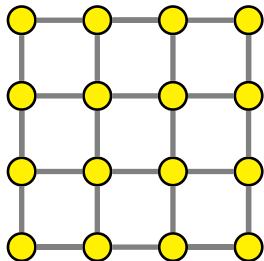


Quickly finding trees with maximum leaves

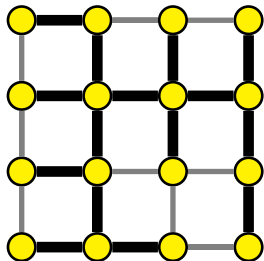
Karsten Behrmann

RWTH Aachen University

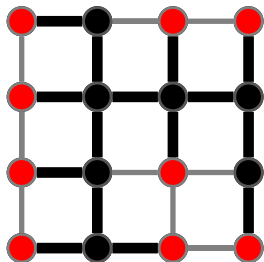
November 14, 2009



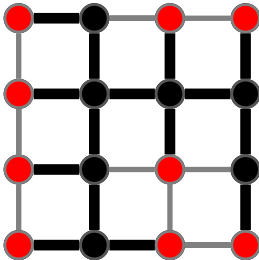
- Given some graph



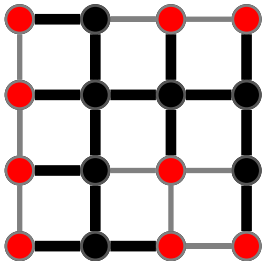
- Given some graph
- Find a spanning tree...



- Given some graph
- Find a spanning tree...
- ... that maximizes the number of leaves



- Given some graph
- Find a spanning tree...
- ... that maximizes the number of leaves
- Considered here: ... that has k or more leaves



- Given some graph
- Find a spanning tree...
- ... that maximizes the number of leaves
- Considered here: ... that has k or more leaves
- Not considered:
Approximate solutions

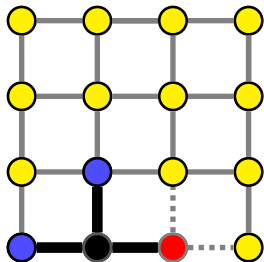
- Most recent work on the subject focused on theoretical bounds
- Unfortunately, those are exponential
- So polynomial factors are ignored

- Most recent work on the subject focused on theoretical bounds
- Unfortunately, those are exponential
- So polynomial factors are ignored
- In an actual implementation, these become relevant
- Our focus: How to make it fast, in practice

- Our problem is NP-hard.

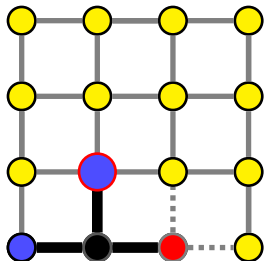
- Our problem is NP-hard.
- Basically, we have to enumerate all trees, counting leaves.
- We do that recursively

- Our problem is NP-hard.
- Basically, we have to enumerate all trees, counting leaves.
- We do that recursively
- And of course a few tricks we can do

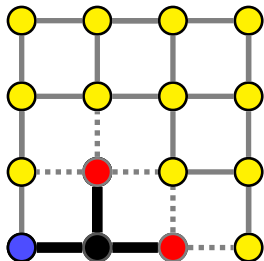


- Grow outwards from some root

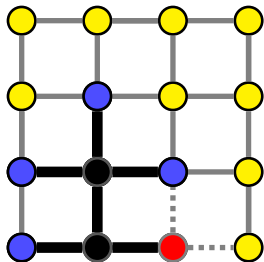
The basic recursion



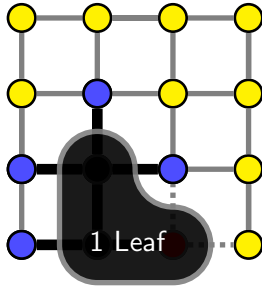
- Grow outwards from some root
- Pick some “visible” node



- Grow outwards from some root
- Pick some “visible” node
- Try it (recursively) as leaf...

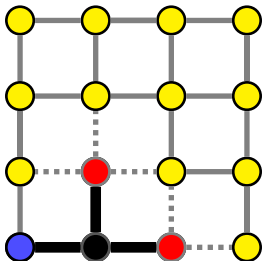


- Grow outwards from some root
- Pick some “visible” node
- Try it (recursively) as leaf...
- And try it as inner node.



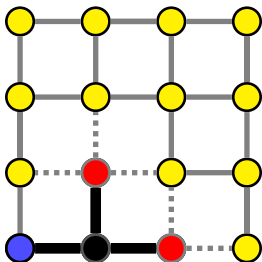
- Grow outwards from some root
- Pick some “visible” node
- Try it (recursively) as leaf...
- And try it as inner node.
- Note: We can contract our known nodes

The Question



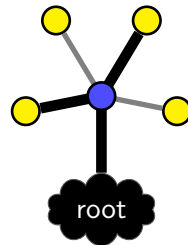
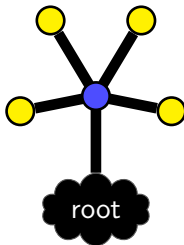
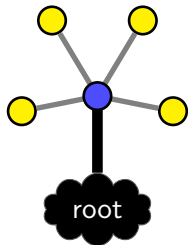
- So how can we quickly detect bad decisions?
- Idea: Our choices define a subset of spanning trees (on the graph)

The Question

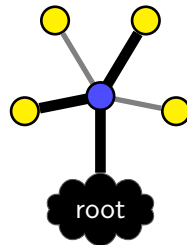
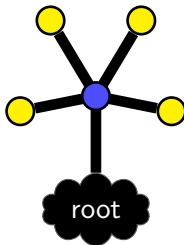
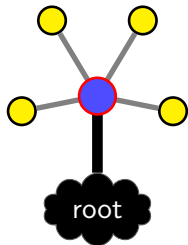


- So how can we quickly detect bad decisions?
- Idea: Our choices define a subset of spanning trees (on the graph)
- We can't quickly check max-leaf of that set...
- ... but maybe we can check something weaker.

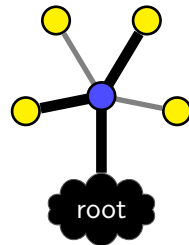
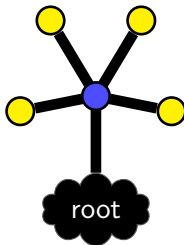
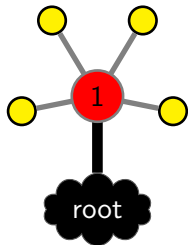
Given a tree, using some of the edges of our graph...



Given a tree, using some of the edges of our graph...

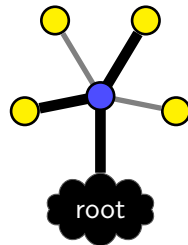
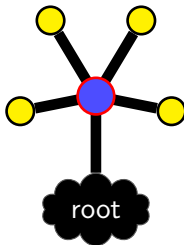
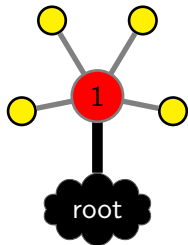


Given a tree, using some of the edges of our graph...



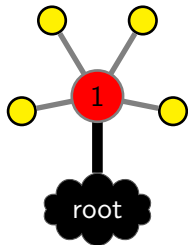
Clearly a leaf.

Given a tree, using some of the edges of our graph...

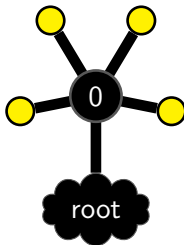


Clearly a leaf.

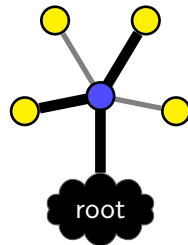
Given a tree, using some of the edges of our graph...



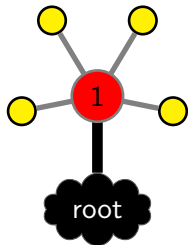
Clearly a leaf.



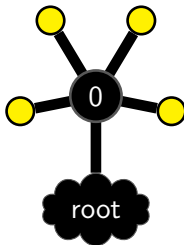
Clearly an inner node.



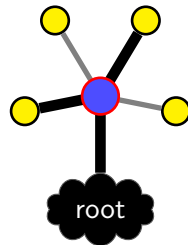
Given a tree, using some of the edges of our graph...



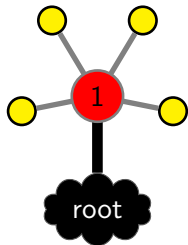
Clearly a leaf.



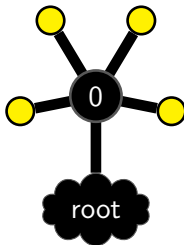
Clearly an inner node.



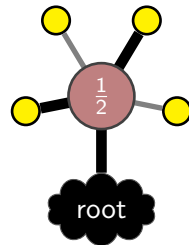
Given a tree, using some of the edges of our graph...



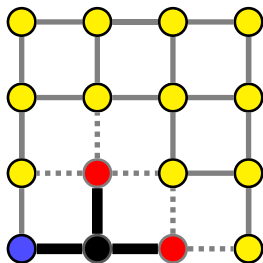
Clearly a leaf.



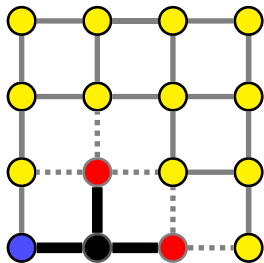
Clearly an inner node.



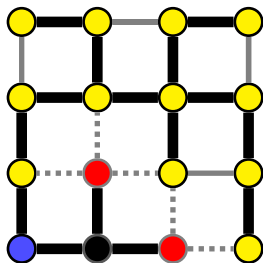
Well... Half a leaf?



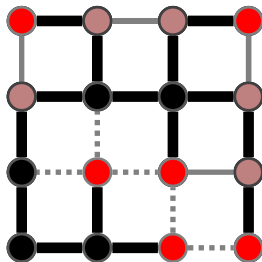
- We can add those “relaxed” leaf counts
- This gives an upper bound on real leaf count



- We can add those “relaxed” leaf counts
- This gives an upper bound on real leaf count
- To find maximum of the relaxed count...

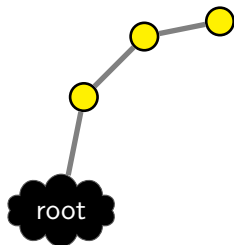


- We can add those “relaxed” leaf counts
- This gives an upper bound on real leaf count
- To find maximum of the relaxed count...
- ...we only need to find a minimum spanning tree
- (weights of edges depend on degree of nodes)

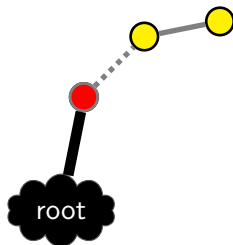


- We can add those “relaxed” leaf counts
- This gives an upper bound on real leaf count
- To find maximum of the relaxed count...
- ...we only need to find a minimum spanning tree
- (weights of edges depend on degree of nodes)
- Bound in this case:

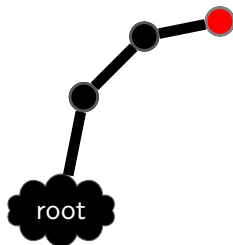
$$6 + 5 \cdot \frac{1}{2} = 8\frac{1}{2}$$



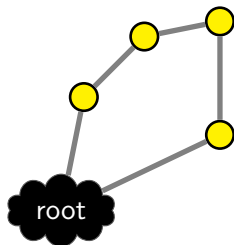
- Problem: Excessive branching



- Problem: Excessive branching
- Trick 1: Connectivity check

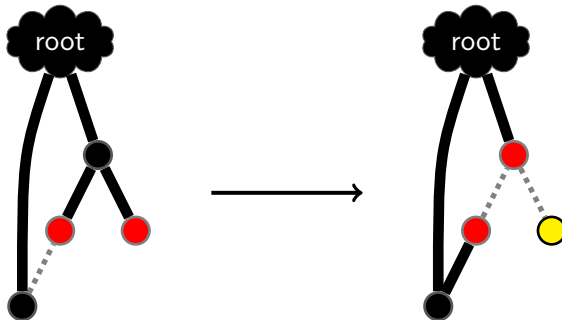


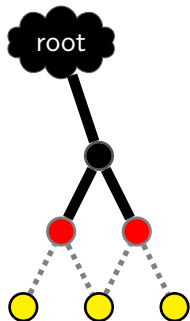
- Problem: Excessive branching
- Trick 1: Connectivity check
- Trick 2: Treat degree-2 nodes specially



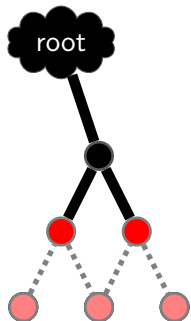
- Problem: Excessive branching
- Trick 1: Connectivity check
- Trick 2: Treat degree-2 nodes specially

Idea: We need not consider all trees if we prove some to be equivalent





- If we find certain patterns...



- If we find certain patterns...
- We can mark nodes as leaves early

- Compared to previous results by T. Fujie:
- (Although we cheated by using a bigger computer)

- Compared to previous results by T. Fujie:
- (Although we cheated by using a bigger computer)
 - Random graphs: Speedup 5,000 – 500,000, iterations reduced by 500 - 1000

- Compared to previous results by T. Fujie:
- (Although we cheated by using a bigger computer)
 - Random graphs: Speedup 5,000 – 500,000, iterations reduced by 500 - 1000
 - 8x8 grid graphs: Speedup 300,000, iterations reduced by 1000

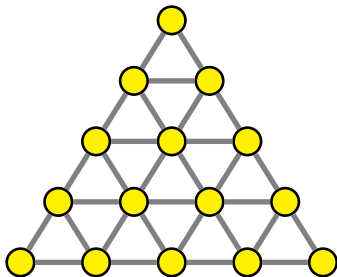
- Compared to previous results by T. Fujie:
- (Although we cheated by using a bigger computer)
 - Random graphs: Speedup 5,000 – 500,000, iterations reduced by 500 - 1000
 - 8x8 grid graphs: Speedup 300,000, iterations reduced by 1000
- Compared to theoretical worst-case:
- Expected slowdown 4^k , experienced slowdown:

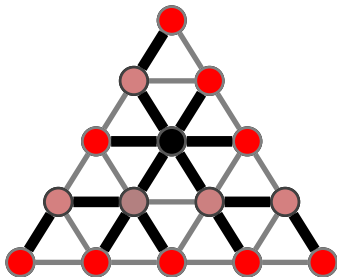
- Compared to previous results by T. Fujie:
- (Although we cheated by using a bigger computer)
 - Random graphs: Speedup 5,000 – 500,000, iterations reduced by 500 - 1000
 - 8x8 grid graphs: Speedup 300,000, iterations reduced by 1000
- Compared to theoretical worst-case:
- Expected slowdown 4^k , experienced slowdown:
- approx. 1.3^k for square grids

- Compared to previous results by T. Fujie:
- (Although we cheated by using a bigger computer)
 - Random graphs: Speedup 5,000 – 500,000, iterations reduced by 500 - 1000
 - 8x8 grid graphs: Speedup 300,000, iterations reduced by 1000
- Compared to theoretical worst-case:
- Expected slowdown 4^k , experienced slowdown:
- approx. 1.3^k for square grids
- around 1.6^k for triangular grids

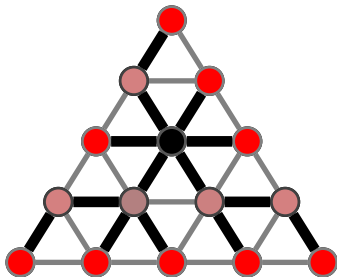
- Compared to previous results by T. Fujie:
- (Although we cheated by using a bigger computer)
 - Random graphs: Speedup 5,000 – 500,000, iterations reduced by 500 - 1000
 - 8x8 grid graphs: Speedup 300,000, iterations reduced by 1000
- Compared to theoretical worst-case:
- Expected slowdown 4^k , experienced slowdown:
- approx. 1.3^k for square grids
- around 1.6^k for triangular grids
- On random graphs: Slight speedup on increasing k
- (But slowdown on increasing n or $n - k$)

- The spanning-tree heuristic should be analyzed
- ideally included in theoretical bound somehow

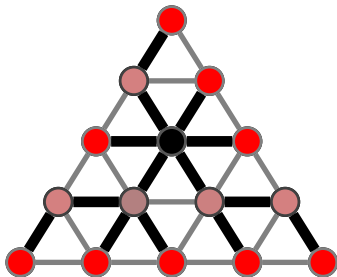




- The spanning–tree heuristic should be analyzed
- ideally included in theoretical bound somehow
- sometimes it significantly overestimates
- Graph on the left has 9 leaves maximum, heuristic estimates 12



- The spanning–tree heuristic should be analyzed
- ideally included in theoretical bound somehow
- sometimes it significantly overestimates
- Graph on the left has 9 leaves maximum, heuristic estimates 12
- Still, heuristic is single most significant speedup



- The spanning–tree heuristic should be analyzed
- ideally included in theoretical bound somehow
- sometimes it significantly overestimates
- Graph on the left has 9 leaves maximum, heuristic estimates 12
- Still, heuristic is single most significant speedup
- Unfortunately, also uses over 90% of CPU time

That's all folks!