Dynamic Mini-Bucket Heuristics for AOBB

William Lam

June 27, 2013
Overview

• Compute MPE using via AOBB, using heuristics to provide pruning in the search space
• Search space size: $O(n \exp(w))$
• Static MBE(i) heuristic
  – Compute once as a preprocessing step
  – Fast: preprocessing takes $O(n \exp(i))$
Static Mini-Bucket Heuristics

h(a, b, c) = h^D(a) + h^D(b, c) + h^E(b, c) ≤ h^*(a, b, c)

Ordering: (A, B, C, D, E, F, G)
Overview

• Dynamic MBE(i) heuristic
  – Compute MBE(i) at each search node on the subproblem induced by the current conditioning
  – Slow: must do $O(n \exp(i))$ work for each search node on the i-cutset so we have $O(n \exp(c+i))$, where $c$ is the size of the i-cutset
  – Accurate: Heuristic tends to be more accurate the deeper we get in the search space
    • Much fewer nodes expanded, but in practice still shows to have significant overhead
      – Pruning needs to reduce nodes searched by at least a factor of $\exp(c)$
Dynamic Mini-Bucket Heuristics

Ordering: (A, B, C, D, E, F, G)

h(a, b, c) = h^D(c) + h^E(c)
= h^*(a, b, c)
Overview

• Dynamic MBE(i) heuristic
  – Conclusions in past work (Marinescu 2008)
    • Cost effective at lower i-bound, so useful when memory is very limited

• Current effort
  – Find a balance between the static and full dynamic schemes
Scheduling

Definition 1. (compatible heuristic) - A MBE heuristic \( h \) is compatible with search node \( n' \) iff \( h \) was computed at \( n \) and \( n \) is an ancestor of \( n' \). Any compatible MBE heuristic \( h \) for a node \( n \) is admissible.

• Generalization of the static scheme
  – Each heuristic corresponds to the static heuristic that would be compiled for the subproblem corresponding to that node
Problem Formulation

$t_e$: time to expand a search node
$t_h$: time to compile a heuristic
$H$: set of heuristics
$N(H)$: number of nodes expanded in search using $H$

- Goal: Minimize $N(H) \times t_e + |H| \times t_h$
- However, computing $N(H)$ requires solving the problem using $H$
- First approaches: focus on minimizing $|H|$
General Criteria

• Computing heuristics every $g$ expansions only
• Only up to a maximum search depth $d$
Problem Specific Features

• Only if the conditioned subproblem has $r$ fewer duplicated variables than the conditioning used for the previous heuristic
  – Since conditioning can reduce the number of mini-buckets, we expect bounds to improve with less approximation
Problem Specific Features

Effectively 8 variables with no conditioning
Problem Specific Features

(Using static heuristic to evaluate)

Effectively 7 variables when evaluating with A=a
(A=a decomposes one copy of D into an independent subproblem)
Problem Specific Features

(Using conditioned dynamic heuristic to evaluate)

Effectively 6 variables when conditioning the model with A=a (bucket merge and conditioning variable)
Problem Specific Features (dynamic)

• Using information available only after compiling a heuristic to make decisions on future computations
  – Relative upper/lower bound gap
    • A good feature for estimating problem complexity (Otten and Dechter, UAI 2012)
    • If the condition is satisfied, then we do not compute any more heuristics for the search space rooted at that node

| Feature $\phi_i$                                      | $|\lambda_i|$ | coo |
|-----------------------------------------------------|--------------|-----|
| Average branching degree in probe                   | 0.57         | 100 |
| Average leaf node depth in probe                    | 0.39         | 87  |
| Subproblem upper bound minus lower bound            | 0.22         | 17  |
| Ratio of nodes pruned by heuristic in probe         | 0.20         | 27  |
| Max. context size minus mini bucket $i$-bound       | 0.19         | 16  |
| Ratio of leaf nodes in probe                        | 0.18         | 10  |
| Subproblem upper bound                              | 0.11         | 7   |
| Std. dev. of subproblem pseudo tree leaf depth      | 0.06         | 2   |
| Depth of subproblem root node in overall space      | 0.05         | 2   |
Problem Specific Features (dynamic)

- If the condition is satisfied, then we do not compute any more heuristics for the search space rooted at that node

\[
\frac{h_c - LB}{(h_c - LB) + (h_c - h_p)} < r
\]

- \(h_c\): current heuristic value
- \(h_p\): previous heuristic value
- \(LB\): current lower bound
- \(r\): relative decrease
Improving Heuristic Computation Time

Definition 2. (exact bucket) - A bucket of a minibucket tree is exact iff it has exactly one minibucket and if it has any child buckets, they are also exact.

Corollary 2. The messages used to bound the cost of a subproblem are exact costs iff the messages were sent from exact buckets.

- As a result, we can completely forego heuristic computation for a node with this condition
- Otherwise, we can share exact buckets between heuristics
- Does not change the worst case complexity for heuristic computation time, but can help in practice
Static Mini-Bucket Heuristics

Ordering: (A, B, C, D, E, F, G)

\[ h(a, b, c) = h_D(a) + h_D(b, c) + h_E(b, c) \leq h^*(a, b, c) \]
Dynamic Mini-Bucket Heuristics

Ordering: (A, B, C, D, E, F, G); Actual buckets processed from scratch: (E,D,C)

\[ h(a, b, c) = h^D(c) + h^E(c) = h^*(a, b, c) \]
Dynamic Mini-Bucket Heuristics

Conditioning: A=a, B=b; Getting bounds for all values of F

Ordering: (A, B, C, D, E, F, G); Actual buckets processed from scratch: None
Algorithm 1 Algorithm Dynamic-MBE

**Input:** Graphical model \( \langle X, D, F, \sum \rangle \); variable order \( \{X_1, \ldots, X_n\} \); parameter \( i \); most recent ancestor buckets \( B' \), variable at root of subproblem \( var \); partial assignment \( A \)

**Output:** cost bound for each subproblem corresponding to an assignment to \( var \)

1. **if** node does not meet requirements for heuristic computation **then**
2. **return** cost bounds from \( B'_{var} \)
3. **else**
4. Remove all irrelevant variables from \( o \) (non-descendants of \( var \))
5. Place each function \( f_o \) in its latest bucket in \( o \)
6. **for** \( i \leftarrow n' \) **downto** 1 (processing bucket \( B_j \)) **do**
7. **if** \( B'_j \) is exact **then**
8. \( \lambda_j \leftarrow \) Extract previously computed message of \( B'_j \) and place it in the bucket corresponding to its latest variable.
9. **else**
10. Condition each function in \( B_j \) with \( A \).
11. Partition functions in \( B_j \) into \( Q_j = \{q^1_j, \ldots, q^p_j\} \), where each \( q^k_j \) has no more than \( i + 1 \) variables.
12. Find the function of each mini-bucket \( q^k_j : F_{jk} \leftarrow \prod_{f \in q^k_j} f \)
13. Generate messages \( \lambda^k_j = \max_{X_j} F_{jk} \) and place each in the latest variable in \( var(q^k_j) \)’s bucket.
14. **end if**
15. **end for**
16. **return** cost bounds from \( B_{var} \)
17. **end if**
Experiments

• Every $g$ nodes ($10^0...10^8$)
• Depth $d$ (0,5,10,15,20,25,30)
• Duplicate reduction (0,5,10,15,20,25,30,35)
• Upper/lower bound threshold (0.6-0.975) (with depth limit 10)
Every g nodes (75-25-5)
Every $g$ nodes (pedigree 9)

![Graph 1](image1)

![Graph 2](image2)

![Graph 3](image3)

![Graph 4](image4)
Depth (75-25-5)
Results (75-25-5)
Results (pedigree9)
Upper/lower bound gap (75-25-5)
Upper/lower bound gap (pedigree19)
Discussion

• Computing every \( g \) nodes/with probability \( p \)
  – Not very effective
  – Computes many heuristics that are not helpful

• Maximum depth \( d \)
  – Generally improves over the static scheme when \( d \) is low
  – Computes the smallest set of heuristics that are compatible with as many nodes as possible
Discussion

• Duplicate reduction $r$
  – Does not provide a good tradeoff
  – May indicate duplicate reduction is not a sufficient feature for detecting a significance decrease in the bound
  – Issues
    • Minibuckets do not necessarily only merge, others may split
    • Different function ordering due to change in function scopes
Discussion

• Upper/lower bound
  – Capable of finding where the heuristic improves significantly if it exists in the problem
  – Essentially finds the best depth to stop at
Discussion

• Estimating $N(H)$
• Previous complexity estimation work in is a special case where $H$ contains only the static MBE heuristic
• More investigation into other possible dynamic features needed