Simple Compact Monotone Tree Drawings

Anargyros Oikonomou[1] and Antonios Symvonis[2(✉)]

[1] School of Electrical and Computer Engineering,
National Technical University of Athens, Athens, Greece
[2] School of Applied Mathematical and Physical Sciences,
National Technical University of Athens, Athens, Greece
symvonis@math.ntua.gr

Abstract. A monotone drawing of a graph G is a straight-line drawing of G such that every pair of vertices is connected by a path that is monotone with respect to some direction.

Trees, as a special class of graphs, have been the focus of several papers and, recently, He and He [6] showed how to produce a monotone drawing of an arbitrary n-vertex tree that is contained in a $12n \times 12n$ grid.

In this paper, we present a simple algorithm that constructs for each arbitrary tree a monotone drawing on a grid of size at most $n \times n$.

1 Introduction

A *straight-line drawing* Γ of a graph G is a mapping of each vertex to a distinct point on the plane and of each edge to a straight-line segment between the vertices. A path $P = \{p_0, p_1, \ldots, p_n\}$ is *monotone* if there exists a line l such that the projections of the vertices of P on l appear on l in the same order as on P. A straight-line drawing Γ of a graph G is *monotone*, if a *monotone* path connects every pair of vertices.

Monotone graph drawing has gained the recent attention of researchers and several interesting results have appeared. Given a planar fixed embedding of a planar graph G, a planar monotone drawing of G can be constructed, but at the cost of some bends on some edges [2]. In the variable embedding setting, we can construct a planar monotone drawing of any planar graph without any bends [8].

One way to find a monotone drawing of a graph is to simply find a monotone drawing of one of its spanning trees. For that reason, the problem of finding monotone drawings of trees has been the subject of several recent papers, starting from the work by Angelini et al. [1] which introduced monotone graph drawings. Angelini et al. [1] provided two algorithms that used ideas from number theory and more specifically Stern-Brocot trees [3,11], [4, Sect. 4.5]. The first algorithm used a grid of size $O(n^{1.6}) \times O(n^{1.6})$ (BFS-based algorithm) while the second one used a grid of size $O(n) \times O(n^2)$ (DFS-based algorithm). Later, Kindermann

The work of Prof. Symvonis was supported by the iRead H2020 research grant (No. 731724).

F. Frati and K.-L. Ma (Eds.): GD 2017, LNCS 10692, pp. 326–333, 2018.
https://doi.org/10.1007/978-3-319-73915-1_26

et al. [9] provided an algorithm based on Farey sequence (see [4, Sect. 4.5]) that used a grid of size $O(n^{1.5}) \times O(n^{1.5})$. He and He [7] gave an algorithm based on Farey sequence and reduced the required grid size to $O(n^{1.205}) \times O(n^{1.205})$, which was the first result that used less than $O(n^3)$ area. Recently, He and He [5] firstly reduced the grid size for a monotone tree drawing to $O(n \log(n)) \times O(n \log(n))$ and, in a sequel paper, to $O(n) \times O(n)$ [6]. Their monotone tree drawing uses a grid of size at most $12n \times 12n$ which turns out to be asymptotically optimal as there exist trees which require at least $\frac{n}{12} \times \frac{n}{12}$ area [6].

Our Contribution: We provide a simple algorithm that given any n-vertex tree T, outputs a monotone drawing of T on a grid of size $n \times n$. Example drawings of our algorithm appear at Figs. 1, 2, 3, 4 and 5. Our algorithm does not employ number theory techniques but a rather simple weighting method and some simple facts from geometry that can be more analytically expressed. Due to space limitation, some proofs appear in the arXiv version of the paper [10].

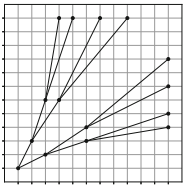

Fig. 1. 3-layer full binary tree (15 nodes).

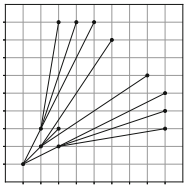

Fig. 2. 2-layer full ternary tree (13 nodes).

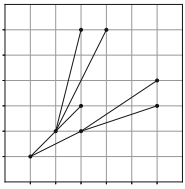

Fig. 3. Tree used in [5,6].

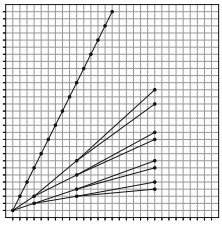

Fig. 4. 3-layer full binary tree plus path (29 nodes).

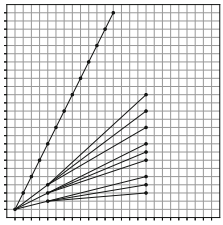

Fig. 5. 2-layer full ternary tree plus path (25 nodes).

2 Definitions and Preliminaries

Let Γ be a drawing of a graph G and (u, v) be an edge from vertex u to vertex v in G. The slope of edge (u, v), denoted by $slope(u, v)$, is the angle spanned by a counter-clockwise rotation that brings a horizontal half-line starting at u and

directed towards increasing x-coordinates to coincide with the half-line starting at u and passing through v. We consider slopes that are equivalent modulo 2π as the same slope. Observe that $slope(u, v) = slope(v, u) - \pi$.

Let T be a tree rooted at a node r. Denote by T_u the subtree of T rooted at a node u. By $|T_u|$ we denote the number of vertices of T_u. In the rest of the paper, we assume that all tree edges are directed away from the root.

In order to simplify the description of our algorithm, we extend the definition of *slope-disjoint* tree drawings given by Angelini et al. [1]. More specifically, a tree drawing Γ of a rooted tree T is called a *non-strictly slope-disjoint* drawing if the following conditions hold:

1. For every node $u \in T$, there exist two angles $a_1(u)$ and $a_2(u)$, with $0 \leq a_1(u) < a_2(u) \leq \pi$ such that for every edge e that is either in T_u or enters u from its parent, it holds that $a_1(u) < slope(e) < a_2(u)$.
2. For every two nodes $u, v \in T$ such that v is a child of u, it holds that $a_1(u) \leq a_1(v) < a_2(v) \leq a_2(u)$.
3. For every two nodes u_1, u_2 with the same parent, it holds that either $a_1(u_1) < a_2(u_1) \leq a_1(u_2) < a_2(u_2)$ or $a_1(u_2) < a_2(u_2) \leq a_1(u_1) < a_2(u_1)$.

The idea behind the original definition of slope-disjoint tree drawings is that all edges in the subtree T_u as well as the edge entering u from its parent will have slopes that *strictly* fall within the angle range $(a_1(u), a_2(u))$ defined for vertex u. $(a_1(u), a_2(u))$ is called the *angle range of* u with $a_1(u)$ and $a_2(u)$ being its *boundaries*. In our extended definition, we allow for angle ranges of adjacent vertices (parent-child relationship) or sibling vertices (children of the same parent) to share angle range boundaries. Note that replacing the "$\leq$" symbols in our definition by the "$<$" symbol gives us the original definition of Angelini et al. [1] for the slope disjoint tree drawings.

Lemma 1. *Every non-strictly slope-disjoint drawing of a tree T is also a slope-disjoint drawing.*

Theorem 1. [1] *Every slope-disjoint drawing of a tree is monotone.*

Theorem 2. *Every non-strictly slope-disjoint drawing of a tree is monotone.*

Based on geometry, we now prove that it is always possible to identify points on a grid that satisfy several properties with respect to their location.

Lemma 2 [See Fig. 6]. *Consider two angles θ_1, θ_2 with $0 \leq \theta_1 < \theta_2 \leq \frac{\pi}{4}$, and let $d = \lceil \frac{1}{\theta_2 - \theta_1} \rceil$. Then, edge e connecting the origin $(0, 0)$ to point $p = (d, \lfloor tan(\theta_1)d + 1 \rfloor)$ satisfies $\theta_1 < slope(e) < \theta_2$.*

Lemma 3 [See Fig. 7]. *Consider angles θ_1, θ_2 with $0 \leq \theta_1 < \theta_2 \leq \frac{\pi}{2}$ and let $d = \lceil \frac{1}{\theta_2 - \theta_1} \rceil$. Then, a grid point p such that the edge e that connects the origin $(0, 0)$ to p satisfies $\theta_1 < slope(e) < \theta_2$, can be identified as follows:*

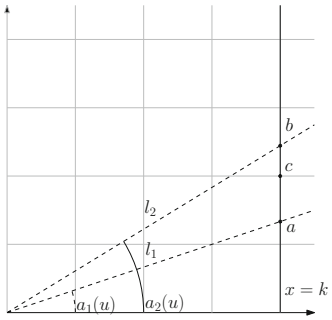

Fig. 6. Geometric representation of Lemma 2.

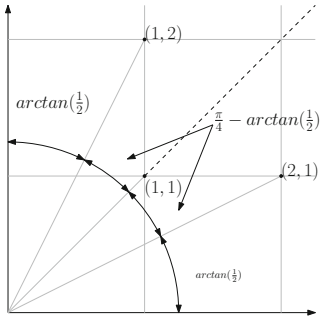

Fig. 7. Point, slopes angular sectors used in Lemma 3.

$$\theta_2 - \theta_1 > \tfrac{\pi}{4} : \quad p = (1,1)$$

$$\tfrac{\pi}{4} \geq \theta_2 - \theta_1 > arctan(\tfrac{1}{2}) : \begin{cases} p = (1,2) & if \ \ \theta_1 \geq \tfrac{\pi}{4} \\ p = (1,1) & if \ \ \tfrac{\pi}{4} > \theta_1 \geq arctan(\tfrac{1}{2}) \\ p = (2,1) & if \ \ arctan(\tfrac{1}{2}) > \theta_1 \end{cases}$$

$$arctan(\tfrac{1}{2}) \geq \theta_2 - \theta_1 : \begin{cases} p = (d, \lfloor tan(\theta_1)d + 1 \rfloor) & if \ \ \tfrac{\pi}{4} \geq \theta_2 > \theta_1 \geq 0 \\ p = (1,1) & if \ \ \theta_2 > \tfrac{\pi}{4} > \theta_1 \\ p = (\lfloor tan(\tfrac{\pi}{2} - \theta_2)d + 1 \rfloor, d) & if \ \ \theta_2 > \theta_1 \geq \tfrac{\pi}{4} \end{cases}$$

Moreover, if $p = (x, y)$ is the identified point, it also holds that:

$$\max(x, y) \begin{cases} \leq \frac{\pi}{2} \frac{1}{\theta_2 - \theta_1} & \text{if } \theta_2 - \theta_1 > arctan(\frac{1}{2}) \\ < \frac{1}{\theta_2 - \theta_1} + 1 & \text{if } arctan(\frac{1}{2}) \geq \theta_2 - \theta_1 \end{cases}$$

3 Monotone Tree Drawing on an $n \times n$ Grid

Our tree drawing algorithm will produce a non-strictly slope-disjoint tree drawing which, by Theorem 2, is monotone. We make the assumption that the given tree is rooted, otherwise, it can be rooted at any arbitrary node. In order to describe a non-strictly slope-disjoint tree drawing, we need to identify for each vertex u of the tree a grid point to draw u as well as to assign to it two angles $a_1(u)$, $a_2(u)$, with $a_2(u) > a_1(u)$. For every tree vertex, the identified grid point and the two angles should be such that the three properties of the non-strictly slope-disjoint drawing are satisfied.

The basic idea behind our algorithm is to split in a balanced way the angle range $(a_1(u), a_2(u))$ of vertex u to its children based on the size of the subtrees rooted at them. The following lemma formalizes this idea.

Lemma 4. *Let u be a node of the rooted tree T such that we already have assigned values for $a_1(u)$ and $a_2(u)$, with $a_1(u) < a_2(u)$. Let $u_1, u_2, \ldots, u_m$, $m \geq 1$, be the children of u in T. Then, the following assignment of a_1, a_2 for the*

children of u satisfies Property-2 and Property-3 of the non-strictly slope disjoint drawing:

$$a_1(u_i) = \begin{cases} a_1(u) & \text{if } i = 1 \\ a_2(u_{i-1}) & \text{if } 1 < i \leq m \end{cases}$$
$$a_2(u_i) = a_1(u_i) + (a_2(u) - a_1(u)) * \frac{|T_{u_i}|}{|T_u|-1}, \quad 1 \leq i \leq m$$

Observation 1. *If a vertex u has only one child, say u_1, then the angle assignment strategy of Lemma 4 assigns $a_1(u_1) = a_1(u)$ and $a_2(u_1) = a_2(u)$, which means that the child "inherits" the angle-range of its parent.*

Algorithm 1 describes our monotone tree drawing algorithm. It consists of three steps: Procedure AssignAngles which assigns angle-ranges to the vertices of the tree according to Lemma 4, Procedure DrawVertices which assigns each tree vertex to a grid point according to Lemma 3 and Procedure BalancedTreeMonotoneDraw which assigns the root to point $(0, 0)$ with angle-range $\left(0, \frac{\pi}{2}\right)$ and initiates the drawing of the tree.

Algorithm 1. Balanced Monotone Tree Drawing algorithm

```
1: procedure BalancedTreeMonotoneDraw
2: Input: An n-vertex tree T rooted at vertex r.
3: Output: A monotone drawing of T on a grid of size at most n × n.
4:     a1(r) ← 0, a2(r) ← π/2
5:     AssignAngles(r, a1(r), a2(r))
6:     Draw r at (0, 0)
7:     DrawVertices(r)
8: procedure AssignAngles(u, a1, a2)
9: Input: A vertex u and the boundaries of the angle-range (a1, a2) assigned to u.
10: Action: It assigns angle-ranges to the vertices of Tu.
11:    for each child ui of u do
12:        Assign a1(ui), a2(ui) as described in Lemma 4.
13:        AssignAngles(ui, a1(ui), a2(ui))
14: procedure DrawVertices(u)
15: Input: A vertex u that has already been drawn on the grid.
16: Action: It draws the vertices of Tu.
17:    for each child ui of u do
18:        Find a valid pair (x, y) as in Lemma 3 where θ1 ← a1(u), θ2 ← a2(u)
19:        If u is drawn at (ux, uy), draw ui at (ux + x, uy + y)
20:        DrawVertices(ui)
```

Lemma 5. *The drawing produced by Algorithm 1 is monotone.*

Proof. The angle-range assignment satisfies Property-2 and Property-3 of the non-strictly slope disjoint drawing as proved in Lemma 4. In addition, the assignment of the vertices to grid points satisfies Property-1 of the non-strictly slope disjoint drawing as proved in Lemma 3. Thus, the produced drawing by Algorithm 1 is non-strictly slope disjoint and, by Theorem 2, it is monotone. □

It remains to establish a bound on the grid size required by Algorithm 1. Our proof will use induction on the number of tree vertices having more than one child. The following lemma will be used as the basis of our induction.

Lemma 6. *Let T be an n-vertex rooted tree in which all vertices have at most one child, i.e., T is a path rooted at one of its endpoints. Then, Algorithm 1 draws T in the diagonal of an $n \times n$ grid.*

Lemma 7. *Let T be a rooted tree in which $k > 0$ of its vertices have at least two children. Let u be a vertex with at least two children and, moreover, every other vertex in T_u has at most one child. Let T' be the tree derived by replacing (in T) the subtree T_u by a path of length $|T_u|$. Then, the size of the grid which Algorithm 1 uses in the worst case for the drawing of T is smaller or equal to the size of the grid it uses in the worst case for the drawing of T'.*

Proof. Let u be a vertex as the one stated in the lemma, i.e., in T_u, u is the only vertex having at least two children. Let $u_1, u_2, \ldots u_m$, $m \geq 2$, be the children of u, and let T_{u_i} be the subtree rooted at u_i. Note that each T_{u_i} is a path. From Observation 1, we recall that for each node in T_{u_i}, the assigned values for a_1 and a_2 by Algorithm 1 will be the same as $a_1(u_i)$ and $a_2(u_i)$. Let $\phi(u) = a_2(u) - a_1(u)$. We consider two subcases based on whether $arctan(\frac{1}{2}) \geq \phi(u)$ or not.

Case-1: $arctan(\frac{1}{2}) \geq \phi(u)$**.** Since Algorithm 1 performs its angle-range assignment based on Lemma 4, for each child of u we have that $\phi(u_i) = a_2(u_i) - a_1(u_i) = \frac{|T_{u_i}|}{|T_u|-1}\phi(u)$. Observe that it also holds that $\phi(u_i) \leq arctan(\frac{1}{2})$.
A node in T_{u_i} is drawn, based on Lemma 3 where $\theta_1 \leftarrow a_1(u_i)$ and $\theta_2 \leftarrow a_2(u_i)$, in a grid of length at most $\frac{1}{\theta_2(u_i)-\theta_1(u_i)} + 1$ if its parent is considered to be drawn at the origin. So, the length of the total grid that is used for the drawing of path T_{u_i} is at most: $|T_{u_i}|(\frac{1}{\theta_2(u_i)-\theta_1(u_i)}+1) = |T_{u_i}|\frac{1}{\frac{|T_{u_i}|}{|T_u|-1}\phi(u)} + |T_{u_i}|$ $= \frac{|T_u|-1}{\phi(u)} + |T_{u_i}| \leq \frac{|T_u|-1}{\phi(u)} + |T_u| - 1 = (|T_u| - 1)(\frac{1}{\phi(u)} + 1)$.
The last term is the maximum grid length dictated by Lemma 3 for the drawing of a path of size $|T_u|$ with $\theta_1 \leftarrow a_1(u)$ and $\theta_2 \leftarrow a_2(u)$. Note also that in Algorithm 1 the largest grid devoted to any of $T_{u_i}, 1 \leq i \leq m$, determines the grid size of the drawing of T_u since the subtrees rooted at children of u are drawn completely inside non-overlapping (but possibly touching) angular sectors. The above statement holds because all the grids that will be used for the subtrees have the same origin (u) and all angular sectors lies in the first quadrant since Algorithm 1 assigns root with angle-range $\left(0, \frac{\pi}{2}\right)$. So, the grid size that is used in the worst case for the drawing of T_u by Algorithm 1 is smaller or equal to that used by it in the worst case for the drawing of a path of length $|T_u|$. Thus, the size of the grid which Algorithm 1 uses in the worst case for the drawing of T is smaller or equal to the size of the grid it uses in the worst case for the drawing of T'.

Case-2: $\phi(u) > arctan(\frac{1}{2})$**.** Let $\phi(u_i) = a_2(u_i) - a_1(u_i), 1 \leq i \leq m$. We consider two subcases based on whether $arctan(\frac{1}{2}) \geq \phi(u_i)$ or not.

Case-2a: $arctan(\frac{1}{2}) \geq \phi(u_i)$**.** A node in T_{u_i} is drawn, based on Lemma 3 where $\theta_1(u_1) \leftarrow a_1(u_i)$ and $\theta_2(u_i) \leftarrow a_2(u_i)$, in a grid of length at most $\frac{1}{\phi(u_i)} + 1 < \frac{\pi}{2}\frac{1}{\phi(u_i)}$, assuming that its parent is drawn at the origin. The last inequality holds for $\phi(u_i) \leq \frac{\pi-2}{2}$, and so it also holds for $\phi(u_i) \leq arctan(\frac{1}{2}) < \frac{\pi-2}{2}$. So, the maximum length of the total grid that is used for the drawing of path T_{u_i} is at most: $|T_{u_i}|\frac{\pi}{2}\frac{1}{\phi(u_i)} = |T_{u_i}|\frac{\pi}{2}\frac{1}{\frac{|T_{u_i}|}{|T_u|-1}\phi(u)} = \frac{\pi}{2}\frac{|T_u|-1}{\phi(u)}$.

Case-2b: $\phi(u_i) > arctan(\frac{1}{2})$**.** A node in T_{u_i} is drawn, based on Lemma 3 where $\theta_1(u_i) \leftarrow a_1(u_i)$ and $\theta_2(u_i) \leftarrow a_2(u_i)$, in a grid of length at most $\frac{\pi}{2} * \frac{1}{\phi(u_i)}$, assuming that its parent is drawn at the origin. So, the maximum length of the total grid that is used for the drawing of path T_{u_i} is: $|T_{u_i}|\frac{\pi}{2}\frac{1}{\phi(u_i)} = |T_{u_i}|\frac{\pi}{2}\frac{1}{\frac{|T_{u_i}|}{|T_u|-1}\phi(u)} = \frac{\pi}{2}\frac{|T_u|-1}{\phi(u)}$.

The last term in both subcases is the maximum grid length dictated by Lemma 3 for the drawing of a path of size $|T_u|$ with $\theta_1 \leftarrow a_1(u)$ and $\theta_2 \leftarrow a_2(u)$ where $\phi(u) > arctan(\frac{1}{2})$. So, the drawing of T_u uses in the worst case a grid length that is smaller or equal to that used in the worst case for the drawing of a path of length $|T_u|$, when both drawings are done by Algorithm 1. Thus, Algorithm 1 uses in the worst case for the drawing of T a grid of size smaller or equal to the one used in the worst case for the drawing of T'. □

Theorem 3. *Given a rooted Tree T, Algorithm 1 produces a monotone grid drawing using a grid of size at most $n \times n$.*

Proof. The monotonicity of the drawing follows directly from Lemma 5. By repeatedly applying Lemma 7, we get that in the worst case the drawing of T uses a grid length that is smaller or equal to the one used in the worst case for the drawing of a path of length $|T|$, when both drawings are done by Algorithm 1. By Lemma 6, we get that the used grid is of size at most $n \times n$. □

References

1. Angelini, P., Colasante, E., Di Battista, G., Frati, F., Patrignani, M.: Monotone drawings of graphs. J. Graph Algorithms Appl. **16**(1), 5–35 (2012)
2. Angelini, P., Didimo, W., Kobourov, S., Mchedlidze, T., Roselli, V., Symvonis, A., Wismath, S.: Monotone drawings of graphs with fixed embedding. Algorithmica **71**(2), 233–257 (2015)
3. Brocot, A.: Calcul des rouages par approximation, nouvelle methode. Revue Chronometrique **6**, 186–194 (1860)
4. Graham, R.L., Knuth, D.E., Patashnik, O.: Concrete Mathematics: A Foundation for Computer Science, 2nd edn. Addison-Wesley Longman Publishing Co. Inc., Boston (1994)
5. He, D., He, X.: Nearly optimal monotone drawing of trees. Theor. Comput. Sci. **654**, 26–32 (2016)
6. He, D., He, X.: Optimal monotone drawings of trees. CoRR abs/1604.03921 (2016)

7. He, X., He, D.: Compact monotone drawing of trees. In: Xu, D., Du, D., Du, D. (eds.) COCOON 2015. LNCS, vol. 9198, pp. 457–468. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21398-9_36
8. Hossain, M.I., Rahman, M.S.: Good spanning trees in graph drawing. Theor. Comput. Sci. **607**, 149–165 (2015)
9. Kindermann, P., Schulz, A., Spoerhase, J., Wolff, A.: On monotone drawings of trees. In: Duncan, C., Symvonis, A. (eds.) GD 2014. LNCS, vol. 8871, pp. 488–500. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45803-7_41
10. Oikonomou, A., Symvonis, A.: Simple compact monotone tree drawings. CoRR abs/1708.09653v2 (2017). http://arxiv.org/abs/1708.09653v2
11. Stern, M.: Ueber eine zahlentheoretische funktion. Journal fur die reine und angewandte Mathematik **55**, 193–220 (1858)