Min-Height-Elimination Tree for Interval Graphs

Abhishek Santhanam

Introduction

In this paper we will be improving the **algorithm**(1) provided by Prof.Bengt Aspvall and Pinnar Heggerness to obtain an O(n^3) algorithm to find the min-height elimination tree of an interval graph.

Explanation

The algorithm used for interval graph in (1) could not be reduced to O(n^3) because though I_{ij} satisfied the hereditary property it was not connected, only the connected components were interval graphs. Hence to avoid this we will be considering forests instead of trees. In this forest each component will be a elimination tree. Hence if we find min-height elimination forest we will in turn have a forest with all trees having min-height.

In this modified algorithm we will store the min height of the elimination forest in Opt. i.e Opt[i,j] will store min height of the elimination forest I_{ij}.

One case we would have to handle is to what minimal seprator to choose when I_{ij} is disconnected. In that case we can choose a temporary Separator of zero size to be the minimal separator.

Algorithm

Listing 1: Modified DynHeight algorithm for min height elimination tree

```
def DynHeightMod(T:Clique Tree, Opt:Array of Integers):
    for i=1 to m do
        for j=1 to m do
            Opt[i,j]= INF
    for i=m downto 1 do
     for j=i to m do
        if(j==i){
            ...(Same as in DynHeight)
        }
        else{
            for k=i to j-1 do
                A= S[i] union S[j-1] //precomputed union values used
                |S1[k]|=|S[k]-A| //(Why?: refer pg 16 (1))
                min_ht=|S1[k]| +max(Opt[i,k]+Opt[k+1,j])
                Opt[i,j]=min(min_ht,Opt[i,j]);
                if |S1[k]|=0 \\ this is the case when it is disconnected
                    break
        }
```