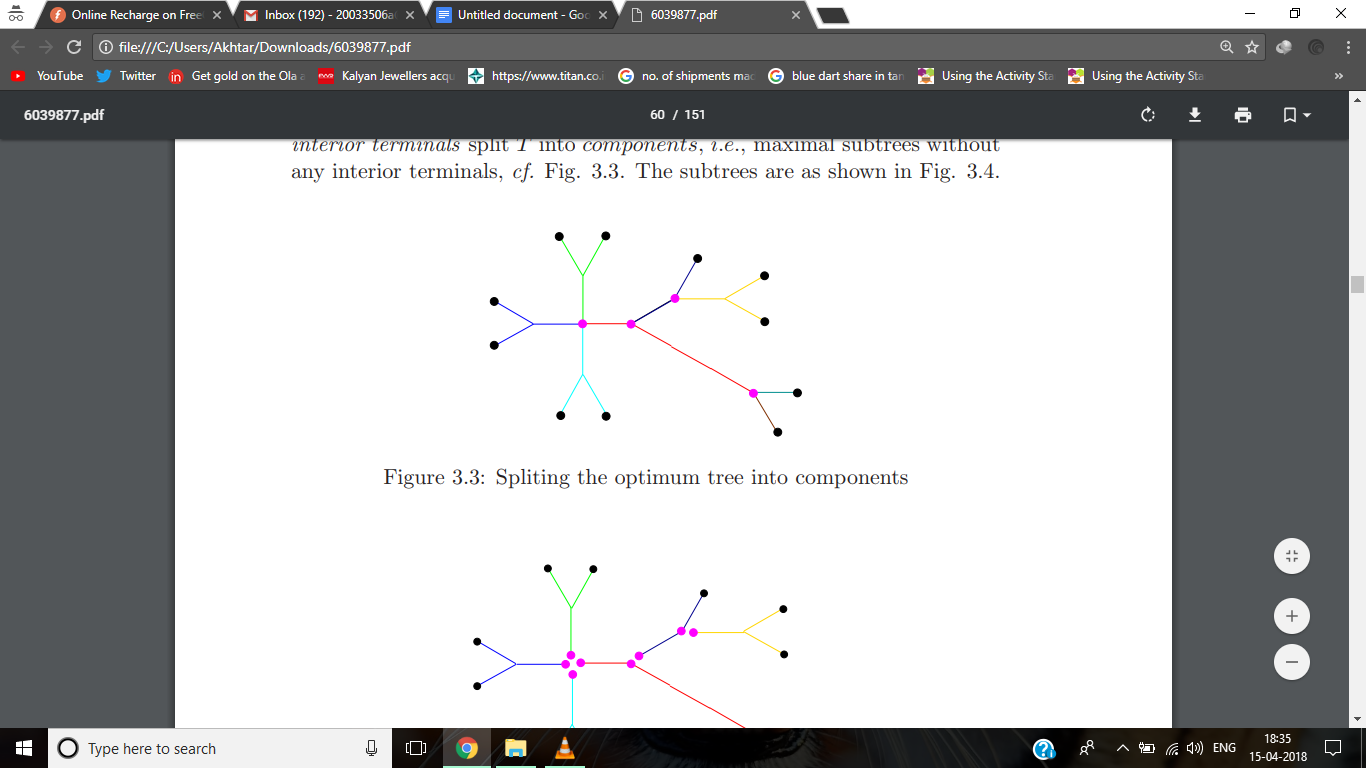
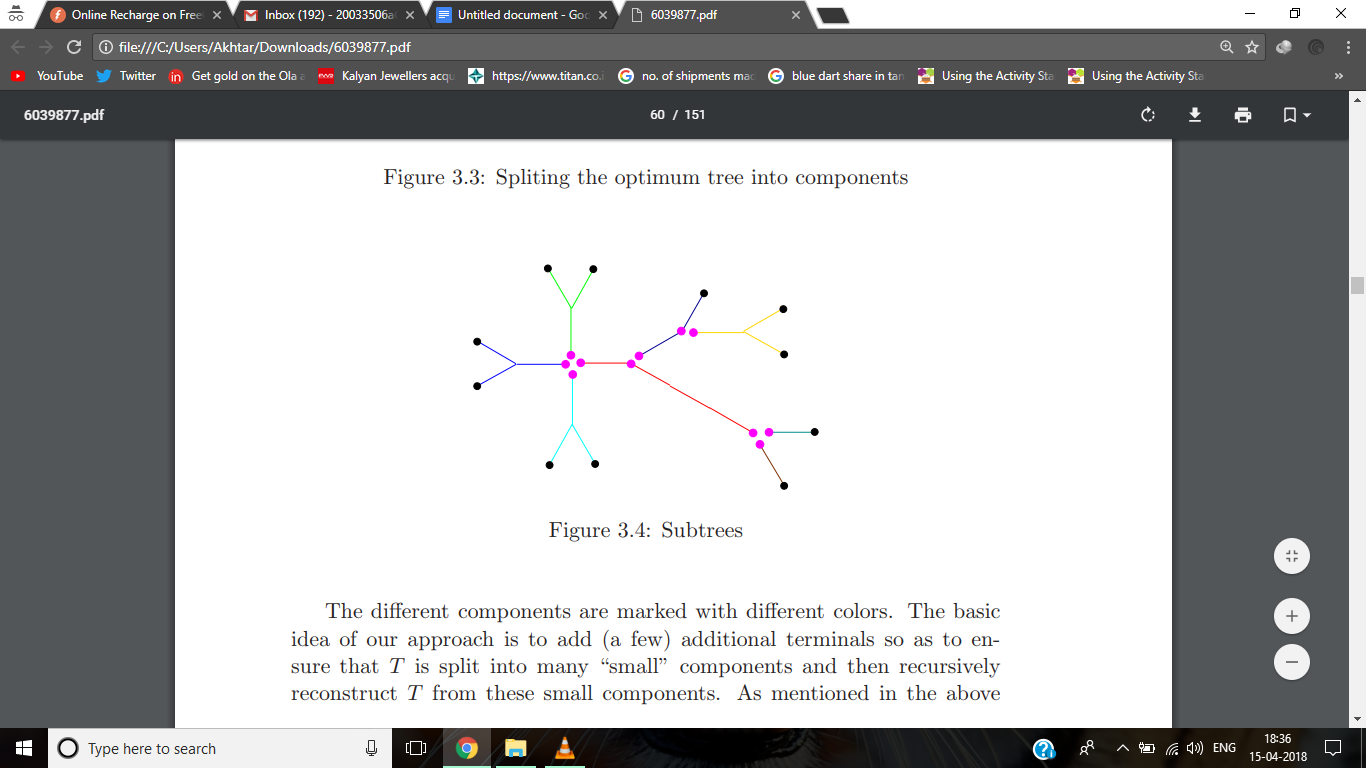
The algorithm in time O∗ (2 + δ) k will be presented for the Steiner tree problem

Let us fix some minimum Steiner tree T = T(X) for X. Every leaf of T is a terminal. In case T has interior nodes which are terminals, these interior terminals split T into components, i.e., maximal subtrees without any interior terminals, cf. Fig. 3.3. The subtrees are as shown in Fig. 3.4.

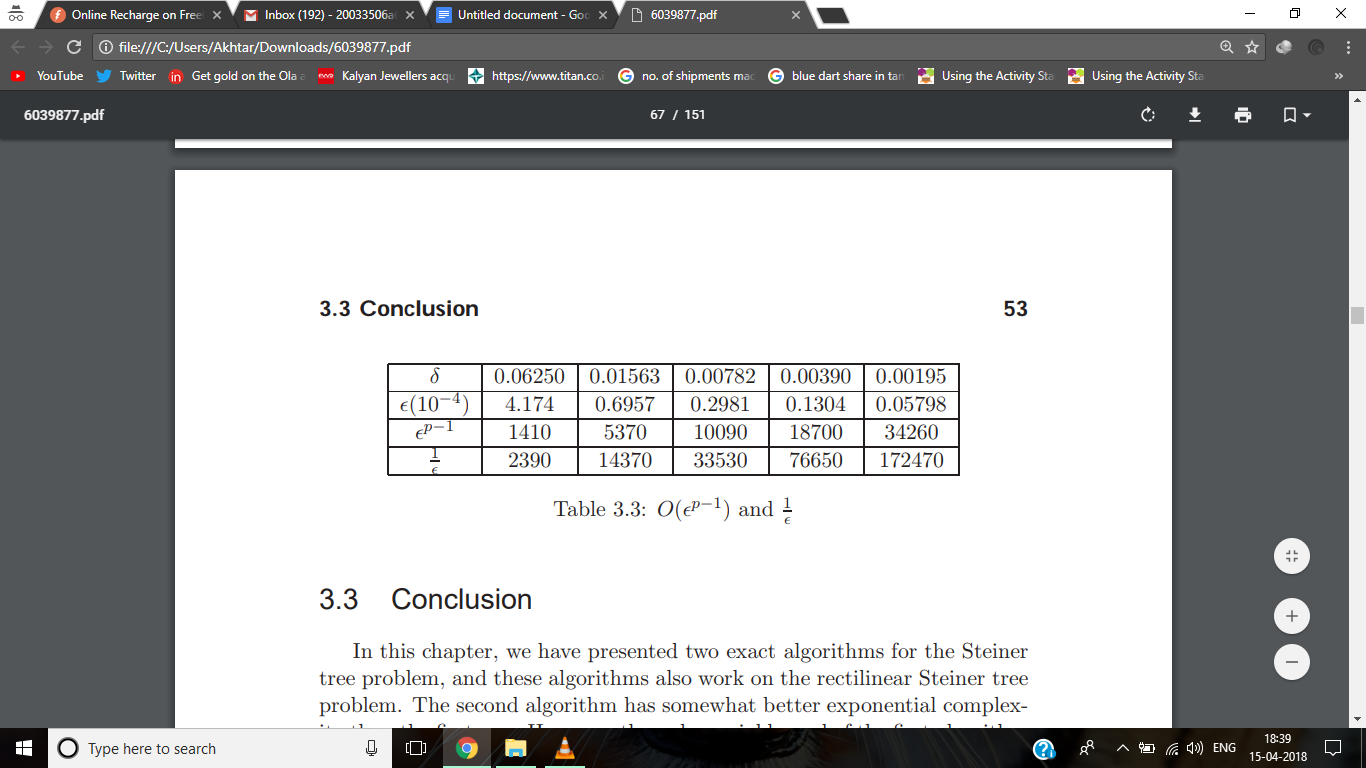




The different components are marked with different colors. The basic idea of our approach is to add (a few) additional terminals so as to ensure that T is split into many “small” components and then recursively reconstruct T from the section, the size of a component equals the number of terminals (leaves) of the component.

Algorithm 3.2.1. (Algorithm ASC (Attach Small Components)) For each Z, Z ˜ ⊆ Z˜ ⊆ V, |Z˜| = k + 1 ǫ do: Step 1. Compute T(X) for all X ⊆ Z, ˜ |X| ≤ ǫk + 1. Step 2. For all X ⊆ Z, ˜ |X| > ǫk + 1, compute T(X) recursively, according to T(X) = min{T(X1) ∪ T(X2) | X = X1 ✶ X2, |X2| ≤ ǫk + 1}. (2) This yields our main result.

CONCLUSION



We have presented two exact algorithms for the Steiner tree problem, and these algorithms also work on the rectilinear Steiner tree problem. The second algorithm has somewhat better exponential complexity than the first one. However, the polynomial bound of the first algorithm is more acceptable. Although we have improved the complexity of the algorithm for the Steiner tree problem, it is still open that whether there exists an algorithm with the complexity less than O∗ (2k ) for the Steiner tree problem.