Look-Ahead with Mini-Bucket Heuristics for MPE

Rina Dechter[1], Kalev Kask[1], **William Lam**[1], Javier Larrosa[2]

[1]University of California, Irvine

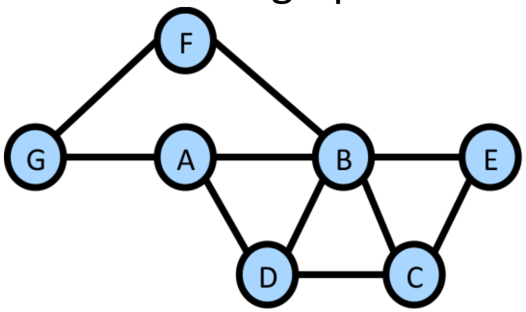
[2]UPC Barcelona Tech

Outline

- Background
  - Graphical models, MPE, Branch-and-Bound
  - Look-ahead
  - Mini-bucket Heuristic
- Bucket Error
- Look-ahead with Bucket Error
- Experiments
- Conclusions

Graphical Models and Finding an Optimal Assignment [Marinescu and Dechter 2008]

A Primal graph



$$f_1(A) + f_2(A,B) + f_3(A,D) + f_4(A,G) + f_5(B,C) + f_6(B,D) + f_7(B,E) + f_8(B,F) + f_9(C,D) + f_{10}(C,E)$$

A	f_1
0	3
1	2

A	B	f_2
0	0	2
0	1	0
1	0	1
1	1	4

A	D	f_3
0	0	3
0	1	0
1	0	0
1	1	1

A	G	f_4
0	0	0
0	1	3
1	0	2
1	1	0

B	C	f_5
0	0	2
0	1	0
1	0	0
1	1	2

B	D	f_6
0	0	0
0	1	1
1	0	2
1	1	4

B	E	f_7
0	0	4
0	1	2
1	0	1
1	1	0

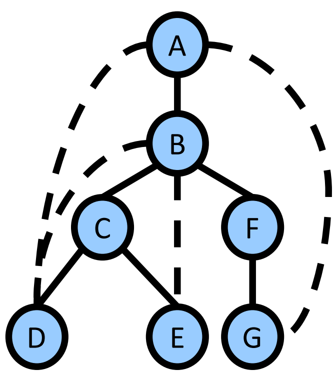
B	F	f_8
0	0	3
0	1	2
1	0	1
1	1	0

C	D	f_9
0	0	1
0	1	4
1	0	0
1	1	0

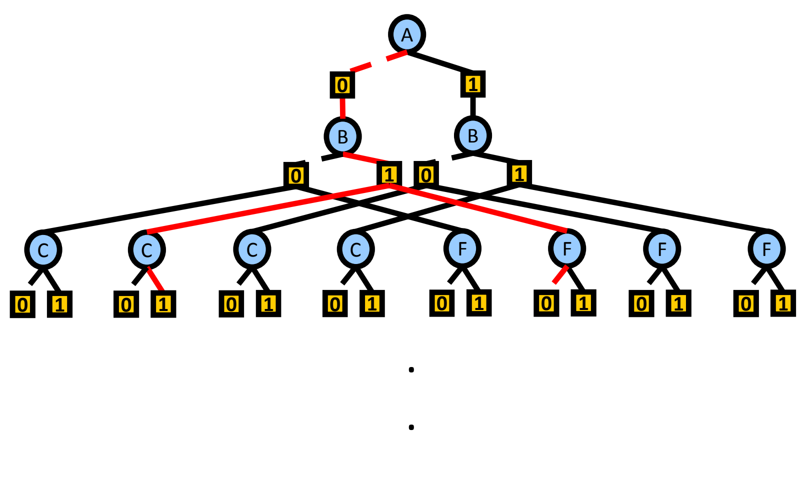
C	E	f_{10}
0	0	1
0	1	0
1	0	0
1	1	2

$$F(\mathbf{X}) = \sum_{i=1}^{10} f_i(\mathbf{X}) \qquad MPE = \min_{\mathbf{x}} F(\mathbf{X})$$

Pseudo-tree



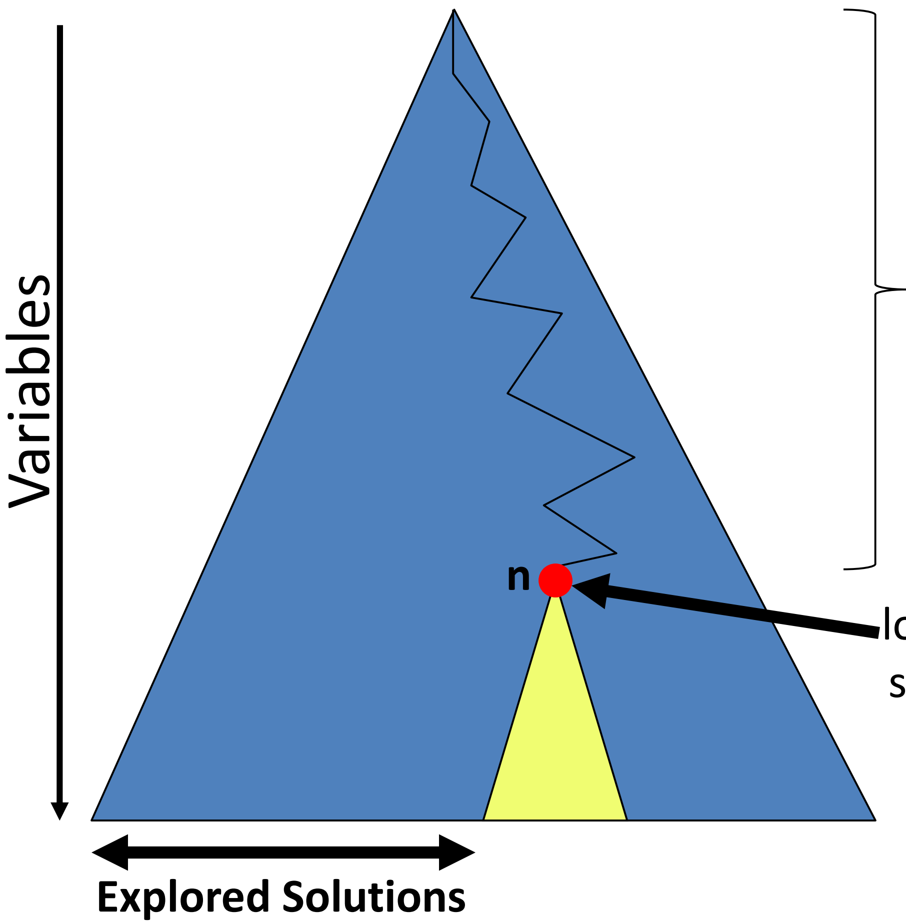
AND/OR search space



Approach: Search

- Depth-first AND/OR Branch and Bound (AOBB)
- Heuristic: mini-bucket elimination (MBE) + variational cost-shifting [Ihler et al 2012, Otten et al 2012]

Depth-First Branch and Bound (DFBB)



Each node *n* is a ***subproblem***
(defined by current conditioning)

g(n)=
cost to get to node n

h(n)=
lower bound of the best solution in the sub-tree

f(n) = g(n)+h(n)
estimate of overall solution given the conditioning to node n

If
f(n) is worse than current best solution,
then prune.

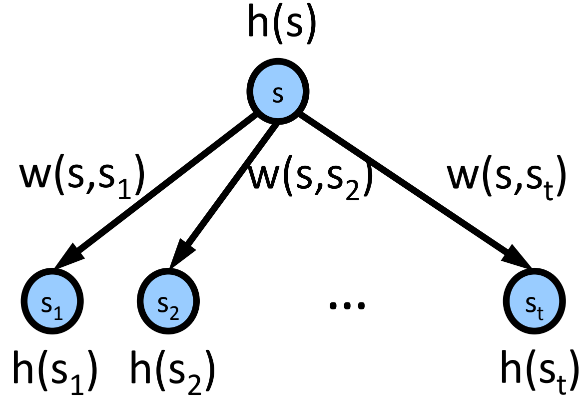
Look-Ahead in Search

- Given that $s_1,\ldots,s_t$ are child nodes of *s* in the search space and $w(s,s_i)$ is the weight of the arc from s to s_i, $h^{lh(d)}$ be the *d*-level lookahead function of *s*, then

$$h^{lh(1)}(s) = \min_{\{s1,\ldots,st\} \text{ in child}(s)} \{w(s, s_i) + h(s_i)\}$$

$$h^{lh(d)}(s) = \min_{\{s1,\ldots,st\} \text{ in child}(s)} \{w(s, s_i) + h^{lh(d-1)}(s_i)\}$$

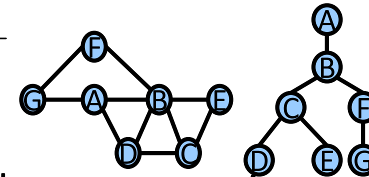
- The (1-level) *residual*: $res_h(s) = h^{lh(1)}(s) - h(s)$
- Can be viewed as a search problem over the next *d* levels



- **Our focus**: Can we cost-effectively improve our heuristic with look-ahead?

Bucket and Mini-Bucket Elimination

$$f_1(A) + f_2(A,B) + f_3(A,D) + f_4(A,G) + f_5(B,C) + f_6(B,D) + f_7(B,E) + f_8(B,F) + f_9(C,D) + f_{10}(C,E)$$



$$F(\mathbf{X}) = \sum_{i=1}^{10} f_i(\mathbf{X})$$

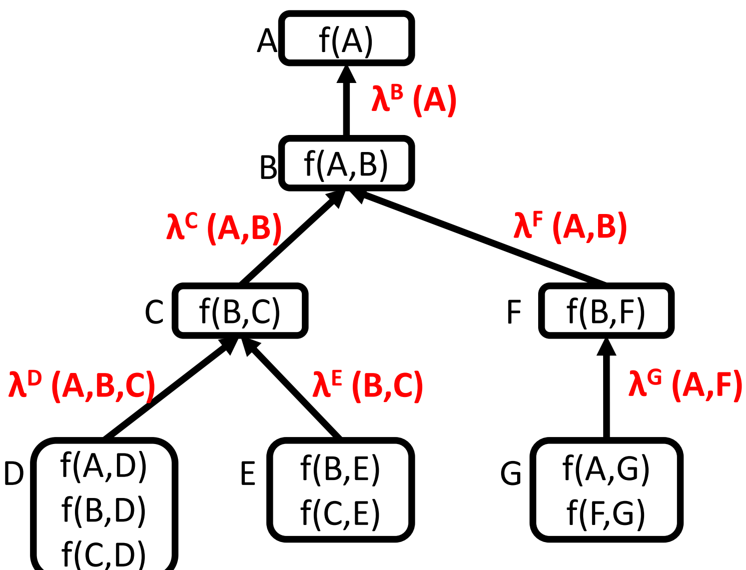
$$MPE = \min_{\mathbf{x}} F(\mathbf{X})$$

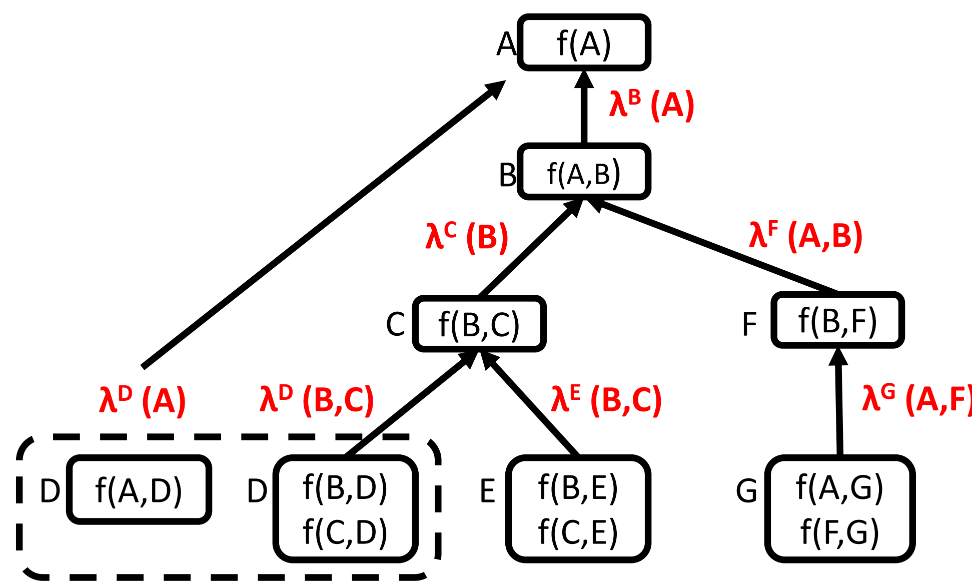
Bucket Elimination (BE) (Dechter 1999)

- Solves the min-sum problem by eliminating variables one at a time.
- Complexity: exponential in the *w** of the underlying primal graph

Mini-Bucket Elimination **(MBE) (Dechter and Rish 2001)**

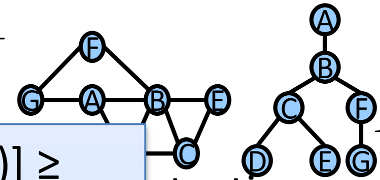
- We can *approximate* BE by solving a relaxation created by duplicating variables
- to bound the w* by a parameter known as the *i-bound.*





Bucket and Mini-Bucket Elimination

$$f_1(A) + f_2(A,B) + f_3(A,D) + f_4(A,G) + f_5(B,C) + f_6(B,D) + f_7(B,E) + f_8(B,F) + f_9(C,D) + f_{10}(C,E)$$



$$F(\mathbf{X}) = \sum_{i=1}^{10} f_i(\mathbf{X})$$

$$MPE = \min_{\mathbf{x}} F(\mathbf{X})$$

Bucket Elimination (BE)

- Solves the min-sum ... at a time.
- Complexity: expon... al graph

Mini-Bucket Eliminatio...

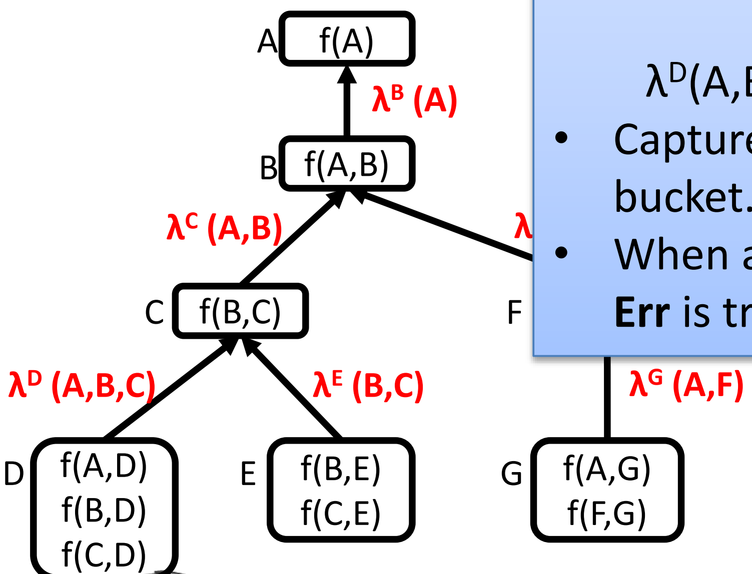
- We can *approxima*... duplicating variables
- to bound the w* b...

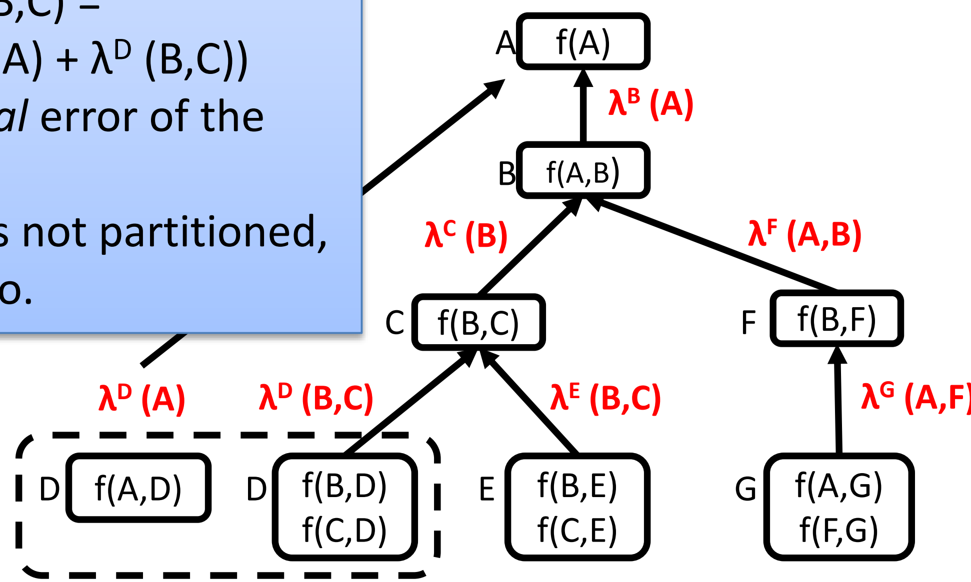
$\min_D [f(A,D) + f(B,D) + f(C,D)] \geq \min_D [f(A,D)] + \min_D [f(B,D) + f(C,D)]$

$\lambda^D(A,B,C) \geq \lambda^D(A) + \lambda^D(B,C)$

- ***Bucket Error***

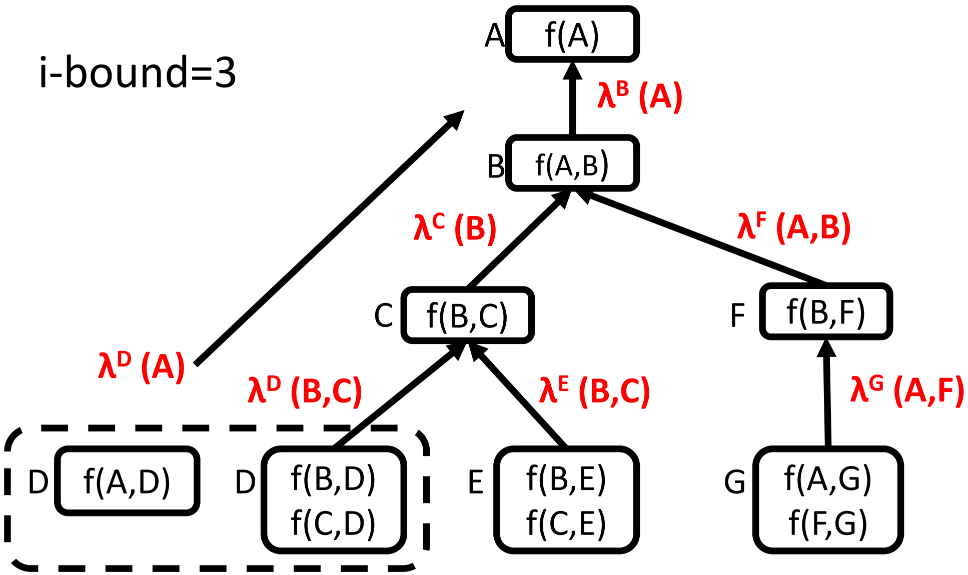
  $\mathbf{Err_D}(A,B,C) = \lambda^D(A,B,C) - (\lambda^D(A) + \lambda^D(B,C))$
- Captures the *local* error of the bucket.
- When a bucket is not partitioned, **Err** is trivially zero.

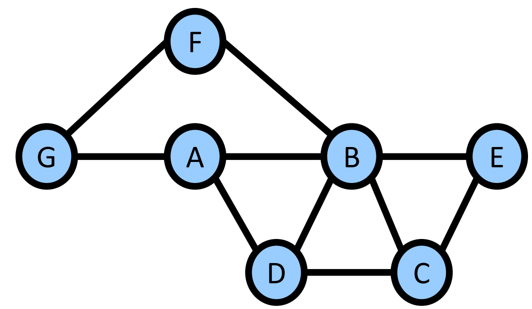




Mini-Bucket Errors and the i-bound

i-bound=3





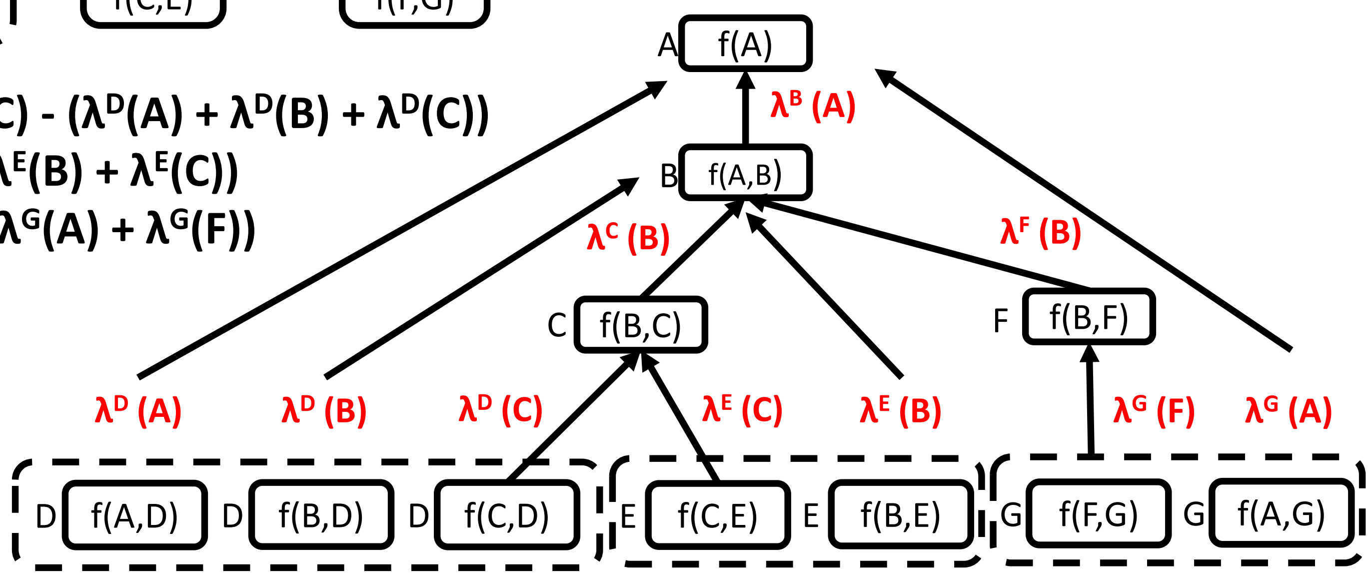
$$Err_D(A,B,C) = \lambda^D(A,B,C) - (\lambda^D(A) + \lambda^D(B,C))$$

$$Err_D(A,B,C) = \lambda^D(A,B,C) - (\lambda^D(A) + \lambda^D(B) + \lambda^D(C))$$

$$Err_E(B,C) = \lambda^E(B,C) - (\lambda^E(B) + \lambda^E(C))$$

$$Err_G(A,F) = \lambda^G(A,F) - (\lambda^G(A) + \lambda^G(F))$$

i-bound=2



Bucket Error Evaluation (BEE)

Algorithm 1: Bucket Error Evaluation (BEE)

Input: A Graphical model $\mathcal{M} = (\mathbf{X}, \mathbf{D}, \mathbf{F})$, a pseudo-tree T, i-bound

Output: The error function Err_k for each bucket

Initialization: Run $MBE(i)$ w.r.t. T.

for *each* $B_k, X_k \in \mathbf{X}$ **do**

  Let $B_k = \cup_k B_k^r$ be the partition used by $MBE(i)$

  $\mu_k = \sum_r (min_{X_k} \sum_{f \in B_k^r} f)$

  $\mu_k^* = min_{X_k} \sum_{f \in B_k} f$

  $Err_k \leftarrow \mu_k^* - \mu_k$

return Err functions

Complexity of BEE

THEOREM 1 (Complexity of BEE)
The complexity of BEE is $O(n \cdot k^{psw(i)})$*, where* n *is the number of variables,* k *bounds the domain size and* $psw(i)$ *is the pseudo-width along* T *relative to MBE(i).*

psw_j: the pseudo-width of bucket *j* is the number of variables in the bucket at the time of processing.

Residual and Bucket Error

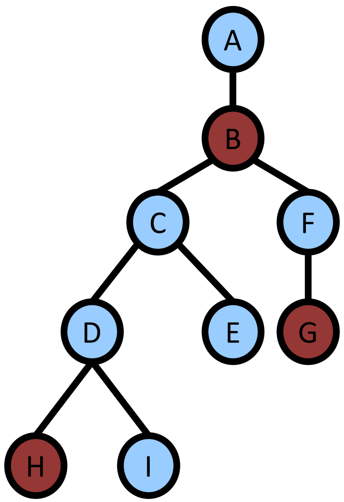
THEOREM **2 (residual and bucket-error)** *Assume an execution of $MBE(i)$ along T yielding heuristic h Then, for every $\bar{x}_p$*

$$res(\bar{x}_p) = \sum_{X_k \in ch(X_p)} Err_k(\bar{x}_p) \tag{9}$$

Minimal Look-ahead Subtree

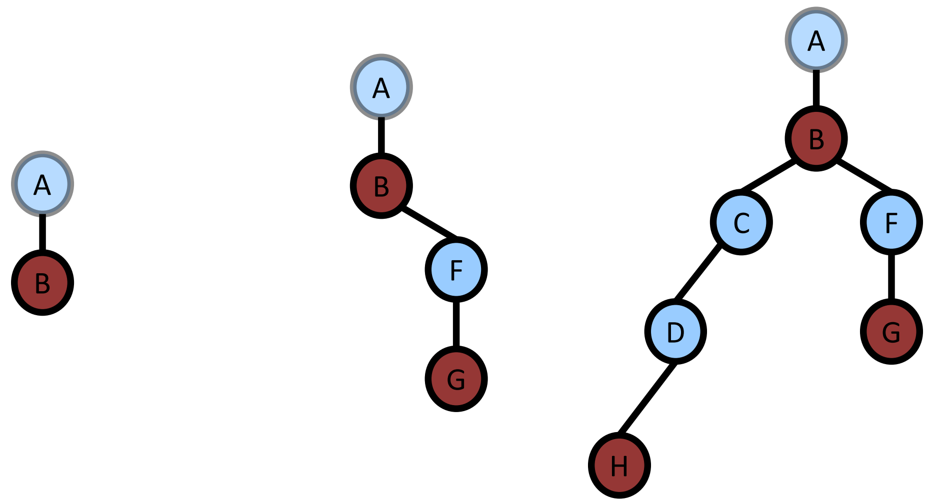
DEFINITION **6 (average relative bucket error)** *The average relative bucket error of X_j given a run of MBE(i) is*

$$\tilde{E}_j = \frac{1}{|dom(B_j)|} \sum_{\bar{x}_j} \frac{Err_j(\bar{x}_j)}{\mu^*(\bar{x}_j)}$$



Pseudo-tree: Red nodes are look-ahead ***relevant*** (relative error above a certain threshold)

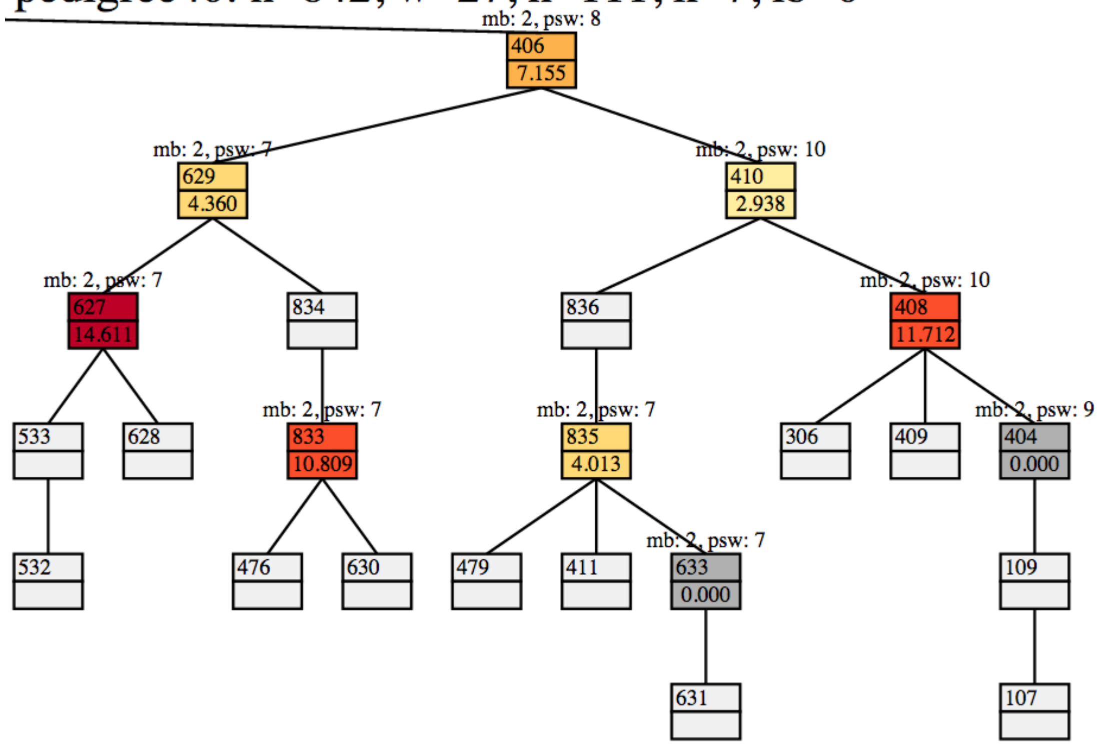
Look-ahead subtrees from A, by depth

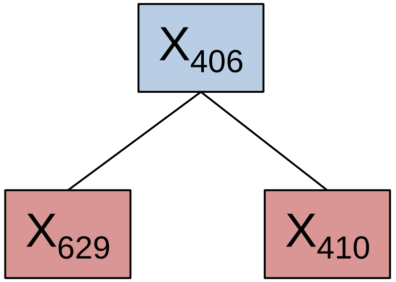


Depth 1 Depth 3 Depth 5

Sample of a part of a pseudotree (pedigree40)

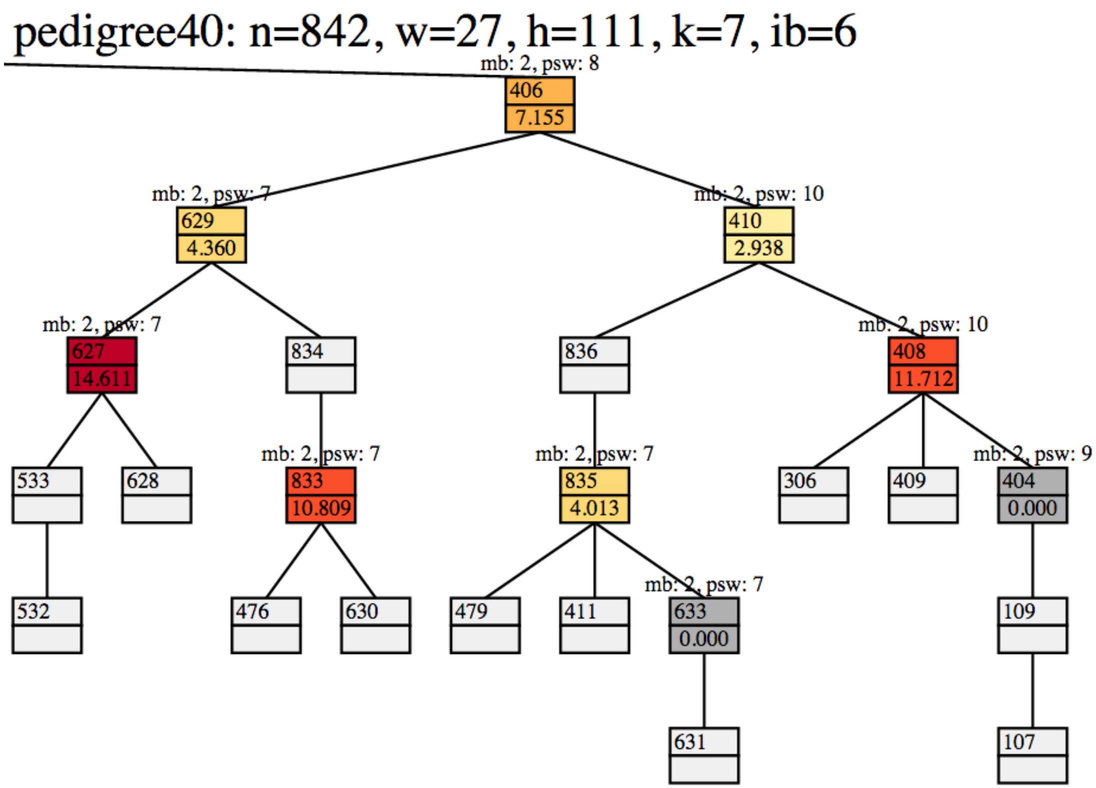
pedigree40: n=842, w=27, h=111, k=7, ib=6

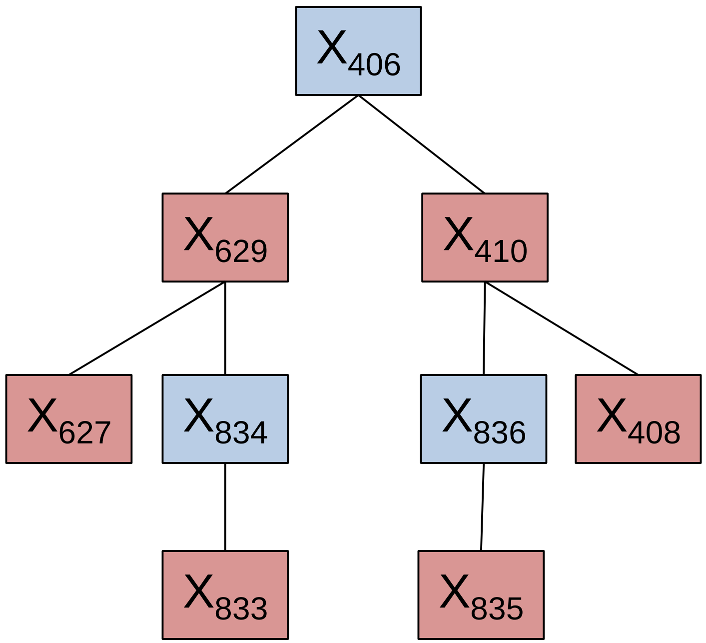




depth = 5,
threshold = 0

Sample of a part of a pseudotree (pedigree40)

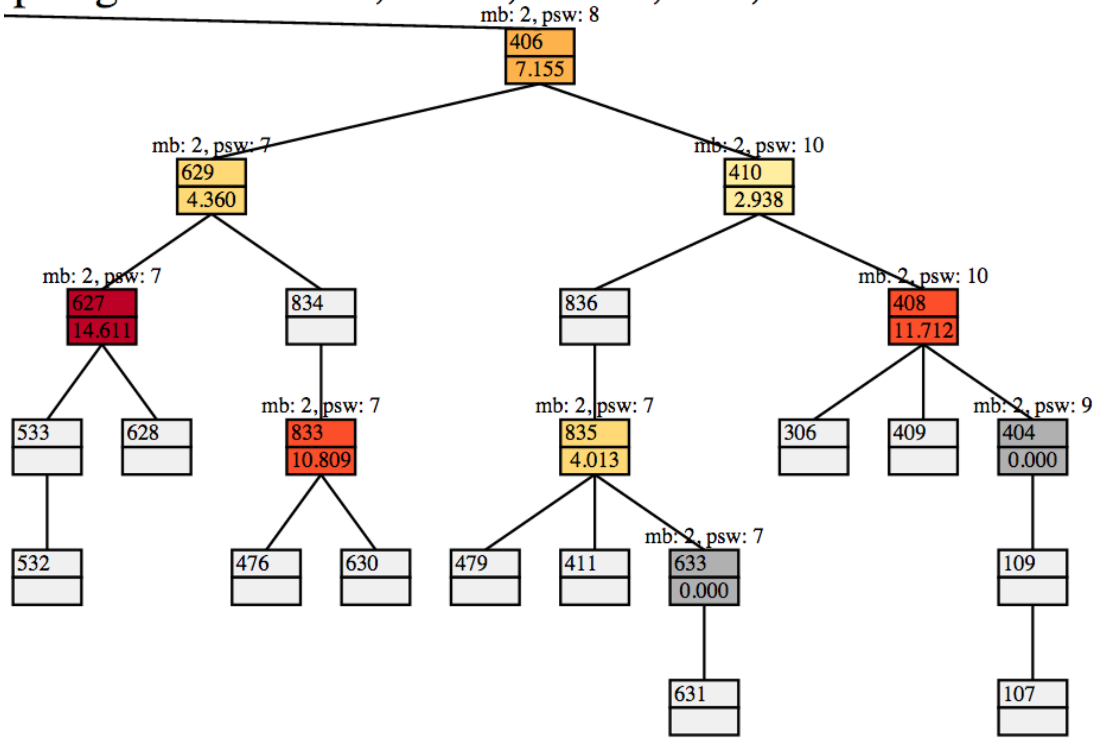


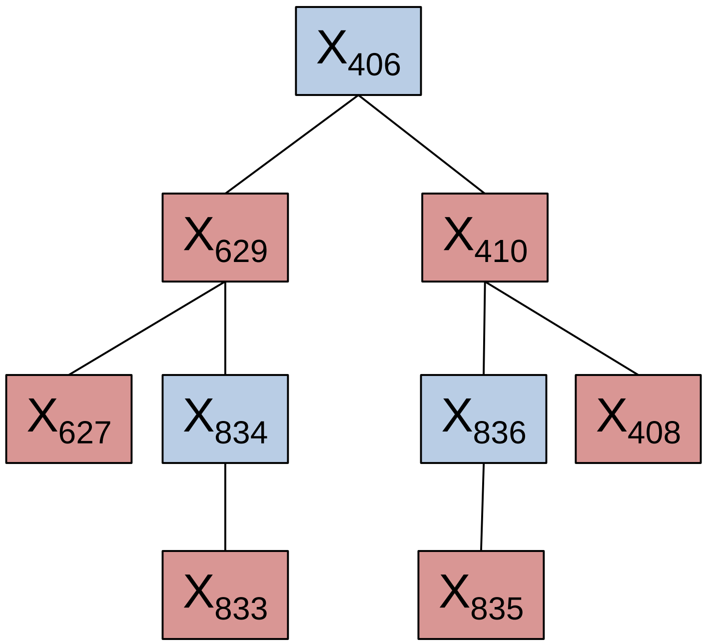


depth = 3,
threshold = 0

Sample of a part of a pseudotree (pedigree40)

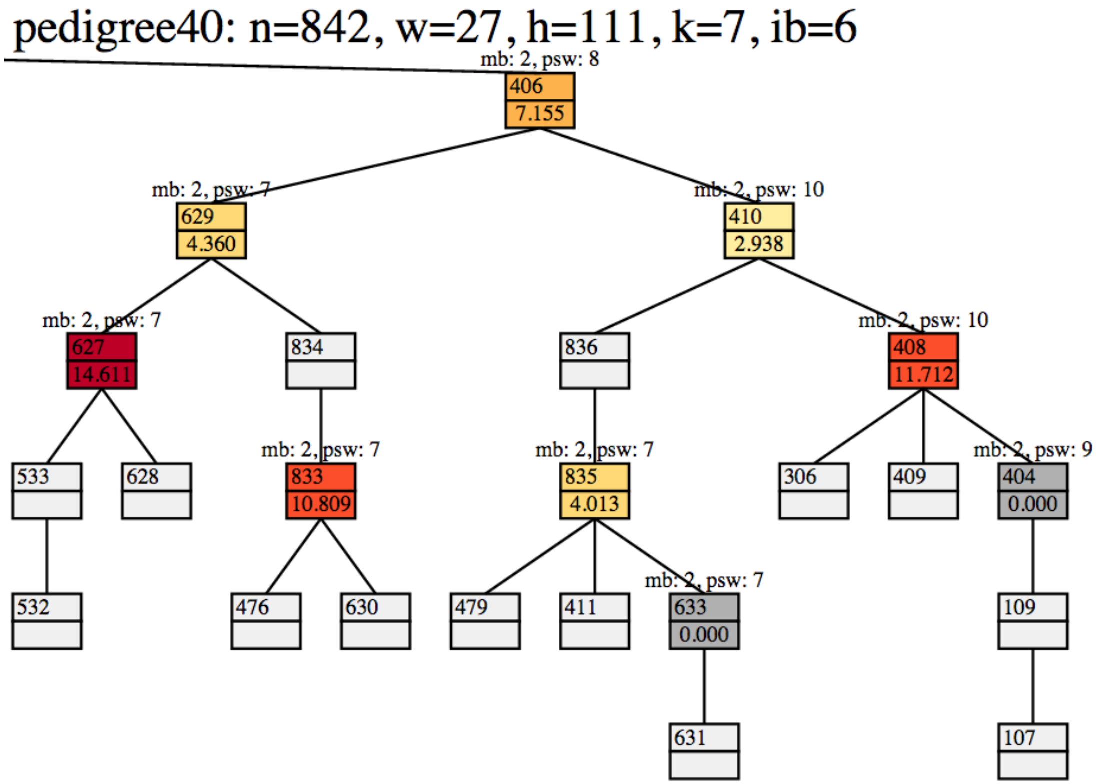
pedigree40: n=842, w=27, h=111, k=7, ib=6

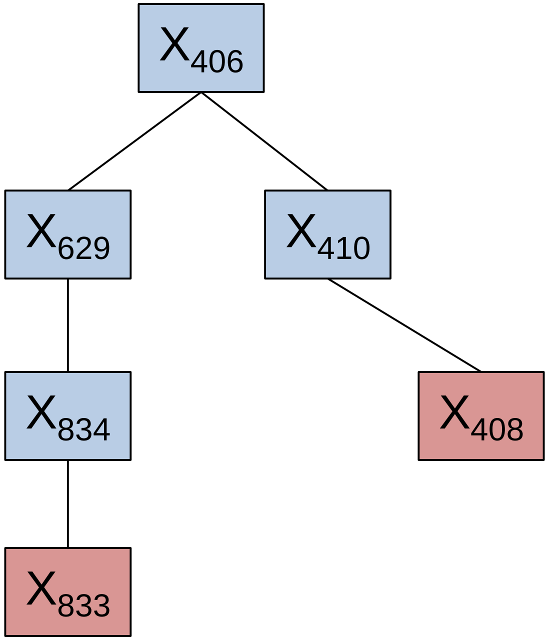




depth = 5,
threshold = 0

Sample of a part of a pseudotree (pedigree40)

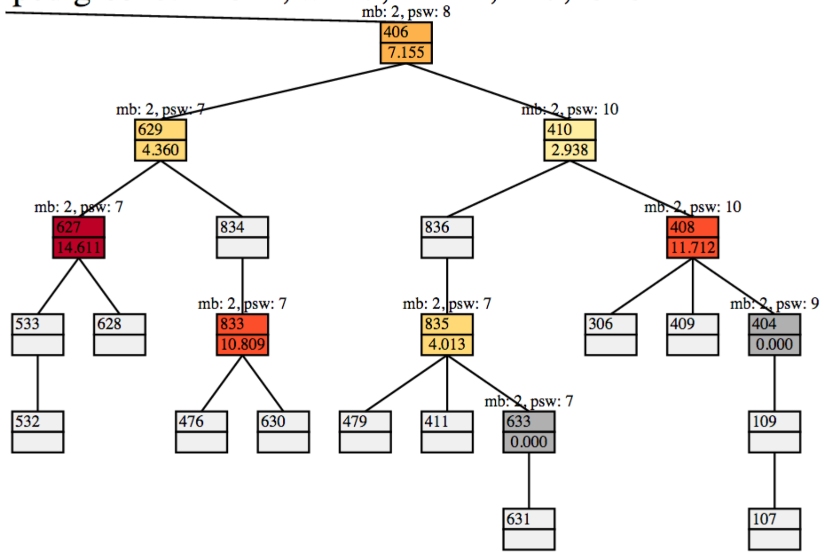




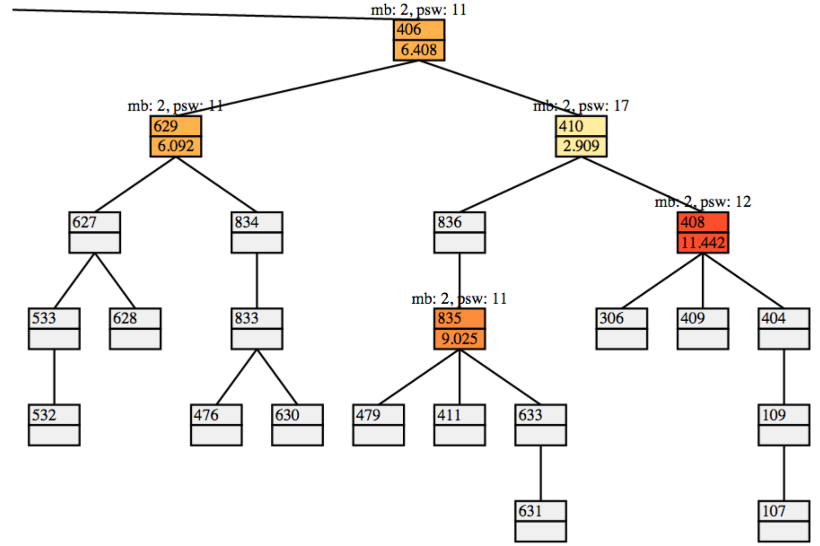
depth ≥ 3,
threshold = 9.5

Bucket-Errors: Across i-bounds (pedigree40)

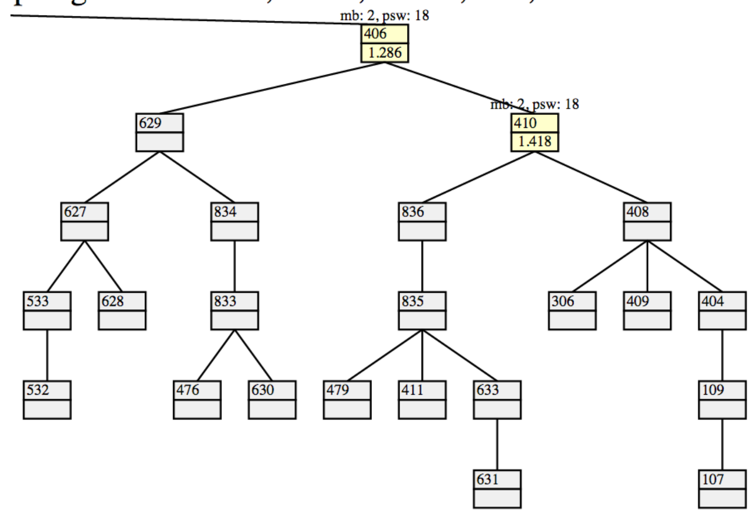
pedigree40: n=842, w=27, h=111, k=7, ib=6



pedigree40: n=842, w=27, h=111, k=7, ib=10



pedigree40: n=842, w=27, h=111, k=7, ib=14



Experiments

- AOBB on the context-minimal AND/OR graph
- MBE-MM heuristic with different i-bounds
- Time limit: 6 hours
- MBE memory: 4GB
- Pruned look-ahead trees with BEE
  - Compute error functions based on sampling at most 10^5 entries. (Exact if there are fewer entries.)
  - Error threshold of 0.01.

Experiments

- Benchmark Statistics

Benchmark	# inst	n	k	w	h	$\|F\|$	a
Pedigree	17	387 1015	3 7	19 39	58 143	438 1290	4 5
LargeFam	14	950 1530	3 3	32 40	66 95	1383 2352	4 4
Promedas	28	735 2113	2 2	41 120	77 180	749 2134	3 3

Experiments

instance	Lookahead depth	time	nodes		time	nodes
		i=5			i=8	
pedigree 7	none	1262	826K		35	23K
	1	912	564K		20	13K
	3	691	311K		***12***	6K
	6	***300***	66K		13	2K
		i=17			i=18	
lf3_11_53	none	10427	6730K		4349	2809K
	1	8611	4875K		3653	2116K
	3	***5481***	1674K		***2750***	901K
	6	20147	583K		10918	323K

Experiments: Summary over Instances

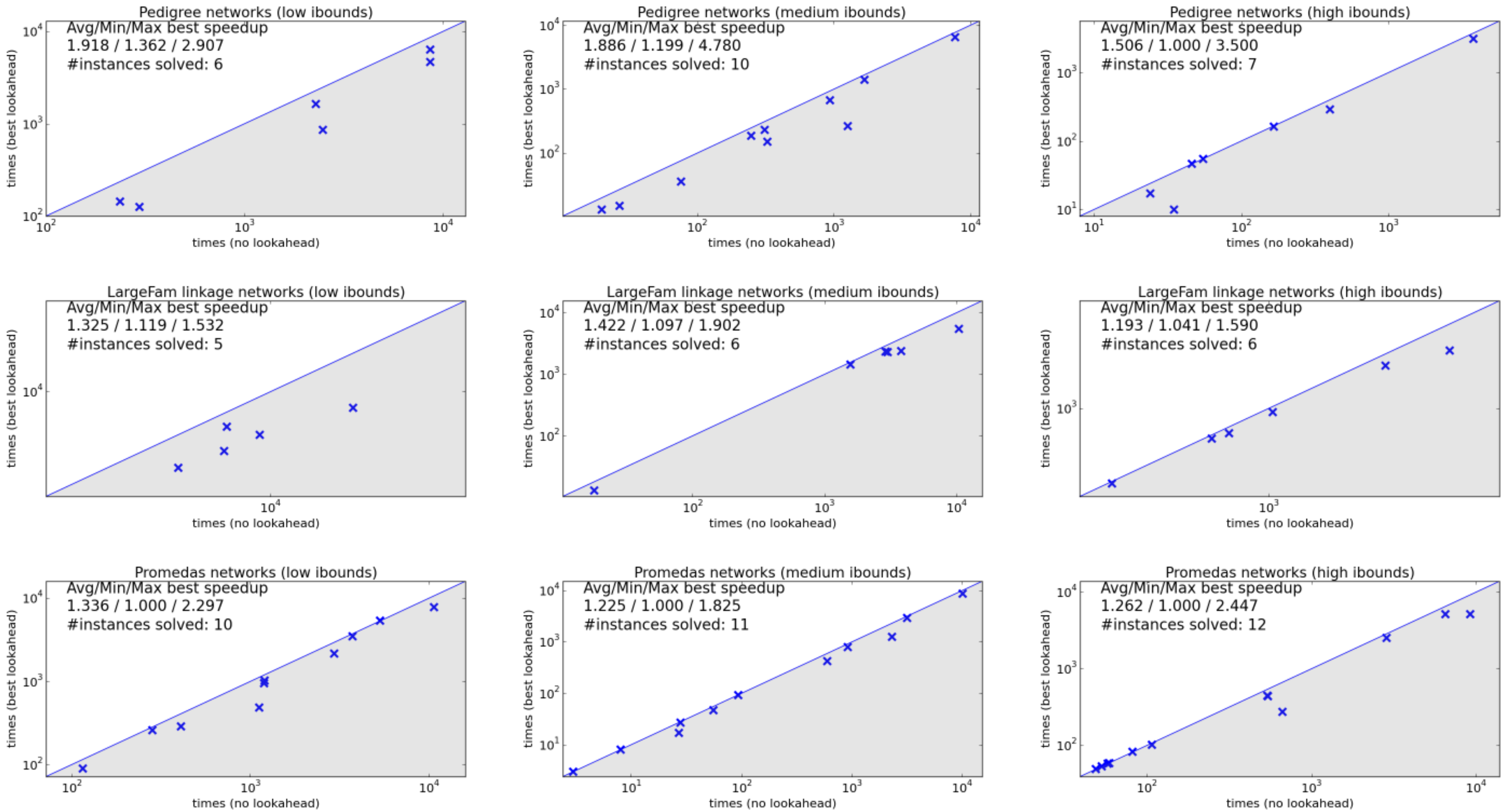
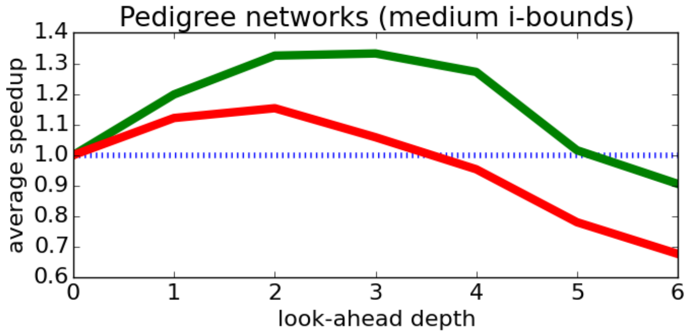
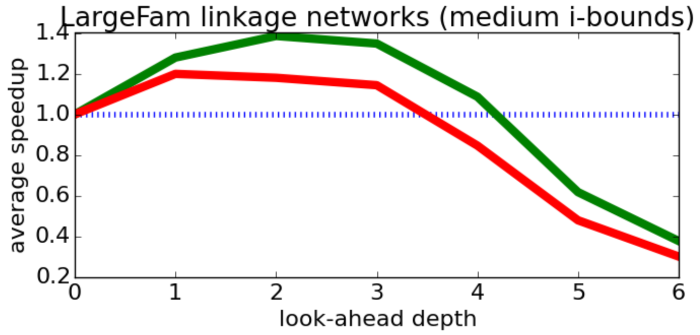
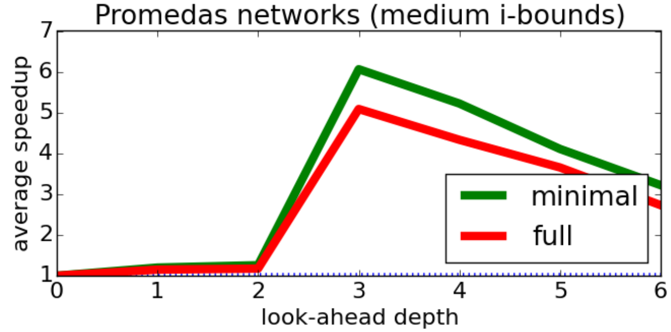


Figure 2: Total CPU times for using no look-ahead plotted against best total CPU times for any level of look-ahead (in log scale). Each point in the gray region represents an instance where the look-ahead heuristic has better time performance for some setting of the depth. We also report the number of instances which were actually solved and the average speedup across these instances.

Experiments: Summary over Instances







Conclusion

- We introduced the notion of bucket error to estimate the accuracy of the MBE heuristic
- We can make look-ahead cost-effective for MBE heuristics using bucket errors
- Future work: Applying the techniques described here to best-first search and work towards anytime algorithms that produce lower and upper bounds.

Thanks!