Report Research Session2
MaxSPED

Birgit Schreiber

11.12.2017

1 Recap last session

We can solve MaxSPED for a intersection graph C which is a tree with $k = 3$ using dynamic programming.

1.1 Definitions

- Graph G and given drawing Γ of G with at most k intersections per edge
- Resulting intersection graph C of Γ (each edge in Γ corresponds to a vertex in C, vertices in C are connected if edges in Γ intersect)
- Stub lengths of edges in Γ O_{in}, O_1, O_2, ...

2 Cacti

In this section cacti with $k \leq 3$ are discussed. A cactus can not have chords and therefore it can be split as shown in figure 1.

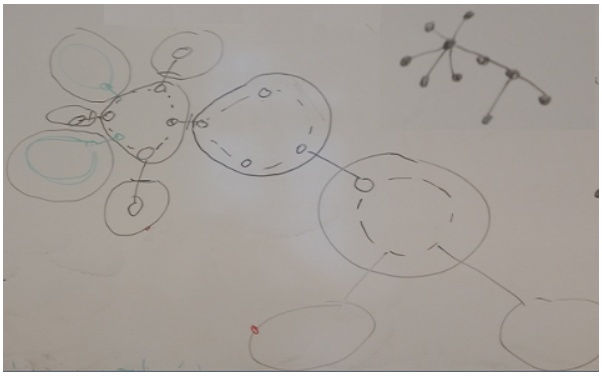

Figure 1: Breakdown of a cactus into circles

We can calculate each circle with the algorithm we have for circles with $k = 2$ from the paper. This algorithm gives us O_{in}, O_1 and O_2 for the vertex which connects the circle to the graph. We interpret each circle as a node in a tree which is traversed bottom up. When a circle has children in this tree the results of this "children-circles" are already known. We use them as the stub length of this vertex and can compute the next circle in the same way. This works because the child does not affect the next step in the circle. A child can only influence the nodes which it is connected to. This is illustrated in figure 2. This allows to solve this problem in polynomial time.

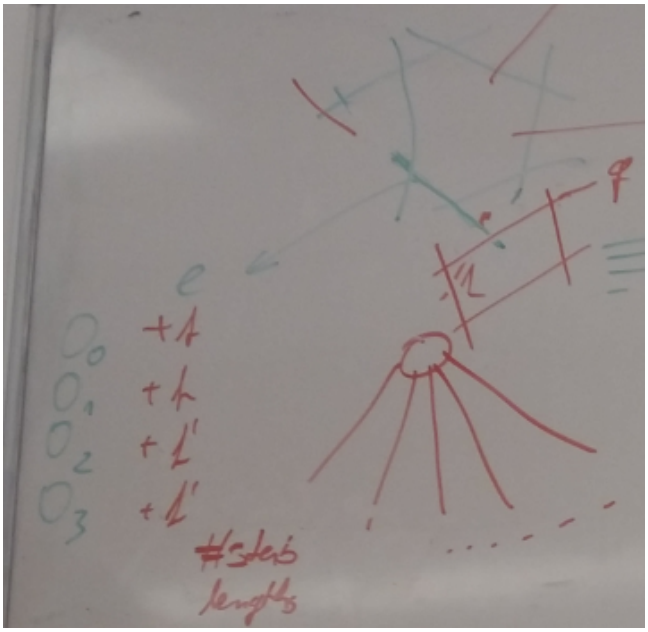

Figure 2: Influence of child

3 General Trees

The runtime guess for trees with degree k is $O(k^2n)$. For each edge in the Graph G k different values have to be calculated (O_{in}, O_1, ..., O_k). Because each edge in G can have at most $n - 1$ intersection $k < n$ holds. Therefore overall kn values are calculated. For each node in the tree which corresponds to an edge in G we have to the sum over the maxima of the children. This is shown in figure 3.

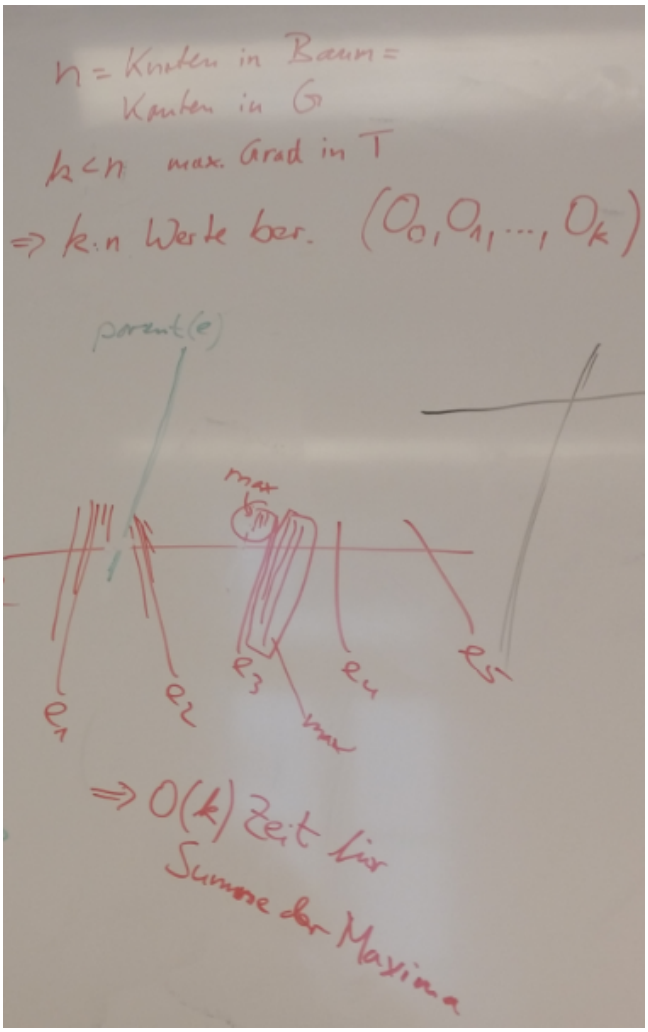

Figure 3: Calculation for one edge

Since each stub is introduced by an edge we can improve the runtime guess if we sort the stubs by their length when calculating the intersection points. The calculation of O_i for $i = 1...k$ is executed in ascending order of the stub lengths. The formula for calculating the values can be seen in figure 4.

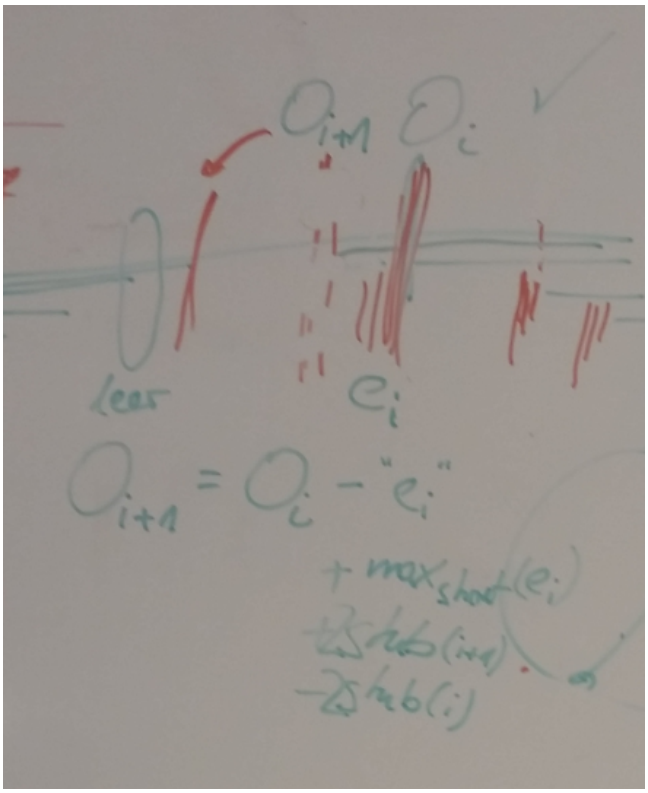

Figure 4: Calculation of O_i

With this all k values for one edge can be calculated in $O(k)$ time which gives us an overall runtime for trees with degree k of $O(nk)$ (if $k > \log n$, otherwise $O(n \log n)$ form calculating intersections).

4 Results

So far we have solutions for the following intersection graphs:

- Trees for $k = 3$ in $O(n)$
- Trees for $k > 3$ in $O(nk)$
- Cacti with $k = 3$ in $O(nk)$
- Bipartite graphs which correspond to unit length grids

5 Open Questions and Possible Next Steps

- We can try in the next step to allow non crossing chords. How can we break this in independent subproblems?
- Is there another planar graph class we can try to solve?
- Can we find an algorithm that works for general planar intersection graphs?
- Can we use the dual graph of the ear decomposition?
- What happens if we allow cacti with $k = 4$?