On the trade-offs of width and height of Pseudo-trees & AND/OR Beam Search

Héctor Otero Mediero - 66558358

CS199: Individual Study - Rina Dechter

Topics covered

1. Introduction to Graphical Models
2. Bayesian Networks (Belief Upd, MPE and MAP)
3. Exact Inference: Bucket Elimination
4. AND/OR Search Spaces
5. Approximation algorithms
6. Mini-Bucket heuristic Search

Topics covered

7. Finding good variable orderings: IGVO, Complete algorithm for treewidth, Enumerating Minimal Triangulations.

8. Trade-offs between height and width.

9. AND/OR Beam Search and Stochastic AND/OR Beam Search.

Topic 1: Height vs Width Trade-offs

Height-Width trade-offs on pseudo-trees

- Pool of pseudo-trees generated by ***Enumerating minimal triangulations of a graph.*** [*Nofar Carmeli, Batya Kenig and Benny Kimelfeld. Efficiently Enumerating Minimal Triangulations. 2017*]

- Compare orderings that generate pseudo-trees with different heights and widths.

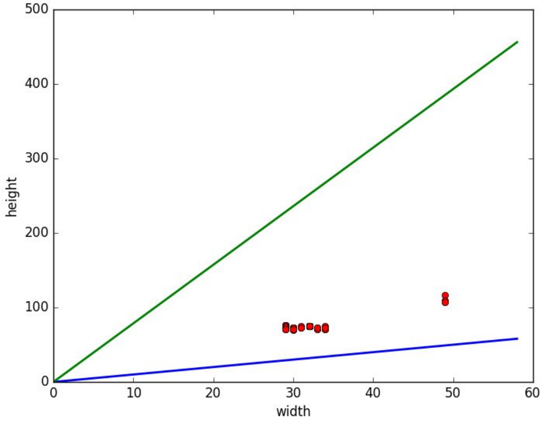
Bounds for the height and width

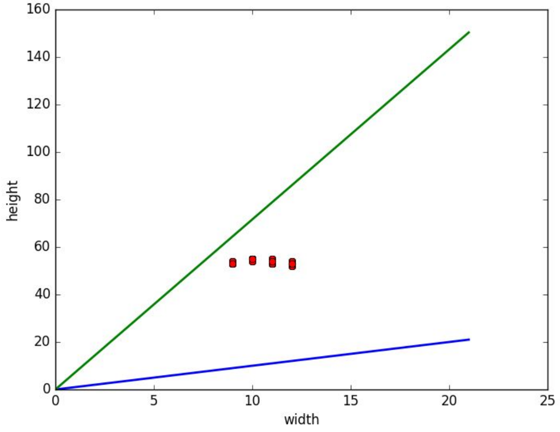
$h \leq w \log n \rightarrow h/w \leq \log n$

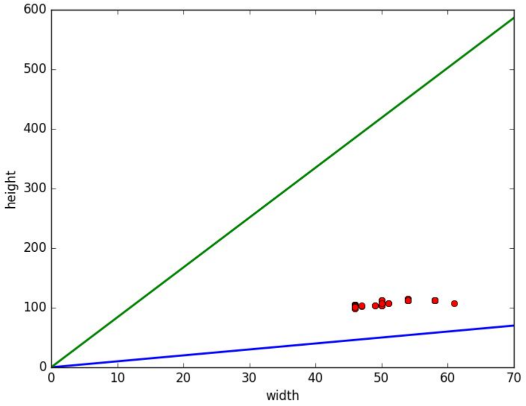
$h \geq w \rightarrow h/w \geq 1$

$$1 \leq h/w \leq \log n$$

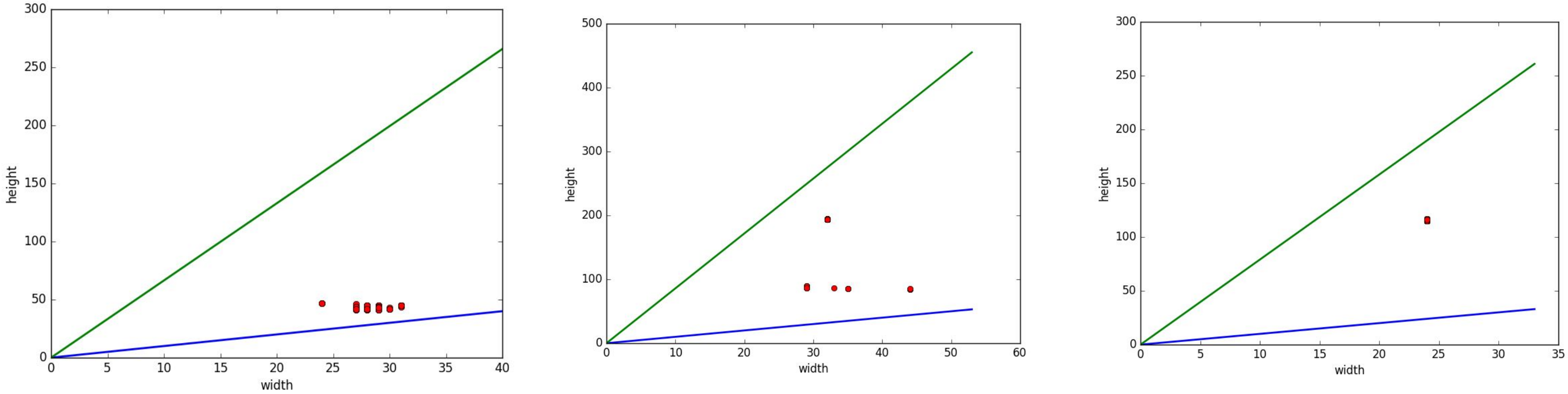
Variability of the orderings (I)







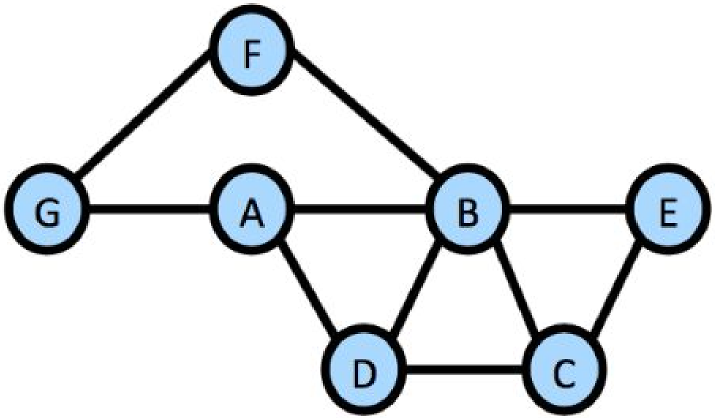
Variability of the orderings (II)



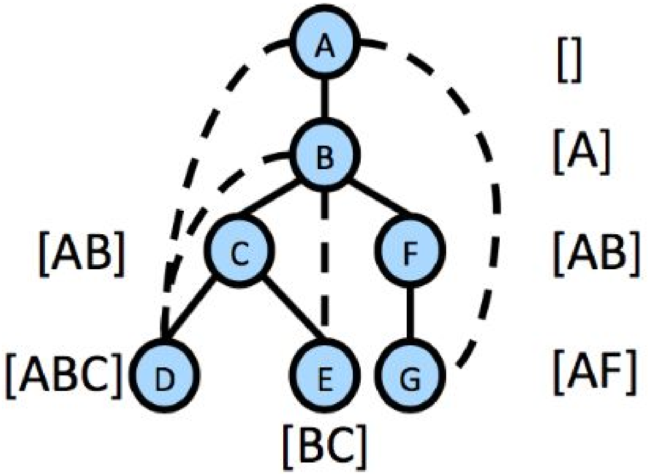
Hypothesis #1

For trees of the same width, a lower height of the trees is better.

Motivation (I)



(a) A graphical model.



(b) Context-labeled pseudo-tree.

Motivation (II)

Dead caches h = w.

If **h = w + δ** with δ small, we're almost in the case where we don't have to cache -> faster

With δ large, we need to check the cache multiple times to check the same problem -> slower.

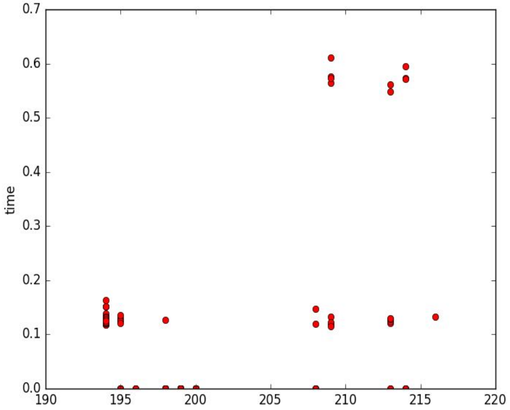
To check the first hypothesis, we plot the execution times for AOBF caching the results on pseudo-trees of equal width and different height.

Characteristics of the networks

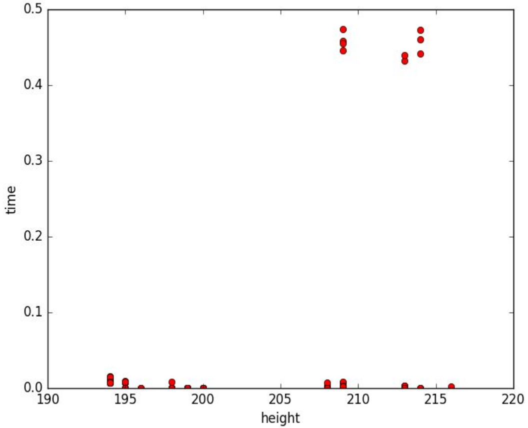
Model	Width	Depth	Number of variables	Average domain size	Max. domain size
BN	14	21	100	2	2
Pedigree	19	61	385	2.06	3
WCSP	9	33	143	2.81	4
Promedas	21	60	1005	2	2

Same width - different height

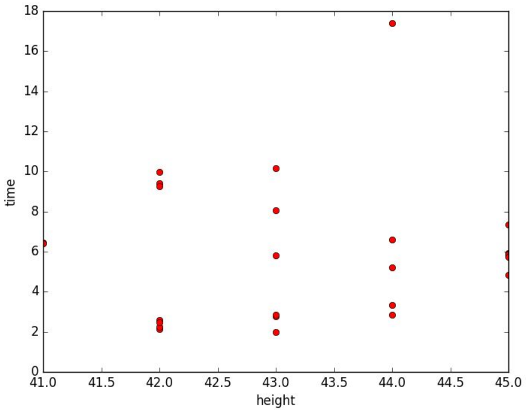
Family2Dominant.1.5loci



Family2Dominant.20.5loci

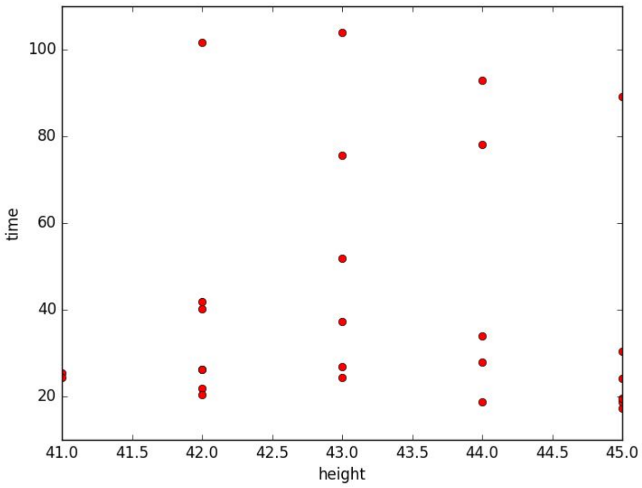


BN_0

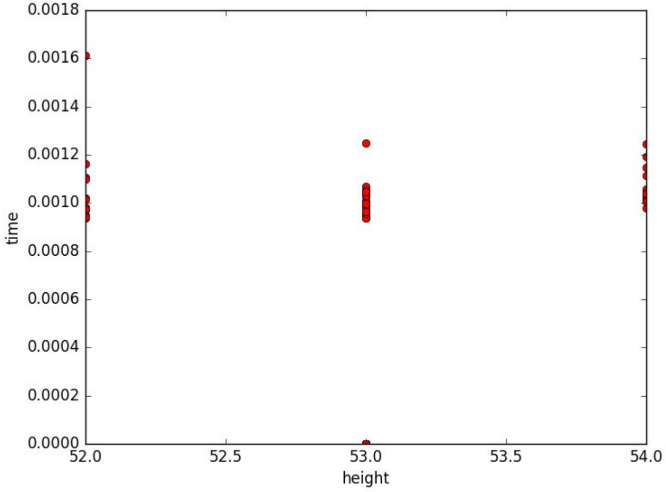


Same width - different height

BN_1



503.wcsp



Same width - different height

- Architecture-dependent

- Pedigree and some examples in BN show better results for shallower pseudo-trees.

- Some BN and WCSP don't.

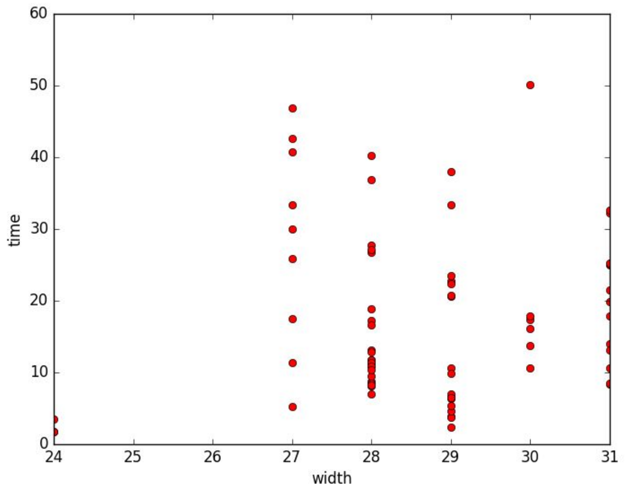
Hypothesis #2

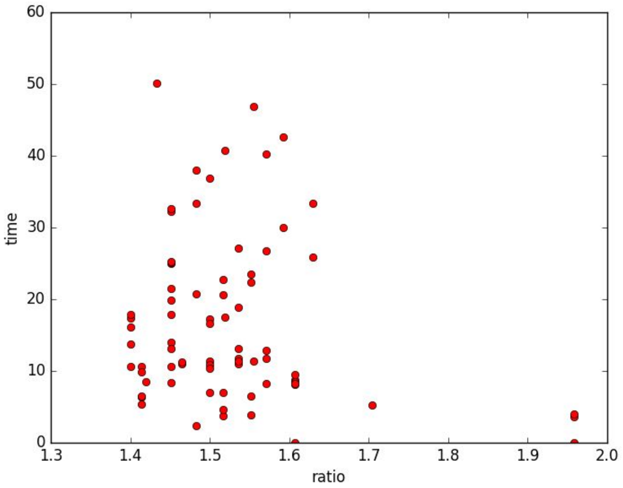
We can use the ratio to generalize for the cases that seem to fulfill the Hypothesis #1.

For these cases, the smaller the ratio, the better the execution time.

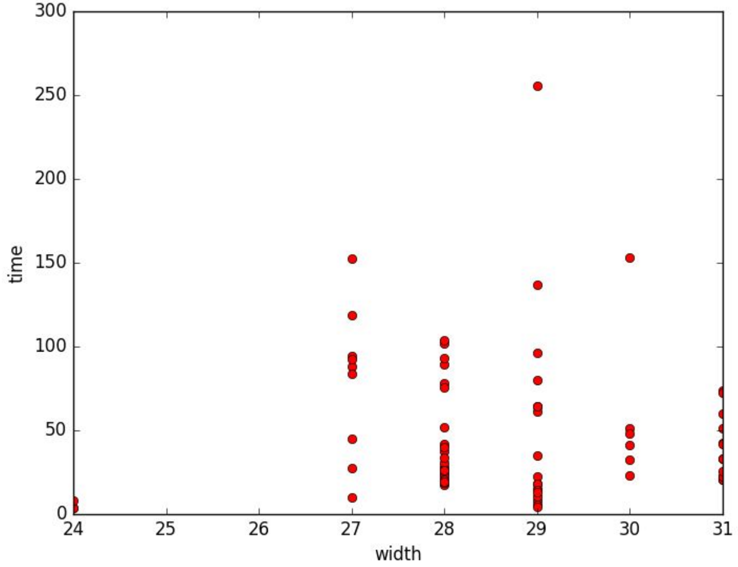
We plot the same execution times of AOBF caching and not caching against the width of the ordering and the ratio.

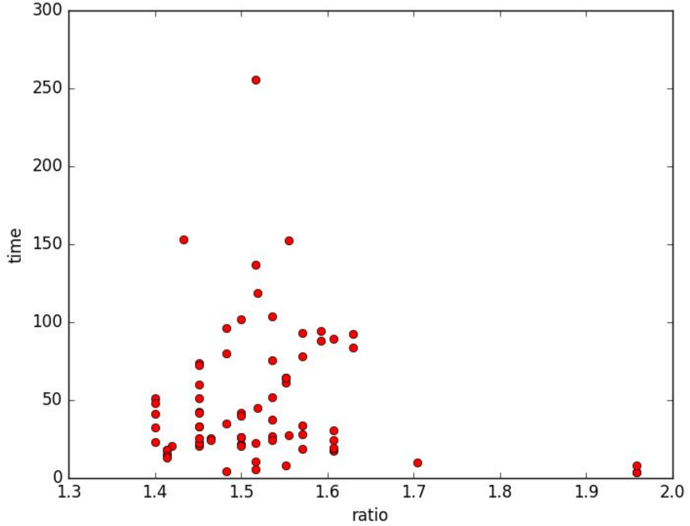
Width-time vs ratio-time (caching) - BN



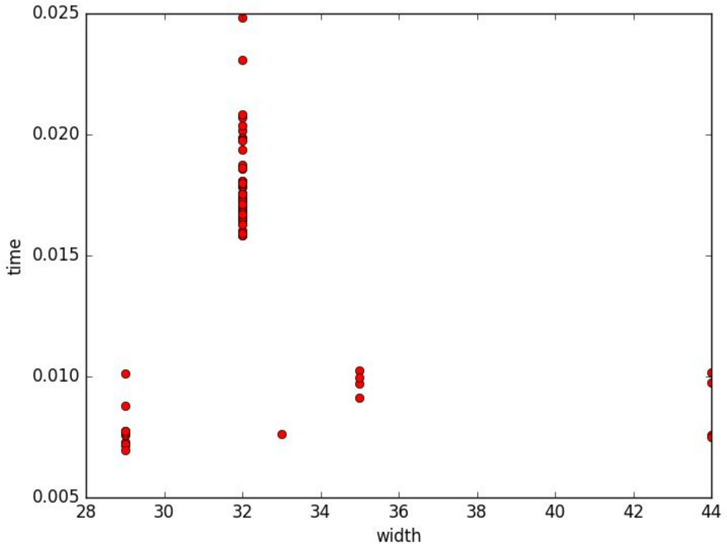


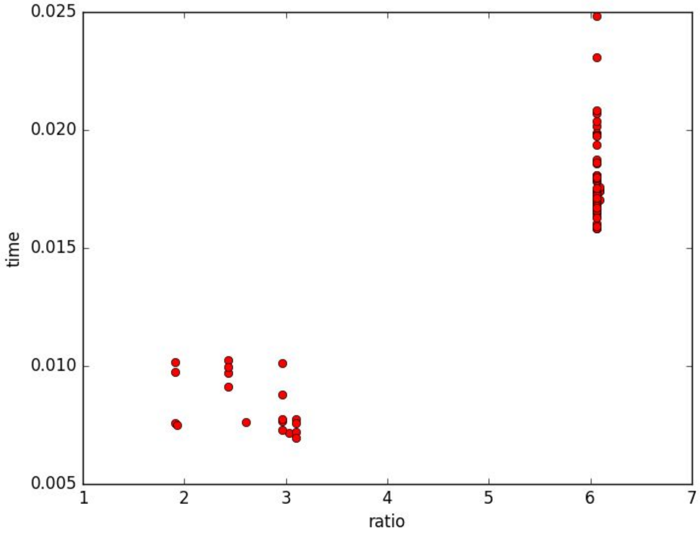
Width-time vs ratio-time (no caching) - BN



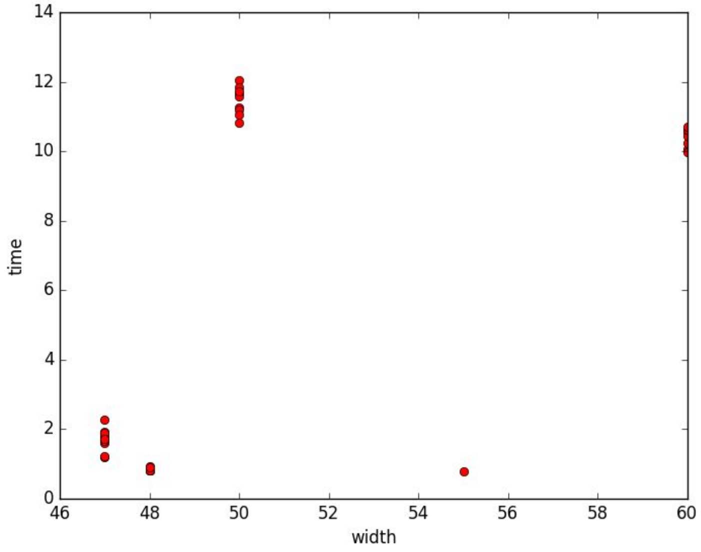


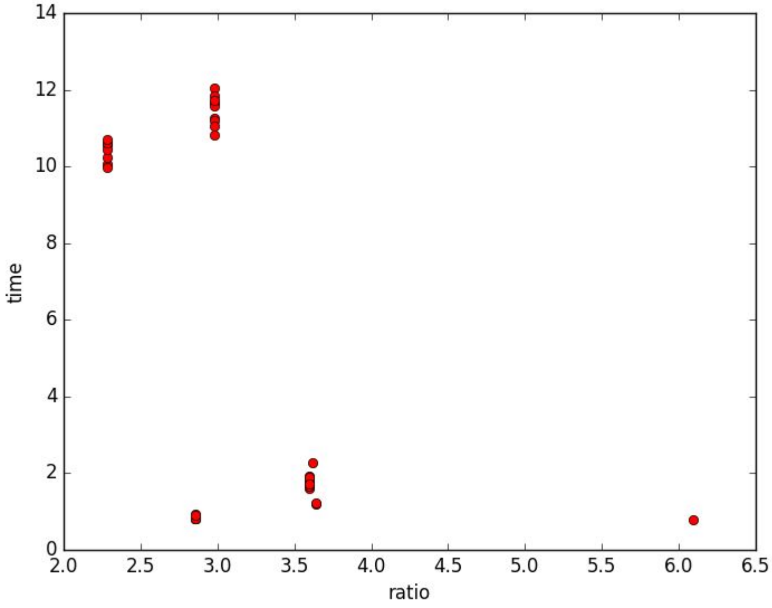
Width-time vs ratio-time (caching) - Pedigree



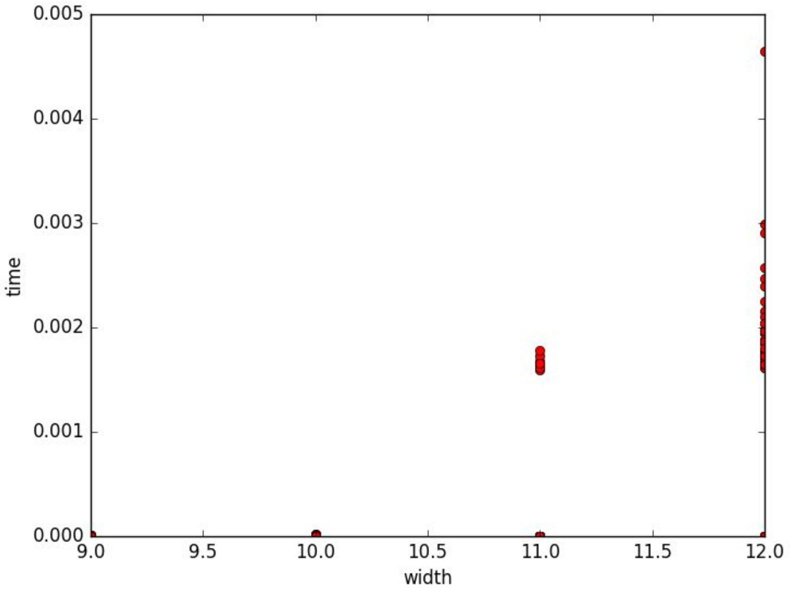


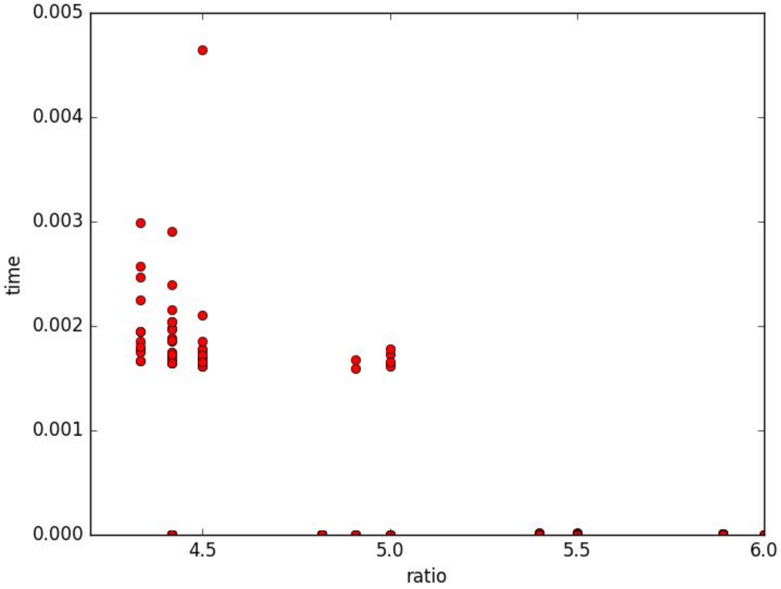
Width-time vs ratio-time (caching) - Or chain





Width-time vs ratio-time (caching) - WCSP





Width-time vs ratio-time

- If width is representative, use width.
- If it's not, use ratio to choose among orderings.
- Problem-dependent

Future work

- Execute same orderings with AOBB
- Execute distributed version of the algorithms.

Topic 2:
AOBeam &
Stochastic AOBeam

Beam Search

- Parameterized by β, beam width.

- At each step of the algorithm we open the best β nodes in terms of the heuristic value.

Beam Search

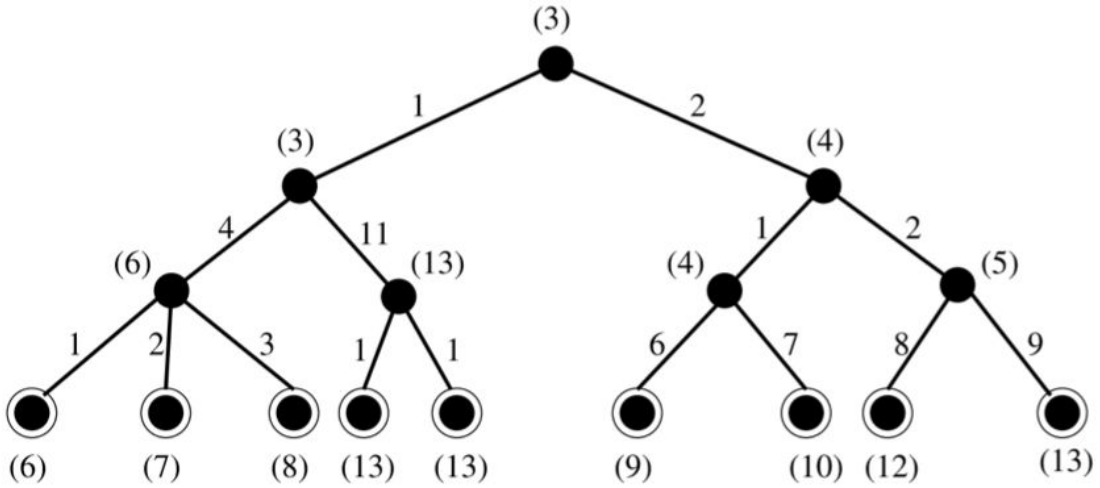
- Doesn't guarantee completeness or optimality.

- Moreover, it's not monotonic with respect to the β parameter.

Beam Search - Monotonicity

$\beta = 2$; $c(sol) = 6$

$\beta = 3$; $c(sol') = 9$



Beam Search - AND/OR Spaces

Adaptation to AND/OR spaces is necessary since:

- Every child of an AND node has to be included.

- For each OR node, we need at least one node in the solution.

Beam Search - AND/OR Spaces

Adaptation -> OR-pruning:

For each OR node, include in the space only the most promising β children.

Motivation

AND/OR Best First Search weaknesses:

- Memory Issues
- Bounds the solution pretty slowly.

AND/OR Beam Search:

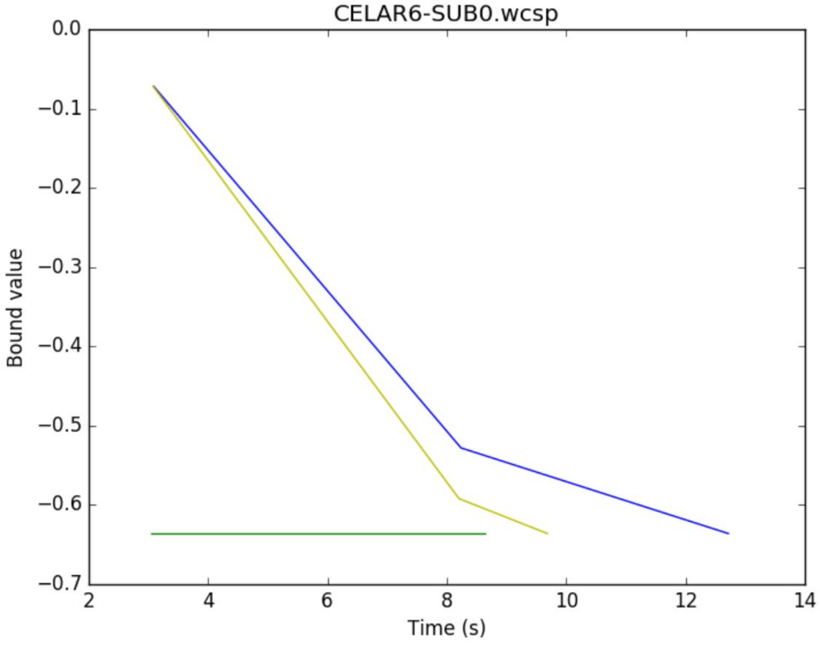
- Less memory usage
- More aggressive pruning sacrifices optimality and completeness but generates bounds faster.

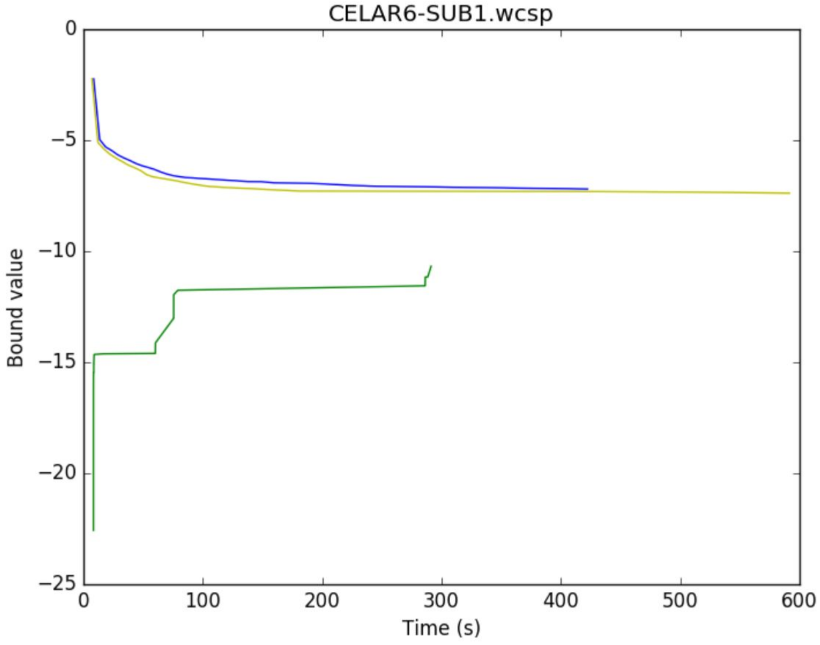
Task computed & algorithms executed

MPE on different UAI inference challenge domains (Pedigree, WCSP, Segmentation, Object detection…)

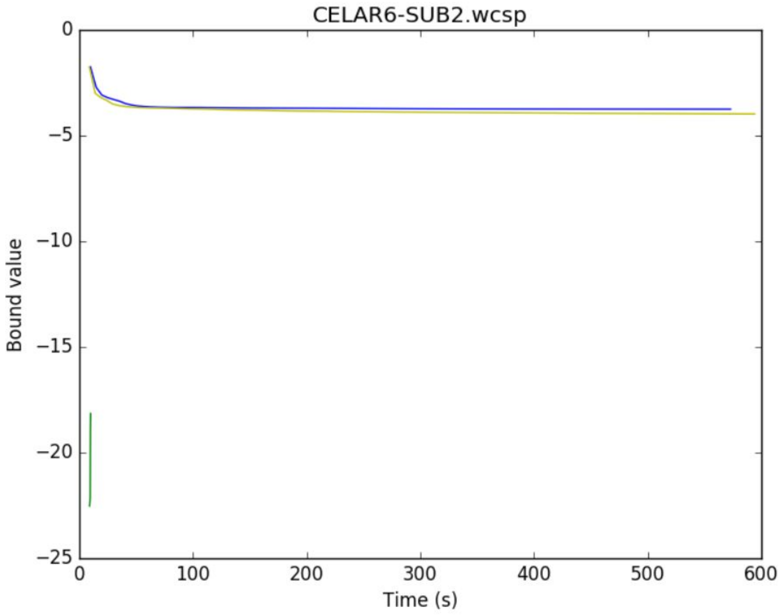
William Lam’s version of DAOOPT that includes AOBF and AOBB.

AOBeam vs AOBF vs AOBB

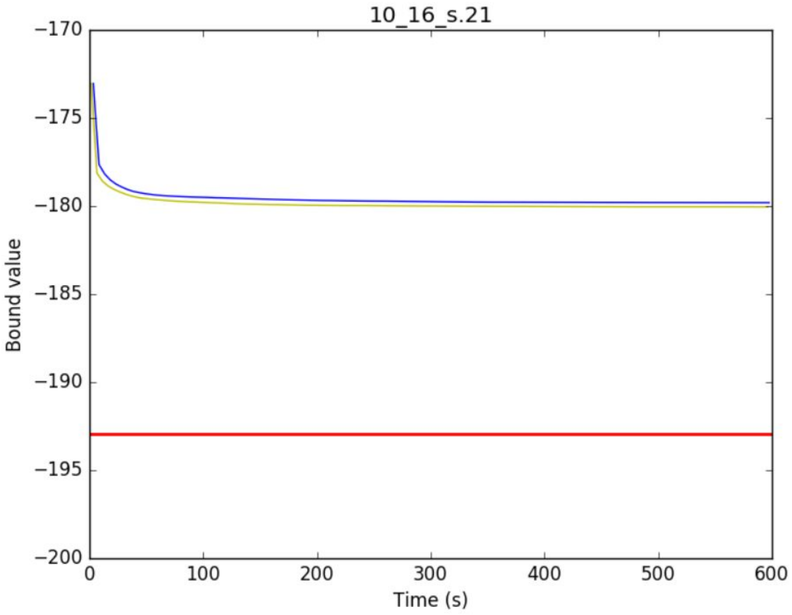




AOBeam vs AOBF vs AOBB

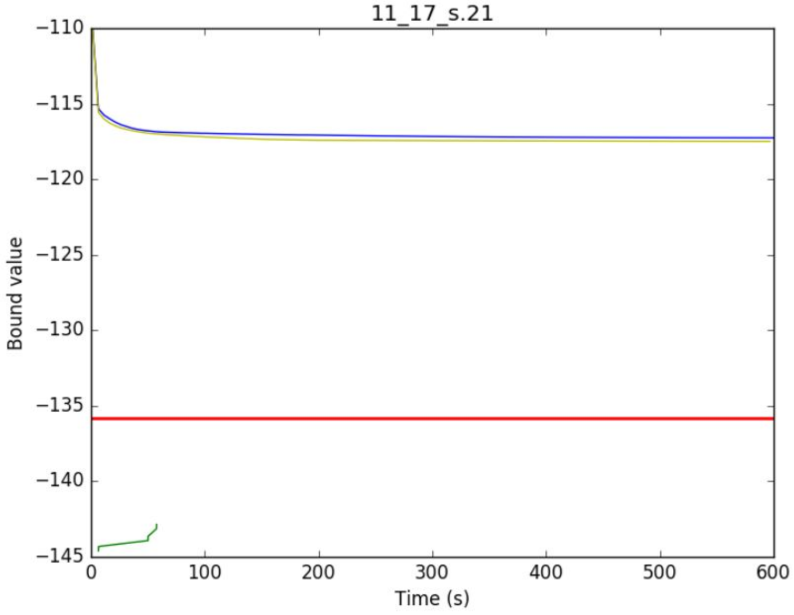


(a) WCSP domain

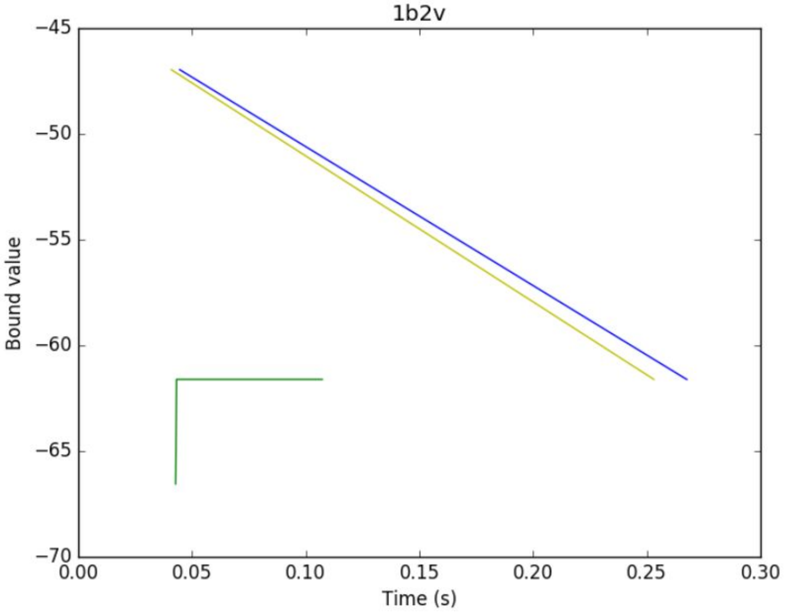


(b) Segmentation domain

AOBeam vs AOBF vs AOBB

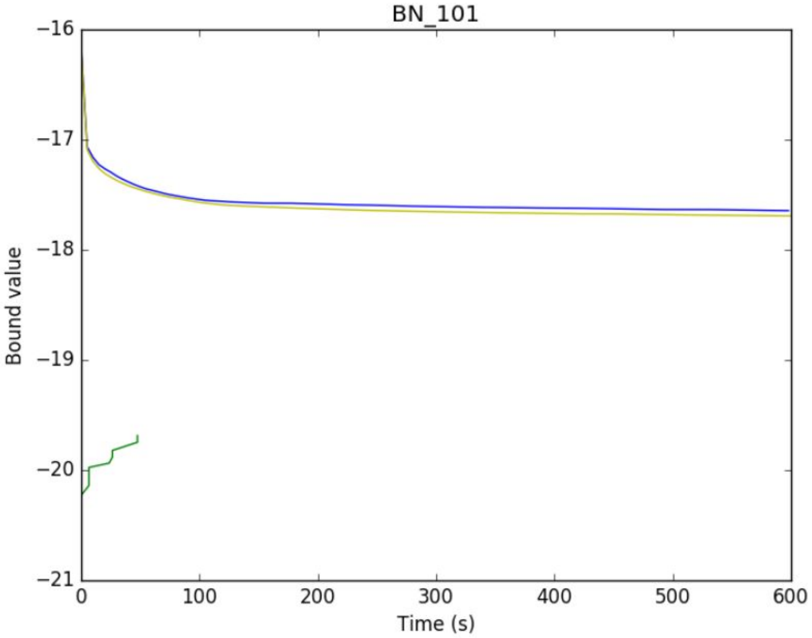


(a) Segmentation domain

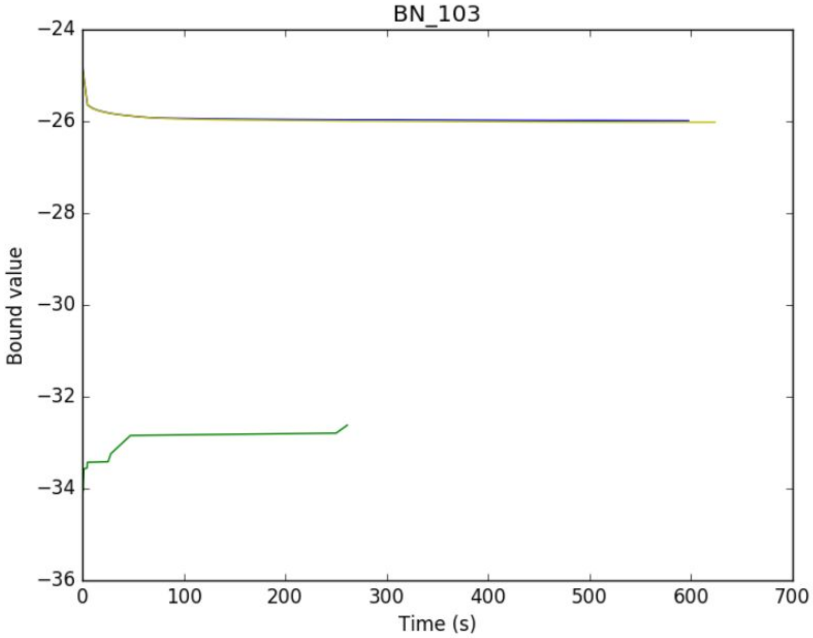


(b) CPD domain

AOBeam vs AOBF vs AOBB

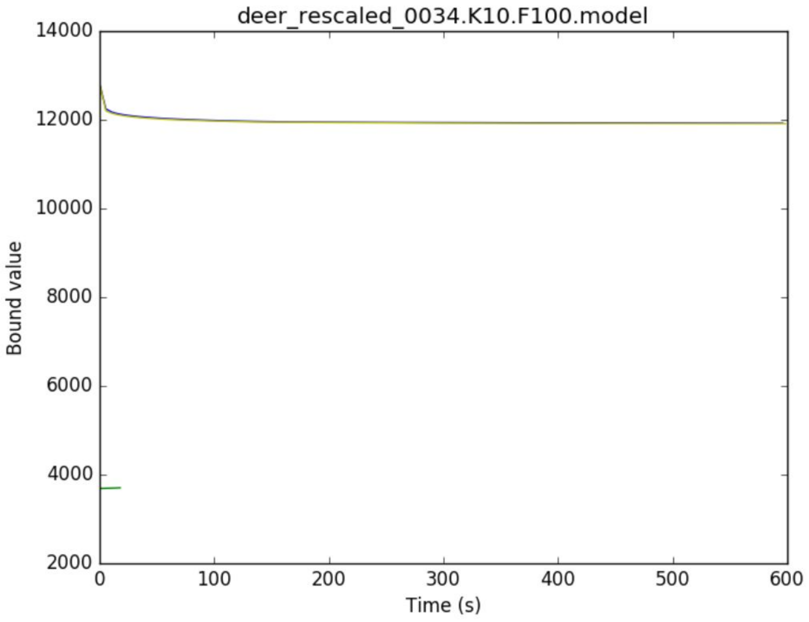


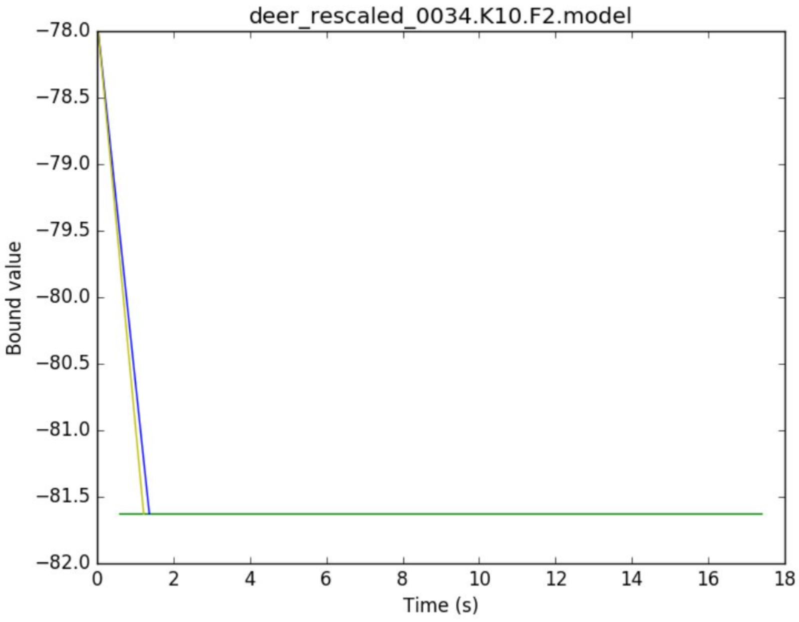
(a) BN domain



(b) BN domain

AOBeam vs AOBF vs AOBB





Comparison with AOBF

- AOBeam prunes faster allowing for faster bounding of the solution.

- The bounding of the solution doesn’t happen as fast as we thought.

Stochastic AOBeam

Idea: reduce the value of β and improve the pruning heuristic

Besides the β-best nodes, include every other node with a probability proportional to its heuristic value.

Stochastic AOBeam

For h’s that are upper bounds.

$$p(n) = \frac{h(n)}{\sum_{n' \in N} h(n')}$$

For h’s that are lower bounds.

$$p(n) = 1 - \frac{h(n)}{\sum_{n' \in N} h(n')}$$

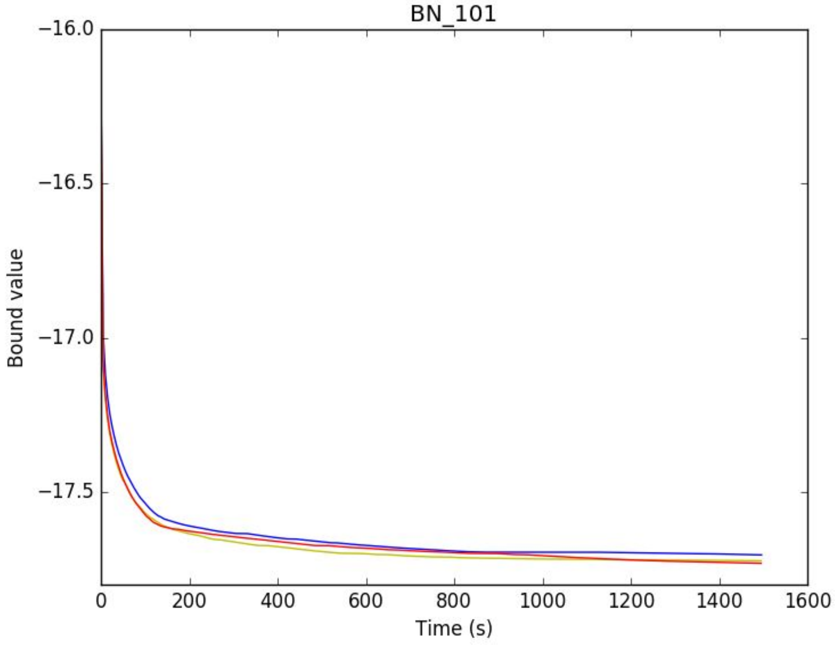
Stochastic AOBeam

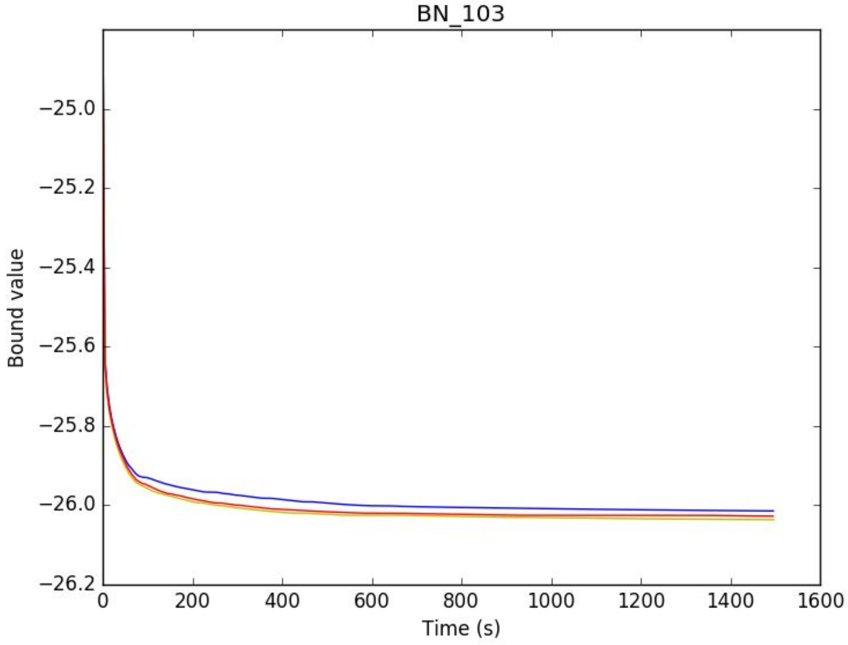
Besides, add an α that adds up to this probability:

$$p'(n) = p(n) + \alpha$$

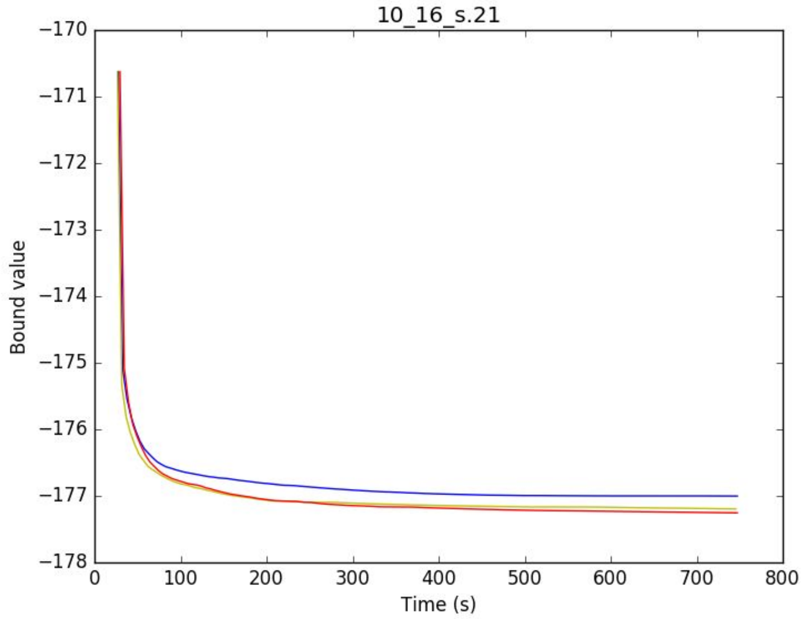
Allows for tuning of the pruning policy.

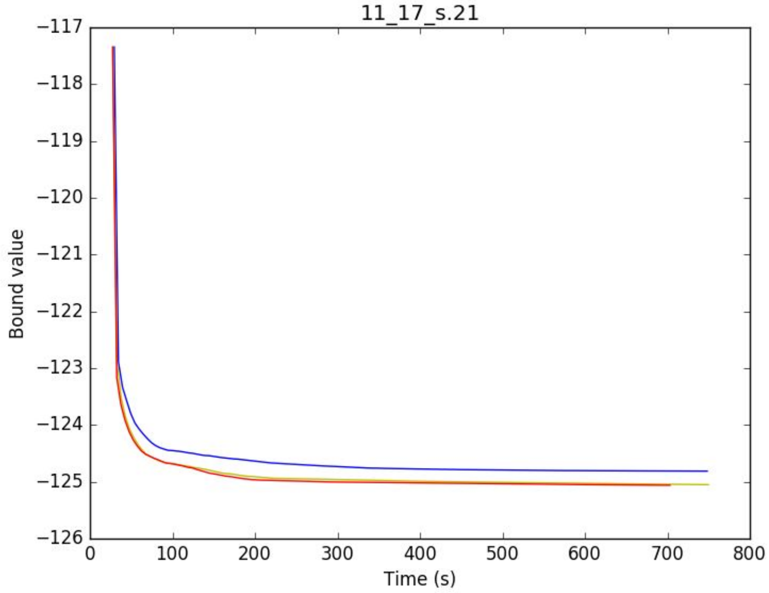
Stochastic AOBeam (β = 1, α=0) vs AOBeam (β = 2) vs AOBF



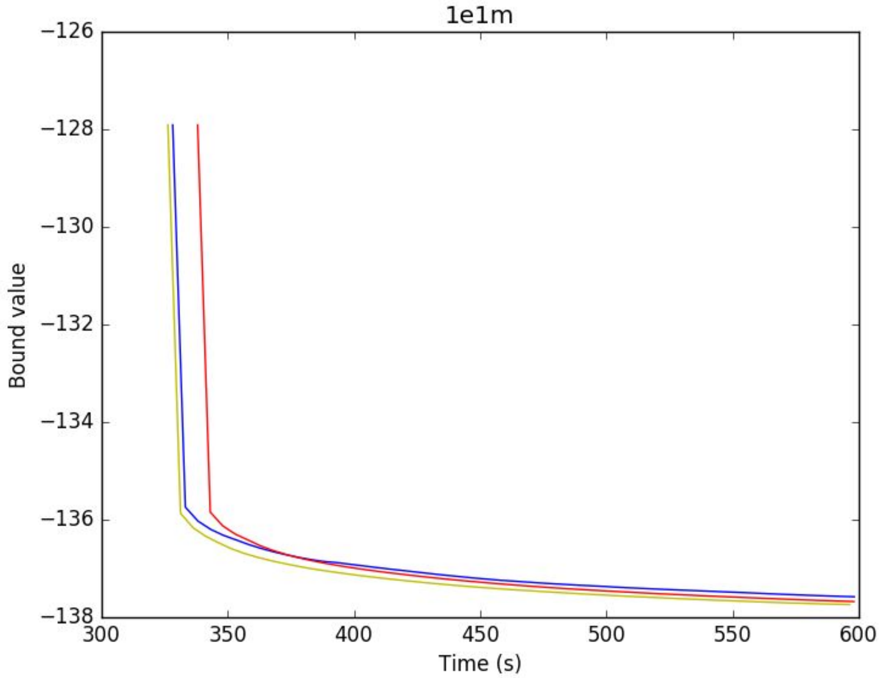


Stochastic AOBeam (β = 1, a = 0) vs AOBeam (β = 2) vs AOBF

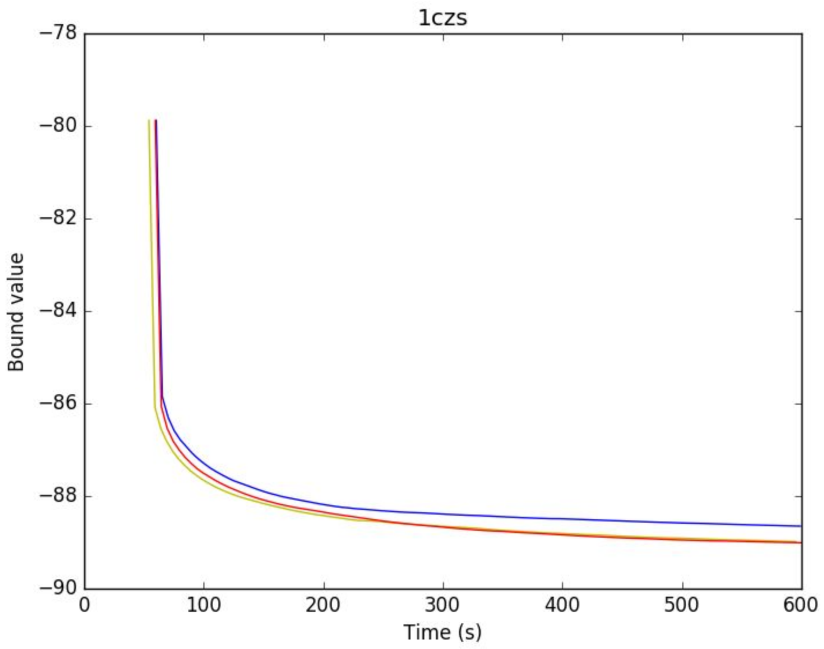


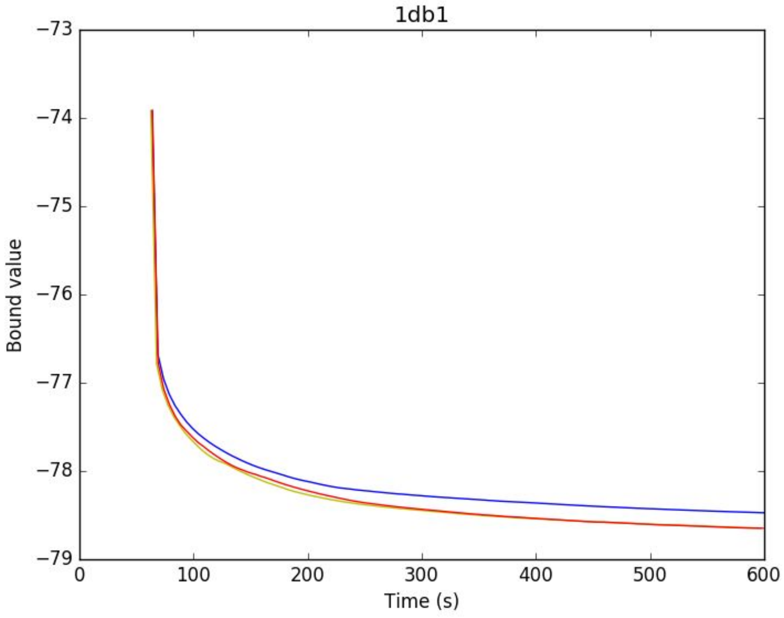


Stochastic AOBeam (β = 1, a = 0) vs AOBeam (β = 3) vs AOBF



Stochastic AOBeam (β = 1, a = 0.05) vs AOBeam (β = 3) vs AOBF





Anytime AOBeam (Theoretical)

Stochastic AOBeam + Anytime ideas

Weaken pruning policy each iteration by:

a) Increasing the value of β (may be too expensive)
b) Increasing the value of α

Anytime AOBeam

ALGORITHM AAOBeam(β, α)

```
v = -∞
s = ∅
while stopping condition not met:
    v', s' = StochasticAOBeam(β, α)
    if v' > v:
        v = v'
        s = s'
    weaken_pruning_policy()
```

Conclusions

- AOBeam's pruning is beneficial but it's not as aggressive as first thought.

- Alternatives should be found to overall have only β open paths overall.

Future work

- Vary α or β during the execution. Allows for faster pruning.

- Make AAOBeam incremental. Using the updated values found in previous searches as heuristics.

This is the end…

Thanks!