"daoopt" – UAI'10 Inference Competition

Team Members: Lars Otten, Rina Dechter*
Additional Contributor: Radu Marinescu†

August 6, 2010

Abstract

This short paper describes the *daoopt* entries to the MPE track of the UAI 2010 Inference competition, where they finished in third place for the three time limits of 20 seconds, 20 minutes, and 1 hour.[1]

1 Summary

daoopt is an exact solver for discrete combinatorial optimization problems, like the task of finding a most probable explanation (MPE) over Bayesian or Markov networks. At its core it is an implementation of a Branch and Bound scheme over the AND/OR search space defined by the given graphical model. It uses a Mini-Bucket heuristic for pruning the search space, as well as Limited Discrepancy Search for finding an initial solution quickly. The solver is written in C++.

2 Main Components

The following are the main components that have been implemented as part of *daoopt*:

- *AND/OR Branch and Bound:* AND/OR search is a general search paradigm for graphical model problems. Special AND nodes are introduced into the search space to capture independence of subproblems, while context-based caching can recognize and merge some unifiable subproblems [1]. It has been shown that the complexity of algorithms that explore this AND/OR context-minimal search graph is time and space exponential in the induced width / treewidth of the problem's underlying graph structure.

  In the context of optimization problems over graphical models, the concept of AND/OR Branch and Bound (AOBB) has been developed and implemented by Marinescu and Dechter [5]. For *daoopt* we reimplemented the algorithm from scratch, with a focus on efficiency.

*UC Irvine
†University College Cork
[1] http://www.cs.huji.ac.il/project/UAI10/

- *Mini Bucket heuristics:* For the pruning step of the branch and bound scheme, *daoopt* implements a Mini-Bucket heuristic. It is based on Mini-Bucket elimination, which is an approximate variant of a variable elimination scheme and computes approximations to reasoning problems over graphical models [2].

  It was shown that the intermediate functions generated by the Mini-Bucket algorithm can be used to derive an admissible heuristic function that (for a maximization problem) always overestimates the optimal cost solution to a subproblem in the search space. Its value for pruning was demonstrated in depth-first and best-first search [4, 5].

  The Mini Bucket algorithm MBE(i) has a control parameter called the i-bound, which controls the time and space complexity of the algorithm. Larger values of i typically lead to better approximations and thus stronger heuristics.

- *Limited Discrepancy Search:* To find and output an initial solution quickly, we implemented a version of Limited Discrepancy Search [3]. LDS in principle explores the same AND/OR context-minimal search graph as AOBB. However, it allows only up to a limited number of "discrepancies" along any search path, where a discrepancy is any branching decision that deviates from what is optimal according to the Mini-Bucket heuristic. Clearly, depending on this limit only a part of the search space is explored, rendering LDS an approximate algorithm.

  We also considered a stochastic local search scheme for initialization, but generally found LDS to be more reliable with respect to the quality of the solutions. Furthermore, for a fixed discrepancy limit the running time of LDS scales with the hardness of the problem, while employing local search adds the additional complexity of determining a good time limit on a per-problem basis.

3 Details

Our implementation uses a minfill heuristic (with random tie breaking) to find a variable ordering for the search process; we select the ordering with lowest induced width over 100 iterations. The i-bound for the Mini-Bucket heuristic is chosen as large as possible, subject to memory constraints: the execution of Mini-Buckets is simulated for decreasing values of i until the total space requirement for the intermediate tables is below the given memory limit. The discrepancy limit for LDS was set to 2.

3.1 Initial Solution

Not outputting any solution within the alloted time was penalized heavily in this competition. Hence, in order to produce at least one solution quickly, we added a preprocessing routine as follows: we compile a Mini-Bucket heuristic

with significantly lower i-bound, allowing quick computation, and then run LDS with discrepancy limit 1. That way we can typically obtain a first solution (which, even though often far from optimal, is a lot better than nothing) within the first 20 seconds. Only then do we compute the full Mini-Bucket heuristic, run LDS with limit 2, and eventually start the complete AOBB solver.

3.2 Anytime Properties

Branch and Bound is in principle an anytime scheme as it outputs improving solutions as the search progresses. However, subproblem decomposition based on the AND/OR framework together with the depth-first nature of the search can compromise this property. Hence, during the month of the evaluation, we implemented an experimental scheme that tries to overcome this potential issue by "rotating" the depth-first expansion over the parts of the search space corresponding to independent subproblems, yielding our entry *daoopt.anytime*. However, the results indicate that the overhead from this additional logic outweighs potential gains in anytime performance; a more in-depth investigation is still pending.

4 Comparison & Conclusion

As an exact Branch and Bound solver, *daoopt* is related to *ToulBar2* that came 2nd, 1st, and 2nd for the three time limits of the MPE track. One of the central differences between the two solvers is the heuristic used for pruning in Branch and Bound: *daoopt* uses Mini-Buckets, while *ToulBar2* uses soft local consistency enforcing. We believe the latter is better suited for many of the problems with large-variable domains in the competition, which probably gave *ToulBar2* the edge over *daoopt*. Another factor may be that *ToulBar2* with local consistency can work with dynamic variable orderings while the pre-compiled Mini-Bucket heuristic requires a static variable ordering.

Overall, however, we were pleasantly surprised to see that these exact algorithms outperformed all but one of the of the dedicated approximate solvers. We believe this speaks for the power and versatility of these general schemes. Finally, it is worth noting that these exact solvers can often provide proofs of optimality, which was not considered in this evaluation.

References

[1] Rina Dechter and Robert Mateescu. AND/OR search spaces for graphical models. *Artif. Intell.*, 171(2-3):73–106, 2007.

[2] Rina Dechter and Irina Rish. Mini-buckets: A general scheme for bounded inference. *Journal of the ACM*, 50(2):107–153, 2003.

[3] William D. Harvey and Matthew L. Ginsberg. Limited discrepancy search. In *IJCAI*, pages 607–615, 1995.

[4] Kalev Kask and Rina Dechter. A general scheme for automatic generation of search heuristics from specification dependencies. *Artif. Intell.*, 129(1-2):91–131, 2001.

[5] Radu Marinescu and Rina Dechter. AND/OR branch-and-bound search for combinatorial optimization in graphical models. *Artif. Intell.*, 173(16-17):1457–1491, 2009.