On the Impact of Subproblem Orderings on Anytime AND/OR Best-First Search for Lower Bounds

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

Abstract. Best-first search can be regarded as anytime scheme for producing lower bounds on the optimal solution, a characteristic that is mostly overlooked. We explore this topic in the context of AND/OR best-first search, guided by the MBE heuristic, when solving graphical models. In that context, the impact of the secondary heuristic for subproblem ordering may be significant, especially in the anytime context. Indeed, our paper illustrates this, showing that the new concept of *bucket errors* can advise in providing effective subproblem orderings in AND/OR search.

1 INTRODUCTION

AND/OR best-first search (AOBF) is guided by two heuristic evaluation functions. The first, f_1, evaluates the potential cost of the best solution extending a current partial solution graph. The second heuristic, f_2, prioritizes the tip nodes of a current partial solution which will be expanded next. Quoting Pearl (page 54): "These two functions, serving in two different roles, provide two different types of estimates: f_1 estimates some properties of the set of solution graphs that may emanate from a given candidate base, whereas f_2 estimates the amount of information that a given node expansion may provide regarding the alleged priority of its hosting graph. Most work in search theory focus on the computation of f_1, whereas f_2 is usually chosen in an ad-hoc manner." [6]

Contributions. We explore the impact of the secondary heuristic in context of AOBF [4] for generating lower bounds in an anytime manner for *min-sum* problems over a graphical model (e.g., MAP in Markov networks [7] and weighted CSPs). Our proposed secondary heuristic for subproblem ordering is based on *bucket errors* [2]. We show empirically on three benchmarks that bucket errors provide relevant information on improving the lower bound when expanding a particular node. In particular, it can improve the anytime lower bound often, when compared to a baseline ordering. As far as we know, this is the first investigation of subproblem ordering in AOBF and of anytime best-first search.

2 BACKGROUND

Graphical Models A *graphical model* is a tuple $\mathcal{M} = (\mathbf{X}, \mathbf{D}, \mathbf{F})$, where $\mathbf{X} = \{X_i : i \in V\}$ is a set of variables indexed by a set V, $\mathbf{D} = \{D_i : i \in V\}$ is a set of finite domains of values for each X_i, and $\mathbf{F}$ is a set of discrete functions over subsets of $\mathbf{X}$. We focus on the *min-sum problem*, $C^* = \min_x \sum_{f \in F} f(x)$, which covers a wide range of reasoning problems such as MAP in graphical models or cost minimization in weighted constraint satisfaction problems.

Primal graph, pseudo-tree. The *primal graph* G of a graphical model $\mathcal{M}$ is a graph where each variable X_i is represented by a node and edges connect variables which appear in the same scope of any $f \in \mathbf{F}$. A *pseudo-tree* $\mathcal{T} = (V, E')$ of the primal graph $G = (V, E)$ is a rooted tree over the same nodes V such that every edge in $E - E'$ connects a node to an ancestor in $\mathcal{T}$.

AND/OR Search. The search space of a graphical model can be guided by the decomposition displayed in the pseudo tree, yielding an AND/OR search space. A (partial) solution of the problem is a subtree of the space known as a *(partial) solution tree*.

AND/OR best-first (AOBF) is a state-of-the-art algorithm, a variant of the AO* algorithm [5], specialized for the AND/OR search spaces over graphical models [4]. Keeping track of the explicated search space in memory, the algorithm maintains the current best partial solution tree at every step, together with an updated current best cost-estimate at each of the nodes, whose value at the root is the current best lower-bound estimate. When expanding the current partial solution tree, the algorithm orders its multiple AND leaf nodes, which corresponds to prioritizing between different subproblems and is where the f_2 heuristic comes into play. In most earlier work, the primary heuristic f_1 is used for subproblem orderings. Also, while the algorithm was considered only as a purely exact algorithm, the sequence of lower bounds generated at every step at the root node, can be seen as providing an anytime lower bound on the solution.

Mini-Bucket Elimination (MBE) [1]. The MBE scheme is an approximation which is obtained by applying the exact bucket elimination algorithm [1] to a relaxation of the problem. The MBE algorithm bounds the buckets' scopes by a parameter i, called i-bound, via partitioning the buckets into *mini-buckets*, which can be viewed as duplicating the bucket's variable. This yields lower bounds which are used as heuristics to guide AND/OR search algorithms. By design, MBE's time and space complexity is exponential in the i-bound.

Bucket Error [2]. The notion of bucket error was introduced recently as a function that captures the local error introduced by the mini-buckets. It is defined as the difference between the function-message that would have been computed in an *individual* bucket without partitioning and the message computed by MBE.

3 BUCKET ERRORS FOR ORDERING

Consider an encountered partial solution tree t that has the currently lowest f_1 but cannot be extended to an optimal solution. If $f_1(t) < C^*$, where C^* is the value of the optimal solution, the selection of which of its subproblems to expand next (the f_2 function) can

[1] University of California, Irvine, USA, {willmlam, kkask, dechter}@ics.uci.edu. Supported in part by NSF grants IIS-1526842, and IIS-1254071, the US Air Force under Contract No. FA8750-14-C-0011 under the DARPA PPAML program.

[2] UPC Barcelona Tech, Spain, larrosa@lsi.upc.edu. Supported in part by MINECO/FEDER under project TIN2015-69175-C4-3-R.

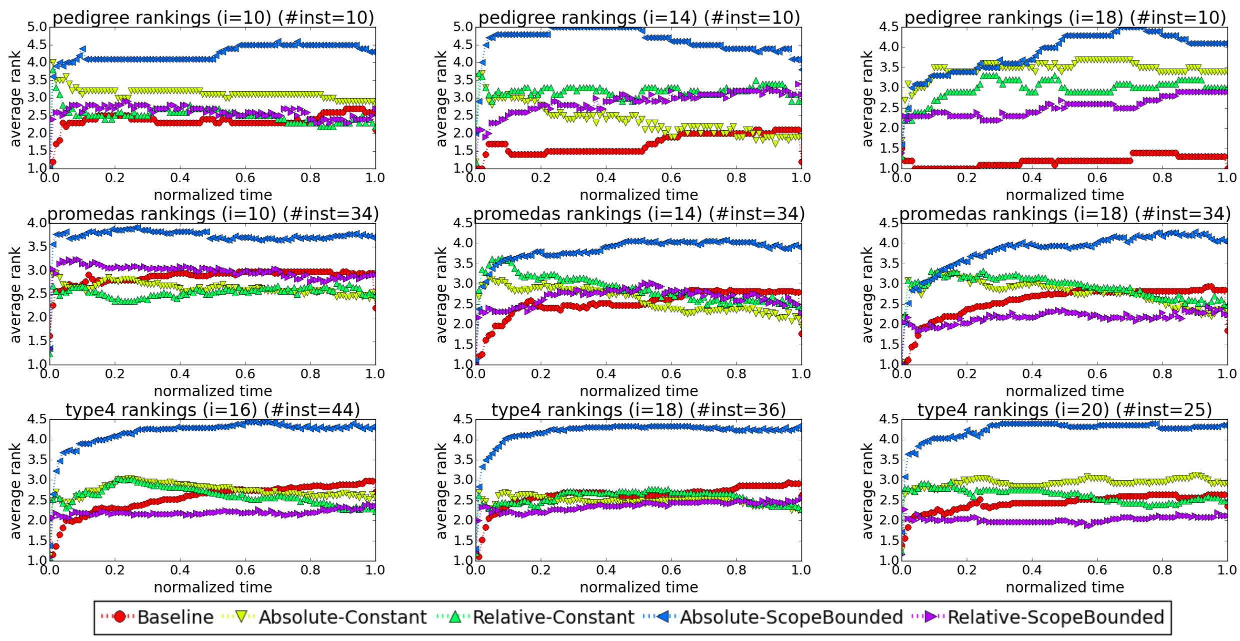

Figure 1: Average rankings over the normalized time relative to the baseline. Lower is better.

influence significantly the number of nodes extending it which will be explored eventually. It is therefore desirable to explore subproblems that yield the largest increase in its heuristic evaluation function first, to discover asap that its least cost extension is actually larger than C^*, thus avoiding the expansion of alternative branches (Best-first search will never expand a partial solution having $f_1 > C^*$.) Overall, we wish to prune partial solution graphs t that lead to sub-optimal solutions in as few node expansions as possible. Therefore, f_2 should prefer a subproblem leading to the largest increase in f_1.

With this observation, the *bucket error* is a natural choice for subproblem ordering, as it can be shown that it approximates the increase in f_1 [2]. We extend this to the notion of *subtree error*, where the bucket error of children are propagated to their ancestors to more accurately capture the mini-bucket error in an entire subproblem. As done before, the bucket-error associated with a variable can be approximated by taking absolute/relative average errors across a sample of the instantiations to the relevant bucket-error function, yielding a single constant value representing the error magnitude for each variable [2]. To generalize, we also use error functions for each variable. Since some error functions may not fit in memory, we bound the scopes of the error functions by removing variables and computing the average/relative error for each instantiation of the remaining variables. This yields 4 variants of our subproblem ordering heuristic (absolute/relative error and constant/scope-bounded error functions).

4 RESULTS AND CONCLUSION

We compare the 4 variants of our approach against the baseline subproblem ordering based on the heuristic evaluation function, f_1. We used mini-bucket elimination with moment-matching (MBE-MM) [3] for all experiments while varying the i-bound to show how differing amounts of error impact the performance. The pseudo-tree was fixed for all settings.

In **Figure 1**, we aggregated the results over each benchmark. We normalized the time scale for each instance to that of the baseline, ranked the bounds yielded by each variant across time, and aggregated across the instances by averaging. The number of instances varies with the different i-bounds since some instances run out of memory when computing the MBE heuristic with higher i-bounds.

We observe that the baseline was better for the pedigree instances, especially at the higher i-bounds. This is due to how the errors in the heuristic here are relatively low, and are therefore less informative. For the more difficult promedas and type4 benchmarks, there is more error in the heuristic, and our proposed subproblem ordering heuristics indeed improve over the baseline ordering. In particular, at the lowest i-bounds, nearly all of our methods improve over the baseline. At higher i-bounds, we see that the *Relative-ScopeBounded* variant outperforms the baseline the most and is the best performing of our 4 variants. We also improve as we move forward in time since the impact of the initial overhead of computing these heuristics becomes less relevant. Overall, our results show that the choice of the subproblem ordering heuristic impacts the performance of AOBF. Our method should be applied to any type of AND/OR best-first search, and future work includes extending to the various memory-efficient A* variants such as IDA*. For more details on the algorithms and experiments, a longer version of this paper is available here.[3]

REFERENCES

[1] Rina Dechter, *Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms*, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2013.

[2] Rina Dechter, Kalev Kask, William Lam, and Javier Larrosa, 'Look-ahead with mini-bucket heuristics for mpe', in *AAAI*, (2016).

[3] Alexander Ihler, Natalia Flerova, Rina Dechter, and Lars Otten, 'Join-graph based cost-shifting schemes', in *Uncertainty in Artificial Intelligence (UAI)*, 397–406, AUAI Press, Corvallis, Oregon, (August 2012).

[4] Radu Marinescu and Rina Dechter, 'Memory intensive and/or search for combinatorial optimization in graphical models', *Artif. Intell.*, **173**(16-17), 1492–1524, (2009).

[5] N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga, Palo Alto, CA, 1980.

[6] J. Pearl, *Heuristics: Intelligent Search Strategies*, Addison-Wesley, 1984.

[7] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, 1988.

[3] `http://www.ics.uci.edu/~dechter/publications.html`