Eberhard Karls Universität Tübingen
Wilhelm Schickard Institut Tübingen

Fachbereich Informatik

On Maximizing The Euclidian Distance Between Vertices In Drawings Of Series Parallel Graphs

Arbeitsbereich Algorithmik

zur Erlangung des akademischen Grades
Master of Science

Autor: Benjamin Çoban
MatNr. 3526251

Version vom: 6. April, 2022

ErstprüferIn: Prof. Dr. Michael Kaufmann
ZweitprüferIn: Prof. Dr. Ulrike von Luxburg

Zusammenfassung

Abstract

Erklärung

Hiermit erkläre ich, dass ich diese schriftliche Abschlussarbeit selbstständig verfasst habe, keine anderen als die angegebenen Hilfsmittel und Quellen benutzt habe und alle wörtlich oder sinngemäß aus andern Werken übernommenen Aussagen als solche gekennzeichnet habe.

Datum, Ort, Unterschrift

Contents

1 Introduction

Thoughts:

1. General graph drawing introduction
2. Graph Drawing Symposium introduction
3. why is maximizing distances between vertices of interest?
4. Figures of readability
5. Thesis structure

The topic of visualization of information relationships occur in various areas of work. Examples of the fields include circuit design, architecture, web science, social sciences, biology, geography, information security and software engineering. Over the last decades, many different efficient algorithms were developed for graph drawings in the Euclidean plane.
Different quality measures for graph drawings have been considered, including area, angular resolution, slope number, average edge length, and total edge length [4], addressing the readability and aesthetics

Starting from a workshop in 1994, the first international conference for *Graph Drawing* was held in Passau in 1995 [12]. The annual symposium covers topics of combinatorical and algorithmic aspects of graph drawing as well as the design of network visualization systems and interfaces.
One part of the symposium is the *Graph Drawing Contest*. The contest consists of two parts - the *Creative Topics* and the *Live Challenge*. The main focus for the Creative Topics lies on the creation of drawings of two given graphs. Aspects to consider for the visualization are clarity, aesthetic appeal and readability.
On the other hand, the Live Challenge is held similar to a programming contest. Participants, usually teams, will get a theme and a set of graphs and will have one hour of processing. The results will be ranked and the team with highest score wins the competition. The teams will be allowed to use any combination of software and human interaction systems in order to produce the best results. Usually, the challenge is derived from a theoretical optimization problem [8].

In 2021, the Live Challenge during the 29th International Symposium on Graph Drawing and Network Visualization held in Tübingen, Germany addressed the optimization of graph drawing edge lengths. An *edge length ratio* of a drawing describes the proportion between the *minimal and maximal edge lengths*. The size of the total area of a drawing affects the maximum edge length. When considering *straight line drawings*, where edges are a single straight line segment, the ratio scales in proportion of the total area size.
For the *Live Challenge*, the goal was to produce a *polyline graph drawing*, where edges are line segments joined together, with uniform edge lengths. The difficulty of this challenge was intensified by constraints on the drawing area and the amount of line segments per edge [9].

In 2022, the 30th International Symposium of Graph Drawing and Network Visualization held in Tokio, Japan [10] addresses an alternation of the Live Challenge from previous year. In contrast to the edge length ratio from 2021, this years ratio describes the proportion of the *maximal polyline edge length* to the *minimal Euclidian distance* between two adjacent vertices.

This thesis contains the examination of maximization of the *Euclidian distance* between two adjacent vertices in small area drawings of certain graph classes. In Section 1, the preliminaries and terminology are defined. In Section 2, the general problem considering the edge length ratio is formalized. Furthermore, the potential for ratio improvement is illustrated by allowing polyline edges. Section 3 describes a drawing algorithm for the graph class of k-*ary trees* which guarantees a satisfying edge length ratio. Section 4 contains drawing algorithms for the graph class of *series parallel graphs*. The subclass of *outerplanar* graphs and 2-*trees* are of particular interest. Those drawings will improve the worst case ratio behaviour described in Section 2. Section 5

2 Preliminaries

2.1 Definitions And Terminology

A *graph* $G = (V, E)$ is a tuple consisting of two sets - the set of vertices $V = V(G)$ and the set of edges $E = E(G)$. An *edge* $e = (v, w), v, w \in V$ is a tuple and describes a connectivity relation between two vertices. If $V' \subseteq V, E' \subseteq E$, then $G' = (V', E')$ is a *subgraph* of G. The *degree* of a vertex states the amount of edges incident to the vertex.
A *path* of length k from a vertex v_1 to v_{k+1} is a sequence of vertices $(v_1, ..., v_{k+1})$ such that (v_i, v_{i+1}) is an edge in G. A path is *simple* if all the vertices in the sequence are distinct. A *cycle* is a path where $v_1 = v_{k+1}$ and has at least one edge. A graph with no cycles is called *acyclic* [5, P. 1170].
Unless otherwise mentioned, the graphs are *undirected*, meaning that the edge (u, v) is identical to the edge (v, u). An undirected graph is *connected* if every vertex is reachable from all the other vertices [5, P. 1170]. A graph is *biconnected* if the removal of any vertex still leaves the graph connected [7, P. 224].
A graph is *simple* if it does contain neighter multiple edges nor self loops. On the other hand, if a graph contains multiple edges or self loops, it is called a *multigraph* [5, P. 1172]. Unless otherwise mentioned, a graph is presumed to be simple.
A graph is *planar* if and only if there exists a crossing-free representation in the plane [5, Page 100]. A *face* is a maximal open region of the plane bounded by edges. The *outer face* is the unbounded face. A bounded face is called *inner face* [6, S. 86].
A multigraph G^* is the *dual graph of* G if and only if there exists a bijective function between G^* and G such that:

1. Every face f in G corresponds to a vertex v_f in G^*
2. For every edge e of G, the corresponding vertices of the faces in G^* incident to e get an edge
3. If e is incident to only one face, a loop is attached to the corresponding vertex in G^*

[6, P. 103] The *weak dual graph* of G is the dual graph of G without considering the outer face.

2.2 Graph Drawing Models And Representations

An $n \times n$ *grid* is a graph consisting of n rows and n columns of vertices. The vertex in the i-th row and j-th column is denoted as (i, j) and is called a *grid point*. All vertices in a grid have exactly four neighbours, except for the boundary vertices [5, P. 760]. One *unit length* values the distance between two adjacent vertices on the grid and is denoted as `UL`. A *drawing* Γ of a graph G is a function, where each vertex is mapped on a unique point $\Gamma(v)$ in the plane and each edge is mapped on an open Jordan curve $\Gamma(e)$ ending in its vertices [7, P. 225]. In this context, a graph will be drawn on an underlying grid. An *embedding* of G is the collection of counter-clockwise circular orderings of edges around each vertex of V, denoted as a sequence of edges.

In a *straight-line drawing*, verticees are points on the grid and edges are straight-line segments.
In a *polyline drawing*, vertices are points on the grid, edges are sequences of contiguous straight-line segments. The transition point between two edge segments is called a *bend*. Like vertices, bends are placed on points on the underlying grid.
A *box* is an axis-parallel rectangle, overlapping vertical and horizontal grid lines. The *width* of a box is one unit smaller than the number of vertical grid lines that are overlapped by it. The *height* of a box is one unit smaller than the number of horizontal grid lines that are overlapped by it. In an *orthogonal box drawing*, vertices are axis-aligned boxes (possibly degenerated to a line segment or a point), edges are sequences of contiguous horizontal or vertical line segments. [3, P. 144ff]
A *layering* is a mapping $L : V \to \mathbb{N}$ and determines the horizontal grid line placement of a vertex. A layering is *valid*, if $|L(u) - L(v)| \geq 1$ for any edge (u, v) [11, P. 4].
A drawing whose minimum enclosing box has width w and height h is called a $w \times h$-drawing and inherits area $w \cdot h$ [3, P. 145].
The *euclidian distance* between two grid points (x_1, y_1) and (x_2, y_2) is defined as $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. The *length of a line segment* is defined as the euclidian distance between the end points. The *length of a polyline* is defined as the sum of the individual line segment lengths.

2.3 The Relationship Between Drawing Models

2.4 Graph Classes

2.4.1 Tree

A graph T is called *tree* if and only if it is connected, acyclic and undirected [5, P. 1172].
A tree is *rooted* if one of its vertices is distinguished from the other ones, called the *root*. When considering a path from the root to any other vertex w, then, the following holds:

1. Besides w, every vertex of this path is an *ancestor* of w
2. For an edge (v_i, v_{i+1}), v_i is called the *parent* of v_{i+1}, and v_{i+1} is a child of v_i.

A vertex with no children is called a *leaf*. Any vertex which is not a leaf is called an *internal node*. The length of the simple path from the root to any vertex v denotes the *depth* of v in T. A *level* of T consists of all vertices of the same depth. The *height* of T is equal to the largest depth of any vertex of T. [5, P. 1176ff]

2.4.2 k-ary Tree

A *k-ary tree* is a rooted tree in which for every vertex has at most k children. A *complete k-ary tree* is a k-ary tree in which all leaves have the same depth and all internal nodes have k children.

2.4.3 Outerplanar Graphs, Series Parallel Graphs and 2-Trees

An *outerplanar graph* is a planar graph that can be drawn such that all vertices are on the outer face. A *maximal outerplanar graph* is an outerplanar graph to which it is not possible to add an edge without destroying the simplicity, planarity or outerplanarity property. A *2-terminal series-parallel graph* with terminals s, t is a recursively defined graph with one of the following three rules:

1. An edge (s, t) is a 2-terminal series-parallel graph
2. If $G_i, i = 1, 2,$ is a 2-terminal series-parallel graph with terminals s_i, t_i, then in the serial composition t_1 is identified with s_2 to obtain a 2-terminal series-parallel graph with s_1, t_2 as terminals
3. If $G_i, i = 1, ..., k,$ is a 2-terminal series-parallel graph with terminals s_i, t_i, then in a parallel composition we identify all s_i into one terminal s and all t_i into the other terminal t and the result is a 2-terminal series-parallel graph with terminals s, t.

A *series-parallel graph*, *SP-graph* in short, is a graph for which every biconnected component is a 2-terminal series-parallel graph. A SP-graph is *maximal* if no edge can be further added while maintaining a SP-graph. [3, P. 143ff]
A *k-tree* is a recursively defined graph with at least $k+1$ vertices. If $n = k+1$, then the k-tree is the complete graph K_{k+1}. If $n > k+1$, start with a K_{k+1} and every vertex added is adjacent to exactly k adjacent neighbours. For a 2-tree, it holds that $k = 2$. The class of 2-trees correspond to the class of maximal SP-graphs [1, Page 2].

2.5 Tools

2.5.1 SPQR Tree

A *cut vertex* in a graph G is a vertex whose removal disconnects G. A *separation pair* in G is a pair of vertices whose removal disconnects G. A *biconnected component* of G is a maximal (in terms of vertices and edges) biconnected subgraph of G. If G contains vertices s, t, then G is *st-biconnectable* if $G \cup \{s, t\}$ is biconnected. A *split pair* of G is either a separation pair or a pair of adjacent vertices of G. A *maximal split component* of G in regard to a split pair $\{u, v\}$ is either an edge (u, v) or a maximal subgraph G' of G such that G' contains u and v and $\{u, v\}$ is not a split pair of G'. A vertex w aside from u and v belongs to exactly one maximal split component. a *split component* of $\{u, v\}$ is defined as the union of any number of maximal split components of $\{u, v\}$. A split pair $\{u, v\}$ is *maximal*, if there is no split pair $\{w, z\}$ in G such that $\{u, v\}$ is contained in a split component of $\{w, z\}$.
The *SPQR-Tree* $\mathcal{T}$ of G is a recursively defined composition of G with respect to its split pairs. $\mathcal{T}$ is a rooted tree with four types of nodes: S, P, Q and R. Any node μ of $\mathcal{T}$ is related to a planar uv-biconnectible multigraph, the so-called *skeleton* of μ, denoted as $sk(\mu)$. $\mathcal{T}$ is recursively defined - Let (s, t) be an edge of G, called the *reference edge*. $\mathcal{T}$ is initialized with a Q node ϕ as root, representing the edge (s, t). The skeleton of ϕ consists of two parallel edges (s, t). One is a *real* edge, one is a *virtual* edge.

After the initialization of $\mathcal{T}$ with an arbitrary reference edge, consider a node ψ of $\mathcal{T}$, $G_\psi = (V(G), E(G) \setminus \{(s,t)\})$ and a pair of vertices $\{u,v\}$ of G_ψ, called the *poles* of ψ.

Trivial case If G_ψ consists of a single edge (u,v), then ψ is a Q-node.

Series case If G_ψ is not a single edge and not biconnected, then ψ is a S-node with at least one cut vertex between the path between u and v.

Parallel case If G_ψ is not a single edge, but biconnected with $\{u,v\}$ as a split pair of G_ψ, then ψ is a P-node.

Rigid case If G_ψ is not a single edge, biconnected with $\{u,v\}$ not being a split pair of G_ψ, then ψ is a R-node.

[2, P. 7-8]

2.5.2 Tree decomposition

Let G be a graph, T a tree, and let $\mathcal{W} = (W_t)_{t \in T}$ be a family of vertex sets $W_t \subseteq V_G$ indexed by the vertices t of W. The pair (TW) is called a *tree decomposition* of G if it satisfies the following three conditions:

1. $V_G = \bigcup_{t \in T} W_t$
2. For every edge $e \in E_G$ there exists a $t \in T$ such that both ends of e lie in W_t
3. For all $v \in V$, there exists a connected subtree T' in T such that $v \in W_{t'}, t' \in V_{T'}$

[6, P. 319] The *width* of a tree decomposition is defined as

$$tw(T, W) = \max\{|W_t|, t \in V_T\} - 1 \tag{1}$$

The *treewidth* of a graph is the least width of any tree decomposition of G [6, P. 321].

3 Initial Situation

3.1 Formalization Of The Problem

3.1.1 The edge-length ratio

Let Γ_G be a given planar polyline drawing. The length of an edge is defined as the sum of $k+1$ line segments, induced by k bends. $l_{\max}$ is the length of the longest edge in Γ_G, $l_{\min}$ is the minimal Euclidian distance between two adjacent vertices in Γ_G. Then, the edge-length ratio r of Γ_G is defined as:

$$r_{\Gamma_G} = \frac{l_{\max}}{l_{\min}} \tag{2}$$

It trivially holds, that $r \geq 1$, since the length of every polyline with at least one bend between two vertices is naturally longer than the Euclidian distance between those. r is said to be *optimal* if $r = 1$. Then, all the edges in a drawing are straight-lines and of the same length.

3.1.2 Upper bound of the ratio

There exist multiple straight-line drawing algorithms which produce a drawing for a planar graph in area $\mathcal{O}(n) \times \mathcal{O}(n)$. The area consumption of a straight-line drawing directly induces the bounds for the ratio. Let $k \times k$ be the area consumption of a bounding square Γ_G is drawn on, $k \in \mathcal{O}(n)$. The maximal length of a straight-line is then bound by $\sqrt{2}k$, from one corner of the grid to the diagonal opposing one, while $l_{\min}$ might value 1 `UL`. The ratio therefore values $\sqrt{2}k \in \mathcal{O}(n)$ in the worst case.
This automatically gives an upper bound for any poly-line drawing Ω_G since a straight-line drawing can be seen as a polyline drawing with zero bends. Including bends in a straight-line drawing enables the possibility to reposition vertices in order to maximize the Euclidian distances.

3.1.3 $\mathcal{NP}$-hardness

3.2 On the edge-length ratio of 2-trees

The class of 2-trees is of particular interest in this thesis due to their balance in restricted properties on the one hand, and having non-trivial approaches and results for general problems on the other hand. This effect is pointed out by previous results regarding the edge-length ratio.
2-trees are biconnected, but not triconnected. This property implies a high amount of possible embeddings for a given 2-tree G, since parallel subgraphs can be permuted and flipped. Therefore, finding drawings with an optimization regarding a specific problem require combinatorial and algorithmic approches.
It was proven that the ratio of straight-line drawings of 2-trees is unbounded, meaning, that for a given constant r, there exists a sufficiently large 2-tree G with $r_G > r$ over all its straight-line drawings. One result states that the ratio lies in the gap between the lower bound $\Omega(\log n)$ and the upper bound $\mathcal{O}(n^{0.695})$ [4, P. 2].

3.3 The Symposium Challenge

The input consists of a JSON file with the following entries:

nodes Every node has an unique ID value between 0 and the amount of nodes - 1, a value for the x and y coordinate each, delimited by the width and height

edges Every edge has an ID for source and destination each and an optional list of bend points, specified in x and y coordinate

width (optional) The maximum x-coordinate of the grid. If unspecified, the width is set to 1,000,000.

height (optional) The maximum y coordinate of the grid. If unspecified, the height is set to 1,000,000.

bends The maximum number of bends allowed per edge

The results of the optimization are also JSON files. The planarity of the graph shall be preserved and the ratio minimized by relocation of the nodes.
For teams participating with their own tools, an embedding might not be given with the input. For participants working manually, an embedding is already given beforehand.

4 k-ary Trees

When analyzing a general problem of graph drawings, considering trees may come in handy due to their assessable properties. The first graph class considered for minimizing the ratio in a drawing is therefore the class of k-ary trees. This section describes a drawing algorithm for a k-ary tree T, guaranteeing a nearly optimal euclidian ratio in the resulting drawing Γ_T. The total area of Γ_T will depend on the height of T.
Since any k-ary tree is acyclic per definition, T is connected, but not biconnected and the treewidth of T equals 1. Having n vertices, T inherits exactly $n-1$ edges. The small bags in a tree decomposition and the low amount of edges makes k-ary trees accessible for a straightforward solution for a given problem.
When working on other graph classes, it may be possible to find a feasible solution in a reduction to an instance of a tree. In fact, this effect will occur in the subsequent section of this thesis.

4.1 The Drawing Algorithm

Lemma 1. *The* height h *of a complete k-ary tree T is in* $\mathcal{O}(\log n)$*.*

Proof.

$$n = \sum_{i=0}^{h} k^i = \frac{k^{h+1}-1}{k-1} \tag{3}$$

$$\Leftrightarrow h = \log_k((k-1)n+1) - 1 \tag{4}$$

$$= \frac{\log((k-1)n+1)}{\log k} - 1 \tag{5}$$

$$\underbrace{\Rightarrow}_{k \text{ constant}} \quad h \in O(\log n) \tag{6}$$

□

Theorem 1. *Every k-ary tree admits a planar straight-line drawing with a nearly optimal ratio, apart from a rounding error, on area $\mathcal{O}(n^2 \log n)$.*

Proof. The following drawing will be constructed from top to bottom, meaning that the y-coordinates of the children of any vertex v are smaller than the y coordinate of v.
Let $r := k^h$. For a vertex v in height i, consider k equidistant columns with x-coordinates between $x(v)-(k-1)\cdot k^{h-h'-1}$ and $x(v)+(k-1)\cdot k^{h-h'-1}$. These are integer coordinates since the distance between two columns next to each other equals $2 \cdot \frac{k^{h-i}}{k}$. Draw a circle around v with radius r. Choose the grid points v_i on the column closest to the resulting intersections with the constraint that $y(v_i) \leq y(v)$ and connect v with its k children with a straight-line.

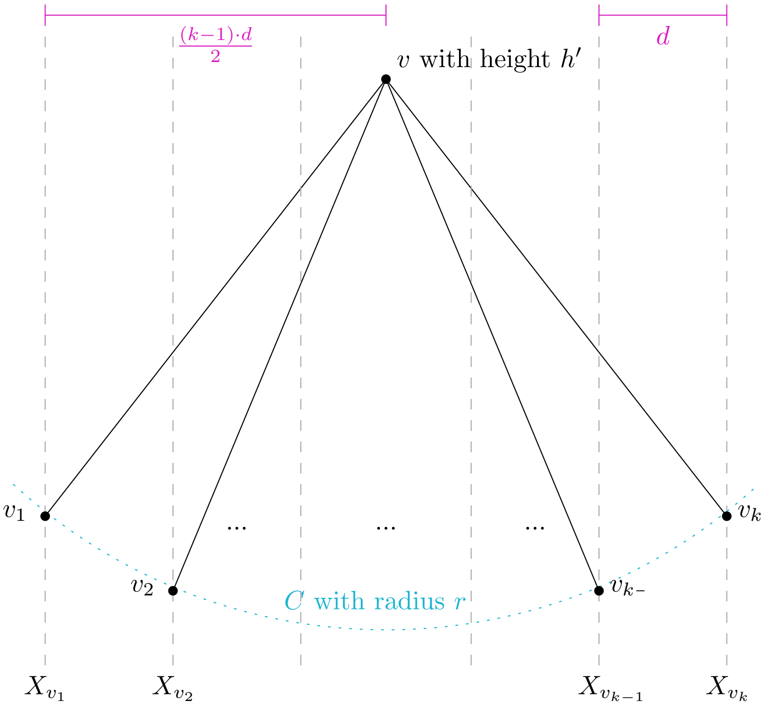

Figure 1: Illustration of drawing algorithm at a vertex v with height h

The distance between two neighbouring columns in height i suffices for the remaining drawing since it holds for the remaining heights:

$$\underbrace{2 \cdot \sum_{j=i}^{h-1} k^{h-j-1}}_{\text{drawn from both columns}} = 2 \cdot \sum_{z=0}^{h-i-1} h^z \tag{7}$$

$$= 2 \cdot \frac{k^{h-i} - 1}{k - 1} < 2 \cdot k^{h-i} \tag{8}$$

The height of the drawing is bound by $h \cdot r = h \cdot k^h \in \mathcal{O}(n \cdot \log n)$. The width is bound by $2 \cdot \sum_{i=0}^{h} k^i = 2 \cdot \frac{k^{h+1}-1}{k-1} \in \mathcal{O}(n)$, resulting in $\mathcal{O}(n^2 \log n)$ area. Since the algorithm works from top to bottom and for height i, the area for every subtree is disjointedly reserved, the resulting drawing is planar. Furthermore, all straight-line edges inherit a length of approximately k^h, no bends were used and the ratio is bound by $1 + \varepsilon, 0 \leq \varepsilon < 1$. $\square$

The following algorithm sums up the approach described above.

Algorithm 1: Drawing algorithm for k-ary trees

Input: complete k-ary tree T,k

Output: Planar drawing of T with nearly optimal ratio

$h \leftarrow$ height of T

$r \leftarrow k^h$

Draw $root(T)$ on any grid point

`Draw`$(root(T), r, h)$

Algorithm 2: `Draw_`k`-ary_Children`(v, r, h)

Input: Already drawn vertex v, radius and height $r, h \in \mathbb{N}$
Output: Coordinates of all the children of v

if v *leaf* **then**
    return
else
    $h' \leftarrow$ `height`(v)
    $d \leftarrow 2 \cdot k^{h-h'-1}$
    $C \leftarrow v$.`DrawCircle`(r)
    `/* Draw circle with radius` r `around` v `*/`
    for $i \in [1..k]$ **do**
        $x(v_i) \leftarrow x(v) - \frac{(k-1)\cdot d}{2} + (i-1) \cdot d$
        `/*` x`-coordinate of` i`-th child of` v `*/`
        $X \leftarrow$ `Column`$(x(v_i))$
        `/* Identify the column at position` $x(v_i)$ `*/`
        $s \leftarrow$ `Intersection`(I, X)
        `/* Calculate the intersection of the circle` C `and the column` X `*/`
        $y(v_i) \leftarrow round(y(s))$
        `DrawStraightLine`(v, v_i)
        `Draw_`k`-ary_Children`(v_i, r, h)

4.2 Example Drawings

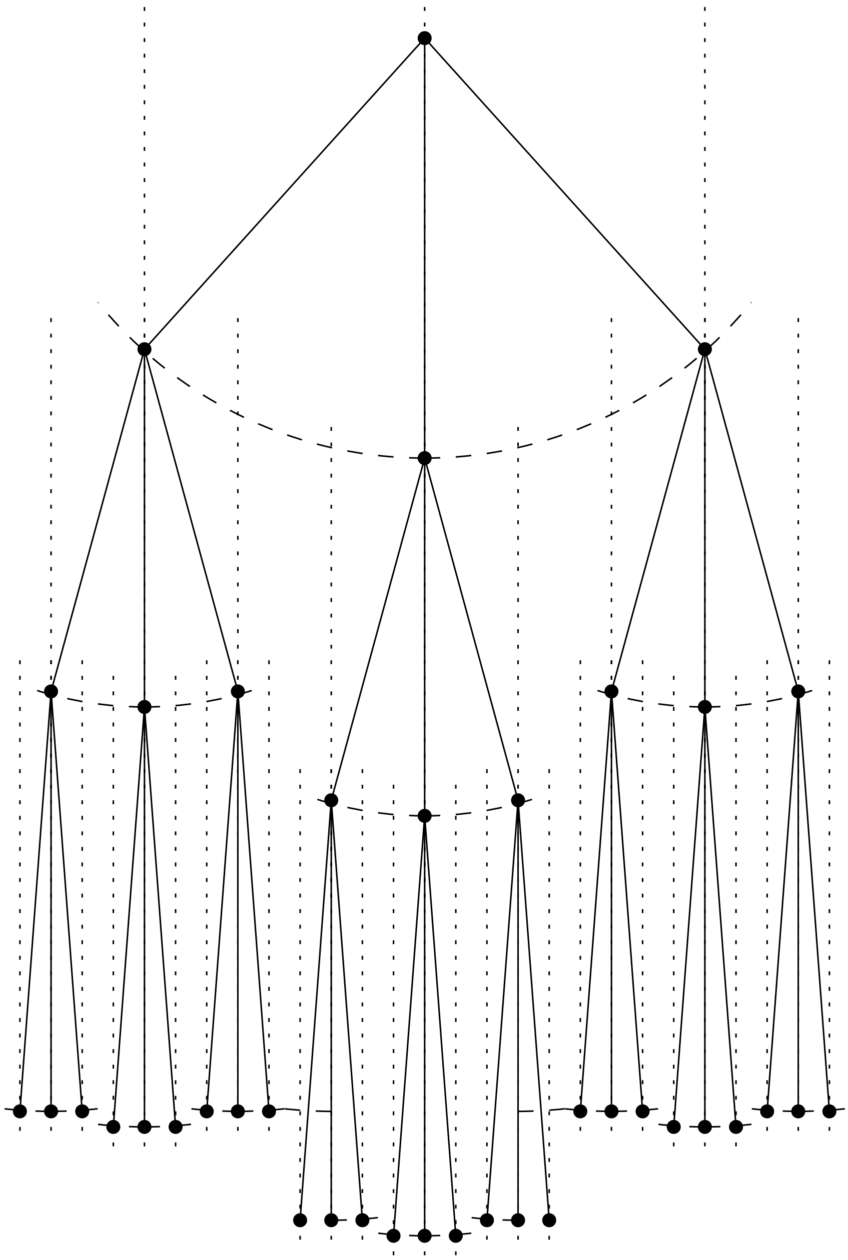

Figure 2: Tertiary tree with height 3

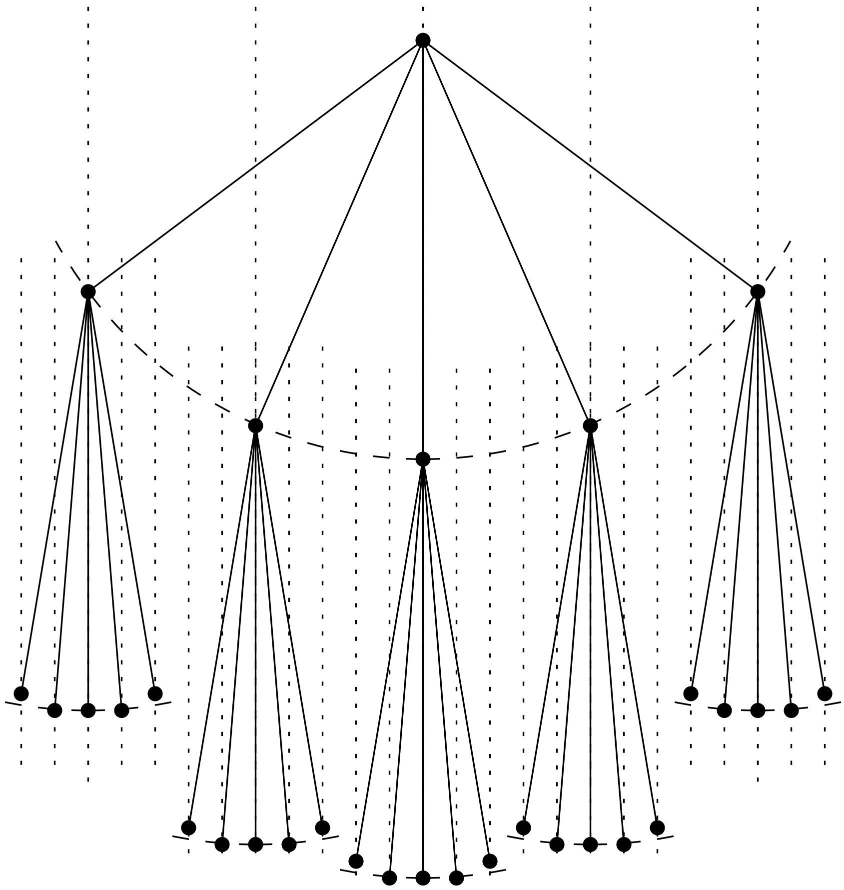

Figure 3: 5-ary tree with height 2

5 Series-Parallel Graphs

This thesis addresses approaches of ratio minimization for drawings of series parallel graphs.

5.1 Outerplanar Graphs

At first, the attention is drawn to *outerplanar graphs* as they are a subclass of SP graphs. There will be two polyline drawing algorithms presented for this class of SP graphs. One resulting drawing algorithm for outerplanar graphs is suited to be extended for 2-trees, since every maximal SP graph contains a maximal outerplanar graph as a subgraph.

5.1.1 Properties Of Outerplanar Graphs

Lemma 2. *A maximal outerplanar graph G inherits triangles as inner faces, except for the outerface.*

Lemma 3. *The weak dual graph G^* of a maximal outerplanar graph G is a simple tree with maximum degree 3 for any vertex.*

Proof. The weak dual graph G^* is connected since G is maximal outerplanar. Suppose, that G^* contains a cycle $\mathcal{C}$. Then, there exists a vertex in G which is enclosed from the outerface by faces according to $\mathcal{C}$ in G^* and G is not outerplanar. This implies that G^* must be acyclic and considering the connectedness, G^* is a tree. Since any face f is a triangle, the degree of v_f in G^* values at most three. The simplicity is derived from the maximal outerplanarity property. If there were multiple edges between vertex v_f and $v_{f'}$ in G^*, then there would be at least one vertex in G which does not lie on the outerface. □

Lemma 4. *Let G be a maximal outerplanar graph with n vertices and G^* the dual graph excluding the outerface, a rooted tree with degree up to three for every vertex v_f. Then, the height of G^* ranges between $\Omega(\log n)$ and $\mathcal{O}(n)$.*

Proof. Since G is a planar graph, it contains $\mathcal{O}(n)$ faces. The rooted tree G^* inherits the following property:

1. The root has at most three children
2. The subtrees rooted at the children of the root are binary

Placing $\mathcal{O}(n)$ vertices in three binary trees connected to a root vertex results in a height of at least $\Omega(\log n)$ due to the k-ary tree height property from Lemma 1. In the worst case, the dual graph will be a chain of vertices, therefore a rooted tree with height $\mathcal{O}(n)$. □

Lemma 5. *A maximal outerplanar graph G can be extended to a maximal outerplanar supergraph G'.*

Proof. A new vertex can be added to G by adding a new vertex v_f in the dual graph G^* so that the degree of G^* is still at most 3. The newly created face f must lie on the outerface and must be a triangle. Otherwise, the outerplanarity property is destroyed. □

When G^* inherits a height of $\mathcal{O}(\log n)$, a new problem emerges. When starting drawing the root of G^*, new vertices are added in all directions, enclosing more and more area along the iterative drawing. This results in short euclidian distances relative to the longest edge, increasing the ratio.
In the worst case, G^* inherits the following properties:

1. The root vertex has exactly three children
2. Every other inner node has exactly two children. In other words, the subtrees adjacent to the root vertex are complete binary trees of height $h-1$

A maximal outerplanar graph with its weak dual graph G^* fulfilling these properties will be referred as *complete*.

5.1.2 Drawing A Complete Maximal Outerplanar Graph With One Bend

A given maximal outerplanar graph can be drawn by using a drawing algorithm for its weak dual graph. In section 4, the k-ary tree drawing algorithm produces a straight-line drawing in $\mathcal{O}(n^2 \log n)$ area. The drawing algorithm 1 can be used with a minor modification to draw the weak dual graph of any complete maximal outerplanar graph G, since G^* is a subtree of a 3-ary tree with the same height.

Algorithm 3: Drawing algorithm for complete maximal outerplanar graphs

Input: A complete maximal outerplanar graph G
Output: Γ_G with one bend

$G^* \leftarrow$ weak dual graph of G with minimal height
$h \leftarrow$ `height(`G^*`)`
`root` $\leftarrow G^*$`.root`
`Draw(root)`
`Draw_3-ary_Children(root,`3^h`,1)`
for $v \in$ *`root.children`* **do**
    `Draw_2-ary_Children(`v`,`2^{h-1}`,`$h-1$`)`
for $v \in G^*$ **do**
    $x(v) \leftarrow 3 \cdot x(v)$
`/*` G^* `is drawn and stretched by a factor of 3. Now, draw` G `by starting with a triangle around the root and for every other node of` G^*`, place bend points and new vertex point */`
`DrawVertex(`x`(root),` y`(root)y+`$h \cdot 3^h$`)`
`DrawVertex(`x`(root) ,` y`(root) )`
`DrawVertex(`x`(root) ,` y`(root) )`
for $v \in G^* \setminus \{$*`root`*$\}$ **do**
    `DrawBendPoint(`x`(v)`-1`,` y`(v))`
    `DrawBendPoint(`x`(v)`$+1$`,` y`(v))`
    `DrawVertex(v.x, v.y`-1`)`

5.1.3 Analysis

5.2 Series Parallel Graphs

5.2.1 Properties Of Series Parallel Graphs

5.2.2 Drawing An Outerplanar Graph With Two Bends

5.2.3 Analysis

5.2.4 Drawing A 2-Tree With Two Bends

5.2.5 Analysis

6 Summary

7 Extensional Work

8 Future Work

9 Acknowledgements

I would like to thank Prof. Michael Kaufmann for instructing this final thesis. Further I appreciate helpful discussions with Dr. Henry Förster. I want to thank especially my family for their patience and encouragement.

References

[1] Carlos Alegria et al. "Planar Straight-line Realizations of 2-Trees with Prescribed Edge Lengths". In: *CoRR* abs/2108.12628 (2021). arXiv: 2108.12628. URL: https://arxiv.org/abs/2108.12628.

[2] Patrizio Angelini et al. "2-Level Quasi-Planarity or How Caterpillars Climb (SPQR-)Trees". In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*. Ed. by Daniel Marx. SIAM, 2021, pp. 2779–2798. DOI: 10.1137/1.9781611976465.165. URL: https://doi.org/10.1137/1.9781611976465.165.

[3] Therese C. Biedl. "Small Drawings of Outerplanar Graphs, Series-Parallel Graphs, and Other Planar Graphs". In: *Discret. Comput. Geom.* 45.1 (2011), pp. 141–160. DOI: 10.1007/s00454-010-9310-z. URL: https://doi.org/10.1007/s00454-010-9310-z.

[4] Vaclav Blazj, Jiri Fiala, and Giuseppe Liotta. "On Edge-Length Ratios of Partial 2-Trees". In: *Int. J. Comput. Geom. Appl.* 31.2-3 (2021), pp. 141–162. DOI: 10.1142/S0218195921500072. URL: https://doi.org/10.1142/S0218195921500072.

[5] Thomas H. Cormen et al. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. ISBN: 978-0-262-03384-8. URL: http://mitpress.mit.edu/books/introduction-algorithms.

[6] Reinhard Diestel. *Graph Theory, 4th Edition*. Vol. 173. Graduate texts in mathematics. Springer, 2012. ISBN: 978-3-642-14278-9.

[7] Christian A. Duncan and Michael T. Goodrich. "Planar Orthogonal and Polyline Drawing Algorithms". In: *Handbook on Graph Drawing and Visualization*. Ed. by Roberto Tamassia. Chapman and Hall/CRC, 2013, pp. 223–246.

[8] *Graph Drawing Symposium 2021*. Graph Drawing Committee. URL: https://algo.inf.uni-tuebingen.de/gd2021/index.php?id=contest (visited on 03/21/2022).

[9] *Graph Drawing Symposium 2021 Contest - Live Challenge*. Graph Drawing Committee. URL: http://mozart.diei.unipg.it/gdcontest/contest2021/index.php?id=live-challenge (visited on 03/21/2022).

[10] *Graph Drawing Symposium 2022 Contest - Live Challenge*. Graph Drawing Committee. URL: http://mozart.diei.unipg.it/gdcontest/contest2022/challenge.html (visited on 03/22/2022).

[11] Ulf Rüegg et al. "A Generalization of the Directed Graph Layering Problem". In: *Graph Drawing and Network Visualization - 24th International Symposium, GD 2016, Athens, Greece, September 19-21, 2016, Revised Selected Papers*. Ed. by Yifan Hu and Martin Nöllenburg. Vol. 9801. Lecture Notes in Computer Science. Springer, 2016, pp. 196–208. DOI: 10.1007/978-3-319-50106-2_16. URL: https://doi.org/10.1007/978-3-319-50106-2%5C_16.

[12] *Symposia*. Graph Drawing Committee. URL: http://www.graphdrawing.org/symposia.html (visited on 03/17/2022).