1 k-trees

A new family of graphs is investigated for its edge-length ratio optimization on a fixed grid. A k-tree is an undirected graph which is incrementally created from a K_{k+1} in a way that each added vertex has exactly k neighbours such that those $k+1$ vertices form a clique. Following this definition, a 4-tree is not planar since it starts with a K_5. Of interest is the 2-tree and the 3-tree.

1.1 2-trees

For a 2-tree, each vertex is added to exactly two neighbours, forming a clique of size 3. Starting at a K_3, adding a vertex will create a new face since it forms a clique. For n vertices, it holds:

$$|E| = 2n - 3 \quad n \geq 3 \tag{1}$$

Proof by induction. IA: K_3 has three edges and three vertices, $3 = 2 \cdot 3 - 3$✓
IV+IS: Suppose, the statement is true for a 2-tree T_i of size i. Adding a vertex to T_i such that T_{i+1} is still a 2-tree will connect the vertex v_{i+1} to two neighbours in T_i, increasing the edge count by two.

$$|E_{T_{i+1}}| = |E_{T_i}| + 2 = 2 \cdot i - 3 + 2 = 2 \cdot (i+1) - 3$$

□

Following Euler's Formula, the amount of faces for a 2-tree of size n are:

$$|F| + |V| - |E| = 2$$
$$|F| = 2 + |E| - |V| = 2 + 2n - 3 - n = n - 1$$

In a planar drawing, every edge is shared by exactly two faces. So when a vertex is added to a 2-tree drawing, there are at least two faces, where the vertex can be placed. In the figure below, an example is illustrated where a vertex can be placed in more than two faces.

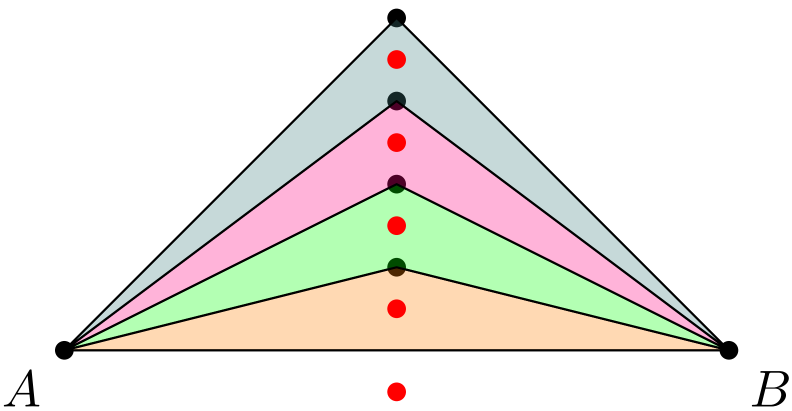


Figure 1: In this drawing, a new vertex, forming a clique with A and B, could be placed in five faces, including the outerface, marked as red dot

The choice of face in which the vertex is drawn into might affect the edge-length ratio directly. One problem of a recursive drawing approach is that we do not know which problems might arise when deciding a face for a placement. So, we need a way to evaluate a suitable embedding for a given 2-tree graph.

Lemma 1 (Expansion Lemma)**.** *If a graph G is k-connected and G' is obtained by adding a vertex with at least k neighbours in G, then G' is k-connected.*

Proof. Since G is k-connected, there are k disjoint paths between every pair of vertices. Adding a vertex v with k neighbours will result in k disjoint paths between v and every other vertex of G' since G was already k-connected. □

Theorem 1. *A 2-tree is biconnected, but not necessarily triconnected.*

Proof. K_3 is biconnected since it is a cycle of size three. Since the 2-trees are recursively defined, adding subsequent vertices with exactly two neighbours will not harm the property of biconnectedness. In Figure 1, this 2-tree serves as a counterexample for triconnectedness. A separation pair is $\{A, B\}$. Therefore, not every 2-tree is triconnected. □

By Spezielle Kapitel der Algorithmik Lecture about graph embeddings, the amount of embeddings for a biconnected graph is quite a lot. With k parallel subgraphs, there are $k!$ permutations and every subgraph can be flipped additionally. Therefore, the amount of embeddings of biconnected graphs lies in $\mathcal{O}(n!2^n)$. Maybe, an SQPR Tree decomposition might help finding a "good"embedding regarding the edge-length ratio.

1.1.1 First approach: Choice of vertex placement

When adding a vertex, it will form a clique with pre-existing neighbours. This very edge is part of two faces. In this approach, the vertex added will be placed in the face with the larger area. The face will then be subdivided into two new faces. What happens, when the vertex is placed on the outerface?

Lemma 2. *Let G be a drawing of a 2-tree and v a vertex added to G connecting to the neighbours defined by an edge on the outerface. Then, presuming, there is enough free area for it, v can be placed on the outerface in a way that the edge-length ratio of G' does increase derived by a rounding error of the grid.*

Proof. Let $l_{\max}$ be the length of the longest edge and $l_{\min}$ be the length of the shortest edge in the 2-tree G. We obtain G' by adding a vertex v. If this vertex' neighbours are on the outerface and there is a free area box of size at least $l_{\min} \times l_{\min}$, then v can be placed on the outerface with a distance between $l_{\min}$ and $l_{\max}$ to its neighbours and the edge-length ratio may increase when the edge-length ratio of G was at 1. In this case, the placement of v might not be possible to achieve edge lengths of $l_{\max}$. There will be a rounding error, depending on the granularity of the grid. □

So, placing vertices on the outerface will not increase the edge-length ratio significantly, presuming, there is enough free area to do so. This should work with any arbitrary number of bends per edge.
Now, we will investigate how the edge-length ratio behaves when a vertex is placed inside of a face. We will start with a drawing of a K_3, one bend allowed. We place a vertex inside of the K_3, creating new edges. Then, we will recursively add vertices to an edge which was created by the previous addition. The following algorithm will draw

this special 2-tree:

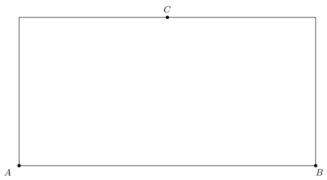
Input: Γ: Drawing of a K_3, $V_{\text{add}} = \{(v_i, e_i)\}$ vertices added and $|V_{\text{add}}| = m$
Output: Drawing of a 2-tree with $m + 3$ vertices

InsertVertex(v,e) = {
  Choose the face adjacent to e with the larger area except for the outerface
  Place the vertex and the edges with bends in this face in a way that the face is subdivided evenly in area
} **while** $V_{add} \neq \emptyset$ **do**
  for $i \in [1..|V_{add}| = m]$ **do**
    if $e_i.drawnIn\Gamma$ **then**
      InsertVertex(v_i,e_i)
      $V_{\text{add}} \leftarrow V_{\text{add}} \setminus \{(v_i, e_i)\}$

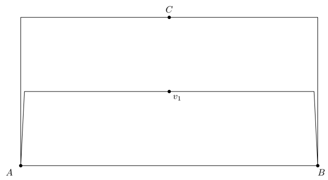
In the special case described above, the set V_{add} is initialized as follows:

$$(v_1, e_1) \in V_{\text{add}} \quad e_1 \text{ arbitrary edge of } K_3$$
$$(v_i, e_i) \in V_{\text{add}} \quad e_i \text{ edge created by inserting } v_{i-1}$$

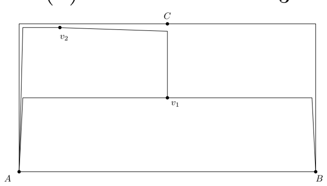
In this special case, the runtime of the algorithm above is in linear time. It is of interest, how the edge-length ratio will behave. In the following figures, vertex and bend placements were done intuitively.



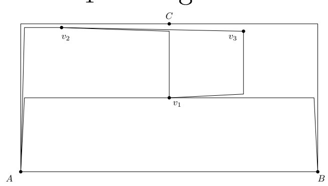
(a) Start with a K_3



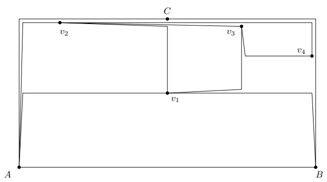
(b) Insert v_1 to neighbours A and B



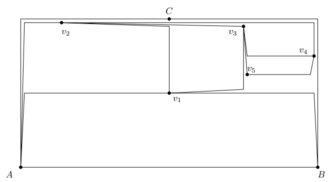
(c) Insert v_2 to neighbours A and v_1



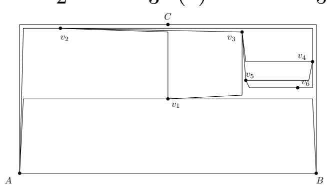
(d) Insert v_3 to neighbours v_1 and v_2



(e) Insert v_4 to neighbours v_2 and v_3



(f) Insert v_5 to neighbours v_3 and v_4



(g) Insert v_6 to neighbours v_4 and v_5

The edge-length ratio worsens with every inserted vertex. The shortest edge created by the i-th insertion has $\approx \frac{3}{4}$ length of the previous shortest edge. Therefore, the edge-length ratio in the i-th insertion values $\left(\frac{4}{3}\right)^i$. For n vertices, the edge-length ratio of a 2-tree lies in $\mathcal{O}(2^n)$ in the worst case with this approach and one bend allowed per edge. Note, that it was not allowed to draw on the outerface.

1.2 Next steps

Improve the worst case result If this approach is a correct one, find a way to improve the exponential worst case of the edge-length.

Evaluate a given 2-tree before drawing it If we knew which embedding would be suitable for a given graph, we could improve the edge-length ratio behaviour by avoiding worst-case scenarioes as far as possible.

Investigate 3-trees Since 3-trees are triconnected, there is one embedding disregarding flip. How does the edge-length ratio behave? Which face would be suitable to put on the outerface?