Approximating the Bandwidth of Caterpillars

Uriel Feige
Weizmann Institute and Microsoft Research
urifeige@microsoft.com

Kunal Talwar
Microsoft Research
kunal@microsoft.com

June 16, 2005

Abstract

A caterpillar is a tree in which all vertices of degree three or more lie on one path, called the backbone. We present a polynomial time algorithm that produces a linear arrangement of the vertices of a caterpillar with bandwidth at most $O(\log n/\log\log n)$ times the *local density* of the caterpillar, where the local density is a well known lower bound on the bandwidth. This result is best possible in the sense that there are caterpillars whose bandwidth is larger than their local density by a factor of $\Omega(\log n/\log\log n)$. The previous best approximation ratio for the bandwidth of caterpillars was $O(\log n)$. We show that any further improvement in the approximation ratio would require using linear arrangements that do not respect the order of the vertices of the backbone.

We also show how to obtain a $(1+\epsilon)$ approximation for the bandwidth of caterpillars in time $2^{\tilde{O}(\sqrt{n/\epsilon})}$. This result generalizes to trees, planar graphs, and any family of graphs with treewidth $\tilde{O}(\sqrt{n})$.

1 Introduction

To set the ground for presenting our results, let us first define the main terms that we use.

Definition 1. *A* linear arrangement *of a graph is a numbering of its vertices from 1 to n. The* bandwidth *of a linear arrangement is the largest difference between the two numbers given to end-points of the same edge. The bandwidth of a graph is the minimum bandwidth over all linear arrangements of the graph.*

Definition 2. *A* caterpillar *is a tree composed of a path, called the* backbone*, and other paths, called* strands*, connected to the backbone. The connection point of a strand to the backbone is called the* root *of the strand. Hence all vertices of degree more than two are roots.*

Definition 3. *The* unfolded bandwidth *of a caterpillar is the minimum bandwidth in a linear arrangement that respects the order of the vertices in the backbone.*

Definition 4. *Let $B(v,r)$ denote the set $\{u \in V : d(u,v) \leq r\}$ where $d(\cdot,\cdot)$ denotes the usual shortest path distance in G. The* local density *of a graph ρ_G is defined as*

$$\rho_G = \max_{v\in V}\max_{r}\frac{|B(v,r)|}{2r}$$

It is easy to check that for any graph G, the local density ρ_G gives a lower bound on the bandwidth of G.

Given an algorithm A for approximating the bandwidth of caterpillars, for every caterpillar G we consider the following four quantities: its local density ρ_G; its bandwidth b_G; its unfolded bandwidth u_G; and the bandwidth of the linear arrangement found by algorithm A, denoted by A_G.

Clearly, $\rho_G \leq b_G \leq u_G \leq A_G$. Previously, it was known [3] that b_G can be as large as $\Omega(\rho_G \log n/\log\log n)$. Also, a simple approximation algorithm [9] obtained $A_G \leq O(\rho_G \log n)$. We present an approximation algorithm for which $A_G = O(\rho_G \log n/\log\log n)$, which is best possible (with respect to ρ_G). We also show that the gap of $\Omega(\log n/\log\log n)$ may be present between every two adjacent quantities in the above list. The gap of $\Omega(\log n/\log\log n)$ between b_G and u_G is especially important, because we show that u_G can in fact be approximated within a constant factor. Without this gap between u_G and b_G, this would improve the approximation ratio for the bandwidth of caterpillars beyond $\log n/\log\log n$.

In a result of a somewhat different nature, we present an algorithm that achieves a $(1+\epsilon)$ approximation ratio for the bandwidth of trees (and hence also caterpillars) in time $2^{\tilde{O}(\sqrt{n/\epsilon})}$. This result generalizes to families of graphs that have recursive decomposition with separators of size $\tilde{O}(\sqrt{n})$, including for example planar graphs. These results show in particular that in any reduction from 3SAT showing the hardness of approximating the bandwidth of caterpillars or trees, the number of vertices in the resulting graph will be at least quadratic in the size of the 3CNF formula, unless 3SAT has subexponential algorithms.

1.1 Related work

Chinn *et al.* [2] showed that trees with constant local density can have bandwidth as large as $\Omega(\log n)$. Chung and Seymour [3] exhibited a family of caterpillars with constant local density and bandwidth $\Omega(\frac{\log n}{\log\log n})$.

Monien [11] showed that the bandwidth problem on caterpillars is NP-hard. We note here that the reduction crucially uses a gadget that forces the bandwidth to be folded. Blache, Karpinski and Jürgen [1] showed that the bandwidth of trees is hard to approximate within some constant factor. Unger [14] claimed (without proof) that the bandwidth of caterpillars is hard to approximate within any constant factor.

Haralambides *et al.* [9] showed that for caterpillars, folding strands to one side is an $O(\log n)$-approximation with respect to the local density. For general graphs, Feige [5] gave a polynomial time $O(\log^{3.5}\sqrt{\log\log n})$ algorithm with respect to the local density lower bound (slightly improved in [10]). A somewhat improved approximation ratio of $O(\log^3 n\sqrt{\log\log n})$ with respect to a semidefinite programming lower bound was given by Dunagan and Vempala [4]. Gupta [8] gave an $O(\log^{2.5} n)$-approximation algorithm on trees, that on caterpillars gives an $O(\log n)$ approximation. Filmus [7] extended this $O(\log n)$-approximation to graphs formed by many caterpillars sharing a backbone vertex.

An exact algorithm for general graphs, running in time approximately n^B was given by Saxe [13] where B is the optimal bandwidth. Feige and Killian [6] gave a $2^{O(n)}$ time exact algorithm.

2 New Results

2.1 More definitions

Definition 5. *A* bucket arrangement *of a graph is a placement of its vertices into consecutive buckets, such that the endpoints of an edge are either in the same bucket or in adjacent buckets. The* bucketwidth *is the number of vertices in the most loaded bucket. The* bucketwidth *of a graph is the minimum bucketwidth of all bucket arrangements of the graph.*

The following lemma is well known.

Lemma 1. *For every graph, its bandwidth and bucketwidth differ by at most a factor of 2.*

Proof. Given a linear arrangement of bandwidth b, make every b consecutive vertices into a bucket. Given a bucket arrangement with bucketwidth b, create a linear arrangement bucket by bucket, where vertices in the same bucket are numbered in arbitrary order. □ □

As we shall be considering approximation ratios which are much worse than 2, we will consider bandwidth and bucketwidth interchangeably.

Definition 6. *The* unfolded bucketwidth *of a caterpillar is the minimum bucketwidth in a bucket arrangement in which every backbone vertex lies in a different bucket. Hence backbone vertices are placed in order.*

Lemma 2. *For every caterpillar, its unfolded bandwidth and unfolded bucketwidth differ by at most a factor of 2.*

Proof. Given an unfolded linear arrangement of bandwidth b, let every backbone vertex start a new bucket, called a backbone bucket. In regions with no backbone vertex (the leftmost and rightmost regions of the linear arrangement), make every b consecutive vertices into a bucket. It may happen that strands (say, k of them) jump over a backbone bucket in this bucket arrangement, but then this backbone bucket has load at most $b - k$. For each such strand, shift nodes so as to move one vertex from the outermost bucket of the strand to the separating backbone bucket. We omit the details from this extended abstract.

Given an unfolded bucket arrangement with bucketwidth b, create a linear arrangement bucket by bucket, where vertices in the same bucket are numbered in arbitrary order. □ □

2.2 Approximating unfolded Bandwidth

Theorem 3. *The unfolded bucketwidth of a caterpillar can be approximated within a constant factor.*

Proof. Up to a factor of two in the resulting bandwidth, we may assume that a strand is folded either to the right, or to the left (but not partly to the right and partly to the left), i.e. all buckets containing a node from a strand H lie on one side of the bucket containing its root. We call such an unfolded bucket arrangement *locally consistent.* We now formulate the problem of finding a locally consistent unfolded bucket arrangement of minimum bucketwidth as an integer program.

The variables of the integer program are of the form x_{ik}, where i specifies a vertex and k specifies a bucket number. Our intention is that $x_{ik} = 1$ if vertex i is in bucket k, and $x_{ik} = 0$ otherwise. Hence we shall have the *integrality constraints*:

$$x_{ij} \in \{0, 1\} \tag{1}$$

For concreteness, we assume that vertices of the caterpillar are named in the following order. First, the backbone vertices are named from left to right as 1 up to ℓ (where ℓ is the number of backbone vertices), and then the strands of the caterpillar are named one by one, where within a strand, vertices are named in order of increasing distance from the root of the strand. (This naming should not be confused with a linear arrangement. It is just a convention used in formulating the integer program.) We now place the vertices of the backbone in consecutive buckets. This gives the *backbone constraints*:

$$x_{ii} = 1 \quad \text{for all} \;\; 1 \leq i \leq \ell \tag{2}$$

Since vertices of strands might be placed in buckets to the left and to the right of the endpoints of the backbone, the parameter k in x_{ik} specifying the bucket number need not be limited to the range 1 up to ℓ, but will be allowed to range between $-n$ and n.

Now we are ready to present the most important set of constraints. They are more complicated than may appear necessary for the integer program, but this will become useful once we relax the integer program to a polynomial time solvable linear program.

Consider two vertices $i, i+1$ on the same strand, rooted at the backbone vertex r. Then we have the *root constraints*:

$$x_{(i+1)r} \leq x_{ir} \tag{3}$$

Moreover, let k be a bucket to the right of bucket r, namely, $k > r$. (Later we will present analogous constraints for $k < r$.) For every $t \geq k$ we have the *right consistency constraints*:

$$\sum_{l=k}^{t} x_{(i+1)l} \leq \sum_{l=k-1}^{t} x_{il} \quad \text{and} \quad \sum_{l=k}^{t} x_{il} \leq \sum_{l=k}^{t+1} x_{(i+1)l} \tag{4}$$

The consistency constraints indicate that $(i+1)$ appears in a bucket to the right of r only if i preceded it, and that i appears only if $(i+1)$ follows it.

Similar constraints are written for buckets to the left of the root. Namely, for $k < r$ and $t \leq k$ we have the *left consistency constraints*:

$$\sum_{l=t}^{k} x_{(i+1)l} \leq \sum_{l=t}^{k+1} x_{il} \quad \text{and} \quad \sum_{l=t}^{k} x_{il} \leq \sum_{l=t-1}^{k} x_{(i+1)l} \tag{5}$$

Finally, we introduce a constraint specifying that the locally consistent unfolded bucketwidth is at most b. Namely, for every bucket k there is the *bucketwidth constraint*:

$$\sum x_{ik} \leq b \tag{6}$$

The above integer program is feasible if and only if there is a locally consistent bucket arrangement of bucketwidth at most b. We now relax the integer program to a linear program by replacing the integrality constraints by nonnegativity constraints $x_{ik} \geq 0$ and *choice constraints*, namely, for every vertex i on a strand of length t rooted at r we have:

$$\sum_{k=r-t}^{r+t} x_{ik} = 1 \tag{7}$$

As the linear program is a relaxation of the integer program, it is feasible whenever the integer program is. As linear programs can be solved in polynomial time, we can obtain a feasible solution to the linear program of bucketwidth at most b^*, where b^* is the minimum locally consistent unfolded bucketwidth. The solution to the linear program is fractional, in the sense the a vertex may fractionally belong to several buckets. We now show how to round the fractional solution to an integer solution, loosing only a constant factor in the bucketwidth.

Consider an arbitrary solution x_{ik} to the linear program. Consider a strand S rooted at r. Consider the smallest $k > r$ such that $\sum_{i \in S} x_{ik} \leq 1/4$. We claim that $\sum_{i \in S, k' \geq k} x_{ik'} \leq |S|/4$. This claim follows from the following inequality:

$$\sum_{k'>k} x_{ik'} \leq \sum_{j \in S; j<i} x_{jk}$$

This inequality can be proved by induction on i. For the first vertex on the strand, this inequality is true because then $\sum_{k'>k} x_{ik} = 0$. Assume now that the inequality was proved for vertex i and then the right consistency constraints and the inductive hypothesis imply for vertex $i+1$ that:

$$\sum_{k'>k} x_{(i+1)k'} \leq \sum_{k' \geq k} x_{ik'} \leq x_{ik} + \sum_{j \in S; j<i} x_{jk} = \sum_{j \in S; j<i+1} x_{jk}$$

Similar to the above, consider the largest $k < r$ with $\sum_{i \in S} x_{ik} \leq 1/4$. It can be shown that $\sum_{i \in S, k' \leq k} x_{ik'} \leq |S|/4$.

It follows that there is some k (w.l.o.g. we will assume here that $k > r$) such that $\sum_{i \in S; r \leq k' < k} x_{ik'}$ is at least $|S|/4$, and $\sum_{i \in S} x_{ik'} \geq 1/4$ for all $r \leq k' < k$. Now place the vertices of S one by one in the buckets starting at r and continuing to the right, putting $\lceil 4 \sum_{i \in S} x_{ik'} \rceil$ vertices in bucket k'. The whole strand can be put in these buckets before reaching bucket k. Each bucket suffered a multiplicative factor of at most 8 (rounding up to the nearest integer contributes a factor of at most 2, because $4 \sum_{i \in S} x_{ik'} \geq 1$).

Finally, there is the following issue that we ignored in the proof above, and this is the fact that for the root bucket r, it might not be the case that $\sum_{i \in S} x_{ir} \geq 1/4$. In that case, the integer solution might put a vertex of S in bucket r that we cannot charge against the fractional values of S that were in bucket r. However, this cannot harm the approximation ratio by more than a constant factor, because the number of vertices added by this effect to bucket r cannot be more than twice the local density at r. □ □

2.3 Upper Bound

In this section, we present an algorithm for the bandwidth problem on caterpillars and show that it is an $O(\frac{\log n}{\log \log n})$ approximation.

We shall give a bucket arrangement of caterpillar T. More precisely, we shall define maps $x : V \to [n]$ and $y : V \to [n]$ such that

- The function $f : V \to [n] \times [n]$ defined as $f(v) = (x(v), y(v))$ is one-one.
- For every edge $(u, v) \in E$, $|x(u) - x(v)| \leq 1$.

In other words, we shall place the vertices of the graph on the integer grid so that adjacent vertices land either in the same column or in adjacent columns. The goal would be to minimize the maximum height of any column.

Let T be a caterpillar with backbone $B = \{1, \ldots, p\}$ and strands $\mathcal{H} = \{H_1, \ldots, H_q\}$, where each H_i can be specified by its root $r_i \in B$ and length l_i. Let $k = 2\frac{\log n}{\log\log n}$ so that $k^k > n$. For an assignment f and for a strand H_i let the *height* α_i of H_i be $\max_{v \in H_i} y(v)$ and let the *range* β_i of H_i be $\max_{v \in H_i} |x(v) - x(r_i)|$.

The algorithm proceeds as follows. Let $\mathcal{H}_s = \{H_i \in \mathcal{H} : k^{s-1} \leq l_i < k^s\}$ (note that $\mathcal{H}_k$ is empty). We start out by setting $f(j) = (j, 1)$ for $j \in B$. The algorithm has k phases. In phase s, we consider the strand $H_i \in \mathcal{H}_s$ in increasing order of r_i (breaking ties arbitrarily). We fold each strand to its right so as to minimize the maximum height of any column, and subject to that, minimize its range. Amongst the arrangements minimizing the height and the range, we break ties to the left, i.e. fill up columns from left to right.

We now show that the mapping f satisfies $\mathtt{bw}_f \leq O(\frac{\log n}{\log\log n}\rho_G)$. We denote by $f^{-1}(X, Y) = \{v : f(v) = (x, y), x \in X, y \in Y\}$ the set of nodes assigned to columns X and rows Y.

Let I_t denote the interval $[4(k + t - 1)\rho_G + 1, 4(k + t)\rho_G]$. For a column j, let $\mathcal{H}_t^j = \{H_i \in \mathcal{H}_t : H_i \cap f^{-1}(\{j\}, I_t) \neq \phi\}$ be the set of hair in $\mathcal{H}_t$ that contribute to $f^{-1}(\{j\}, I_t)$.

The tie breaking rule ensures the following.

Observation 1. *Just after H_i is assigned, if H_i contributes to columns j and j', $j < j'$, then the height of j' is no larger than that of j.*

Next we show a few simple lemmas

Lemma 4. *Every strand $H_i \in \mathcal{H}_t^j$ has range β_i at most k^{t-1}.*

Proof. For every $H_i \in \mathcal{H}_t^j$, it must be the case that for all $j' : r_i \leq j' \leq j$, the column j' has height at least $4k\rho_G$ after H_i was considered. Consider the set of vertices $f^{-1}([r_i, j], [1, 4k\rho_G])$ assigned to rows r_i through j and columns 1 through $4k\rho_G$ in this partial assignment. Because of the order in which we consider strands, each of these vertices comes from a strand of length less than k^t and the hence the root of every such strand must lie in $[r_i - k^t, j]$. Thus every such vertex lies in $B(r_i, 2k^t)$ and there are at least $2k\rho_G|j - r_i| = 2k\rho_G\beta_i$ such vertices. Since $|B(r_i, 2k^t)| \leq 4k^t\rho_G$, it follows that $\beta_i \leq k^{t-1}$ for every $H_i \in \mathcal{H}_t^j$. □ □

Corollary 5. *For every j, t, $|\mathcal{H}_t^j| < 4\rho_G$.*

Proof. Every strand $H_i \in \mathcal{H}_t^j$ has its root in $[j - k^{t-1}, j]$. Since each has length at least k^{t-1}, $B(j, 2k^{t-1})$ has size at least $(1 + |\mathcal{H}_t^j|k^{t-1})$. Hence, $|\mathcal{H}_t^j| < 4\rho_G$. □ □

Lemma 6. *At the end of phase s of the algorithm, the height of (partial) assignment is at most $4(k + s)\rho_G$.*

Proof. We show the claim by induction on s. In the base case $s = 0$, each column has one node and since ρ_G is at least 1, the claim holds.

Suppose the claim is true at end of phase $(t-1)$. We wish to show that it is also true at the end of phase t.

Consider column j and let $\mathcal{H}_t^j$ be as above. We first argue that each strand in $\mathcal{H}_t^j$ contributes at most one vertex to column j. Assume the contrary and let H_i be the first strand in $\mathcal{H}_t^j$ that contributes at least 2 vertices to column j. Further, let j be the first such column. Then there is a column $j' > j$ such that just before H_i was assigned, j' had height at least one more than that of j. Since we look at strand in each class from left to right, every strand contributing to $f^{-1}(j', I_t)$ at this point has its root to the left of j. Thus by observation 1, j' has height no larger than j, contradicting the assumption. Hence the claim.

By corollary 5, $\mathcal{H}_t^j$ has fewer than $4\rho_G$ strands and hence the induction holds. The claim follows. □ □

We remark that we have actually shown an $O(\log h/\log\log h)$-approximation, where h is the longest strand length. Note that our algorithm outputs an unfolded arrangement. We show in the next section that it is not possible to beat the $O(\frac{\log n}{\log\log n})$ barrier when outputting an unfolded arrangement.

Finally, we note while this algorithm never folds the backbone, it could be far from optimal for the unfolded bandwidth problem.

Proposition 7. *The algorithm of the above theorem may output a linear arrangement with bandwidth $\Omega(\log n/\log\log n)$ times the unfolded bandwidth.*

Proof. The instance has a backbone of length $2^{k+1}-1$ and a strand of length $i2^i$ at vertex 2^i and vertex $2^{k+1}-2^i$. Folding (one vertex per bucket) the first half of the strands to the right and the the second half of the strands to the left gives an arrangement with unfolded bucketwidth $O(\log k)$. On the other hand, our algorithm (in fact any algorithm folding all strands to the right) gives a bucketwidth of k. Since k is $O(\log n)$, the claim follows. □ □

2.4 Gap between bandwidth and unfolded bandwidth

We construct a sequence of caterpillars $C_1, C_2, \ldots$ such that C_k has bucketwidth $O(k)$ and unfolded bucketwidth $\Omega(k^2)$.

C_1 is a caterpillar with a backbone of length three and p strands of length 1 attached to the middle node. The first and last nodes on the backbone are designated s_1 and t_1 respectively (see figure 1).

C_k is constructed by joining in series, a path P_1 of length $(l_k + w_k)$, a copy T_1 of C_{k-1}, a path P_2 of length $2w_k$, another copy T_2 of C_{k-1} and finally a path P_3 of length $(l_k + w_k)$. Moreover, p strands each of length l_k are attached at the first and last vertices of P_2 (see figure 1). The first vertex of P_1 and the last vertex of P_3 are named s_k and t_k respectively. We refer to the backbone vertices in T_1, P_2 and T_2 as the *core* of C_k. The strands attached to P_2 are referred to as *central* strands.

Here w_k and l_k are parameters that we define as follows: $l_1 = 1$,$w_1 = 0$, $w_k = 2l_{k-1} + 2w_{k-1}$, $l_k = 6kl_{k-1}$. Note that the length of the backbone satisfies the relation $B_k = 4w_k + 2l_k + 2B_{k-1}$. It follows that

Observation 2. *The length of the backbone of C_k is at most $B_k \leq 4l_k$.*

We first show that there is an arrangement with small bucketwidth.

Lemma 8. *There is a valid bucket arrangement of C_k with bucketwidth $p+2k$ into at most $w_{k+1} = 2l_k + 2w_k$ buckets with s_k and t_k in the first and last buckets respectively.*

Proof. The proof uses induction. The base case is immediate.

Suppose that there is such a bucket arrangement for C_{k-1}. We use it to construct a bucket arrangement for C_k (see figure 2). We place the first path P_1 into buckets 1 through $(l_k + w_k)$. The recursive arrangement of T_1 is placed in buckets $(l_k + w_k + 1)$ through $(l_k + 2w_k)$. We place the path P_2 in buckets $(l_k + 2w_k)$ to $(l_k + 1)$. T_2 is placed similarly in buckets $(l_k + 1)$ to $(l_k + w_k)$ and we place P_3 starting at $(l_k + w_k + 1)$ and ending at bucket $2(l_k + w_k)$. Each central strand rooted at the endpoint of P_2 in bucket $(l_k + 2w_k)$ spans buckets $(l_k + 2w_k + 1)$ through $2(l_k + w_k)$ and each central strand rooted at the endpoint in bucket $l_k + 1$ goes into buckets l_k through 1. It is easy to verify that this is a legal bucket arrangement. Moreover, the maximum height of any bucket increases by at most 2, and hence the induction holds. □ □

We now show that the unfolded bucketwidth of C_k is large. Consider an unfolded bucket arrangement f. Without loss of generality, s_k falls to the left of t_k. A bucket is called a *core* bucket if it contains a vertex from the core. Other buckets are referred to as *peripheral.* Note that the core buckets fall between the left set and the right set of peripheral buckets.

Lemma 9. *In any unfolded bucket arrangement of C_k, some core bucket has load more than $\frac{pk}{3}$.*

Proof. We prove this by induction on k

In the base case $k = 1$, we simply use a local density argument. There are $p+3$ nodes and since the diameter is 2, they all fall in at most 3 buckets. The claim follows.

Suppose the claim holds for C_{k-1} and let f be an unfolded bucket arrangement for C_k. Suppose that f has bucketwidth less than $\frac{pk}{3}$. Consider the $2p$ central strands of G_k.

Proposition 10. *In any arrangement with bucketwidth less than $(\frac{pk}{3})$ at least $(\frac{2p}{3})$ central strands extend to peripheral buckets.*

Proof. Suppose not. Then at least $\frac{4p}{3}$ central strands are wholly contained in the core buckets. The number of core bucket is at most $2(w_k + B_{k-1})$ and each has at most $\frac{pk}{3}$ nodes in the arrangement f. However, since each strand has length l_k and $(\frac{4p}{3})l_k > 2(w_k + B_{k-1})(\frac{kp}{3})$, we get a contradiction. □ □

Thus $\frac{2p}{3}$ strands extend to peripheral buckets. Without loss of generality, at least $(\frac{p}{3})$ of these strands extend to the left set of peripheral buckets. Each of these strands contributes at least 1 to the buckets containing the backbone of T_1. Since f also induces an unfolded bucket arrangement of T_1, the induction hypothesis implies that one of the core buckets of T_1 must have load at least $\frac{p(k-1)}{3}$ from vertices in T_1. Adding the load due to the $\frac{p}{3}$ crossing strands, we get an overall load of at least $\frac{pk}{3}$. The claim follows. □ □

Taking p to be k, we get a gap of $\Omega(k)$. Since $l_k \leq 6^k k!$, the number of vertices in C_k is $2^{O(k \log k)}$. Hence we get a gap of $\Omega(\frac{\log n}{\log \log n})$. Since (unfolded) bucketwidth and (unfolded) bandwidth are within a constant factor of each other, the gap between unfolded bandwidth and bandwidth is also $\Omega(\frac{\log n}{\log \log n})$.

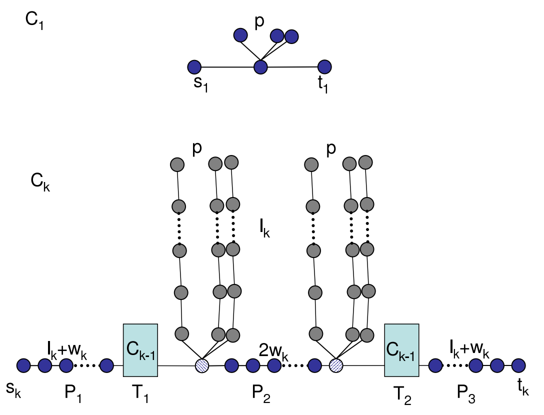

Figure 1: Construction of caterpillar C_k

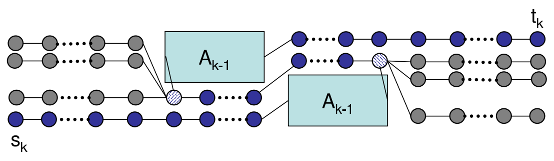

Figure 2: Low bucketwidth arrangement of caterpillar C_k

3 An approximation scheme

In this section, we give an approximation scheme for the bandwidth problem on trees that gives a $(1+\epsilon)$-approximation in time $2^{\tilde{O}(\sqrt{\frac{n}{\epsilon}})}$. We note that the algorithm described here is more complicated than needed; the extra work here enables us to extend the algorithm to a larger class of graphs without any modifications.

We first guess the bandwidth B. If $B \leq \sqrt{\frac{n}{\epsilon}}$, the dynamic programming algorithm of Saxe [13] finds the optimal bandwidth arrangement in time $n^{O(B))}$ which is $2^{\tilde{O}(\sqrt{\frac{n}{\epsilon}})}$ for B as above.

In case $B \geq \sqrt{\frac{n}{\epsilon}}$, we run a different algorithm. We shall construct an assignment f of V into K buckets $1, \ldots, K$ such that

- For any bucket i, the number of nodes assigned to it $|f^{-1}(i)|$ is at most ϵB.
- For any $u, v \in V$ such that $(u, v) \in E$, $f(u)$ and $f(v)$ differ by at most $\frac{1}{\epsilon}$.

It is easy to see that if G has bandwidth B, such an assignment always exists for $K = (\frac{n}{\epsilon B})$. Moreover, given such as assignment, we can find a bandwidth $(1+\epsilon)B$ arrangement of G by picking any ordering that respects the ordering defined by f.

Note that any tree T has a vertex v_T such that deleting v from T breaks it up in components of size at most $\frac{2n}{3}$; we call such a v a *balanced separator* of T. We recurse on each component until each component is a leaf. This sequence of decompositions can be represented in the form of a rooted *decomposition tree* τ with the following properties:

- Each node x of the decomposition tree τ corresponds to a subtree T_x and a balanced separator v_x of T_x.

- The children of an internal node x correspond exactly to the components resulting from deleting v_x from the tree T_x.
- The root node r corresponds to T and the leaves correspond to singletons.
- The depth of the tree is $O(\log n)$.

A partial assignment is a partial function f that maps a subset $D_f \subseteq V$ into K buckets $1, \ldots, K$. We say f is feasible if for every $u, v \in D_f$, $(u, v) \in E$, $f(u)$ and $f(v)$ differ by at most $\frac{1}{\epsilon}$. We say g *extends* f to D' if $D_g = D_f \cup D'$ and f agrees with g on D_f. Given a partial assignment f, its *profile* p_f is defined by the K-tuple $(|f^{-1}(1)|, |f^{-1}(2)|, \ldots, |f^{-1}(K)|)$. For the purposes of our algorithm two partial assignments are considered equivalent if they have the same profile.

Our algorithm does dynamic programming on the decomposition tree τ of T. Let x be an internal node of τ with children $y_1, \ldots, y_k$. Let $f_x^1, \ldots, f_x^{n_x}$ be the set of all feasible partial assignments with domain consisting of all the separator nodes on the r-x path in τ, i.e. f_x^a has domain $P_x = \{v_z : z \text{ is on } r\text{-}x \text{ path in } \tau\}$. Given x, $a \le n_x$ and $j \le k$, we compute the list $L_{x,j}^a$ of all possible profiles of a feasible extension g of f_x^a to $\cup_{i \le j} T_{y_i}$. We use the notation L_x^a for $L_{x,k}^a$ if x has k children in τ.

We populate the dynamic programming table from the leaves moving up the tree as follows. We start with the obvious setting of L_x^a for each leaf x of τ, for each $a \in [n_x]$. For an internal node x, clearly $L_{x,0}^a$ is just the profile of f_x^a. Given $L_{x,(j-1)}^a$ and $L_{y_j}^b$ for every feasible extension $f_{y_j}^b$ of f_x^a to v_{y_j}, we show how to compute $L_{x,j}^a$. The crucial fact here, ensured by our construction of the decomposition tree, is that all the edges leaving T_{y_j} are incident on P_x. Thus given a feasible extension g_1 of f_x^a to $\cup_{i<j} T_{y_i}$ and another feasible extension g_2 of f_x^a to T_{y_j}, they can be combined to give a feasible extension g_3 of f_x^a to $\cup_{i \le j} T_{y_i}$. Thus for every profile p_1 in $L_{x,(j-1)}^a$ and for every profile p_2 in any of the lists $L_{y_j}^b$ where $f_{y_j}^b$ is a feasible extension of f_x^a, we get a profile p_3 to be placed in $L_{x,j}^a$.

Finally, if T has bandwidth B, at least one of the lists L_r^a contains a profile p_f with $|f^{-1}(i)| \le \epsilon B$ for each i.

It remains to bound the running time of our algorithm. Since the depth of the tree is $O(\log n)$, for every node x in τ, the number of partial assignments n_x is at most $K^{O(\log n)}$. The number of nodes in τ is clearly at most n and hence the algorithm only need compute $nK^{O(\log n)}$ table entries. The size of each list $L_{x,j}^a$ is bounded by the total number of possible profiles, which is no more than n^K. Each entry of the table can thus be computed in time $O(n^{2K})$. Thus the overall running time of the algorithm is $2^{O(K \log n)}$. Since $K = \frac{n}{\epsilon B}$ and $B \ge \sqrt{\frac{n}{\epsilon}}$, the running time of our algorithm is $2^{\tilde{O}(\sqrt{\frac{n}{\epsilon}})}$. Thus we have shown that

Theorem 11. *The algorithm described above computes a $(1+\epsilon)$ approximation to the bandwidth of a tree in time $2^{\tilde{O}(\sqrt{\frac{n}{\epsilon}})}$.*

We note that it is easy to modify the algorithm to also give an assignment having such a profile.

Moreover, note that the only property of the tree we have used is the existence of small separators. The algorithm can be naturally modified to work with any family of graphs with (recursive) small separators—the number of partial assignments to be considered for a table entry now goes up $K^{O(t \log n)}$ where t is an upper bound on the size of the separator. This gives a $2^{\tilde{O}(t+\sqrt{\frac{n}{\epsilon}})}$ time $(1+\epsilon)$-approximation algorithm for graphs with tree-width t. Recall that planar graphs and other excluded minor families of graphs have separators of size $O(\sqrt{n})$. Thus the running time of our algorithm for such graphs is $2^{\tilde{O}(\sqrt{\frac{n}{\epsilon}})}$.

Acknowledgements

We thank Moses Charikar, Marek Karpinski and Ryan Williams.

References

[1] G. Blache, M. Karpinski, and J. Wirtgen. On approximation intractability of the bandwidth problem. Technical report, University of Bonn, 1997.

[2] P. Chinn, J. Chvatálová, A. Dewdney, and N. Gibbs. The bandwidth problem for graphs and matrices - survey. *Journal of Graph Theory*, 6(3):223–254, 1982.

[3] F. R. Chung and P. D. Seymour. Graphs with small bandwidth and cutwidth. *Discrete Mathematics*, 75:113–119, 1989.

[4] J. Dunagan and S. Vempala. On euclidean embeddings and bandwidth minimization. In *APPROX '01/RANDOM '01*, pages 229–240, 2001.

[5] U. Feige. Approximating the bandwidth via volume respecting embeddings. *J. Comput. Syst. Sci.*, 60(3):510–539, 2000.

[6] U. Feige. Coping with the NP-hardness of the graph bandwidth problem. In *SWAT*, pages 10–19, 2000.

[7] Y. Filmus. Master's thesis, Weizmann Institute, 2003.

[8] A. Gupta. Improved bandwidth approximation for trees. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 788–793, 2000.

[9] J. Haralambides, F. Makedon, and B. Monien. Bandwidth minimization: An approximation algorithm for caterpillars. *Mathematical Systems Theory*, pages 169–177, 1991.

[10] R. Krauthgamer, J. Lee, M. Mendel, and A. Naor. Measured descent: A new embedding method for finite metrics. In *FOCS*, pages 434–443, 2004.

[11] B. Monien. The bandwidth minimization problem for caterpillars with hair length 3 is NP-complete. *SIAM J. Algebraic Discrete Methods*, 7(4):505–512, 1986.

[12] C. Papadimitriou. The NP-completeness of the bandwidth minimization problem. *Computing*, 16:263–270, 1976.

[13] J. Saxe. Dynamic-programming algorithms for recognizing small-bandwidth graphs in polynomial time. *SIAM J. Algebraic Discrete Methods*, 1:363–369, 1980.

[14] W. Unger. The complexity of the approximation of the bandwidth problem. In *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, pages 82–91, 1998.