Real-time Heuristic Search for Combinatorial Optimization and Constraint Satisfaction

Wheeler Ruml
Division of Engineering and Applied Sciences
Harvard University
33 Oxford Street, Cambridge, MA 02138 USA
ruml@eecs.harvard.edu

Abstract

We summarize a new approach to real-time heuristic tree search for problems with a fixed goal depth, such as combinatorial optimization and constraint satisfaction problems. Best-leaf-first search (BLFS) is a general and complete algorithm that visits leaves in an order that efficiently approximates increasing predicted cost. It can adapt its search order on-line to the current problem. Empirical results on a variety of challenging synthetic benchmarks suggest that BLFS yields competitive or superior performance and is more robust than previous methods.

Motivation

Many applications of artificial intelligence technologies impose strict real-time requirements. Examples include intelligent user interfaces, natural language generation, computer-aided planning and scheduling, decision support systems, model-based control, and medical diagnosis. Unfortunately, many of the AI problems underlying these applications are fundamentally combinatorial in nature and are NP-hard. While real-time heuristic search for shortest-path problems has been addressed extensively in the literature (Korf 1990), less work has been done on tree search problems with a fixed maximum goal depth, such as combinatorial optimization and constraint satisfaction. These problems are typically formulated as trees in which one selects a variable at each node and branches on its possible values. The goal is to find the leaf with the lowest cost or with the fewest violated constraints. Because these trees grow exponentially with problem size, complete enumeration of the leaves is often infeasible even without 'real-time' constraints. While iterative improvement search methods can sometimes be used to provide acceptable solutions, many domains benefit from representation as tree search problems. For instance, in a constraint satisfaction problem, propagation techniques can be used to derive values for many problem variables given assignments to only a small fraction of them.

When exhaustive enumeration is infeasible, a rational strategy visits leaves in increasing order of predicted cost. Previous systematic algorithms for this setting follow a predetermined search order, thereby making strong implicit assumptions about predicted cost and using problem-specific information inefficiently. We introduce a search framework, *best-leaf-first search* (BLFS), in which assumptions are explicitly represented in the form of a predictive model of leaf costs. Because the search uses an explicit model, it can even depend on model parameters which are estimated on-line from the search tree itself, rather than assumed beforehand. The central idea of BLFS is to visit leaves in an order that approximates increasing predicted cost. This is achieved by visiting all leaves whose predicted cost falls within a fixed bound, and then iteratively raising the bound. BLFS is analogous to the iterative-deepening A* (IDA*) algorithm for shortest-path problems (Korf 1985). Their common framework of single-agent rationality provides a clean unification of search for combinatorial optimization and constraint satisfaction with the tradition of heuristic search in AI for shortest-path problems. Further details of this work are given by Ruml (2002).

BLFS(*root*)
1. Visit a few leaves
2. *Nodes-desired* ← number of nodes visited so far
3. Loop until time runs out:
4. Double *nodes-desired*
5. Estimate cost bound that visits *nodes-desired* nodes
6. Call BLFS-expand(*root*, *bound*)

BLFS-expand(*node*, *bound*)
7. If leaf(*node*), visit(*node*)
8. else, for each *child* of *node*:
9. If best-completion(*child*) $\leq$ *bound*
10. BLFS-expand(*child*, *bound*)

Figure 1: Simplified pseudo-code for best-leaf-first search.

Best-Leaf-First Search

The basic structure of BLFS is a simple loop in which we carry out successive depth-first searches. Pseudo-code is shown in Figure 1. Each search visits all leaves whose costs are predicted to fall within a cost bound. In these respects, BLFS is similar to IDA*. IDA* controls expansion of a node n using a prediction of the cost of the best path to the nearest goal that goes through n. Analogously, BLFS uses the predicted cost of the best leaf below n. This allows the algorithm to avoid descending into any subtree that does not

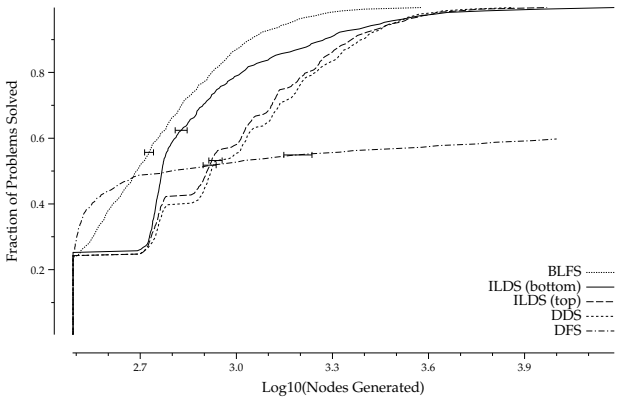

Figure 2: Distribution of search times for latin squares.

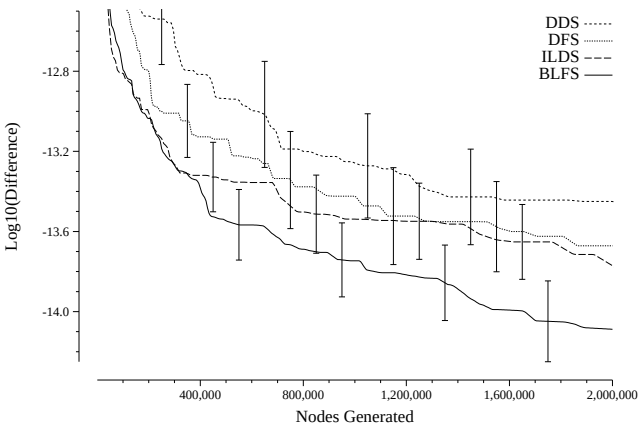

Figure 3: Anytime profiles for partitioning 256 numbers

contain a leaf we wish to visit on this iteration (step 9). However, BLFS uses an explicit representation of its predictive model, rather than a black-box function suppled by the user, as in IDA*. Being able to choose a simple model leads to two important advantages over IDA*. First, we can choose a model that will give *consistent* predictions. That is to say, we can ensure that for every node there exists a child whose best descendant will have the same evaluation. This means that BLFS is certain to reach leaves on every iteration, never expanding a node unnecessarily and never overlooking a node that has a descendant within the cost bound. IDA* does not generate any leaves until it has generated all internal nodes whose cost is less than that of the optimal leaf, and is therefore useless for real-time applications. The second advantage that BLFS enjoys over IDA* is that the cost bound can be updated optimally. Because the predicted costs are generated by a known model, we can choose cost bounds that can be expected to cause twice as many nodes to be visited as on the previous iteration (step 5). By approximately doubling the number of nodes visited on each iteration, BLFS limits its overhead to a factor of less than three in the worst-case situation in which the entire tree must be searched.

Selected Results

A latin square is an n by n array in which each cell has one of n colors. Each row and column must contain each color exactly once. We preassigned 30% of the cells. We tested depth-first search (DFS), two version of Korf's improved limited discrepancy search (ILDS, Korf, 1996), one taking discrepancies at the top first and the other taking them at the bottom first, depth-bounded discrepancy search (DDS, Walsh, 1997), and BLFS. The number of variables left unassigned when the first constraint violation was detected was used as the cost of each leaf. Space limitations prevent description of the leaf cost model.

The performance of the algorithms is shown in Figure 2 in terms of the fraction of problems solved within a given number of node generations. From the figure, we see that 25% of the problems were solved by visiting a single leaf (the greedy solution). But depth-first search, the most common tree search technique, is amazingly brittle and becomes hopeless lost on many problems. The discrepancy search algorithms immediately retreat to the root. BLFS first explores all ties, which may occur at intermediate levels of the tree, and it solves all the problems within 4,000 nodes (note the logarithmic scale). Its relatively compact run-time distribution is superior for real-time applications. The advantage of BLFS search over the discrepancy methods seemed to increase as problems grew larger (not shown here).

In number partitioning, one attempts to divide a given set of numbers into two disjoint groups such that the difference between the sums of the two groups is as small as possible (Korf 1995). The logarithm of this difference was used as the leaf cost. A predictive model of leaf cost was learned during the search. Figure 3 compares the anytime performance profile of BLFS with those of DFS, ILDS, and DDS. The vertical axis represents the partition difference of the best solution found so far, which we are attempting to minimize. Error bars represent 95% confidence intervals on the mean. BLFS surpasses the other algorithms throughout, and its advantage increases with problem size (not shown here).

Acknowledgments

Thanks to Stuart Shieber, the Harvard AI Research Group, and NSF grants CDA-94-01024 and IRI-9618848.

References

Korf, R. E. 1985. Depth-first iterative-deepening: An optimal admissible tree search. *AIJ* 27(1):97–109.

Korf, R. E. 1990. Real-time heuristic search. *AIJ* 42:189–211.

Korf, R. E. 1995. From approximate to optimal solutions: A case study of number partitioning. In *Proc IJCAI-95*.

Korf, R. E. 1996. Improved limited discrepancy search. In *Proceedings of AAAI-96*, 286–291.

Ruml, W. 2002. Heuristic search in bounded-depth trees: Best-leaf-first search. TR-01-02, Harvard University.

Walsh, T. 1997. Depth-bounded discrepancy search. In *Proceedings of IJCAI-97*.