Empowering Mini-Bucket in Anytime Heuristic Search with Look-Ahead: Preliminary Evaluation

William Lam, Kalev Kask, and Rina Dechter
University of California, Irvine
{willmlam,kkask,dechter}@ics.uci.edu

Abstract

The paper explores the potential of look-ahead methods within the context of AND/OR search in graphical models using the Mini-Bucket heuristic for combinatorial optimization tasks (e.g., weighted CSPS or MAP inference). We study how these methods can be used to compensate for the approximation error of the initially generated Mini-Bucket heuristics, within the context of anytime Branch-And-Bound search.

Introduction

This paper investigates the effect of look-ahead on the anytime performance of search solving the combinatorial optimization task of MPE in graphical models when using the mini-bucket heuristic (Dechter, Kask, and Lam 2015). There, we introduced the approach and showed a relationship between the residual of the mini-bucket heuristic and a newly defined notion of bucket-error.

Contributions. Look-ahead can be carried out to varying levels during a search process. When we have well behaved monotone heuristics, a more aggressive look-ahead yields a more accurate heuristic but requires more time. This paper builds on our recent work on look-ahead in which we developed the notion of a bucket-error. We showed that it coincides with the residual and can therefore predict one-level look-ahead when using the mini-bucket heuristic, facilitating an algorithm that computes look-ahead in this context efficiently. Here we provide preliminary empirical evaluation exploring the potential of look-ahead for anytime branch and bound search.

Background

A *graphical model* is $\mathcal{M} = (\mathbf{X}, \mathbf{D}, \mathbf{F})$, where $\mathbf{X} = \{X_i : i \in V\}$ is a set of variables indexed by a set V and $\mathbf{D} = \{D_i : i \in D\}$ is the set of finite domains of values for each X_i. $\mathbf{F} = \{f_\alpha : \alpha \in F\}$ is a set of discrete functions, where $\alpha \subseteq V$ and $X_\alpha \subseteq X$ is the scope of f_α. The functions' scopes imply a *primal graph* G where each variable X_i is a node and an edge (X_i, X_j) is in G iff the pair of variables appears in the scope of any f_α. We focus here on the *min-sum problem*, which is to compute $C^* = \min_{\mathbf{X}} \sum_{\alpha \in F} f_\alpha(X_\alpha)$, and the assignment that achieves this minimum.

Bucket Elimination (BE (Dechter 1999)) solves the min-sum problem by eliminating variables, one at a time, in sequence. It works on a structure called *pseudo-tree* T also known as a bucket-tree. The complexity of BE is time and space exponential in *induced (tree) width* w^* of the underlying primal graph of the problem (Dechter 1999). For more information and on how BE works, see the reference.

Mini-Bucket Elimination $MBE(i)$ is an approximation of BE which solves a relaxation of the problem. The relaxed problem is created by duplicating certain variables so that the functions generated during BE will be bounded by a parameter i, or i-bound. The relaxed problem has a treewidth of i. BE differs from MBE in that, when processing the functions defined on each variable (i.e., its bucket) the mini-bucket partitions the bucket and processes each partition separately. The mini-bucket generates bounds (lower-bounds for minimization) used as heuristics during search.

AND/OR search The search space of a graphical model can be guided by the decomposition that is suggested by the pseudo-tree, yielding an AND/OR search space.

In the rest of the paper we will use the following notation.

- B_Y is the bucket of variable Y.
- ψ_Y - sum of all the original functions in B_Y. ψ_Y^r - sum of those in the r^{th} mini-bucket MB_Y^r of B_Y.
- $\bar{X}_{1..p}$ - A path of variables from X_1 to X_p. $\bar{x}_{1..p}$ - An assignment of values to the variables in $\bar{X}_{1..p}$.

MBE Look-ahead for Search

Since the AND/OR search space is guided by the pseudo-tree many of the concepts of look-ahead can be viewed relative to the pseudo-tree as well. A d-level look-ahead computation can be organized over variables of the pseudo-tree T that are within distance d of the current variable X_p.

DEFINITION 1 (**d-level subtree** $T_{(X_p,d)}$) *Given a pseudo-tree T and an integer d, $T_{(X_p,d)}$ is the set of variables in the d-level* subtree *of T rooted at X_p;*

DEFINITION 2 **(Look-ahead MBE-heuristic in AND/OR)** *Given an AND/OR search space guided by a pseudo-tree T and a corresponding MBE heuristic function h^{mbe}, the*

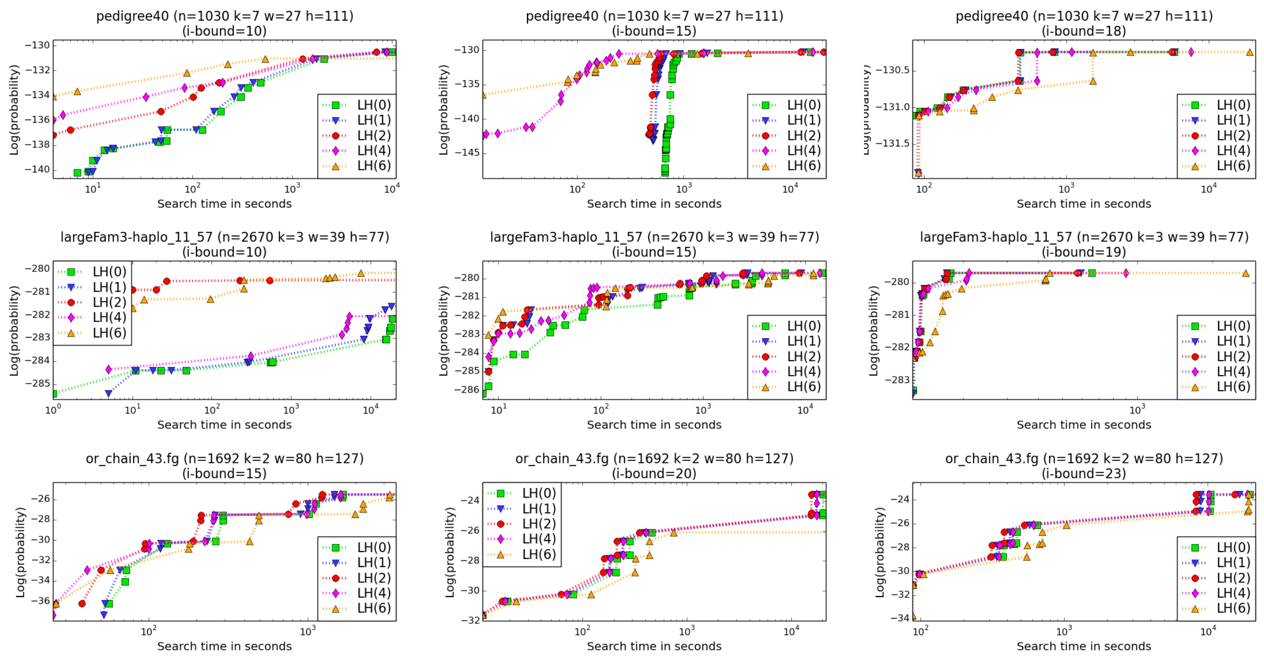

Figure 1: Selected instances. The solution quality is plotted as a function of time per instance (in log scale).

d-level look-ahead heuristic of h^{mbe} at X_p is a function, whose scope is subsumed in $\bar{X}_{1..p}$, denoted $h^{lh(d)}_{X_p}(s_p)$, $s_p = \bar{x}_{1..p}$

- $h^{lh(0)}_{X_p}(s_p) = h^{mbe}_{X_p}(s_p)$
- $h^{lh(d)}_{X_p}(s_p) = \sum_{Y \in child(X_p)} \min_Y(\psi_Y(s_p, y) + h^{lh(d-1)}_Y(s_p, y))$

The d-level residual at X_p is

- $res^{lh(d)}_{X_p}(s_p) = h^{lh(d)}_{X_p}(s_p) - h^{mbe}_{X_p}(s_p)$

Experiments

We ran AND/OR Branch and Bound (AOBB) (Marinescu and Dechter 2009) using the state-of-the-art MBE-MM heuristic (Ihler et al. 2012) on 3 different benchmarks with a time bound of 6 hours for each instance. To evaluate the general impact of look-ahead across a range of depths d, we collected the solutions reported by our solver at various time snapshots and reported those in instance by instance anytime graphs.

Figure 1 plots the time vs. solution quality for an instance from each benchmark. Points that are higher and to the left indicate superior anytime performance. If a line runs off the right end of the plot, it indicates that the problem was not solved within the time bound. Each row of the figure corresponds to an particular problem instance and the columns correspond to different i-bounds. In this small sample of our results, we see indeed that the look-ahead scheme has a positive impact on anytime performance at earlier times, especially when the i-bound is lower. However, as the search time and/or i-bound increase, shallower look-ahead depths can become more preferable. In results on exact search (Dechter, Kask, and Lam 2015), a look-ahead depth of 2-3 often performed the best even for the same set of lower i-bounds, with deeper look-ahead yielding decreased performance due to time overhead. Here, we see that deeper look-ahead can give superior early anytime performance.

Conclusion

We performed a preliminary evaluation of the effect of look-ahead on the anytime performance of solving MPE. Our results show that look-ahead can be a cost effective method of achieving higher quality solutions sooner, especially when the heuristic is weak. The search time is also a factor, suggesting that it may also be beneficial to dynamically change the look-ahead depth over time. Future work in estimating the residual and its relative effect during search can guide us towards more flexible look-ahead subtrees for this purpose.

References

Dechter, R.; Kask, K.; and Lam, W. 2015. Some properties of look-ahead with mini-bucket heuristics. Technical report, Unpublished, UC Irvine, ICS.

Dechter, R. 1999. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence* 113:41–85.

Ihler, A.; Flerova, N.; Dechter, R.; and Otten, L. 2012. Join-graph based cost-shifting schemes. In *Uncertainty in Artificial Intelligence (UAI)*. Corvallis, Oregon: AUAI Press. 397–406.

Marinescu, R., and Dechter, R. 2009. And/or branch-and-bound search for combinatorial optimization in graphical models. *Artif. Intell.* 173(16-17):1457–1491.