# Bandwidth Approximation of a Restricted Family of Trees

**Abstract**

Bandwidth is one of the canonical NP-complete problems [17]. It is NP-hard to approximate within any constant, even on trees [22].

Gupta gave a randomized approximation algorithm [8] for bandwidth on trees, which has an approximation ratio of $O(\log^{2.5} n)$. This algorithm has the best currently known approximation ratio on trees.

Gupta showed that his algorithm has an approximation ratio of $O(\log n)$ on caterpillars, a restricted family of trees. We show that Gupta's algorithm has an approximation ratio of $O(\log n)$ on many-caterpillars, an extension of caterpillars. We also simplify and derandomize Gupta's algorithm on many-caterpillars.

## 1 Introduction

Bandwidth is one of the well-known NP-complete problems on graphs. Given an undirected graph $G = (V, E)$, an ordering $f$ is a one-to-one mapping from $V$ to $\{1, \ldots, n\}$, where $n = |V|$. The stretch of the edge $(x, y) \in E$ in the ordering $f$ is $|f(x) - f(y)|$. The bandwidth of the ordering $f$ is $B(f) = \max_{(x,y)\in E} |f(x) - f(y)|$. Finally, the bandwidth of $G$ is $B(G) = \min_f B(f)$, where $f$ goes over all possible orderings of $G$. For example, the bandwidth of a path is 1, as figure 1 shows.

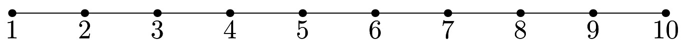

Figure 1: An optimal ordering of $P_{10}$.

The bandwidth of a cycle is 2, as figures 2 and 3 demonstrate.

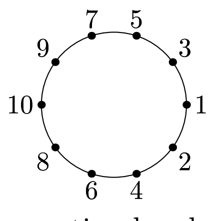

Figure 2: An optimal ordering of $C_{10}$.

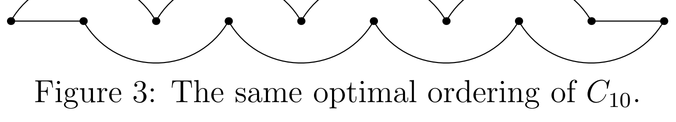

Figure 3: The same optimal ordering of $C_{10}$.

The bandwidth of a complete $k$-ary tree with $n$ vertices is $\Theta(n/\log_k n)$. The exact value was found by Smithline [19], who also gave an algorithm to construct an optimal ordering. See also [11] for a discussion about full binary trees.

Bandwidth arises in several contexts. According to [12], the problem originated in the 1960s in JPL (the Jet Propulsion Laboratory) in the context of coding (see [12] for references). A canonical application of bandwidth is in linear algebra. A sparse matrix can be stored and processed more efficiently if its non-zero entries are concentrated in a narrow band around the diagonal (hence the name bandwidth). Given a sparse symmetric matrix $A_{n\times n}$, one seeks a permutation matrix $P$ that minimizes the bandwidth of $PAP^{-1}$. This is equivalent to finding an optimal ordering of the graph $G$ whose vertices are $\{1, \ldots, n\}$, and $(i, j)$ is an edge whenever $A_{ij} \neq 0$. For details see [3].

Bandwidth generalizes in several ways. It can be generalized to directed acyclic graphs by requiring the ordering to be topological (see [7]). In section 2.3 we describe bandwidth as an embedding problem. In technical terms, bandwidth is the minimal dilation of an embedding of a graph into the infinite path. In VLSI layout, the target graph is a square mesh representing an integrated circuit [21].

Bandwidth is one of several layout problems. One of them, minimum linear arrangement, also originated in JPL, according to [12]. The problem is to find an ordering of the vertices of a graph that minimizes the sum of the stretches of all edges. Minimum linear arrangement can also be seen as an embedding problem, and is useful in the area of VLSI.

## 1.1 NP-hardness results

Saxe's algorithm [18] decides whether $B \leq k$ in time $O(n^k)$, using dynamic programming. Another algorithm [7] tests whether $B \leq 2$ in linear time. Both algorithms also output an optimal ordering of the graph.

Bandwidth is NP-complete [17]. Contrary to many other NP-complete problems, bandwidth remains NP-complete even when restricted to trees. In fact, bandwidth is NP-complete [7] on trees with maximum degree 3. Moreover, bandwidth remains NP-complete [16] when restricted to caterpillars with maximum degree 3 or maximum hair length 3 (caterpillars are described later in the introduction).

Bandwidth is not only hard to compute exactly, but also hard to approximate. A reduction from 3SAT shows [1] that it is NP-hard to approximate bandwidth to within $\frac{3}{2}$ on general graphs, and to within $\frac{4}{3}$ on trees (the difference results from replacing cliques by stars). Unger [22] has recently published an extended abstract claiming that it is NP-hard to approximate bandwidth within any constant factor, even on caterpillars with maximum degree 3.

Unger [22] further claims that for every $x > 1$ and $\epsilon > 0$ there is a family of graphs on which bandwidth is approximable within $x + \epsilon$, yet NP-hard to approximate within $x - \epsilon$. Examples of such families for $x = 2$ are unit circular-arc graphs (intersection graphs of a set of unit arcs on a circle), and two restricted families of ringed caterpillars (caterpillars whose spines are completed to cycles).

## 1.2 Approximation algorithms

Several approximation algorithms have been proposed for bandwidth, most of them randomized. Many of the algorithms express their performance in terms of the local density, which is the maximum of the ratio $(|G'| - 1)/\operatorname{diam}(G')$ over all connected subgraphs $G'$ of the given graph. The local density of a graph $G$ is customarily denoted $D(G)$. In section 2.2 there is a related notion of graphical density.

Local density serves as a lower bound to the bandwidth. To see this, suppose a graph $G$ has a subgraph $G'$ consisting of $n' + 1$ vertices and having diameter $d'$ . Consider an optimal ordering $f$ of $G$. Let $v_l$ be the leftmost vertex of $G'$ in this ordering, and let $v_r$ be the rightmost. On the one hand, $|f(v_r) - f(v_l)| \geq n'$, since $G'$ has $n' + 1$ vertices. On the other hand, $|f(v_r) - f(v_l)| \leq d'B(G)$, since the distance between $v_l$ and $v_r$ is at most $d'$. It follows that $B(G) \geq n'/d' = (|G'| - 1)/\operatorname{diam}(G')$, hence $B(G) \geq D(G)$.

A constant factor approximation algorithm for dense graphs is given in [12]. The randomized algorithm produces w.h.p. (with high probability) an ordering with bandwidth at most $3B(G)$ in time $O(n^{1/\delta})$, where the minimum degree of $G$ is at least $\delta n$. Another constant factor approximation algorithm [13] works on asteroidal triple-free graphs (see [13] for definition). In view of Unger's results [22], we should not expect such an algorithm for general graphs, or even for trees.

The first polylogarithmic approximation algorithm for bandwidth on general graphs was given by Feige [6]. The randomized algorithm uses volume respecting embeddings, inspired by [14]. First the graph is embedded into a high dimensional Euclidean space, then it is projected on a random line through the origin. The algorithm outputs the ordering induced by this line. The bandwidth of the resulting ordering is w.h.p. $O(D \log^{3.5} n \sqrt{\log \log n})$.

Around the same time, a different approximation algorithm was proposed in [2]. The randomized algorithm uses a semi-definite program to embed the graph into a high dimensional Euclidean space. The result is projected on a random line through the origin, resulting in an ordering whose bandwidth is w.h.p. $O(\sqrt{nB}\log n)$.

The two approaches were combined by Vempala [23, 5] to give the best approximation algorithm currently known for general graphs. Vempala's randomized algorithm first uses the semi-definite program of [2], then embeds the result using a volume respecting embedding [6], and finally performs a projection onto a random line through the origin. With high probability, the resulting ordering has bandwidth $O(B\log^3 n\sqrt{\log\log n})$.

Vempala's algorithm [23] also approximates the minimum linear arrangement. Furthermore, it generalizes to produce embeddings into $d$-dimensional infinite grids instead of the 1-dimensional infinite path.

Approximation algorithms with better approximation ratios are known for trees. An extremely simple approximation algorithm [10] approximates bandwidth on caterpillars. Caterpillars are trees formed by picking a path called the spine, and attaching to it other paths called hairs. All hairs emanate from the spine. See figure 6 for an example. The algorithm produces an ordering by sorting the vertices with respect to their depths (distances from the root). The resulting ordering has bandwidth $O(D\log n)$. See section 3.1 for a description and analysis of this algorithm.

A more complicated algorithm [20, 9] approximates bandwidth on a type of trees called height-balanced trees (see [9] for a definition). The algorithm is greedy, and it outputs an ordering with bandwidth $O(D\log n)$.

Gupta's algorithm [8] is a polylogarithmic approximation algorithm for general trees. The randomized algorithm uses the concept of caterpillar decomposition. A caterpillar decomposition of a rooted tree is a decomposition of the tree into edge-disjoint paths. The decomposition has dimension $\kappa$ if each route from the root to a leaf is composed of at most $\kappa$ different paths. The dimension of the lowest-dimensional caterpillar decomposition of a tree $T$ is called its caterpillar dimension $\kappa(T)$. The caterpillar dimension, along with an optimal decomposition, can be found using dynamic programming in polynomial time [15]. It turns out [15] that $\kappa \leq \log n$. Gupta's algorithm produces w.h.p. an ordering with bandwidth $O(D\log^2 n\sqrt{\kappa}) = O(D\log^{2.5} n)$.

Gupta's algorithm is easy to describe, though harder to analyze. Given an optimal caterpillar decomposition, stretch each path independently by a random factor between 1 and 2. Sort the vertices with respect to their (stretched) distances from the root, and output the resulting ordering. On caterpillars, Gupta's algorithm is nearly equivalent to the one in [10], hence (as Gupta showed) it produces an ordering with bandwidth $O(D\log n)$, independently of the stretching.

## 1.3 Our contribution

The main result in this paper is the following theorem.

**Theorem 1.** *There is a deterministic polynomial-time algorithm (described in section 4.1) that, given a many-caterpillar $T$ with density $D$ and height $h$, constructs an ordering of $T$ with bandwidth $O(D \log h)$.*

Theorem 1 is a reformulation of theorem 7 in the main body. Here we explain the context in which theorem 1 was proved.

We tried to improve the analysis of Gupta's algorithm [8], suspecting that the algorithm produces an ordering with bandwidth $O(D \log n)$ on all trees.

**Conjecture 1.** *Given a tree $T$ with $n$ vertices, Gupta's algorithm [8] produces an ordering of $T$ with bandwidth $O(D(T) \log n)$.*

To describe our approach, we define two notions. For any $\kappa$, a tree $T$ for which $\kappa(T) \leq \kappa$ is called a $\kappa$-tree. A $\kappa$-tree whose root has degree 1 is called a $\kappa$-caterpillar. Using this terminology, caterpillars can be described as 2-caterpillars. General 2-trees are given the name 'many-caterpillars'. For an example of a many-caterpillar which is not a caterpillar, see figure 7.

Our plan was to prove conjecture 1 by induction on the caterpillar dimension of the tree. The induction base is either paths (1-caterpillars) or many-paths (1-trees). Many-paths can be described as stars with extended paths, see figure 4. It is easy to see that given a many-path, Gupta's algorithm outputs an ordering with bandwidth $O(D)$.

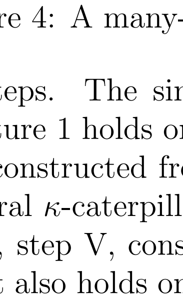
Figure 4: A many-path

There are two induction steps. The simpler step, step H, consists of showing that given that conjecture 1 holds on $\kappa$-caterpillars, it also holds on $\kappa$-trees. Note that $\kappa$-trees are constructed from $\kappa$-caterpillars 'horizontally', by identifying the roots of several $\kappa$-caterpillars.

The more complicated step, step V, consists of showing that given that conjecture 1 holds on $\kappa$-trees, it also holds on $(\kappa+1)$-caterpillars. Note that

$(\kappa+1)$-caterpillars are constructed from $\kappa$-trees 'vertically', by letting several $\kappa$-trees emanate from the vertices of a path, the spine.

When trying to execute our plan, we had to strengthen the induction hypothesis (see later for a formulation). Moreover, we could only prove induction step H. We then proved the strengthened induction hypothesis for caterpillars, using a regularization process. As a result, we have shown that Gupta's algorithm outputs an ordering with bandwidth $O(D \log n)$ on many-caterpillars.

The strengthened induction hypothesis implies the stronger conclusion that Gupta's algorithm outputs an ordering with bandwidth $O(D \log h)$ on a many-caterpillar with height $h$. This conclusion is the best possible in the following sense. For each $D > 1$ there exists a family of caterpillars (constructed in section 3.1 under the name 'triangle caterpillars') of density $\Theta(D)$, for which Gupta's algorithm outputs an ordering with bandwidth $\Omega(D \log h)$. Note, however, that the bandwidth of a triangle caterpillar $T$ is $\Theta(D(T))$, and an optimal ordering can be constructed efficiently.

Having restricted our scope to many-caterpillars, we simplified Gupta's algorithm and derandomized it (see section 4.1).

The proof of theorem 1 can be extended to trees which are not many-caterpillars, yet have similar structure. In particular, theorem 1 extends to the trees used in [1] to show that bandwidth is hard to approximate within $\frac{4}{3}$.

Our framework is summarized by the following three theorems.

**Definition 1** (Layout Function)**.** *Let $T = (V, E)$ be a tree of height $h$. A layout function of $T$ is any function $f\colon V \to [1, 2h]$ satisfying $|f(x) - f(y)| \le 2$ for each edge $(x, y) \in E$. Note $[1, 2h]$ denotes a closed interval on the real line.*

**Definition 2** (Property Good)**.** *Let $T$ be a tree of height $h$. The tree $T$ has property Good if there exists a constant $c$ and a distribution $\mathcal{F}(T)$ of layout functions of $T$ satisfying the following property. For every integers $k \ge 1$ and $z \in \{1, \ldots, 2h\}$,*

$$E_{f\in\mathcal{F}}[|f^{-1}(I_z)|^k] \le \frac{n^{[z]}(T)}{z} c^k (D(T)k)^{k-1},$$

*where $n^{[z]}(T)$ is the number of vertices of $T$ of depth $\le z$, and $I_z$ is the interval $[z, z+1)$.*

*A family $\mathcal{T}$ of trees has property Good if every tree $T \in \mathcal{T}$ in the family has property Good, with the same constant $c$ for all trees in the family.*

**Theorem 2.** *Caterpillars have property Good. Given a caterpillar $T$ of height $h$, the distribution $\mathcal{F}(T)$ consists of $h$ equally probable functions $f_1$*

*up to $f_h$. The function $f_i$ is defined by $f_i(v) = (1 + i/h)d(v)$, where $d(v)$ is the depth of $v$ (the distance from $v$ to the root).*

**Theorem 3.** *Let $\mathcal{T}$ be a family of trees. A tree is horizontally derived from $\mathcal{T}$ if it is the result of identifying the roots of several trees belonging to $\mathcal{T}$. The family $H(\mathcal{T})$ consists of all trees horizontally derived from $\mathcal{T}$.*

*If the family $\mathcal{T}$ has property Good with constant $c$, then so does $H(\mathcal{T})$ with constant $O(c)$. Given a tree $T \in \mathcal{T}$ which is the result of identifying the roots of $T_1$ up to $T_d$, the distribution $\mathcal{F}(T)$ is defined as follows. To sample $\mathcal{F}(T)$, sample a layout function $f_i \in \mathcal{F}(T_i)$ for each $i \in \{1, \ldots, d\}$. The resulting sample $f \in \mathcal{F}(T)$ is defined by $f(r) = 0$ if $r$ is the root, and $f(v) = f_i(v)$ if $v \in T_i$.*

**Theorem 4.** *If $T$ has property Good then $B(T) = O(D(T) \log h)$.*

Theorem 2 is a reformulation of corollary 21. The theorem is proved using the results of section 3. Theorem 3 is a reinterpretation of lemma 24. Finally, theorem 4 is a corollary of lemma 14. It is based on similar results in [6, 8].

Theorem 2 represents a simplification of Gupta's original algorithm. However, theorem 2 is also true with respect to the original algorithm.

In the main body of this paper, no attempt is made to formulate the results in terms of the general framework. Instead, we concentrate on proving the main result, theorem 1.

### 1.4 Directions for further research

We believe that the analysis of Gupta's algorithm can be greatly improved by employing our framework. The following conjecture (induction step V) needs to be proven.

**Conjecture 2.** *Let $\mathcal{T}$ be a family of trees. A tree is vertically derived from $\mathcal{T}$ if it is the result of letting several trees belonging to $\mathcal{T}$ hang from vertices of a path (called the spine). The family $V(\mathcal{T})$ consists of all trees vertically derived from $\mathcal{T}$.*

*If the family $\mathcal{T}$ has property Good, then so does $V(\mathcal{T})$. Given a tree $T \in \mathcal{T}$ which is the result of letting the trees $T_1$ up to $T_d$ hang from depths $h_1$ to $h_d$, the distribution $\mathcal{F}(T)$ is defined as follows. To sample $\mathcal{F}(T)$, sample a layout function $f_i \in \mathcal{F}(T_i)$ for each $i \in \{1, \ldots, d\}$, and an integer $s \in \{1, \ldots, h\}$, where $h$ is the length of the spine. The resulting sample $f \in \mathcal{F}(T)$ is defined by $f(s_i) = (1+s/h)i$ if $s_i$ is the spinal vertex at depth $i$, and $f(v) = f(s_{h_i}) + f_i(v)$ if $v \in T_i$.*

Proving conjecture 2 would imply that the performance of Gupta's algorithm is $O(Dc_\kappa \log h)$, where $c_\kappa$ is a constant depending only on the caterpillar dimension, probably of the form $c^\kappa$. Further refining could then possibly be used to remove the dependency on the caterpillar dimension.

Moreover, we follow others and conjecture that $B = O(D \log n)$, at least for trees. One can prove such a relation constructively using approximation algorithms. For example, Feige's algorithm [6] shows that $B = O(D \log^{3.5} n \sqrt{\log \log n})$ for general graphs. Perhaps $B = O(D \log n)$ can be proved for trees using Gupta's algorithm.

We should mention that there are examples where $B = \Omega(D \log n)$. An expander family with $B = \Omega(D \log n)$ can be found in [14]. A tree family with $B = \Omega(D \log n)$ was constructed by Chvátalová [4]. The trees are constructed recursively as follows. Let $T_0$ consist of a single vertex. The tree $T_{i+1}$ is constructed from two copies of $T_i$ by putting each copy down a path of length $2^i$ emanating from the new root. The density of each tree is constant, whereas the bandwidth is $\Omega(i) = \Omega(\log n)$. The tree $T_2$ appears in figure 5.

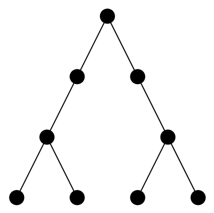
Figure 5: The tree $T_2$

Note that ordinary full $k$-ary trees have $B = O(D)$, as proved by Smithline [19]. The triangle caterpillars mentioned in section 1.3 also have $B = O(D)$.

## 2 Preliminaries

We define bandwidth and many-caterpillars, as well as some notions of density, in the present section. In section 3 we describe a certain regularization process on caterpillars, and draw some conclusions. In section 4 we present and analyze the approximation algorithm.

We fix once and for all some terminology. First, $\log n = \log_2 n$, that is, all our logarithms are base 2. Second, some graphical terminology is needed.

**Notation 3** (Graphical terminology)**.** *Let $G$ be a graph. The vertex-set of $G$ is denoted $V(G)$. The edge-set of $G$ is denoted $E(G)$. If $T$ is a rooted tree, the root of $T$ is denoted $r(T)$, and the number of non-root vertices of $T$ is $n(T) = |V(T) \setminus r(T)| = |V(T)| - 1$.*

Note the unusual definition of $n(T)$.

### 2.1 Caterpillars and many-caterpillars

Caterpillars and many-caterpillars are certain types of rooted trees having a simple structure.

**Definition 4** (Caterpillar)**.** *A caterpillar is a rooted tree $T$ which is composed of a spine and hairs, all rooted paths. The spine contains the root $r(T)$. The caterpillar $T$ is obtained by identifying the root of each hair $H$ with a spinal vertex $A(H)$. The hair $H$ is said to emanate from $A(H)$.*

Note that the decomposition of a caterpillar into spine and hairs is not necessarily unique. Figure 6 shows a caterpillar whose spine is the vertical path.

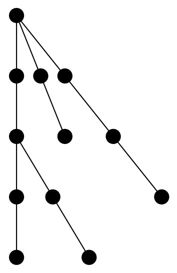
Figure 6: A caterpillar

**Definition 5** (Many-caterpillar)**.** *A many-caterpillar is a rooted tree formed by identifying the roots of several caterpillars.*

Any caterpillar is also a many-caterpillar. Figure 7 shows a many-caterpillar which is not a caterpillar.

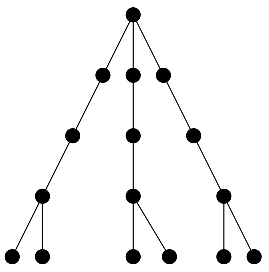
Figure 7: A many-caterpillar

Many-caterpillars can be used to construct more complicated rooted trees. This is done by taking a rooted tree, and identifying each vertex with the root of a many-caterpillar. If a rooted tree can be constructed by taking a single vertex and iterating this process $d \geq 1$ times, the tree is said to have caterpillar dimension $d + 1$. Clearly, many-caterpillars are the trees of caterpillar dimension 2.

Alternatively, the caterpillar dimension of a tree is $\leq d$ if the tree can be decomposed into edge-disjoint paths in such a way that every root-to-leaf route is composed of at most $d$ paths.

The caterpillar dimension of an arbitrary tree can be found using dynamic programming. It turns out that the caterpillar dimension of a rooted tree having $n$ vertices is at most $\log n$. The caterpillar dimension of a binary tree of height $h$ is $h$. In other words, a binary tree having $n$ vertices has caterpillar dimension $\Omega(\log n)$.

## 2.2 Density

We describe two related notions of graphical density. The first notion is geared towards relating to bandwidth. The second notion is defined only on rooted trees, and agrees with the first notion up to a multiplicative constant. The first notion is 'density', and the second notion is 'thickness'.

Before describing the notions of density and thickness, we fix three bits of terminology.

**Notation 6** (Distance)**.** *The shortest distance between two vertices $x$ and $y$ in a graph is denoted $d(x, y)$.*

**Notation 7** (Subtree)**.** *Each vertex $v$ in a rooted tree is a root of a subtree whose vertices (including $v$) are denoted $V_{\downarrow}(v)$.*

**Notation 8** (Ancestor)**.** *Each vertex $v$ in a rooted tree $T$ other than the root has an ancestor denoted $A(v)$. We also define $A(r(T)) = r(T)$ (recall $r(T)$ is the root of $T$). The vertex $A^d(v)$ is the result of applying $d$ times in succession the ancestor function $A$ on $v$.*

Density measures the relative size of neighborhoods.

**Definition 9** (Neighborhood)**.** *Let $v$ be a vertex in a graph $G$. The $d$th neighborhood of $v$ is*

$$N(v,\, d) = \{x \in V(G) : 0 < d(x,\, v) \leq d\}.$$

Note that in our definition, $v \notin N(v,\, d)$.

**Definition 10** (Density)**.** *A graph $G$ is $D$-dense if for every vertex $v$ and integer $d \geq 0$ we have $|N(v,\, d)| \leq 2dD$. The density $D(G)$ of $G$ is the minimal number $D$ such that $G$ is $D$-dense.*

Thickness is the analog of density when neighborhoods are replaced by downward neighborhoods.

**Definition 11** (Downward neighborhood)**.** *Let $v$ be a vertex in a rooted tree. The $d$th downward neighborhood of $v$ is*

$$N_{\downarrow}(v,\, d) = \{x \in V_{\downarrow}(v) : 0 < d(x,\, v) \leq d\}.$$

**Definition 12** (Thickness)**.** *A rooted tree $T$ is $\Theta$-thick if for every vertex $v$ and integer $d \geq 0$ we have $|N_{\downarrow}(v,\, d)| \leq d\Theta$. The thickness $\Theta(T)$ of $G$ is the minimal number $\Theta$ such that $T$ is $\Theta$-thick.*

Note that whereas $|N(v,\, d)|/d \leq 2D(G)$ for every neighborhood $N(v,\, d)$ of $G$, here $|N_{\downarrow}(v,\, d)|/d \leq \Theta(T)$ for every downward neighborhood $N_{\downarrow}(v,\, d)$ of $T$.

In a caterpillar, only certain downward neighborhoods are required to find the thickness.

**Lemma 1.** *A caterpillar $T$ is $\Theta$-thick if and only if for every spinal vertex $v$ and integer $d \geq 0$ we have $|N_{\downarrow}(v,\, d)| \leq d\Theta$.*

*Proof.* The only if part is clear. So suppose that for every spinal vertex $v$ and integer $d \geq 0$ we have $|N_{\downarrow}(v,\, d)| \leq d\Theta$. If $n(T) = 0$ then clearly $T$ is $\Theta$-thick, so suppose $n(T) \geq 1$. In this case $|N_{\downarrow}(r(T),\, 1)| \geq 1$ and so $\Theta \geq 1$. If $u$ is a non-spinal vertex then clearly $|N_{\downarrow}(u,\, d)| \leq d \leq d\Theta$ for every $d \geq 0$. It follows that $T$ is $\Theta$-thick. $\square$

Density and thickness are equivalent up to a multiplicative constant.

**Lemma 2.** *Let $T$ be a rooted tree. Then $D(T) \leq \Theta(T) \leq 2D(T)$.*

*Proof.* Since $N_{\downarrow}(v,\, d) \subset N(v,\, d)$, if $T$ is $D$-dense then $T$ is also $2D$-thick. Therefore $\Theta(T) \leq 2D(T)$.

For the other direction, note that for some neighborhood $N(v,\, d)$ we have $|N(v,\, d)| = 2dD(T)$. Let $x = A^d(v)$. Clearly $N_{\downarrow}(x,\, 2d) \cup x \supset N(v,\, d) \cup v$, hence $|N_{\downarrow}(x,\, 2d)| = 2dD(T)$. It follows that $\Theta(T) \geq D(T)$. $\square$

## 2.3 Bandwidth

The notion of bandwidth arises when attempting to evaluate an embedding of a graph onto the infinite path. One possible evaluation is in terms of the distortion of distance. Specifically, we would like distances to be not much larger than in the original graph. The notion of bandwidth is one possible formalization of this idea.

**Definition 13** (Bandwidth)**.** *Let $G$ be a graph. A layout $f$ of $G$ is a one-to-one function mapping vertices of $G$ to integers. The bandwidth of a layout $f$ is $B(f) = \max_{(x,\,y)\in E(G)} |f(x) - f(y)|$. The bandwidth of the graph $G$ is $B(G) = \min_f B(f)$, where $f$ goes over all layouts of $G$.*

A layout of a graph is simply an embedding into the infinite path, and the bandwidth of a layout measures the worst multiplicative factor by which distances are stretched (this measure is known as the dilation of the embedding). The definition is only concerned with the stretching of edges, yet the two notions are equivalent.

**Lemma 3.** *Let $f$ be a layout of a connected graph $G$. Then*

$$B(f) = \max_{x\neq y} \frac{|f(x) - f(y)|}{d(x,\,y)}.$$

*Proof.* Let $B^*(f) = \max_{x\neq y} |f(x) - f(y)|/d(x,\,y)$. Clearly $B^*(f) \geq B(f)$. On the other hand, if $d(x,\,y) = d$ then $|f(x) - f(y)| \leq dB(f)$ by considering the path of length $d$ from $x$ to $y$. Hence also $B^*(f) \leq B(f)$. $\square$

The bandwidth of a graph measures the amount by which distances are stretched in any embedding of the graph into the infinite path. If there is a large neighborhood with small diameter, it is evident that distances must be stretched to accommodate this neighborhood.

**Lemma 4.** *Let $G$ be a connected graph. Then $B(G) \geq D(G)$.*

*Proof.* There is a neighborhood $N(v,\,d)$ such that $|N(v,\,d)| = 2dD(G)$. Let $f$ be any layout of $G$. By lemma 3, for each $x \in N(v,\,d)$ we have $|f(x)-f(v)| \leq dB(f)$. There are $2dB(f)$ integers $i \neq f(v)$ satisfying $|i - f(v)| \leq dB(f)$. Since $f$ is one-to-one it follows that $|N(v,\,d)| \leq 2dB(f)$. Hence $B(f) \geq D(G)$, and so $B(G) \geq D(G)$. $\square$

## 3 Regularization of caterpillars

Caterpillars have a very simple structure. Yet for some purposes, it is advantageous to consider caterpillars having an even simpler structure. In section 3.1 we describe a regularization process, which transforms a given caterpillar into another caterpillar of extremely simple form, conserving certain important statistics. In section 3.2 we draw some conclusions.

We summarize some useful terminology in the following definition.

**Definition 14** (Depth, Height, Tail vertex)**.** *Let $T$ be a rooted tree with root $r$. The depth of a vertex $x$ is $d(x) = d(x, r)$. The height $h(T)$ of the tree is the depth of the deepest vertex. A vertex of maximal depth is called a tail vertex.*

**Lemma 5.** *Let $T$ be a rooted tree. Then $n(T) \leq \Theta(T)h(T)$.*

*Proof.* Clearly $n(T) = |N_{\downarrow}(r(T), h(T))|$, where $r(T)$ is the root of $T$. The lemma now follows from the definition of thickness. □

### 3.1 Regularization process

Before seeking a regularization process, one must ask what statistics are to be conserved by the process. Three natural statistics are the number of vertices, the height and the thickness. Our regularization process conserves these three statistics, as well as a fourth one. To motivate the fourth statistic, we look at a bandwidth approximation algorithm for caterpillars.

The best bandwidth approximation algorithm currently known for caterpillars is roughly as follows [10].

**Algorithm 1.** *Given a caterpillar, order its vertices in such a way that if $d(x) < d(y)$ then $x$ precedes $y$ (recall $d(v)$ is the depth of $v$). There are many possible such orders, and the exact order chosen is not important. The algorithm produces a layout $f$ defined by $f(x) = k$, where $x$ is the $k$th vertex in the order.*

**Theorem 5.** *Let $f$ be a layout constructed as described for a caterpillar $T$. Then $B(f) = O(\Theta(T) \log h(T))$.*

*Proof.* It is easy to see that $B(f)$ is bounded by twice the maximal number of vertices at given depth. Let $n(T, h)$ be the number of vertices at depth $h$. If there is no spinal vertex at depth $h-1$ then clearly $n(T, h) \leq n(T, h-1)$. So we upper bound the number of vertices at depth $h$, assuming there is a spinal vertex at depth $h - 1$.

For any vertex $v$, let $d(v, S)$ be its distance from the spine. For integral $k \leq h$, let

$$V(k) = \{x : d(x) = h \text{ and } \frac{k}{2} \leq d(v, S) \leq k\}.$$

For integral $0 \leq k < h$, let $S_k$ be the spinal vertex at depth $k$. Evidently $|N_\downarrow(S_{h-k}, k)| \geq |V(k)|k/2$. Since $|N_\downarrow(S_{h-k}, k)| \leq \Theta(T)k$, it follows that $|V(k)| \leq 2\Theta(T)$.

Let $K = \{2^0, 2^1, \ldots, 2^{\lceil \log h \rceil}\}$. Each non-spinal vertex at depth $h$ belongs to $V(k)$ for some $k \in K$. The set $K$ contains $O(\log h)$ integers, hence there are at most $1 + O(|K|\Theta(T)) = O(\Theta(T) \log h)$ vertices at depth $h$. The proposition follows. □

By lemmas 2 and 4 we have $\Theta(T) \leq 2D(T) \leq 2B(T)$. Hence algorithm 1 has an approximation ratio of $O(\log h)$, where $h$ is the height of the caterpillar.

The proof of theorem 5 points to the importance of the number of vertices at given depth. In the proof of the theorem, only the maximal number was used. However, other results require more data.

**Definition 15** (Inventory). *Let $T$ be a rooted tree. For integral $h \leq h(T)$, define $n(T, h)$ as the number of vertices at depth $h$. The inventory of $T$ is the multiset*

$$I(T) = \{n(T, h) : 1 \leq h \leq h(T)\}.$$

Note that since the root is always the only vertex at depth 0, there is no need to include $n(T, 0)$ in the definition. We emphasize that the inventory is a multiset, a term meaning that the number of occurrences of each element is important, but the elements are not ordered.

The two statistics $n(T)$ and $h(T)$ are determined by the inventory.

**Lemma 6.** *Let $T$ be a rooted tree. Then $n(T) = \sum_{x \in I(T)} x$ and $h(T) = |I(T)|$.*

*Proof.* The lemma is a direct consequence of the definition of $I(T)$. □

As a consequence of lemma 6, whenever the inventory is conserved, so are the height and the number of vertices.

We can restate theorem 5 as $\max I(T) = O(\Theta(T) \log h(T))$, if $T$ is a caterpillar. The following family of caterpillars [10], called 'triangle caterpillars', satisfy $\max I(T) = \Omega(\Theta(T) \log h(T))$.

Fix the desired thickness to be an odd $\Theta \geq 3$. Choose a height $h = 2^l \geq 1$. The spine consists of the root $S_0$ and $h$ other vertices $S_1, \ldots, S_h$, such that the depth of $S_i$ is $i$. Let $\delta = (\Theta - 1)/2$. There are $2\delta$ hairs of length 1 emanating from $S_{h-1}$. For each integer $j \in [1, l]$, there are $\delta$ hairs of length $2^j$ emanating from $S_{h-2^j}$. The resulting tree for $\Theta = 3$ and $h = 2^2$ is shown in figure 8.

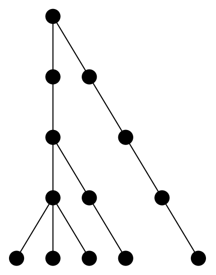
Figure 8: A triangle caterpillar

It can be checked that the thickness of the resulting tree $T$ is $\Theta$, and $n(T, h) = \delta(\log h + 2) + 1$ is the maximum of the inventory. The properties of triangle caterpillars are summarized in proposition 1.

One feature of the triangle caterpillars is that the spine and hairs all have tail vertices (vertices of depth $h(T)$). Such caterpillars are easy to analyze, since their structure is wholly contained in their inventory.

**Definition 16** (Regular Caterpillar)**.** *A regular caterpillar is a caterpillar whose spine and hairs each contain a tail vertex.*

**Lemma 7.** *For each multiset $I$ of positive integers there is a unique (up to isomorphism) regular caterpillar $T$ such that $I(T) = I$.*

*Proof.* Given a multiset $I$, the unique regular caterpillar $T$ is constructed as follows. Order $I = \{I_1, \ldots, I_h\}$ so that $I_i \leq I_{i+1}$. The caterpillar $T$ has a spine of length $h$. There are $I_1 - 1$ hairs emanating from the root. For each $i \in [1, h-1]$, there are $I_{i+1} - I_i$ hairs emanating from depth $i$. Clearly, $T$ is the unique regular caterpillar such that $I(T) = I$. $\square$

At this point, it is useful to increase our terminology involving spines and hairs.

**Definition 17** (Spine Terminology)**.** *Let $S$ be the spine of a caterpillar. Define $A(S)$ as the root of the caterpillar, and denote the deepest vertex of $S$ by $B(S)$. The depth of $A(S)$ is denoted* $\operatorname{top}(S)$ *(it is always zero), and the depth of $B(S)$ is denoted* $\operatorname{bot}(S)$. *A spine is long if it contains a tail vertex. Otherwise it is short. The spinal vertex at depth $i$ is denoted $S_i$. If $i < \operatorname{bot}(S)$ then the successor of $S_i$ is* $\operatorname{suc}(S_i) = S_{i+1}$.

**Definition 18** (Hair Terminology)**.** *For a hair $H$ of a caterpillar $T$, recall $A(H)$ denotes the spinal vertex from which $H$ emanates. The depth of $A(H)$ is denoted* $\operatorname{top}(H)$*. The deepest vertex of $H$ is denoted $B(H)$, and its depth is denoted* $\operatorname{bot}(H)$*. A long hair is any hair having a tail vertex. A short hair is any hair not having a tail vertex. A deep hair is any hair $H$ with* $\operatorname{bot}(H) > \operatorname{bot}(S)$*, where $S$ is the spine of $T$.*

The reader is cautioned that depth grows downwards in our figures, so that vertices at maximal depth appear at the bottom.

The regularization process conserves the inventory. It might, however, decrease the thickness. In our application, this is in fact a blessing.

**Theorem 6** (Regularization process)**.** *Let $T$ be any caterpillar. There is a regular caterpillar $T'$ satisfying $I(T') = I(T)$ and $\Theta(T') \le \Theta(T)$.*

*Proof.* The regularization process is algorithmic. It starts by performing 'spine elongation', to insure that the spine is long (has a tail vertex). It then repeatedly performs two operations called 'normalization' and 'shifting'. All three operations change the form of the caterpillar, conserving the inventory. All operations also do not increase the thickness, but some may decrease it.

**Spine elongation.** Suppose the spine $S$ is short. Hence some hair is deep (extends below the spine). Of all the deep hairs, choose a hair $H$ with maximal $\operatorname{top}(H)$. The spine is lengthened by shortening $H$ to a hair $H'$ such that $\operatorname{bot}(H') = \operatorname{bot}(S)$, and extending the spine by adding $\operatorname{bot}(H) - \operatorname{bot}(S)$ new vertices to the deep end of the spine.

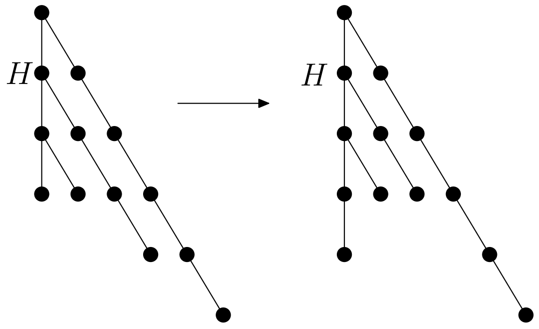

Figure 9: Spine lengthening

This exchange does not change the inventory.

Moreover, if $T$ is the original caterpillar and $T'$ is the new one, then $\Theta(T') \le \Theta(T)$. Indeed, let $S$ be the spine of $T$, and let $S'$ be the spine of $T'$. Pick a spinal vertex $S'_i$ of $T'$, and an integer $d \ge 0$.

1. If $i \le \operatorname{top}(H)$ then $|N_{\downarrow}(S'_i, d)| = |N_{\downarrow}(S_i, d)| \le d\Theta(T)$.

2. If $\mathrm{top}(H) < i$ and $i+d \leq \mathrm{bot}(S)$ then again $|N_{\downarrow}(S'_i,\, d)| = |N_{\downarrow}(S_i,\, d)| \leq d\Theta(T)$.

3. If $\mathrm{top}(H) < i \leq \mathrm{bot}(S)$ and $i + d > \mathrm{bot}(S)$ then

$$|N_{\downarrow}(S'_i,\, d)| = |N_{\downarrow}(S_i,\, \mathrm{bot}(S) - i)| + |N_{\downarrow}(v,\, d - (\mathrm{bot}(S) - i))| \leq d\Theta(T),$$

where $v$ is the vertex of $H$ at depth $\mathrm{bot}(S)$.

4. Finally, if $\mathrm{bot}(S) < i \leq \mathrm{bot}(S')$ then $|N_{\downarrow}(S'_i,\, d)| = |N_{\downarrow}(u,\, d)| \leq d\Theta(T)$, where $u$ is the vertex of $H$ at depth $i$.

By lemma 1, $T'$ is $\Theta(T)$-thick and so $\Theta(T') \leq \Theta(T)$.

Spine elongation is performed by the following algorithm: as long as the spine is short, perform spine lengthening (as just described). Since each spine lengthening actually increases the length of the spine, this algorithm terminates. After spine elongation, the spine is long. Since all other operations preserve this property, spine elongation is performed only once.

**Normalization.** Let $H$ and $H_1$ be two hairs such that $\mathrm{top}(H) < \mathrm{top}(H_1) \leq \mathrm{bot}(H) < \mathrm{bot}(H_1)$.

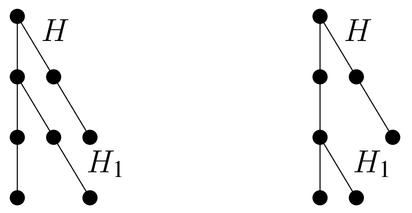

Figure 10: Conflicting pairs

Such a pair of hairs is called a conflicting pair. This pair is normalized by shortening $H_1$ to a hair $H'_1$ such that $\mathrm{bot}(H'_1) = \mathrm{bot}(H)$, and extending $H$ to a hair $H'$ such that $\mathrm{bot}(H') = \mathrm{bot}(H_1)$.

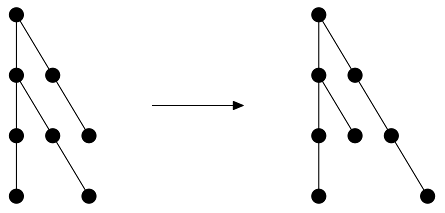

Figure 11: Normalizing a conflicting pair

This exchange does not change the inventory. It also keeps the spine long. Moreover, if $T$ is the original caterpillar and $T'$ is the new one, then $\Theta(T') \leq \Theta(T)$. Indeed, for every spinal vertex $S'_i$ of $T$ and integer $d \geq 0$ we have $|N_{\downarrow}(S'_i,\, d)| \leq |N_{\downarrow}(S_i,\, d)| \leq d\Theta(T)$, where $S$ and $S'$ are the spines of $T$ and $T'$, respectively.

Note that the reverse exchange preserves the inventory, but might increase the thickness.

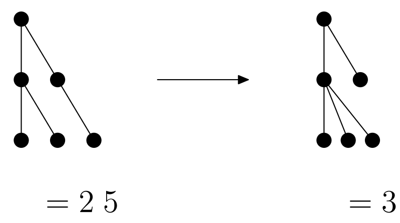

Figure 12: The reverse of normalization

For this reason, pair normalization is a natural operation. Note, however, that when performing spine elongation, the reverse exchange is performed and justified.

Normalization of an entire caterpillar is performed by the following algorithm: as long as there is a conflicting pair, normalize that pair. To see that the algorithm terminates, consider the potential function $\Phi = \sum_v \mathrm{top}(H(v))$, where $v$ goes over all non-spinal vertices, and $H(v)$ is the hair containing $v$. Every iteration decreases $\Phi$. Since $\Phi \geq 0$, the algorithm terminates. The result is a caterpillar without conflicting pairs. Note that normalization preserves the length of the spine.

**Shifting.** Let $T$ be a normalized caterpillar having some short hair $H$. The hair $H$ is part of a group of short hairs delimited by long hairs above and below it (if they exist). To determine the group, we determine its delimiters Top and Bot, which are simply depths. The top delimiter is $\mathrm{Top} = \max_{H'} \mathrm{top}(H')$, where the maximum is over all long hairs $H'$ such that $\mathrm{top}(H') \leq \mathrm{top}(H)$. If no such hairs exist, $\mathrm{Top} = 0$. The bottom delimiter is $\mathrm{Bot} = \min_{H'} \mathrm{top}(H')$, where the minimum is over all long hairs $H'$ such that $\mathrm{top}(H') > \mathrm{top}(H)$, or equivalently $\mathrm{top}(H') > \mathrm{bot}(H)$ (recall $T$ is normalized). If no such hairs exist, $\mathrm{Bot} = h(T)$.

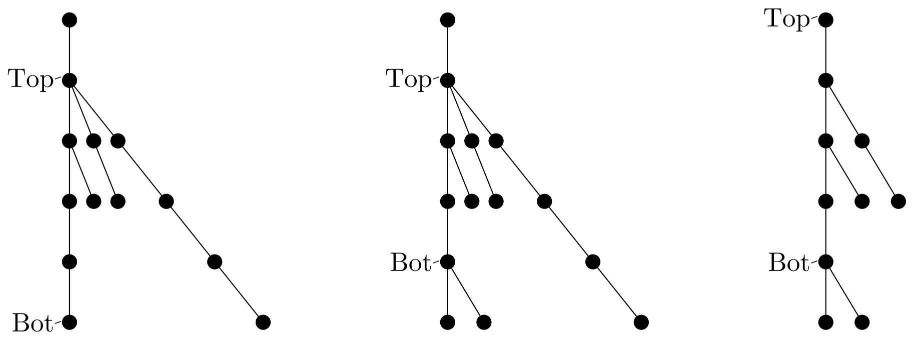

Figure 13: An illustration of Top and Bot

The local group of short hairs containing $H$ is the set $\mathcal{H}$ consisting of all short hairs $H'$ such that $\mathrm{Top} \leq \mathrm{top}(H') < \mathrm{Bot}$, or equivalently $\mathrm{Top} <$

$\mathrm{bot}(H') < \mathrm{Bot}$. Shifting $\mathcal{H}$ is done by letting each hair $H' \in \mathcal{H}$ be attached to $\mathrm{suc}(A(H))$ instead of $A(H)$. In other words, the group $\mathcal{H}$ is shifted done by one vertex. Note that $\mathrm{suc}(A(H))$ exists since the spine is long.

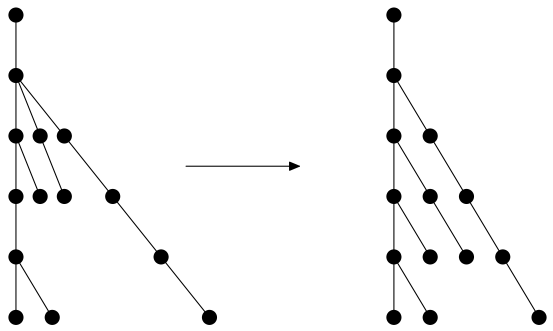
Figure 14: An example of shifting

It is easy to see that since the original caterpillar is normalized, shifting conserves the inventory. Shifting also keeps the spine long.

Moreover, if $T$ is the original caterpillar and $T'$ is the new one, then $\Theta(T') \leq \Theta(T)$. Indeed, let $S$ be the spine of $T$, and $S'$ be the spine of $T'$. Pick a spinal vertex $S'_i$ of $T'$ and an integer $d \geq 0$. If $i \leq \mathrm{Top}$ or $i \geq \mathrm{Bot}$ then it is easy to see that $|N_\downarrow(S'_i, d)| \leq |N_\downarrow(S_i, d)| \leq d\Theta(T)$. Next suppose $\mathrm{Top} < i < \mathrm{Bot}$. If $i + d \leq \mathrm{Bot}$ then $|N_\downarrow(S'_i, d)| \leq |N_\downarrow(S_{i-1}, d)| \leq d\Theta(T)$. If $i + d > \mathrm{Bot}$ then

$$|N_\downarrow(S'_i, d)| \leq |N_\downarrow(S_{i-1}, \mathrm{Bot} - i)| + |N_\downarrow(S_{\mathrm{Bot}}, d - (\mathrm{Bot} - i))| \leq d\Theta(T).$$

It follows by lemma 1 that $\Theta(T') \leq \Theta(T)$.

Shifting consists of the operation just described, that is shifting a single group of short hairs.

**Regularization.** The regularization process executes the following algorithm: first, perform spine elongation; second, as long as there are short hairs, normalize and then shift. To see that the algorithm terminates, consider the potential function $\Psi = \sum_H (h(T) - \mathrm{top}(H))$, where $H$ goes over all hairs. Normalization does not increase $\Psi$ (it can decrease it), and shifting decreases $\Psi$, unless all hairs are long. Since $\Psi \geq 0$, the algorithm terminates. $\square$

Figure 15 shows an example of an execution of the regularization process. The letters N and S stand for normalization and shifting, respectively.

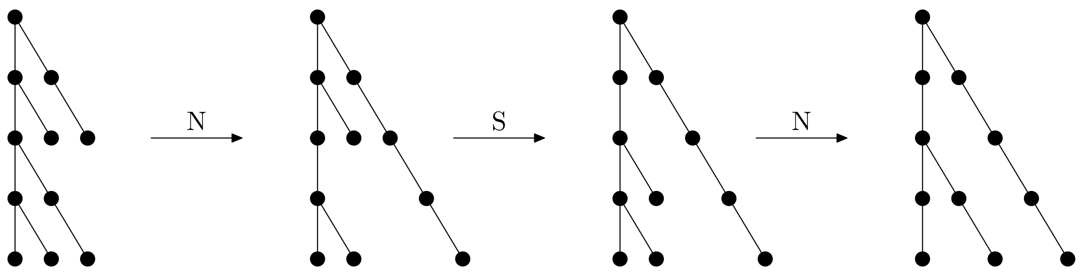

Figure 15: Regularization of a caterpillar

Recall that by lemma 7, for each caterpillar $T$ there is a unique regular caterpillar $T'$ with $I(T') = I(T)$. The lemma also provides an easy construction of $T'$. Hence what the regularization process really shows is that $\Theta(T') \leq \Theta(T)$. An example where $\Theta(T') < \Theta(T)$ is presented in figure 16.

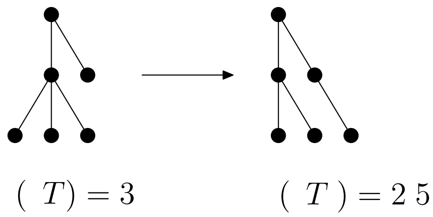

Figure 16: Regularization might decrease the thickness

## 3.2 Consequences

The inventories of regular caterpillars are easy to analyze. Results proven about the inventories of regular caterpillars can be extended to all caterpillars using theorem 6.

A natural question to ask about an inventory is the size of its largest element. This question is answered in theorem 5. For convenience, we reproduce the proof here.

**Lemma 8.** *Let $T$ be a caterpillar. Then* $\max I(T) = O(\Theta(T) \log h(T))$.

*Proof.* By theorem 6, we can assume that $T$ is regular.

Suppose the maximum $M = \max I(T)$ is achieved at depth $h \leq h(T)$. Let $\mathcal{H}$ denote the multiset $\mathcal{H} = \{h - \mathrm{top}(H) : \mathrm{top}(H) < h\}$, so that $M = 1 + |\mathcal{H}|$. It is easy to find $O(\log h)$ intervals of the form $[k/2,\, k]$ that cover $[0,\, h-1]$. One of these intervals $[k/2,\, k]$ contains $\Omega(M/\log h)$ elements of $\mathcal{H}$. Hence $|N_{\downarrow}(S_{h-k},\, k)| = \Omega(kM/\log h)$. It follows that $M/\log h = O(\Theta(T))$, and so $M = O(\Theta(T) \log h)$. □

A generalization of the previous question asks how many elements of the inventory can be greater than a given number.

**Definition 19.** *Let $I$ be a multiset. The number of elements of $I$ which are greater than $r$ is denoted $|I|_{>r}$.*

**Lemma 9.** *Let $T$ be a caterpillar. Then for integral $s \geq 1$ we have*

$$|I(T)|_{>2\Theta(T)s} \leq \frac{n(T)}{\Theta(T)} 2^{-s}.$$

*Proof.* By theorem 6, we can assume that $T$ is regular.

Order the hairs of $T$ in order of increasing $\operatorname{top}(H)$. Let $\operatorname{len}(k)$ denote $h(T) - \operatorname{top}(H_k)$, where $H_k$ is the $k$th hair in the order. Since $T$ is regular, $|I(T)|_{>k} = \operatorname{len}(k)$, if $H_k$ exists, and $|I(T)|_{>k} = 0$ otherwise.

Choose any $k > 2\Theta(T)$. The $2\Theta(T)$ hairs $H_k$ to $H_{k-2\Theta(T)+1}$ all have length at least $\operatorname{len}(k)$. This implies that $|N_\downarrow(S_{h(T)-\ell}, \ell)| \geq 2\Theta(T)\operatorname{len}(k)$, where $\ell = \operatorname{len}(k - 2\Theta(T))$. Hence $\operatorname{len}(k - 2\Theta(T)) \geq 2\operatorname{len}(k)$. It follows that $\operatorname{len}(k - 2\Theta(T)r) \geq 2^r \operatorname{len}(k)$ for $r \geq 0$.

The first $2\Theta(T)$ hairs $H_1$ to $H_{2\Theta(T)}$ all have length at least $\operatorname{len}(2\Theta(T)) \geq 2^{s-1}\operatorname{len}(2\Theta(T)s)$. Hence $n(T) \geq 2^s\Theta(T)\operatorname{len}(2\Theta(T)s)$. The lemma follows. □

**Corollary 10.** *Let $T$ be a caterpillar. Then*

$$\max I(T) = O\Big(\Theta(T) \log \frac{n(T)}{\Theta(T)}\Big).$$

*Proof.* Define $h' = n(T)/\Theta(T)$, and let $s = \lceil \max I(T)/2\Theta(T)\rceil - 1$. Then $|I(T)|_{>2\Theta(T)s} \geq 1$, and so $h' \geq 2^s$ by lemma 9. It follows that $s \leq \log h'$ and so $\max I(T) = O(\Theta(T)\log h')$. □

Recall that by lemma 5, $n(T) \leq \Theta(T)h(T)$. Hence corollary 10 is stronger than lemma 8. Both the corollary and the lemma are tight in the following sense. The triangle caterpillars defined in section 3.1 satisfy $\max I(T) = \Omega(\Theta(T)\log h(T))$ and $h(T) = n(T)/\Theta(T)$.

The following lemma is a consequence of lemma 9. This lemma (or rather its corollary) is the only result from this section which is needed in section 4.

**Lemma 11.** *Let $T$ be a caterpillar. For integral $k \geq 1$, we have the following inequality, for some constant $C_1$:*

$$\sum_{x \in I(T)} x^k = O(n(T)\Theta(T)^{k-1}(C_1 k)^k).$$

*Proof.* We divide the sum $\sum_{x\in I(T)} x^k$ into two parts $S_L$ and $S_H$, according to the size of $x$. The sum $S_L$ contains all summands $x^k$ where $x \leq 2\Theta(T)$. The sum $S_H$ contains all summands $x^k$ where $x > 2\Theta(T)$.

We begin by estimating the sum $S_L$. Let $m_i$ denote the number of summands $i^k$ in $S_L$, so that $S_L = \sum_{i=1}^{2\Theta(T)} m_i i^k$. Denote $m = \sum_{i=1}^{2\Theta(T)} m_i i$. Clearly $m \leq n(T)$. For $i \in [1, 2\Theta(T)]$, let $m'_i = m_i i$, so that $m = \sum_i m'_i$ and $S_L = \sum_i m'_i i^{k-1}$. Since $i \leq 2\Theta(T)$, evidently $S_L \leq m(2\Theta(T))^{k-1} \leq n(T)(2\Theta(T))^{k-1}$.

We turn to estimate the sum $S_H$. Clearly

$$S_H \leq \sum_{s=2}^{\infty} (2\Theta(T)s)^k |I(T)|_{>2\Theta(T)(s-1)}.$$

Using lemma 9, we can estimate

$$S_H \leq 2^k n(T)\Theta(T)^{k-1} \sum_{s=2}^{\infty} \frac{s^k}{2^{s-1}}.$$

It remains to estimate the sum $\sum_{s=2}^{\infty} s^k/2^s$:

$$\begin{aligned}\sum_{s=2}^{\infty} \frac{s^k}{2^s} &= \sum_{s=2}^{2k} \frac{s^k}{2^s} + \sum_{s=2k+1}^{\infty} \frac{s^k}{2^s} \\ &\leq (2k)^k \sum_{s=2}^{2k} 2^{-s} + \frac{(2k)^k}{2^{2k}} \sum_{t=1}^{\infty} \Big(\frac{(2k+1)/2k}{2}\Big)^t \\ &\leq (2k)^k + (k/2)^k \sum_{t=1}^{\infty} \Big(\frac{3}{4}\Big)^t = (2k)^k + 3(k/2)^k = O((2k)^k).\end{aligned}$$

The lemma follows with $C_1 = 4$. □

**Corollary 12.** *Let $T$ be a caterpillar. For integral $k \geq 1$, we have the following inequality, for some constant $C_2$:*

$$\sum_{x\in I(T)} x^k = O(n(T)(C_2\Theta(T))^{k-1}k!).$$

*Proof.* The Stirling formula states that $k! = \Theta(\sqrt{k}(k/e)^k)$. It follows that $k^k = O(e^k k!)$. The corollary follows by letting $C_2 = eC_1$. □

The inequality in lemma 11 is asymptotically tight for the triangle caterpillars discussed in section 3.1, in a sense made precise by corollary 13.

**Proposition 1.** *Let $T$ be a triangle caterpillar constructed for thickness $\Theta$ and height $h = 2^l$ (the construction is described in section 3.1). Then $T$ is a regular caterpillar with $n(T) = h\Theta$, $h(T) = h$, $\Theta(T) = \Theta$ and $B(T) = D(T) = (\Theta + 1)/2$. Moreover, the inventory $I(T)$ is*

$$\{(\delta + 1)^{2^{l-1}}, (2\delta + 1)^{2^{l-2}}, \ldots, (l\delta + 1)^{2^0}, ((l + 2)\delta + 1)^{2^0}\},$$

*where $\delta = (\Theta - 1)/2$.*

**Corollary 13.** *Let $T$ be a triangle caterpillar constructed for thickness $\Theta$ and height $h = 2^l$. If $l \geq k$ then*

$$\sum_{x \in I(T)} x^k \geq n(T)\Theta(T)^{k-1}(k/6)^k.$$

*Proof.* Since $l \geq k$, $I(T)$ contains $2^{l-k}$ times the element $(k\delta + 1)$, where $\delta = (\Theta - 1)/2 \geq \Theta/3$. Hence

$$\sum_{x \in I(T)} x^k \geq 2^{l-k}(k\delta)^k = h(k\delta/2)^k \geq n(T)\Theta(T)^{k-1}(k/6)^k. \qquad \square$$

It can be shown that if $T$ is a triangle caterpillar then $B(T) = D(T) = (\Theta(T) + 1)/2$. Details omitted.

# 4 Approximation Algorithm

An approximation algorithm for bandwidth was proposed in [10]. The algorithm operates only on caterpillars, and its approximation ratio is $O(\log n)$. It is described and analyzed in section 3.1. Gupta [8] presented a randomized approximation algorithm for trees with approximation ratio $O(\log^{2.5} n)$. On caterpillars, the algorithm operates in a manner reminiscent of the algorithm in [10], and its approximation ratio is also $O(\log n)$. Moreover, Gupta showed that his algorithm has approximation ratio $O(\log^2 n)$ on many-caterpillars (and more generally, on trees of constant caterpillar dimension).

In section 4.1 we present a randomized approximation algorithm for many-caterpillars, which is based on Gupta's algorithm. We outline its analysis, and show how to derandomize it. In section 4.2 we complete the analysis of our algorithm, showing that it has an approximation ratio of $O(\log n)$. It follows from our analysis that the approximation ratio of Gupta's algorithm on many-caterpillars is also $O(\log n)$.

## 4.1 Algorithm

Recall algorithm 1 described in section 3.1. The layout produced by this algorithm is equivalent to an ordering of the vertices with respect to their depths. The algorithm for many-caterpillars uses this ordering for each individual caterpillar. The different orderings are first 'stretched' randomly, then combined together.

**Algorithm 2** (Randomized version)**.** *Let $T$ be a many-caterpillar obtained by identifying the roots of the caterpillars $T_1$ up to $T_d$, where $n(T_i) > 0$. For each caterpillar $T_i$, choose independently a random integer $\sigma_i$ from $[1, h(T_i)]$ uniformly, and let $s_i = 1 + \sigma_i / h(T_i)$ be the stretch factor of $T_i$.*

*For each vertex $v$ of $T_i$, recall $d(v)$ is the depth of $v$. Let $p(v) = s_i d(v)$ be the placement of $v$. Order the vertices of $T$ in order of ascending $p(v)$: $v_0 = r(T), v_1, \ldots, v_{n(T)}$. Output the layout $f(v_j) = j$.*

Algorithm 1 is a special case of algorithm 2, obtained when $d = 1$.

The analysis in section 3.1 estimates the maximal number of vertices having the same depth. This is useful since each edge connects vertices of adjacent depths. In algorithm 2, each edge is stretched by some factor between 1 and 2. Hence it is natural to look at the number of vertices falling within a constant width interval.

**Definition 20** (Unit interval, Unit statistic)**.** *A unit interval is any interval of the form $[z, z + 1)$, where $z$ is an integer.*

*Let $T$ be a many-caterpillar. The zth unit statistic $X_z(T)$ is the random variable counting the number of vertices $v \in V(T)$ such that $p(v) \in [z,\, z+1)$ in an execution of algorithm 2 on $T$. The notation $X_z^k(T)$ means $(X_z(T))^k$.*

Using unit statistics, we can adapt the analysis in section 3.1 to algorithm 2.

**Lemma 14.** *Let $T$ be a many-caterpillar. Suppose that in some execution of algorithm 2, the inequality*

$$\sum_{z=1}^{2h(T)} X_z^{k(T)}(T) \leq (C_0\Theta(T)g(T))^{k(T)}$$

*holds for some constant $C_0$ and functions $g(T)$ and $k(T)$. Then the layout $f$ produced by the algorithm has bandwidth $B(f) = O(\Theta(T)g(T))$.*

*Proof.* The given inequality implies that $X_z(T) \leq C_0\Theta(T)g(T)$ for each $z \in [1,\, 2h(T)]$. Also, $X_0^{k(T)}(T) = 1$. By definition of $B(f)$, there is some edge $(x,\, y) \in E(T)$ such that $f(y) = f(x) + B(f)$. Hence there are at least $B(f)$ vertices $v$ that are placed in the interval $[p(x),\, p(y)]$. Since $p(y) \leq p(x) + 2$, the interval $[p(x),\, p(y)]$ is covered by at most 3 unit intervals. One of these interval $[z,\, z+1)$ contains at least $B(f)/3$ vertices. Thus $X_z(T) \geq B(f)/3$ and so $B(f) \leq 3C_0\Theta(T)g(T)$. □

Proposition 1 demonstrates that algorithm 1 may produce an arrangement with $B(f) = \Omega(\Theta(T)\log h(T))$, for example on triangle caterpillars. Since algorithm 2 extends algorithm 1, we aim for $g(T) = \log h(T)$.

In section 4.2 we show that the inequality needed for lemma 14 holds in expectation, with $g(T) = \log h(T)$. Given this fact, it is easy to use the method of conditional expectations to derandomize algorithm 2. They key is the following lemma.

**Lemma 15.** *Let $Y_1$ to $Y_d$ be mutually independent random variables, and let $K \geq 0$ be an integer. Suppose that for integral $k \in [0, K]$, the expectation $E[Y_1^k]$ can be computed. Then $E[(Y_1 + \cdots + Y_d)^K]$ can be computed in time polynomial in $d$ and $K$.*

*Proof.* Let $Z_i = Y_1 + \cdots + Y_i$. We are given $E[Z_1^k]$ for all integral $k \in [0,\, K]$. We show how to compute $E[Z_{i+1}^k]$ from the values of $E[Z_i^l]$ for integral $l \in [0,\, k]$ in time polynomial in $k$. The result follows.

Let $k$ be an integer in $[0,\, K]$, and suppose that $E[Z_i^l]$ is given for all $l \in [0,\, k]$. The random variables $Z_i$ and $Y_{i+1}$ are mutually independent.

Using the binomial theorem, linearity of expectation, and the independence of $Z_i$ and $Y_{i+1}$, we get

$$E[Z_{i+1}^k] = \sum_{l=0}^{k} \binom{k}{l} E[Z_i^l] E[Y_{i+1}^{k-l}].$$

The last sum can be computed from known expectations in time polynomial in $k$. □

Using lemma 15 it is easy to derandomize algorithm 2.

**Lemma 16** (Derandomizing algorithm 2)**.** *Let $T$ be a many-caterpillar obtained by identifying the roots of the caterpillars $T_1$ up to $T_d$. Let*

$$X = \sum_{z=1}^{2h(T)} X_z^{k(T)}(T)$$

*for some integral function $k(T)$. Let $X(\tau)$ denote the value of $X$ if $\sigma_i = \tau_i$. Integers $\tau_1$ up to $\tau_d$ such that $X(\tau) \leq E[X]$ can be found in time polynomial in $k(T)$ and $n(T)$.*

*Proof.* The expectations $E[X_z^k(T_i)]$ for integers $z \in [1, 2h(T)]$, $k \leq k(T)$ and $i \in [1, d]$ can all be found in polynomial time. The value of $X_z^k(T_i)$ given $\sigma_i = \tau_i$ can also be computed in polynomial time. Let $C$ represent any constraint on the $\sigma_i$. Since $X_z(T) = \sum_{i=1}^{d} X_z(T_i)$ for $z \geq 1$, lemma 15 can be used to compute $E[X|C]$ in polynomial time.

The assignment $\tau$ is found using the method of conditional expectations. First $\tau_1$ is found, then $\tau_2$, and so on. To determine $\tau_i$, compute $E[X|C_i(\tau^*)]$ for all possible $\tau^*$, where $C_i(\tau^*)$ is the constraint "$\sigma_j = \tau_j$ for all $j < i$, and $\sigma_i = \tau^*$". Pick some $\tau_i$ that satisfies $E[X|C_i(\tau_i)] \leq E[X]$. □

Lemma 16 directly transforms algorithm 2 into a deterministic algorithm.

**Algorithm 3** (Deterministic version)**.** *Let $T$ be a many-caterpillar obtained by identifying the roots of the caterpillars $T_1$ up to $T_d$, where $n(T_i) > 0$. Use lemma 16 to find an assignment $\tau$. Let $s_i = 1 + \tau_i/h(T_i)$ be the stretch factor of $T_i$. Let $p(v) = s_i d(v)$ be the placement of $v$. Order the vertices of $T$ in order of ascending $p(v)$: $v_0 = r(T), v_2, \ldots, v_{n(T)}$. Output the layout $f(v_j) = j$.*

The analysis outline of lemma 14 easily adapts to algorithm 3.

**Lemma 17.** *Let $T$ be a many-caterpillar. Suppose that for each $z \in [1,\, 2h(T)]$ the inequality*

$$E[X_z^{\log h(T)}(T)] \leq (c_0 \Theta(T) g(T))^{\log h(T)}$$

*holds, for some constant $c_0$ and function $g(T)$. Then the layout $f$ produced by algorithm 3 has bandwidth $B(f) = O(\Theta(T)g(T))$.*

*Proof.* The inequality

$$E\left[\sum_{z=1}^{2h(T)} X_z^{\log h(T)}(T)\right] \leq (C_0 \Theta(T) g(T))^{\log h(T)}$$

holds for $C_0 = 4c_0$. By lemma 16, an execution of algorithm 3 is equivalent to an execution of algorithm 2 for which

$$\sum_{z=1}^{2h(T)} X_z^{\log h(T)}(T) \leq (C_0 \Theta(T) g(T))^{\log h(T)}.$$

The lemma now follows from lemma 14 using $k(T) = \log h(T)$. □

## 4.2 Analysis of the algorithm

To complete the analysis of algorithm 3, we have to show that if we run algorithm 2 on a many-caterpillar $T$, then for all $z \in [1,\, 2h(T)]$ the inequality

$$E[X_z^{\log h(T)}(T)] \leq (c_0 \Theta(T) \log h(T))^{\log h(T)}$$

holds, for some constant $c_0$. We prove a more general inequality on caterpillars, then deduce the same inequality, with a different constant, for many-caterpillars.

When dealing with $X_z(T)$, only vertices of $T$ whose depth is at most $z$ matter, since only such vertices can be placed in $[z,\, z+1)$. This leads to the following definition.

**Definition 21.** *Let $T$ be a rooted tree, and $z \geq 0$ be an integer. Define*

$$V^{[z]}(T) = \{v \in V(T) : d(v) \leq z\} = N_{\downarrow}(r(T),\, z) \cup r(T).$$

*Let $T^{[z]}$ be the subtree of $T$ induced by $V^{[z]}(T)$, and denote $n(T^{[z]}) = n^{[z]}(T)$.*

**Lemma 18.** *Let $T$ be a many-caterpillar. Then $X_z(T) = X_z(T^{[z]})$.*

*Proof.* Recall $X_z(T)$ denotes the number of vertices $v$ of $T$ such that $p(v) \in [z,\, z+1)$ in an execution of algorithm 2. Recall next that $p(v) = \sigma_i d(v)$ for some $\sigma_i \geq 1$. Hence if $p(v) \in [z,\, z+1)$ then $d(v) \leq z$, so that $v \in V^{[z]}(T)$. The lemma follows from the nature of algorithm 2. $\square$

We can bound $n^{[z]}(T)$ using the thickness of $T$.

**Lemma 19.** *Let $T$ be a rooted tree. For every integer $z \geq 0$ we have $n^{[z]}(T) \leq z\Theta(T)$.*

*Proof.* By definition of $\Theta(T)$, $|N_{\downarrow}(r(T),\, z)| \leq z\Theta(T)$. The lemma follows. $\square$

In section 3.1 we defined the inventory of a caterpillar (definition 15). Recall that $n(T,\, h)$ is the number of vertices of $T$ at depth $h$. It is easy to estimate $E[X_z^k(T)]$ for a caterpillar $T$ using $n(T,\, h)$ for various $h$.

**Lemma 20.** *Let $T$ be a caterpillar, $z$ be an integer in $[1,\, 2h(T)]$ and $k \geq 1$ be an integer. Then*

$$E[X_z^k(T)] \leq \frac{8}{z} \sum_{h=\lceil z/2 \rceil}^{z} n(T,\, h)^k \leq \frac{8}{z} \sum_{x \in I(T)} x^k.$$

*Proof.* The random variable $X_z(T)$ depends on the random stretch $s$. The random stretch $s$ gets with equal probability the $h(T)$ values $s_\sigma = 1 + \sigma/h(T)$ for $\sigma \in [1,\, h(T)]$. We first determine the value of $X_z(T)$ given $s = s_\sigma$.

Recall that $X_z(T)$ counts the number of vertices $v$ such that $p(v) \in [z,\, z+1)$. Also recall that $p(v) = s_\sigma d(v)$, where $d(v)$ is the depth of $v$. Hence $X_z(T)$ counts the number of vertices $v$ such that $d(v) \in [z/s_\sigma,\, (z+1)/s_\sigma)$. The interval $I_\sigma = [z/s_\sigma,\, (z+1)/s_\sigma)$ is of length $1/s_\sigma$. Since $s_\sigma \geq 1$, $I_\sigma$ contains at most one integer $\lceil z/s_\sigma \rceil$ (it might contain no integers). Hence if $s = s_\sigma$ then $X_z(T) \leq n(T,\, \lceil z/s_\sigma \rceil)$.

If $z = 1$ then $X_z(T) \leq n(T,\, 1)$, and the lemma follows. Hence we can assume that $z \geq 2$. Let $z_\sigma = \lceil z/s_\sigma \rceil$. Then

$$E[X_z^k(T)] \leq \frac{1}{h(T)} \sum_{\sigma=1}^{h(T)} n(T,\, z_\sigma)^k.$$

The $z_\sigma$ range from $\lceil z/2 \rceil$ to $z$. Let $w$ be an integer in $[z/2,\, z]$. We determine how often $z_\sigma = w$. If $w = 1$ then $z = 2$ and so $z_\sigma = 1$ only for $\sigma = 2$. Next assume $w \geq 2$. The equation $z_\sigma = w$ is equivalent to $w - 1 < z/s_\sigma \leq w$. The last inequality is equivalent to $z/w \leq s_\sigma < z/(w-1)$. The length of the interval $I_w = [z/w,\, z/(w-1))$ is at most $z/w(w-1) \le 2z/w^2 \le 8/z$. Hence $I_w$ contains at most $8h(T)/z$ points of the form $1 + \sigma/h(T)$. Summarizing,

$$E[X_z^k(T)] \le \frac{8}{z} \sum_{h=\lceil z/2 \rceil}^{z} n(T,\, h)^k. \qquad \square$$

**Corollary 21.** *Let $T$ be a caterpillar, $z$ be an integer in $[1,\, 2h(T)]$ and $k \ge 1$ be an integer. Then*

$$E[X_z^k(T)] = O\Big(\frac{n^{[z]}(T)}{z}(C_2\Theta(T))^{k-1}k!\Big)$$

*for some constant $C_2$.*

*Proof.* Follows directly from lemma 18 and corollary 12. $\square$

Gupta's algorithm [8] differs from algorithm 2 by stretching the hairs independently of the spine. The proof of lemma 20 can be extended to Gupta's algorithm, and so corollary 21 is also true for that algorithm (with a different constant). Details omitted.

An analog of corollary 21 holds for many-caterpillars. This is proved using the relation $X_z(T) = X_z(T_1) + \cdots + X_z(T_d)$, employing the multinomial theorem. We begin by stating the multinomial theorem.

**Proposition 2** (Multinomial theorem). *Given $d$ numbers $x_1$ up to $x_d$ and an integer $k \ge 0$, the following identity holds:*

$$(x_1 + \cdots + x_d)^k = \sum_{\substack{p_1 + \cdots + p_d = k \\ p_i \ge 0 \text{ an integer}}} \frac{k!}{p_1! \cdots p_d!} x_1^{p_1} \cdots x_d^{p_d}.$$

We think of $p_1$ up to $p_d$ as a vector denoted $\hat{p}$. We need to differentiate vectors according to the number of indexes $i$ such that $p_i \ge 1$. The reason is that corollary 21 holds only for $k \ge 1$.

**Definition 22** (Multinomial vectors, Strength, Support). *A multinomial vector is a vector of non-negative integers. The elements of a multinomial vector $\hat{p}$ of length $d$ are denoted $p_1$ to $p_d$. The set of all multinomial vectors of length $d$ and sum $k$ is denoted* $\mathrm{Vec}(d,\, k)$.

*The strength of a multinomial vector $\hat{p}$ is the number* $\mathrm{str}(\hat{p})$ *of indexes $i$ such that $p_i \ge 1$. The support of $\hat{p}$ is the set* $\sup(\hat{p})$ *consisting of these indexes. The subset of* $\mathrm{Vec}(d,\, k)$ *consisting of vectors of strength $s$ is denoted* $\mathrm{Vec}(d,\, k;\, s)$.

We reformulate the multinomial theorem using the new notation.

**Proposition 3** (Multinomial theorem, reformulation)**.** *Given $d$ numbers $x_1$ up to $x_d$ and an integer $k \geq 0$, the following identity holds:*

$$(x_1 + \cdots + x_d)^k = k! \sum_{\hat{p} \in \mathrm{Vec}(d,\, k)} \prod_{i \in \sup(\hat{p})} \frac{x_i^{p_i}}{p_i!}.$$

Before proving the analog of corollary 21 on many-caterpillars we need two auxiliary lemmas. The first estimates the size of $\mathrm{Vec}(d,\, k;\, s)$.

**Lemma 22.** *Let $d \geq 1$, $k \geq 0$ and $s \in [1,\, d]$ be integers. The cardinality of* $\mathrm{Vec}(d,\, k;\, s)$ *is at most $2^k(ed/s)^s$.*

*Proof.* A multinomial vector $\hat{p}$ of strength $s$ can be reduced naturally to a multinomial vector $\mathrm{red}(\hat{p})$ of length $s$ by listing all the non-zero elements of $\hat{p}$ in their order. On the other hand, each positive vector in $\mathrm{Vec}(s,\, k)$ corresponds to $\binom{d}{s}$ vectors in $\mathrm{Vec}(d,\, k;\, s)$. Each positive vector in $\mathrm{Vec}(s,\, k)$ is an ordered partition of $k$. Since there are $2^{k-1}$ ordered partitions of $k$, $\mathrm{Vec}(s,\, k)$ contains at most $2^{k-1}$ positive vectors. Hence $|\mathrm{Vec}(d,\, k;\, s)| \leq 2^{k-1}\binom{d}{s} < 2^k(ed/s)^s$. ☐

The second auxiliary lemma shows that the point where certain symmetric functions are maximal is itself symmetric.

**Lemma 23.** *Let $n_1,\, \ldots,\, n_d$ be non-negative real variables constrained by the inequality $n_1 + \cdots + n_d \leq n$. For an integer $s \in [0,\, d]$, the maximum of*

$$\mathrm{sym}(\hat{n};\, s) = \mathrm{sym}(n_1,\, \ldots,\, n_d;\, s) = \sum_{\substack{I \subset \{1,\, \ldots,\, d\} \\ |I| = s}} \prod_{i \in I} n_i$$

*is attained when $n_i = n/d$. Moreover, the maximum point is unique if $s \geq 2$.*

*Proof.* Note that the domain in question is compact, and so the continuous function $\mathrm{sym}(\hat{n};\, s)$ attains a maximum under the constraints (including non-negativity). If $s = 0$ then clearly $\mathrm{sym}(\hat{n};\, s) = 1$ regardless of $\hat{n}$. For $s > 0$, it is clear that if $n_1 + \cdots + n_d < n$ then increasing $n_1$ increases sym. Hence we can assume that $n_1 + \cdots + n_d = n$. If $s = 1$ then $\mathrm{sym}(\hat{n};\, s) = n_1 + \cdots + n_d = n$ regardless of $\hat{n}$. So we can also assume that $s \geq 2$.

Consider an assignment $\hat{n}$ different from $n_i = n/d$. Thus $n_j \neq n_k$ for some $j$ and $k$. Define a new assignment $\hat{m}$ by letting $m_i = n_i$ for $i \notin \{j,\, k\}$, and $m_j = m_k = (n_j + n_k)/2$. Note that the $m_k$ are non-negative and that $m_1 + \cdots + m_d = n$. Define

$$\mathrm{sym}'(\hat{m};\, t) = \sum_{\substack{I \subset \{1,\, \ldots,\, d\} \setminus \{j,\, k\} \\ |I| = t}} \prod_{i \in I} m_i,$$

and define $\mathrm{sym}'(\hat{n};\, t)$ the same way. Note that $\mathrm{sym}'(\hat{m};\, t) = \mathrm{sym}'(\hat{n};\, t)$ for all $t$. Note also that $m_j + m_k = n_j + n_k$. Since $n_j \neq n_k$, we have $(n_j - n_k)^2 > 0$. Written differently, this is $(n_j + n_k)^2 > 4n_j n_k$ or $m_j m_k > n_j n_k$. The new assignment satisfies

$$\begin{aligned}
\mathrm{sym}(\hat{m};\, s) &= \sum_{\substack{I \subset \{1,\, \ldots,\, d\} \\ |I| = s}} \prod_{i \in I} m_i \\
&= \mathrm{sym}'(m;\, s) + (m_j + m_k)\,\mathrm{sym}'(m;\, s-1) + m_j m_k\,\mathrm{sym}'(m;\, s-2) \\
&\geq \mathrm{sym}'(n;\, s) + (n_j + n_k)\,\mathrm{sym}'(n;\, s-1) + n_j n_k\,\mathrm{sym}'(n;\, s-2) \\
&= \sum_{\substack{I \subset \{1,\, \ldots,\, d\} \\ |I| = s}} \prod_{i \in I} n_i = \mathrm{sym}(\hat{n};\, s).
\end{aligned}$$

Moreover, if either $s = 2$ or $n_i \neq 0$ for some $i \notin \{j,\, k\}$, then $\mathrm{sym}'(n;\, s-2) \neq 0$ and so $\mathrm{sym}(\hat{m};\, s) > \mathrm{sym}(\hat{n};\, s)$. It follows that the unique maximal assignment is $n_i = n/d$. □

We are now ready to present the many-caterpillar analog of corollary 21.

**Lemma 24.** *Let $T$ be a many-caterpillar, $z$ be an integer in $[1,\, 2h(T)]$ and $k \geq 1$ be an integer. Then for some constant $C_3$,*

$$E[X_z^k(T)] \leq O\Big(\frac{n^{[z]}(T)}{z}(C_3\Theta(T))^{k-1}k!\Big).$$

*Proof.* Suppose $T$ is obtained by identifying the roots of the caterpillars $T_1$ to $T_d$, where $n(T_i) > 0$. Clearly $n^{[z]}(T_1) + \cdots + n^{[z]}(T_d) = n^{[z]}(T)$. Moreover, the thickness of each $T_i$ is at most $\Theta(T)$.

To compute $E[X_z^k(T)]$ we use the equality $X_z(T) = X_z(T_1) + \cdots + X_z(T_d)$. We expand $E[X_z^k(T)]$ using the multinomial theorem (proposition 3) and the mutual independence of the $X_z(T_i)$:

$$E[X_z^k(T)] = k! \sum_{\hat{p} \in \mathrm{Vec}(d,\, k)} \prod_{i \in \mathrm{sup}(\hat{p})} \frac{E[X_z^{p_i}(T_i)]}{p_i!}. \tag{1}$$

By corollary 21, for some constant $c_2$ we have the inequality

$$E[X_z^p(T_i)] \le \frac{n^{[z]}(T_i)}{z} c_2^p \Theta(T_i)^{p-1} p! \le \frac{n^{[z]}(T_i)}{z} c_2^p \Theta(T)^{p-1} p!\,. \tag{2}$$

Substituting inequality (2) into (1) we get (recall $\operatorname{str}(\hat{p}) = |\sup(\hat{p})|$)

$$\begin{aligned} E[X_z^k(T)] &\le k! \sum_{\hat{p}\in \operatorname{Vec}(d,\,k)} \prod_{i\in\sup(\hat{p})} \frac{n^{[z]}(T_i)}{z} c_2^{p_i} \Theta(T)^{p_i-1} \\ &= k!(c_2\Theta(T))^k \sum_{\hat{p}\in \operatorname{Vec}(d,\,k)} (z\Theta(T))^{-\operatorname{str}(\hat{p})} \prod_{i\in\sup(\hat{p})} n^{[z]}(T_i) \\ &= k!(c_2\Theta(T))^k \sum_{s=1}^{d} (z\Theta(T))^{-s} \sum_{\hat{p}\in \operatorname{Vec}(d,\,k;\,s)} \prod_{i\in\sup(\hat{p})} n^{[z]}(T_i). \end{aligned} \tag{3}$$

For each $s$, the corresponding summand in the expression (3) is clearly a multiple of the function $\operatorname{sym}(n^{[z]}(T_1),\,\ldots,\,n^{[z]}(T_d);\,s)$ mentioned in lemma 23. Since $n^{[z]}(T_1) + \cdots + n^{[z]}(T_d) = n^{[z]}(T)$, it follows from lemma 23 that

$$\begin{aligned} E[X_z^k(T)] &\le k!(c_2\Theta(T))^k \sum_{s=1}^{d} (z\Theta(T))^{-s} \sum_{\hat{p}\in \operatorname{Vec}(d,\,k;\,s)} \prod_{i\in\sup(\hat{p})} \frac{n^{[z]}(T)}{d} \\ &= k!(c_2\Theta(T))^k \sum_{s=1}^{d} \left(\frac{n^{[z]}(T)}{d\,z\,\Theta(T)}\right)^s |\operatorname{Vec}(d,\,k;\,s)|. \end{aligned} \tag{4}$$

By lemma 19, $n^{[z]}(T) \le z\Theta(T)$. Using this and the estimate given by lemma 22, we obtain

$$\begin{aligned} E[X_z^k(T)] &\le k!(2c_2\Theta(T))^k \sum_{s=1}^{d} \left(\frac{e\,n^{[z]}(T)}{s\,z\,\Theta(T)}\right)^s \\ &\le k!(2c_2)^k \Theta(T)^{k-1} \frac{e\,n^{[z]}(T)}{z} \sum_{s=1}^{d} \left(\frac{e\,n^{[z]}(T)}{s\,z\,\Theta(T)}\right)^{s-1} \\ &\le k!(2c_2)^k \Theta(T)^{k-1} \frac{e\,n^{[z]}(T)}{z} \sum_{s=1}^{d} \left(\frac{e}{s}\right)^{s-1}. \end{aligned} \tag{5}$$

It is easy to see that the infinite series $\sum_{s=1}^{\infty} (e/s)^{s-1}$ converges. Hence the lemma follows from inequality (5) with $C_3 = 2c_2$. $\square$

Combining lemma 24 with lemma 17, we show that algorithm 3 has an approximation ratio of $O(\log h)$.

**Theorem 7.** *The layout $f$ produced by running algorithm 3 on a many-caterpillar $T$ satisfies $B(f) = O(\Theta(T)\log h(T))$.*

*Proof.* By Lemma 24, for each $z \in [1,\, 2h(T)]$ the inequality

$$E[X_z^{\log h(T)}(T)] \leq C_4 \frac{n^{[z]}(T)}{z}(C_3\Theta(T))^{\log h(T)-1}(\log h(T))!$$

holds for an execution of algorithm 2 on $T$ (assuming $\log h(T)$ is integral, a technicality), for some constant $C_4$. By lemma 19 we have $n^{[z]}(T)/z \leq \Theta(T)$, hence

$$E[X_z^{\log h(T)}(T)] \leq C_4(C_3\Theta(T))^{\log h(T)}(\log h(T))!$$

holds. Using $(\log x)! \leq (\log x)^{\log x}$ we infer that

$$E[X_z^{\log h(T)}(T)] \leq C_4(C_3\Theta(T)\log h(T))^{\log h(T)} \leq (C_5\Theta(T)\log h(T))^{\log h(T)},$$

where $C_5 = C_3C_4$ is a constant. The theorem now follows from lemma 17 using $g(T) = \log h(T)$ . □

**Corollary 25.** *Algorithm 3 has an approximation ratio of $O(\log h(T))$ on a many-caterpillar $T$.*

*Proof.* By lemma 2, $\Theta(T) \leq 2D(T)$. By lemma 4, $\Theta(T) \leq 2B(T)$. Hence algorithm 3 produces a layout $f$ with $B(f) = O(B(T)\log h(T))$. □

Since corollary 21 holds also for Gupta's algorithm, it follows that Gupta's algorithm also has an approximation ratio of $\log h$ on many-caterpillars.

## 5 References

1. G. Blache, M. Karpinski and J. Wirtgen, *On approximation intractability of the bandwidth problem*, ECCC TR98-014.

2. A. Blum, G. Konjevod, R. Ravi and S. Vempala, *Semi-definite relaxations for Minimum Bandwidth and other vertex-ordering problems*, STOC 30 (1998), 100–105.

3. P. Chinn, J. Chvátalová, A. K. Dewdney and N. Gibbs, *The bandwidth problem for graphs and matrices — a survey*, Journal of Graph Theory 6 (1982), 223–254.

4. J. Chvátalová, *On the bandwidth problem for graphs*, Ph.D. dissertation, University of Waterloo, 1980.

5. J. Dunagan and S. Vempala, *On Euclidean embeddings and bandwidth minimization*, RANDOM-APPROX 2001, 229–240.

6. U. Feige, *Approximating the bandwidth via volume respecting embeddings*, JCSS 60(3) (2000), 510–539.

7. M. R. Garey, R. L. Graham, D. S. Johnson and D. E. Knuth, *Complexity results for bandwidth minimization*, SIAM J. Appl. Math. 34 (1978), 477–495.

8. A. Gupta, *Improved bandwidth approximation for trees and chordal graphs*, J. Algorithms 40(1) (2001), 24–36.

9. J. Haralambides and F. Makedon, *Approximation algorithms for the bandwidth minimization problem for a large class of trees*, Theory Comput. System 30 (1997), 67–90.

10. J. Haralambides, F. Makedon and B. Monien, *Bandwidth minimization: an approximation algorithm for caterpillars*, Math. Systems Theory 24 (1991), 169–177.

11. R. Heckmann, R. Klasing, B. Monien and W. Unger, *Optimal embedding of complete binary trees into lines and grids*, J. Par. Dist. Comp. 49(1) (1998), 40–56.

12. M. Karpinski, J. Wirtgen and A. Zelikovski, *An approximation algorithm for the bandwidth problem on dense graphs*, ECCC TR97-017.

13. T. Kloks, D. Kratsch and H. Müller, *Approximating the bandwidth for asteroidal triple-free graphs*, ESA 95 (Lecture Notes in Computer Science 979), 434–447.

14. N. Linial, E. London and Y. Rabinovich, *The geometry of graphs and some of its algorithmic applications*, Combinatorica 15 (1995), 215–245.

15. J. Matoušek, *On embedding trees into uniformly convex Banach spaces*, Israel J. Math. 114 (1999), 221–237.

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

17. C. H. Papadimitriou, *The NP-completeness of the bandwidth minimization problem*, Computing 16 (1976), 263–270.

18. J. Saxe, *Dynamic programming algorithms for recognizing small-bandwidth graphs in polynomial time*, SIAM Journal on Algorithmic Methods 1 (1980), 363–369.

19. L. Smithline, *Bandwidth of the complete $k$-ary tree*, Disc. Math. 142 (1995), 203–212.

20. M. M. Sysło and J. Zak, *The bandwidth problem for ordered caterpillars, Report CS-80-065* at the Computer Science Department of Washington State University, 1980.

21. J. D. Ullman, *Computational aspects of VLSI*, Computer Science Press, 1984.

22. W. Unger, *The complexity of the approximation of the bandwidth problem*, FOCS 39 (1998), 82–91.

23. S. Vempala, *Random Projection: a new approach to VLSI layout*, FOCS 39 (1998), 389–395.