Cluster trees

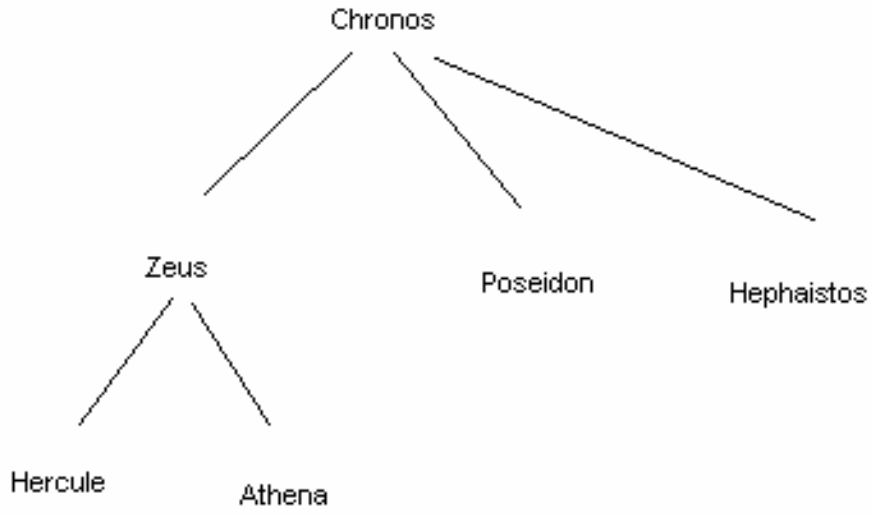
Some algorithms to generate them.

Graph

- A graph is a set of vertex V together with a binary relation E that we could read as „is connected by an Edge to“.

Tree

- A tree is an acyclic graph.
- We will only speak of rooted tree here, so that the root is a special vertex and every other vertex (called nodes) is seen as “subordinate“ to another if the other is nearer the root.



The main Problem

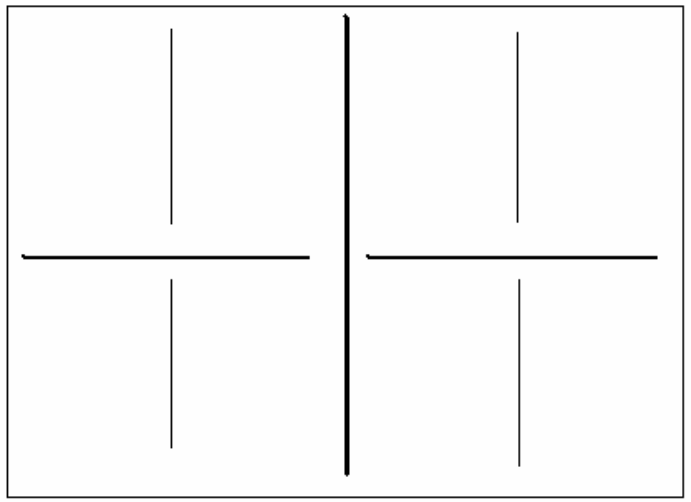
- We are given a set of n point in d dimension and we would like to put them in a cluster tree.
- We can achieve that by repetitive splitting of the set of points. The way we split the points will determine our algorithm.

Properties we would like to have.

- A lot of cluster to choose from
- Small diameters
- (of course) reasonnable running time
- Ease of implemenation
- (perhaps) regularity

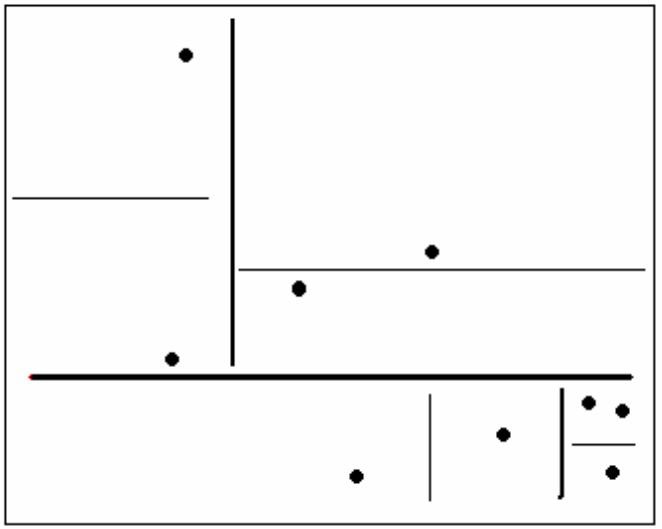
Three approaches

- Splitting the points (or the space) in a predetermined way.



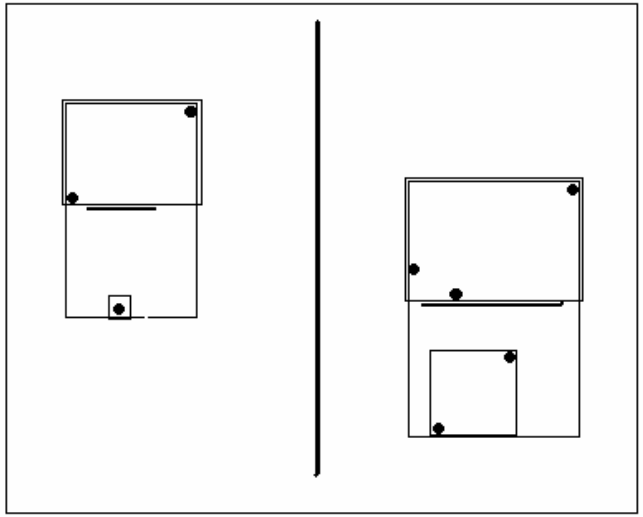
Getting balanced trees

- Dividing in near sized subsets.



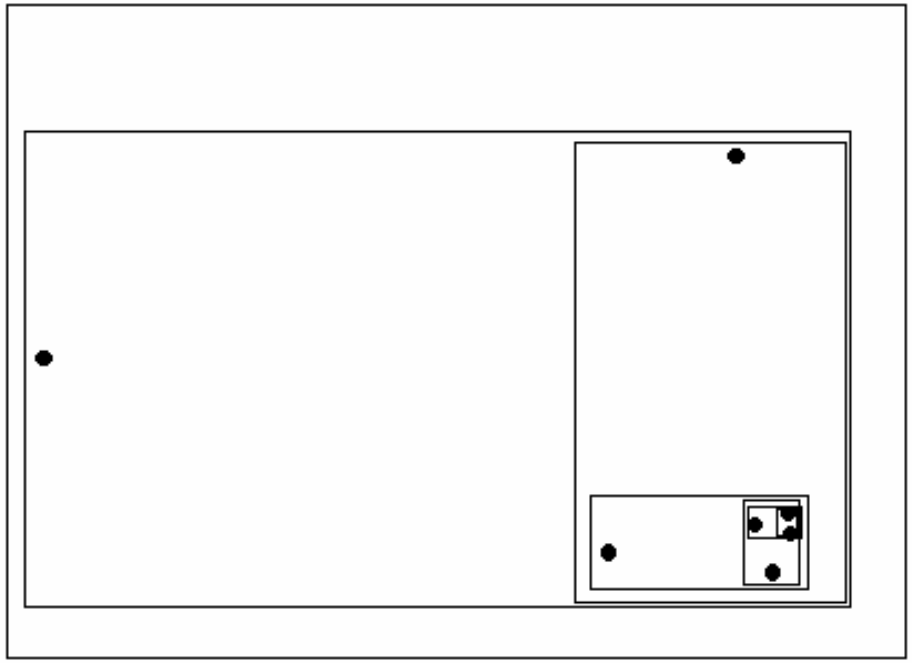
Using bounding boxes

- Each time we have a subset of point we enclose it in the smallest possible box, then we split this box along the longest direction.

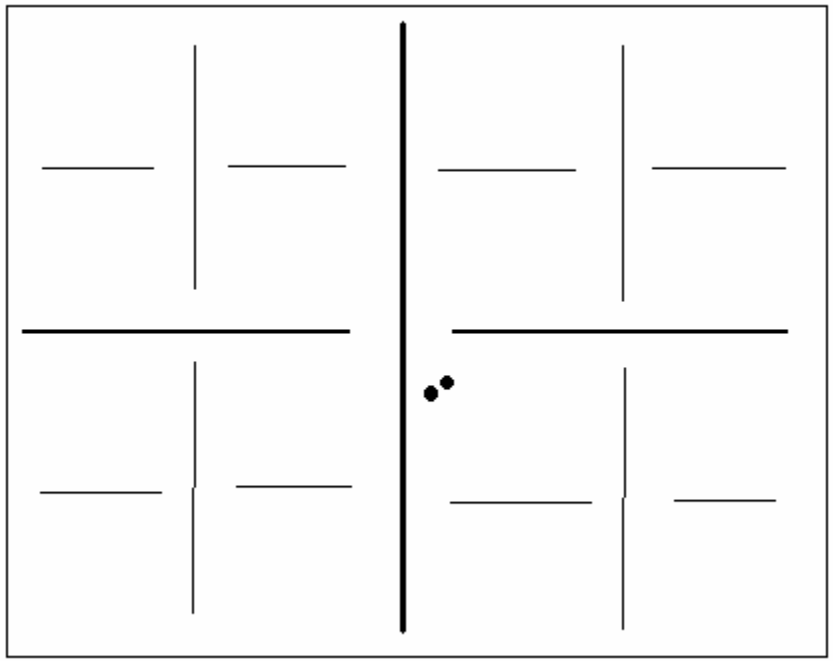


The Worst cases

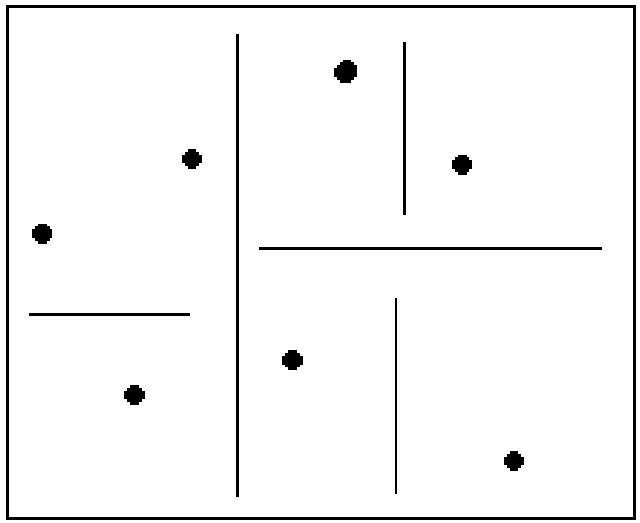
- The Bounding box reliant algorithm



The regular algorithm worst case



The balanced algorithm (absence of) worst case



The average Case of the Regular algo.

- We reformulate the problem : how many bit do we need to look at to differentiate n uniformly (between [0,1]) distributed numbers
- We get prob(s bit are sufficient)< ((2^s)*(2^s-1)*(2^s-2)*…(2^s-n+1))/ (2^(sn))

- That is by 1-n^2*2^(-s) approximable with error of O(n^4/(2^-2s))
- It implies taking s> C+d*log n with c and d big enough, gave us 1/(2^m) chance not to be finished, with d linear in m.
- So that the number of „refinement we need to make in average is logarithmic
-