such that $|cut(S_{x+1}, V' - S_{x+1})| \leq c \cdot \sqrt{g} \cdot \log n'$, as required. □

We now prove that failure of test-decompose(V', g, w) implies that $\mathrm{MDS}(V') > g$. This claim holds regardless of whether or not a weight function was given as an actual parameter. Notice that the guarantee is one sided; a success might be obtained even if g is very small. (For example, consider a balanced binary tree as an input graph. All cuts have size 1, so g can be sub-linear, i.e., $(\frac{4}{3})^{\log_2 n}$, but the drawing size is linear.)

Lemma 8 *If the procedure* test-decompose(V', g, w) *fails, then* $g < \mathrm{MDS}(V')$.

Proof: The proof is by induction on $|V'|$. The induction basis is trivial (see Step 0). In the induction step, we consider the steps in which a failure can be decided upon. Assume that the lemma holds for all graphs with less than $|V'|$ vertices, and assume that $g \geq \mathrm{MDS}(V')$ and test-decompose(V', g, w) fails.

1. A failure in Step 2 contradicts the assumption because of the guarantee of the partitioning algorithm.

2. A failure in Step 4 Case 2 contradicts the assumption as follows. By the induction hypothesis, failure of the procedure test-decompose$(V_i', g/2)$ implies that $\mathrm{MDS}(V_i') > g/2$, for $i = 1, 2$. From the Super-additivity Property (Proposition 1) it follows that $\mathrm{MDS}(V') > g$, a contradiction.

3. A failure in Step 4.1 implies, by the induction hypothesis, that $\mathrm{MDS}(V_2') > g$. Since $\mathrm{MDS}(V') \geq \mathrm{MDS}(V_2')$, we reach a contradiction.

4. The harder case to prove is when a failure is decided upon in Step 4.2. Consider an optimal drawing D' of G', the subgraph of G induced by V'. Every point in D' is either a crossing point between two edges of G' or a vertex in V'. Let D_{x-1} denote the restriction of D' to a drawing of F_{x-1}. Recall that test-decompose$(F_{x-1}, g/2)$ failed. Therefore, by the induction hypothesis, $\mathrm{MDS}(F_{x-1}) > g/2$. Since D_{x-1} is a drawing of the subgraph induced by F_{x-1}, it follows that D_{x-1} contains more than $g/2$ points.

The points in D_{x-1} can be partitioned as follows: (a) points of the restriction of D_{x-1} to S_x; (b) points of the restriction of D to S_{x+1}; and (c) crossings of edges in the cut $cut(S_x, S_{x+1})$. Let P denote the points in D_{x-1} of type (b) or (c). Without loss of generality, the number of points in P is at least half the number of points in D_{x-1}, and hence, greater than $g/4$.

Let $D_{0..x}$ denote the restriction of D' to a drawing of $S \cup S_x$. The points in the drawing $D_{0..x}$ are contained in the points of the drawing D' that are not points in P. Since the number of points in P is more than $g/4$, we get that the number of points in $D_{0..x}$ is strictly less than $\mathrm{MDS}(G') - g/4$. By our assumption $g \geq \mathrm{MDS}(G')$, and thus the number of points in $D_{0..x}$ is strictly less than $\frac{3}{4}g$. Since $D_{0..x}$ is a drawing of $S \cup S_x$, $\mathrm{MDS}(S \cup S_x) < \frac{3}{4}g$, and by the induction hypothesis, test-decompose$(S \cup S_x, \frac{3}{4}g)$ succeeds. Hence, failure is not decided upon in Step 4.2, a contradiction. □

The following claim shows that the computed decomposition tree with estimators satisfies Properties P1-P3.

Lemma 9 *Let T' denote the decomposition tree with estimators $\phi(t)$ returned by a successful call to* decompose(G', g). *Then T and $\phi(t)$ satisfy Properties P1-P3 with respect to G'.*

Proof: Follows by induction on $|V|$. □

Polynomial running time : Let $F(n, g)$ and $G(n, g)$ denote the running times of decompose and test-decompose when the subgraph has n nodes and the guess equals g. Note that the weight function does not affect the running time. Let $p(n)$ denote a polynomial bounding the running time of the partitioning algorithm and the lines in test-decompose that do not call decompose and test-decompose. The functions $F(n, g)$ and $G(n, g)$ satisfy the following recurrences: (We make a relaxed upper bound, in assuming that the functions $F(\cdot)$ and $G(\cdot)$ are monotone in both parameters.)

$$F(n,g) \leq \sum_{i=0}^{\lceil \log_2 g \rceil} G(n, g/2^i)$$

$$G(n,g) \leq p(n) + 2F(\frac{3}{4}n, g/2) + G(\frac{3}{4}n, g) + 2F(n, \frac{3}{4}g).$$

The following claim proves that the running time of the decomposition algorithm is polynomial. Recall that g is always polynomial in n.

Lemma 10 *There exists constants τ, γ, δ such that $F(n, g) \leq \tau \cdot p(n) \cdot n^{\gamma} \cdot g^{\delta}$.*

Proof Idea: One may simply prove this by induction. The reason the proof works is that at each recursion step there are either: (i) a constant number of recursive calls in which one of the parameters (the number of vertices or the guess value) decreases by a constant factor; or (i) a logarithmic number of recursive calls in which the guess parameters form a geometric sequence. For this reason it was imperative that we went down a path with geometrically decreasing number of vertices. It was also important that in Step 4.2 once the chain of failures ends with two successful siblings, we were able to argue that the guess value should decrease by a constant factor.

5 Approximating $\sqrt{2}$-bifurcators

An α-bifurcator is defined to be decomposition tree in which $|cut_t| \leq |cut_r| \cdot \alpha^{-depth(t)}$, for every tree node t, where r is the root of the tree. An optimal α-bifurcator is an α-bifurcator with the minimum $|cut_r|$. In our algorithm we require that the bounds on the cut sizes decrease geometrically by some factor f relative to *every* node and not only the root. This means that the bound on a cut size of a child is guaranteed to be at least f times smaller than the bound on the cut size of its parent. Such a decomposition tree can be define to be a *hereditary bifurcator.* In fact we have also constructed a $2/\sqrt{3}$ bifurcator. The following corollary is shown in [BL84, LR98].

Corollary 11 *Given an $O(\log^k n)$ approximation for the size of drawing on the plane, there exists an $O(\log^{(k+1)/2} n)$ approximation algorithm for finding the optimal $\sqrt{2}$-bifurcator.*

Thus we improve the approximation factor of $\sqrt{2}$-bifurcators by a $\sqrt{\log n}$ factor to $O(\log^2 n)$. As mentioned in the introduction, Bhatt and Leighton, [BL84], demonstrated that $\sqrt{2}$-bifurcators are useful for minimizing capacitive delay, producing fault tolerant layouts (layouts using prefabricated chip, regular layouts), minimizing wire crossing in layouts and more.

6 VLSI Layouts

In this section we present a VLSI layout algorithm for graphs of degree at most 4 in a grid with an aspect ratio (i.e., the ratio of its short dimension to its long one) of at least $\frac{1}{3}$. The algorithm can embed a graph G in any host grid of size $O(\mathrm{AREA}(G) \cdot \log^4 n)$ provided that the grid's aspect ratio is at least $\frac{1}{3}$.

An $O(\log^5 n)$-approximate VLSI layout area can be derived by applying planar graph embedding algorithms on the drawing computed in Sec. 4, the size of which is $O(\mathrm{MDS} \cdot \log^3 n)$ [L80, V81]. This is an improvement by a logarithmic factor over the approximation factor of the VLSI layout algorithm of Bhatt and Leighton [BL84]. We show how an additional logarithmic factor can be saved.

6.1 Weighted version of Leiserson's Algorithm

Leiserson's algorithm for computing layouts of graphs is based on the graphs satisfying a separator theorem. The separator theorem is stated in terms of the *number* of vertices. Namely, every n-vertex graph in a given family can be separated into two disjoint graphs, each containing at least a constant fraction of the vertices, by removing at most $f(n)$ edges. For example, in bounded degree planar graphs, the separator is guaranteed to be bounded by the square root of the number of vertices and the size of each part is bounded by $\frac{2}{3}$ of the original number of vertices. Planar graph embedding algorithms require $O(n \log^2 n)$ area.

We consider a "weighted" version of Leiserson's algorithm in which subgraphs are assigned weights. The input to the weighted version of Leiserson's algorithm consists of a graph and a decomposition tree with weights $Wt(t)$ defined over the tree nodes. Loosely speaking, the cut size in every internal tree node is bounded by the square-root of the weight of the tree node, and the weight of each node is divided roughly equally between its two children. The weighted version of Leiserson's algorithm returns a VLSI layout of area $O(Wt(r) \cdot \log^2 Wt(r))$, where r denotes the root of T. If $Wt(r) = O(\text{MDS}(G) \cdot \log^2 n)$, then $\log Wt(r) = O(\log n)$, and by Preposition 2 an approximation factor of $O(\log^4 n)$ follows for the VLSI layout.

The following properties of the decomposition tree T and the tree node weights $Wt(t)$ are sufficient for the "success" of the weighted version of Leiserson's algorithm:

Non-triviality: For each leaf ℓ, $Wt(\ell) \geq 1$.

Super-additivity: For any two siblings in the tree t_ℓ and t_r, with a parent t,

$$Wt(t_\ell) + Wt(t_r) \leq Wt(t).$$

Balance: For any two siblings in the tree t_ℓ and t_r,

$$\max\{Wt(t_\ell), Wt(t_r)\} \leq 2 \cdot \min\{Wt(t_\ell), Wt(t_r)\}.$$

(Note that the above two conditions imply that both the weights $Wt(t_\ell)$ and $Wt(t_r)$ are at most $\frac{2}{3}Wt(t)$.)

Separator: For any internal tree node t, $|cut_t| = O(\sqrt{Wt(t)})$.

We outline the weighted version of Leiserson's algorithm. First, the algorithm defines a function $A(t)$ over the tree nodes that specifies the area allocated for embedding G_t. The function $A(t)$ is defined as follows:

$$A(t) = \begin{cases} 1 & \text{if } t \text{ is a leaf} \\ \left(\sqrt{A(t_\ell) + A(t_r)} + 2\sqrt{3} \cdot |cut_t|\right)^2 & \text{otherwise,} \end{cases}$$

where t_ℓ and t_r denote the children of t. In Claim 12 it is proved that $A(t) = O(Wt(t) \cdot \log^2 Wt(t))$.

Given a rectangle R_t with aspect ratio at least $\frac{1}{3}$ and area $A(t)$, the algorithm computes an embedding of G_t in R_t as follows. A sub-rectangle R'_t of area $A(t_\ell) + A(t_r)$ that is similar to R_t is allocated for embedding G_ℓ and G_r. The sub-rectangle R'_t is divided between G_{t_ℓ} and G_{t_r} in proportion to the weights $Wt(t_\ell)$ and $Wt(t_r)$. After G_{t_ℓ} and G_{t_r} are embedded in the sub-rectangles allocated for them, the edges of cut_t are embedded using the unused rows and columns of R_t (at most 2 columns and 2 rows are used for embedding each edge in the cut). In Claim 12 it is proved that there is a sufficient amount of unused rows and columns left in R_t for routing the edges of cut_t.

We briefly point out why each of the properties is required: Non-triviality and Super-additivity guarantee that the size of a rectangle allocated for a subgraph is not smaller than the size of the subgraph. Super-additivity also makes sure that the sub-rectangle allocated for the children of t is not larger than the rectangle allocated for t. The Balance Property makes sure that the division of the rectangle does not create sub-rectangles whose aspect ratio is below $\frac{1}{3}$. The Separator Property is used to solve the recurrence and prove the bound on the required area.

The following claim shows that the weighted version of Leiserson's algorithm computes a layout of area $O(Wt(r) \cdot \log^2 Wt(r))$. The proof follows the same arguments as in the unweighted case.

Claim 12 *Let T denote a decomposition tree with node weights $Wt(t)$ of a graph G of degree 4. Suppose that the node weights satisfy the Non-triviality, Super-additivity, Balance, and Separator Properties. There exists a constant c_A such that if R is a rectangle of area at least $c_A \cdot Wt(r) \cdot \log^2 Wt(r)$ (where r denotes the root of T) and aspect ratio at least $\frac{1}{3}$, then the weighted version of Leiserson's algorithm succeeds in embedding G in R.*

6.2 Computing a weighted decomposition tree

Given a graph G, we show how to construct a decomposition tree T and a weight function $Wt(\cdot)$ that obeys the properties sufficient for the success of the weighted version of Leiserson's algorithm. Observe that recursively separating an optimal drawing of G generates a decomposition tree that satisfies these properties if the weight function equals the size of the sub-drawing. We show how to compute an "approximation" of such a tree in which the weight of the root is $O(\text{MDS}(G) \log^2 n)$.

At first glance, it is tempting to use the decomposition tree computed for the minimal drawing as the decomposition tree required here and to use the estimator function computed before as the required weight function. Two obstacles obstruct this approach: (a) The decomposition tree is not balanced since the ratio of estimators of two siblings is not bounded. (b) The decomposition tree may not obey the Super-additivity Property, as we may have two siblings with estimators that add up to more than the estimator of the parent (i.e., in Step 4.2, the sum of the estimators of the children is 1.25 times estimator of the parent). Violating Balance is solved by re-balancing, but violating Super-additivity makes $\phi(t)$ a bad candidate for a weight function. Super-additivity is obtained by presenting a weight function that is computed bottom-up (to ensure Super-additivity). Balance is obtained afterwards by re-balancing the tree with respect to the weights in a top-down fashion.

6.2.1 The new weight function

Let T denote the decomposition tree of G with estimator $\phi(t)$ computed in Section 4. Let $CW(t)$ be the maximum sum of cut sizes along a path from t to a leaf in the subtree rooted at t, where the maximum is taken over all such paths. The weight function $Wt(t)$ is defined as follows:

$$Wt(t) = \begin{cases} 1 & \text{if } t \text{ is a leaf} \\ \max\{Wt(t_\ell) + Wt(t_r), CW^2(t)\} & \text{otherwise.} \end{cases}$$

The following lemma bounds the weight of the root of T.

Lemma 13 *For some constant c_W, $Wt(t) \leq c_W \text{MDS}(G_t) \cdot \log^2(n_t + 1)$.*

Proof: The proof is by induction. The induction basis for the leaves is obvious. Consider an internal tree node t. If $Wt(t) = Wt(t_\ell) + Wt(t_r)$, then the claim follows by the super-additivity of the minimum drawing size.

If $Wt(t) = CW^2(t)$, note that the upper bounds on the sizes of the cuts along any path from t to any of its leaves decay exponentially and that the size of the first cut (and hence sum of all the cuts along a path from t to a leaf) is $O(\sqrt{\phi(t)} \log n_t) = O(\sqrt{\text{MDS}(G_t)} \log n_t)$. It follows that the square of the sum of these cuts is $O(\text{MDS}(G_t) \log^2 n_t)$. For an appropriately chosen c_W, the claim follows. □

The weight function $Wt(t)$ satisfies all the properties required except for the Balance Property. We show how to achieve this in the following subsection.

6.2.2 Re-balancing the tree

Re-balancing proceeds in a top down procedure recursively. The process will be performed in two steps. The second step will be the mere inflation of weights of some nodes in the tree. The recursive procedure used for the first step is rebalance($\mathcal{T}$). The input to this procedure is a collection of trees (inclusive of the case of a single tree) denoted by $\mathcal{T}$, where the initial input consists of a single tree. The procedure computes two collections $\mathcal{T}_1$ and $\mathcal{T}_2$ and calls rebalance recursively for each one of them. Given a collection $\mathcal{T}$ of trees, let $Wt(\mathcal{T}) = \sum_{T' \in \mathcal{T}} Wt(T')$.

The procedure rebalance$(\mathcal{T})$**.** Assume $\mathcal{T} = \{T_1, \ldots, T_p\}$.

STEP 1: If there exists any j such that $\sum_{i=1}^{j} Wt(T_i)$ is between $\frac{1}{3}$ and $\frac{2}{3}$ of $Wt(\mathcal{T})$, we create the two collections $\mathcal{T}_1 = \{T_1, \ldots, T_j\}$ and $\mathcal{T}_2 = \{T_{j+1}, \ldots, T_k\}$. The separating cut is of size 0.

STEP 2: If there is no such a j, then there exists a j such that $\sum_{i=1}^{j-1} Wt(T_i)$ and $\sum_{i=j+1}^{p} Wt(T_i)$ are both less than $\frac{1}{3}Wt(\mathcal{T})$. Starting at the root $r(T_j)$ of T_j walk down to the subtree T' such that T' is the first subtree whose weight is less than $\frac{2}{3}Wt(T_j)$ and T' is heavier than its sibling. Let the subtrees hanging from the path from $r(T_j)$ to T' (other than T') be $T'_1, \ldots, T'_q$ (where T'_q denotes the sibling of T'). By Lemma 14 below $\sum_{i=0}^{q} Wt(T'_i) < \frac{2}{3}Wt(T_j)$.

STEP 3: We have four collections:

$$\mathcal{T}_a = \{T_1, \ldots, T_{j-1}\} \qquad \mathcal{T}_b = \{T_{j+1}, \ldots, T_p\}$$
$$\mathcal{T}_c = \{T'\} \qquad \mathcal{T}_d = \{T'_1, \ldots, T'_q\}$$

Let $MIN(a,b) \in \{a,b\}$ be the index of the collection with the smaller weight between collections $\mathcal{T}_a$ and $\mathcal{T}_b$. Recall the weight of a collection $\mathcal{T}$ is the sum of the weights of the trees in $\mathcal{T}$. Similarly, $MAX(a,b) \in \{a,b\}$ is defined to be index of the larger weighted collection. Define $MIN(c,d)$ and $MAX(c,d)$ similarly. Let

$$\begin{aligned} \mathcal{T}_1 &= \mathcal{T}_{MIN(a,b)} \cup \mathcal{T}_{MAX(c,d)} \\ \mathcal{T}_2 &= \mathcal{T}_{MAX(a,b)} \cup \mathcal{T}_{MIN(c,d)}. \end{aligned}$$

STEP 4: Call rebalance$(\mathcal{T}_1)$ and rebalance$(\mathcal{T}_2)$, and let S_i denote the decomposition tree computed for $\mathcal{T}_i$. Let S be a decomposition tree for $\mathcal{T}$ obtained by connecting S_1 and S_2 to a root s. Set $Wt(s) = Wt(\mathcal{T})$.

Lemma 14 *In Step 2 above* $\sum_{i=0}^{q} Wt(T'_i) < \frac{2}{3}Wt(T_j)$.

Proof: Consider two cases: (a) If $Wt(T') > Wt(T_j)/3$, then it follows by super-additivity. (b) If $Wt(T') \leq Wt(T_j)/3$, then also $Wt(T'_q) \leq Wt(T_j)/3$. Since the weight of the parent of T' and T'_q is greater than $\frac{2}{3}Wt(T_j)$, it follows that $\sum_{i=0}^{q-1} Wt(T'_i) < Wt(T_j)/3$, and the claim follows. □

The Separator Property holds after re-balancing. A cut is either zero or is bounded by $CW(r(T_j))$ where $r(T_j)$ is the root of T_j, which is bounded by $\sqrt{Wt(T_j)}$. In addition, the Non-triviality Property and the Super-additivity Property hold. The subtrees returned have weights $Wt(\mathcal{T}_1)$ and $Wt(\mathcal{T}_2)$. The following lemma is proved in the full paper.

Lemma 15 *In the decomposition tree returned, for any two siblings t_ℓ and t_r, both the weights $Wt(\mathcal{T}_1)$ and $Wt(\mathcal{T}_2)$ are at most $\frac{2}{3}$ of $Wt(\mathcal{T})$.*

Inflating weights and achieving balance: The Balance Property is obtained by inflating weights using the following local transformation. For every non-root internal tree node t_ℓ, let t_r denote its sibling, and assume that $Wt(t_\ell) \leq Wt(t_r)$. Inflate $Wt(t_\ell)$ as follows:

$$Wt(t_\ell) = \max\{Wt(t_\ell), \frac{1}{2}Wt(t_r)\}.$$

After the inflation all the other properties are preserved. For example, the Super-additivity Property is preserved because if $Wt(t_\ell)$ is increased by inflation, then after inflation $Wt(t_\ell) + Wt(t_r) \leq \frac{3}{2}Wt(t_r) \leq \frac{3}{2} \cdot \frac{2}{3}Wt(t) = Wt(t)$, where t denotes the parent of t_ℓ and t_r.

Having computed a decomposition tree with weights that satisfy all the properties, we conclude with the following theorem.

Theorem 16 *A graph G of degree 4 can be embedded in polynomial time in any grid of aspect ratio at least $\frac{1}{3}$ and area $O(\text{AREA}(G)\log^4 n)$.*

Acknowledgments

The authors would like to thank Seffi Naor, János Pach, Satish Rao, and Farhad Shahrokhi for discussions about these problems.

References

[BL84] S.N. Bhatt and F.T. Leighton, "A framework for solving VLSI graph layout problems", J. of Computer and System Sciences, 28:300–343, 1984.

[ENRS99] G. Even, J. Naor, S. Rao and B. Schieber, "Fast approximate graph partitioning algorithms", SIAM J. on Computing, 28(6):2187–2214, 1999.

[G00] S. Guha, "Nested dissection and approximation algorithms", Manuscript.

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

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

[L80] C.E. Leiserson. "Area-efficient graph layouts (for VLSI)". In 21st Annual Symposium on Foundations of Computer Science, pp. 270-281, Syracuse, New York, 1980.

[L83] F.T. Leighton, *Complexity Issues in VLSI*, MIT Press, 1983.

[LR98] F.T. Leighton and S. Rao, "Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms", J. of the ACM, Upcoming Articles, `http://www.acm.org/jacm/Upcoming.html`

[LT79] R.J. Lipton and R.E. Tarjan, "A separator theorem for planar graphs", SIAM J. on Applied Mathematics, 36(2):177–189, 1979.

[SSSV97] F. Shahrokhi, O. Sýkora, L.A. Székely, and I. Vrťo, "Crossing Numbers: bounds and applications", Intuitive Geometry, Bolyai Society Mathematical Studies 6, Budapest, pp. 179–206, 1997.

[U84] J.D. Ullman, *Computational Aspects of VLSI*, Computer Science Press, 1984.

[V81] L.G. Valiant, "Universality considerations in VLSI circuits", IEEE Transactions on Computers, C-30(2):135–140, 1981.