To appear in the Proceedings of IJCAI-01.

Incomplete Tree Search using Adaptive Probing

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

Abstract

When not enough time is available to fully explore a search tree, different algorithms will visit different leaves. Depth-first search and depth-bounded discrepancy search, for example, make opposite assumptions about the distribution of good leaves. Unfortunately, it is rarely clear *a priori* which algorithm will be most appropriate for a particular problem. Rather than fixing strong assumptions in advance, we propose an approach in which an algorithm attempts to adjust to the distribution of leaf costs in the tree while exploring it. By sacrificing completeness, such flexible algorithms can exploit information gathered during the search using only weak assumptions. As an example, we show how a simple depth-based additive cost model of the tree can be learned on-line. Empirical analysis using a generic tree search problem shows that adaptive probing is competitive with systematic algorithms on a variety of hard trees and outperforms them when the node-ordering heuristic makes many mistakes. Results on boolean satisfiability and two different representations of number partitioning confirm these observations. Adaptive probing combines the flexibility and robustness of local search with the ability to take advantage of constructive heuristics.

1 Introduction

Consider the problem of searching a finite-depth tree to find the best leaf. Many search trees arising in practical applications are too large to be explored completely. Given a limited amount of time, one can only hope to search the tree in such a way that leaves with a greater chance of being optimal are encountered sooner. For instance, when a node-ordering heuristic is available, a depth-first search can expand the children of a node in the order in which they are preferred by the heuristic. However, the backtracking order of depth-first search will visit the second-ranked child of the last internal branching node before reconsidering the choice at the next to last branching node. Each decision at which a non-preferred child is chosen is called a discrepancy [Harvey and Ginsberg, 1995]. Depth-first search will visit the leaf whose path from the root has all discrepancies below depth i before visiting the leaf with a single discrepancy at depth i. This corresponds to an implicit assumption that a single discrepancy at depth i will lead to a worse leaf than taking discrepancies at every deeper depth.

Limited discrepancy search [Harvey and Ginsberg, 1995; Korf, 1996] was designed with a different assumption in mind. It assumes that discrepancies at any depth are equally disadvantageous and so visits all leaves with k discrepancies anywhere in their paths before visiting any leaf with $k + 1$ discrepancies. Depth-bounded discrepancy search [Walsh, 1997] uses a still different assumption: a single discrepancy at depth i is worse than taking discrepancies at all depths shallower than i. Motivated by the idea that node-ordering heuristics are typically more accurate in the later stages of problem-solving, when local information better reflects the remaining subproblem, this assumption is directly opposed to the one embodied by depth-first search.

When faced with a new search problem, it is often not obvious which algorithm's assumptions most accurately reflect the distribution of leaf costs in the tree or even if any of them are particularly appropriate. In this paper, we investigate an adaptive approach to tree search in which we use the costs of the leaves we have seen to estimate the cost of a discrepancy at each level. Simultaneously, we use these estimates to guide search in the tree. Starting with no preconceptions about the relative advantage of choosing a preferred or non-preferred child node, we randomly probe from the root to a leaf. By sharpening our estimates based on the leaf costs we observe and choosing children with the probability that they lead to solutions with lower cost, we focus the probing on areas of the tree that seem to contain good leaves.

This stochastic approach is incomplete and cannot be used to prove the absence of a goal leaf. In addition, it generates the full path from the root to every leaf it visits, incurring overhead proportional to the depth of the tree when compared to depth-first search, which generates roughly one internal node per leaf. However, the problem-specific search order of adaptive probing has the potential to lead to better leaves much faster. Since an inappropriate search order can trap a systematic algorithm into exploring vast numbers of poor leaves, adaptive probing would be useful even if it only avoided such pathological performance on a significant fraction of problems.

After describing the details of an algorithm based on this adaptive approach, we investigate the algorithm's performance using the abstract tree model of Harvey and Ginsberg [1995]. We find that adaptive probing outperforms systematic methods on large trees when the node-ordering heuristic is moderately inaccurate, and exhibits better worst-case performance whenever the heuristic is not perfect at the bottom of the tree. To confirm these observations, we also test the algorithm on two different representations of the combinatorial optimization problem of number partitioning and on the goal-search problem of boolean satisfiability. It performs well on satisfiability and the naive formulation of number partitioning, but is competitive only for long run-times when using the powerful Karmarkar-Karp heuristic.

2 An Adaptive Probing Algorithm

Within the general approach outlined above, there are many ways to extract information from observed leaf costs. In this paper, we will evaluate one simple model. We will assume that the cost of every leaf is the sum of the costs of the actions taken to reach it from the root. Each position in the ordered list of children counts as a distinct action and actions at different levels of the tree are modeled separately. So a tree of depth d and branching factor b requires db parameters, one for each action at each level. The model assumes, for instance, that the effects of choosing the second-most-preferred child at level 23 is the same for all nodes at level 23. This is just a generalization of the assumption used by discrepancy search algorithms. In addition, we will estimate the variance of the action costs by assuming that each estimated cost is the mean of a normal distribution, with all actions having the same variance.

This model is easy to learn during the search. Each probe from the root corresponds to a sequence of actions and results in an observed leaf cost. If $a_j(i)$ is the cost of taking action i at depth j and l_k is the cost of the kth leaf seen, probing three times in a binary tree of depth three might give the following information:

$$
\begin{array}{llllllll}
a_0(0) & & + & a_1(0) & & + & & a_2(1) & = l_0 \\
a_0(0) & & + & & a_1(1)+ & a_2(0) & & & = l_1 \\
& a_0(1)+ & & a_1(0) & & + & a_2(0) & & = l_2
\end{array}
$$

We can then estimate the $a_j(i)$ using a least squares regression algorithm. In the experiments reported below, a perceptron was used to estimate the parameters [Cesa-Bianchi *et al.*, 1996]. This simple gradient descent method updates each cost according to the error between a prediction of the total leaf cost using the current action estimates, $\hat{l}_k$, and the actual leaf cost, l_k. If d actions were taken, we update each of their estimates by

$$\eta \frac{(l_k - \hat{l}_k)}{d}$$

where η controls the learning rate (or gradient step-size). All results reported below use $\eta = 0.2$, although similar values also worked well. (Values of 1 and 0.01 resulted in reduced performance.) This update requires little additional memory, takes only linear time, adjusts d parameters with every leaf, and often performed as well as an impractical $O(d^3)$ singular value decomposition estimator. It should also be able to track changes in costs as the probing becomes more focussed, if necessary.

Because we assume that it is equal for all actions, the variance is straightforward to estimate. If we assume that the costs of actions at one level are independent from those at another, then the variance we observe in the leaf costs must be the sum of the variances of the costs selected at each level. The only complication is that the variance contributed by each level is influenced by the mean costs of the actions at that level—if the costs are very different, then we will see variance even if each action has none. More formally, if X and Y are independent and normally distributed with common variance σ^2_{XY}, and if W takes its value according to X with probability p and Y with probability $1-p$, then

$$
\begin{aligned}
\sigma^2_W &= E(W^2) - \mu^2_W \\
&= p(\mu^2_X + \sigma^2_{XY}) + (1-p)(\mu^2_Y + \sigma^2_{XY}) - \\
&\quad (p\mu_X + (1-p)\mu_Y)^2 \\
&= \sigma^2_{XY} + p\mu^2_X + (1-p)\mu^2_Y - (p\mu_X + (1-p)\ \mu_Y)^2
\end{aligned}
$$

Since we can easily compute p by recording the number of times each action at a particular level is taken, and since the action costs are estimates of the μ_i, we can use this formula to subtract away the effects of the different means. Following our assumption, we can then divide the remaining observed variance by d to distribute it equally among all levels.

Using the model during tree probing is also straightforward. If we are trying to minimize the leaf cost, then for each decision, we want to select the action with the lower expected cost (i.e., the lower mean). As our estimates may be quite inaccurate if they are based on few samples, we don't always want to select the node with the lower estimated cost. Rather, we merely wish to select each action with the probability that it is truly best. Given that we have estimates of the means and variance of the action costs and we know how many times we have tried each action, we can compute the probability that one mean is lower than another using a standard test for the difference of two sample means. We then choose each action according to the probability that its mean cost is lower. To eliminate any chance of the algorithm converging to a single path, the probability of choosing any action is clamped at $0.05^{1/d}$ for a depth d tree, which ensures at least one deviation on 95% of probes.

Now we have a complete adaptive tree probing algorithm. It assumes the search tree was drawn from a simple model of additive discrepancy costs and it learns the parameters of the tree efficiently on-line. Exploitation of this information is balanced with exploration according to the variance in the costs and the number of times each action has been tried. The method extends to trees with large and non-uniform branching factors and depths. The underlying model should be able to express assumptions similar to those built into algorithms as diverse as depth-first search and depth-bounded discrepancy search, as well as many other weightings not captured by current systematic methods.

3 Empirical Evaluation

We first investigate the performance of this adaptive probing algorithm using an abstract model of heuristic search. This gives us precise control over the density of good nodes and the accuracy of the heuristic. To ensure that our conclusions apply to more complex domains, we will also evaluate the algorithm using two NP-complete search problems: the combinatorial optimization problem of number partitioning and the goal-search problem of boolean satisfiability.

3.1 An Abstract Tree Model

In this model, introduced by Harvey and Ginsberg [1995] for the analysis of limited discrepancy search, one searches for goal nodes in a binary tree of uniform depth. Goals are distributed according to two parameters: m, which controls goal density, and p, which controls the accuracy of the heuristic. Each node either has a goal below it, in which case it is *good*, or does not, in which case it is *bad*. Clearly, the root is good and bad nodes only have bad children. The probabilities of the other configurations of parent and children are:

P(good $\rightarrow$ good good) = $1 - 2m$
P(good $\rightarrow$ bad good) = $1 - p$
P(good $\rightarrow$ good bad) = $2m - (1 - p)$

The expected number of goal nodes is $(2 - 2m)^d$, where d is the depth of the tree.

Following Walsh's [1997] analysis of depth-bounded discrepancy search, we will estimate the number of leaves that each algorithm must examine before finding a goal using empirical measurements over lazily (but deterministically) generated random trees. To provide a leaf cost measure for adaptive probing, we continue the analogy with constraint satisfaction problems that motivated the model and define the leaf cost to be the number of bad nodes in the path from the root. (If we were able to detect failures before reaching a leaf, this would be the depth remaining below the prune.) The results presented below are for trees of depth 100 in which $m = 0.1$. The probability that a random leaf is a goal is 0.000027. By investigating different values of p, we can shift the locations of these goals relative to the paths preferred by the heuristic.

Figure 1 shows the performance of depth-first search (DFS), Korf's [1996] improved version of limited discrepancy search (ILDS), depth-bounded discrepancy search (DDS), and adaptive probing on 2,000 trees. A heuristic-biased probing algorithm is also shown. This algorithm selects the preferred child with the largest probability that would be allowed during adaptive probing. Following Walsh, we raise the accuracy of the heuristic as depth increases. At the root, $p = 0.9$ which makes the heuristic random, while at the leaves $p = 0.95$ for 75% accuracy. ILDS was modified to incorporate this knowledge and take its discrepancies at the top of the tree first.

Adaptive probing quickly learns to search these trees, performing much better than the other algorithms. Even though DDS was designed for this kind of tree, its assumptions are too strong and it always branches at the very top of the tree. ILDS wastes time by branching equally often at the bottom where the heuristic is more accurate. The *ad hoc* biased probing algorithm, which branches at all levels, is competitive

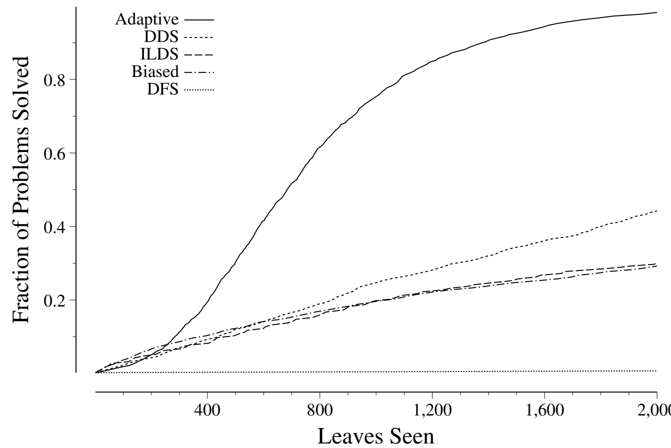

Figure 1: Probability of finding a goal in trees of depth 100 with m = 0.1 and p linearly varying between 0.9 at the root and 0.95 at the leaves.

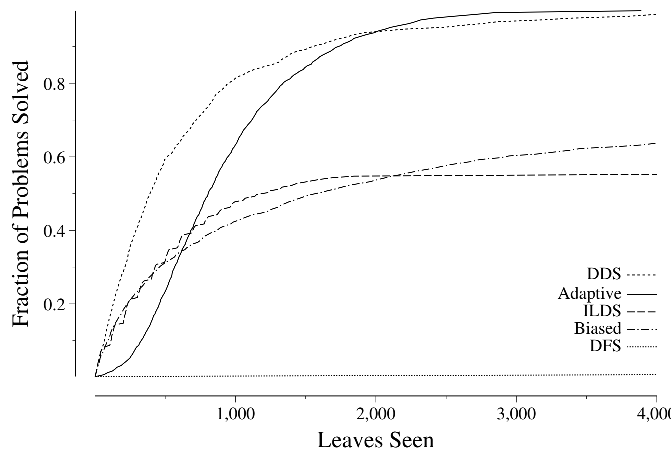

Figure 2: Performance on trees of depth 100, m = 0.1, and p varying from 0.9 at the root to 0.98 at the leaves.

with ILDS (and will actually surpass it, given more time) but fails to exploit the structure in the search space. DFS vainly branches at the bottom of the tree, ignorant of the fatal mistake higher in the tree, and solves almost no problems within 2,000 leaves.

DDS does better when the heuristic is more accurate, since its steadfast devotion to the preferred child in the middle and bottom of the tree is more often correct. Figure 2 shows the algorithms' performance on similar trees in which the heuristic is accurate 90% of the time at the leaves. DDS has better median performance, although adaptive probing exhibits more robust behavior, solving all 2,000 problems within 4,000 leaves. DDS had not solved 1.4% of these problems after 4,000 leaves and did not complete the last one until it had visited almost 15,000 leaves. In this sense, DDS has a heavier tail in its cost distribution than adaptive probing. Similar results were obtained in trees with uniform high p. Adaptive probing avoids entrapment in poor parts of the tree at the expense of an initial adjustment period.

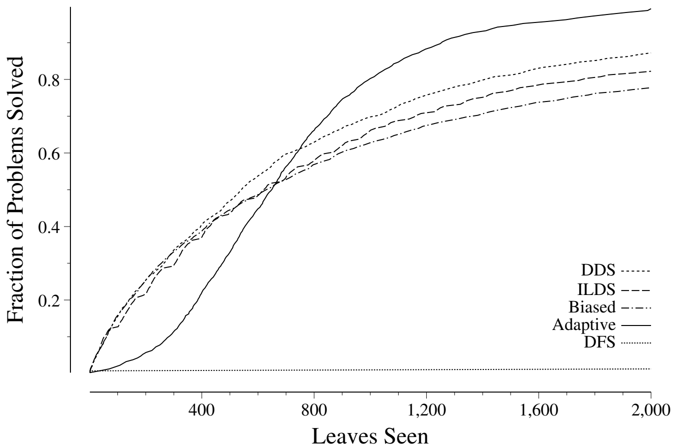

Figure 3: Performance on trees of depth 100, m = 0.1, and p varying from 0.98 at the root to 0.9 at the leaves.

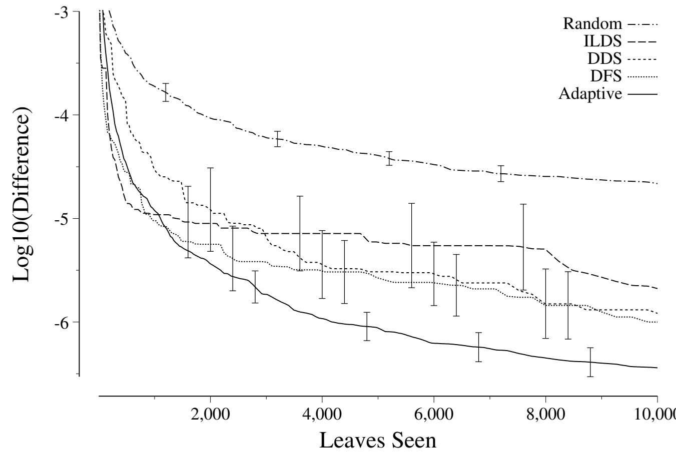

Figure 4: Searching the greedy representation of number partitioning. Error bars indicate 95% confidence intervals around the mean over 20 instances, each with 128 44-digit numbers.

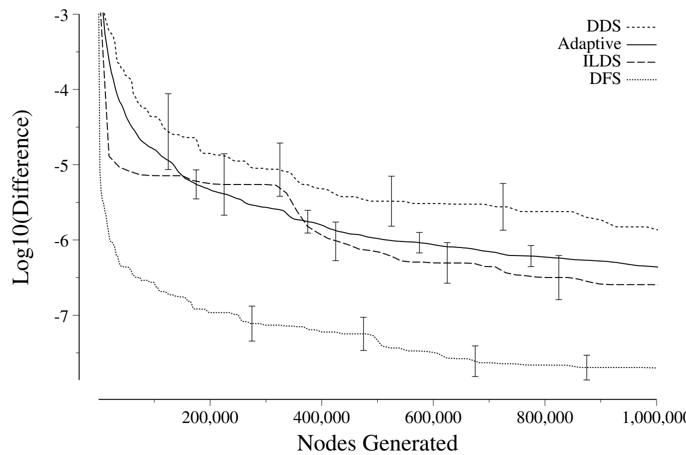

Figure 5: Performance on the greedy representation of number partitioning as a function of nodes generated.

Even with an accurate heuristic, however, the assumptions of DDS can be violated. Figure 3 shows what happens in trees in which the heuristic is accurate at the top of the tree and random at the very bottom. DDS still has an advantage over ILDS because a single bad choice can doom an entire subtree, but adaptive probing learns a more appropriate strategy.

To ensure that our insights from experiments with the abstract tree model carry over to other problems, we also evaluated the algorithms on three additional kinds of search trees.

3.2 Number Partitioning

The objective in a number partitioning problem is 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. It was used by Johnson et al. to evaluate simulated annealing [1991], Korf to evaluate his improvement to limited discrepancy search [1996], and Walsh to evaluate depth-bounded discrepancy search [1997]. To encourage difficult search trees by reducing the chance of encountering a perfectly even partitioning [Karmarkar *et al.*, 1986], we used instances with 64 25-digit numbers or 128 44-digit numbers.[1] (Common Lisp, which provides arbitrary precision integer arithmetic, was used to implement the algorithms.) Results were normalized as if the original numbers had been between 0 and 1. To better approximate a normal distribution, the logarithm of the partition difference was used as the leaf cost.

The Greedy Representation

We present results using two different representations of the problem. The first is a straightforward greedy encoding in which the numbers are sorted in descending order and then each decision places the largest remaining number in a partition, preferring the partition with the currently smaller sum. Figure 4 compares the performance of adaptive tree probing with depth-first search (DFS), improved limited discrepancy search (ILDS), depth-bounded discrepancy search (DDS), and completely random tree probing. To provide a comparison of the algorithms' search orders, the horizontal axis represents the number of leaves seen. Adaptive probing starts off poorly, like random sampling, but surpasses all other algorithms after seeing about 1,000 leaves. It successfully learns an informative model of the tree and explores the leaves in a more productive order than the systematic algorithms.

However, recall that adaptive tree probing suffers the maximum possible overhead per leaf, as it generates each probe from the root. (This implementation did not attempt to reuse initial nodes from the previous probe.) The number of nodes (both internal and leaves) generated by each algorithm should correlate well with running time in problems in which the leaf cost is computed incrementally or in which the node-ordering heuristic is expensive. Figure 5 compares the algorithms on the basis of generated search nodes. (To clarify the plot, DFS and ILDS were permitted to visit many more leaves than the other algorithms.) In a demonstration of the importance of overhead, DFS dominates all the other algorithms in

[1] These sizes also fall near the hardness peak for number partitioning [Gent and Walsh, 1996], which specifies $log_{10}2^n$ digits for a problem with n numbers.

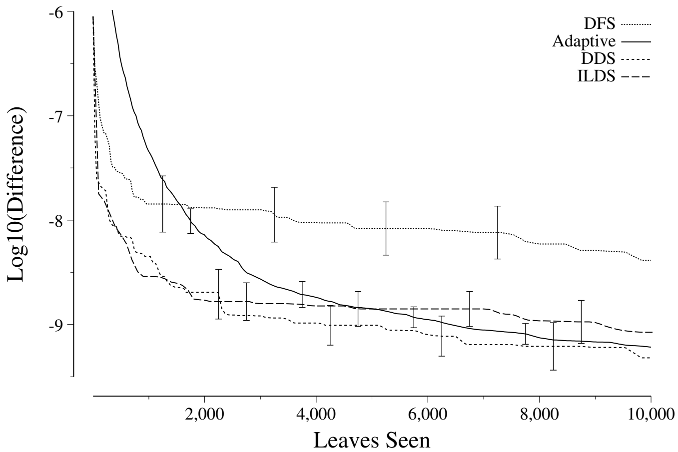

Figure 6: Searching the CKK representation of number partitioning. Each instance had 64 25-digit numbers.

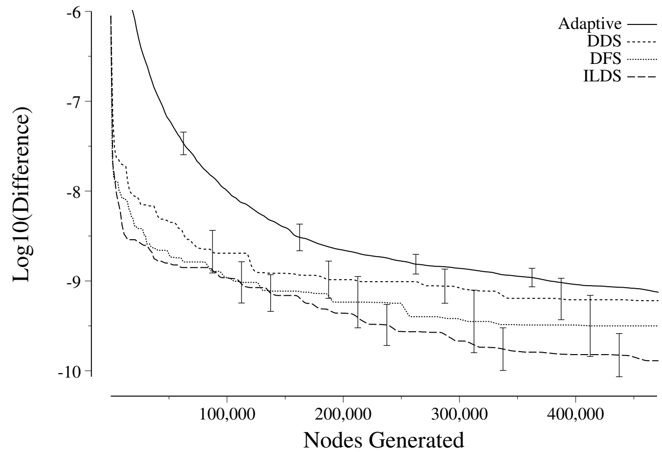

Figure 7: Performance on the CKK representation of number partitioning as a function of nodes generated.

this view, and ILDS performs comparably to adaptive probing. DFS reuses almost all of the internal nodes on each leaf's path, generating only those just above the leaves. Since ILDS needs to explore discrepancies at every level of the tree, it will usually need to generate a significant fraction of the path down to each leaf. DDS, which limits its discrepancies to the upper levels of the tree, incurs overhead similar to that of adaptive probing because it never reuses internal nodes in the middle of the tree.

On instances using 64 numbers, adaptive probing again dominated DDS, but was clearly surpassed by ILDS. (It performed on par with a version of ILDS that visited discrepancies at the top of the tree before those at the bottom.) This suggests that, in these search trees, the advantage of adaptive probing over ILDS and DDS increases with problem size.

The CKK Representation

A more sophisticated representation for number partitioning was suggested by Korf [1995], based on the heuristic of Karmarkar and Karp [1982]. The essential idea is to postpone the assignment of numbers to particular partitions and merely constrain pairs of number to lie in either different bins or the same bin. Numbers are considered in decreasing order and constrained sets are reinserted in the list according to the remaining difference they represent. This representation creates a very different search space from the greedy heuristic.

Figure 6 shows the performance of the algorithms as a function of leaves seen. DDS has a slight advantage over ILDS, although adaptive probing is eventually able to learn an equally effective search order. DFS and random sampling too often go against the powerful heuristic. As in the greedy representation, however, interior node overhead is an important consideration. Figure 7 shows that DDS and adaptive probing are not able to make up their overhead, and results using 128 numbers suggest that these difficulties increase on larger problems. Bedrax-Weiss [1999] argues that the KK heuristic is extraordinarily effective at capturing relevant information and that little structure remains in the space. These results are consistent with that conclusion, as the uniform and limited discrepancies of ILDS appear best.

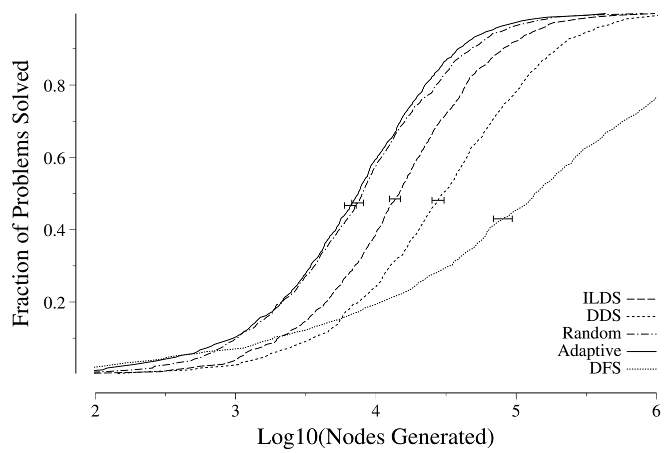

Figure 8: Fraction of random 3-satisfiability problems solved. Error bars indicate 95% confidence intervals around the mean over 1000 instances, each with 200 variables and 3.5 clauses per variable. (The DFS and DDS means are lower bounds.)

3.3 Boolean Satisfiability

We also tested on instances of boolean satisfiability. Following Walsh [1997], we generated problems according to the random 3-SAT model with 3.5 clauses per variable and filtered out any unsatisfiable problems. All algorithms used unit propagation, selected the variable occurring in the most clauses of minimum size, and preferred the value whose unit propagation left the most variables unassigned. The cost of a leaf was computed as the number of variables unassigned when the empty clause was encountered.

Figure 8 shows the percentage of 200-variable problems solved as a function of the number of nodes generated. Although Walsh used these problems to argue for the suitability of DDS, we see that both ILDS and purely random sampling perform significantly better. (Crawford and Baker [1994] similarly found random sampling effective on scheduling problems that had been converted to satisfiability problems.) DFS performs very poorly. Adaptive probing performs slightly better than random sampling (this is most noticeable at the extremes of the distribution). Although slight, this advantage persisted at all problem sizes we examined (100, 150, 200, and 250 variables).

To summarize: in each search space we examined, the systematic search algorithms ranked differently in performance. This makes it difficult to select which algorithm to use for a new problem. Even when taking its overhead into account, adaptive probing seemed to perform respectably in every search space. The only space in which it was not the best or near the best was the CKK space for number partitioning, in which the node-ordering heuristic is very accurate. Of course, further work is needed to assess its performance in very different domains, such as those with a high branching factor, and against additional methods, such as interleaved depth-first search [Meseguer, 1997].

4 Related Work

Abramson [1991] used random sampling in two-player game trees to estimate the expected outcome of selecting a given move. He also discussed learning a model off-line to predict outcome from static features of a node. In an optimization context, Juillé and Pollack [1998] used random tree probing as a value choice heuristic during beam search, although no learning was used.

Bresina [1996] used stochastic probing for scheduling, introducing a fixed *ad hoc* bias favoring children preferred by the node-ordering heuristic. Adaptive probing provides a way to estimate that bias on-line, rather than having to specify it beforehand, presumably using trial and error. By removing this burden from the user, it also becomes feasible to use a more flexible model.

The DTS system of Othar and Hansson [1994] uses learning during search to help allocate effort. Their method learns a function from the value of a heuristic function at a node to the node's probability of being a goal and the expected effort required to explore the node's subtree. It then explores nodes with the greatest expected payoff per unit of effort. In a similar vein, Bedrax-Weiss [1999] proposed weighted discrepancy search, which uses a training set of similar problems to estimate the probability that a node has a goal beneath it, and uses the distribution of these values to derive an optimal searching policy. Adaptive probing is less ambitious and merely estimates action costs rather than goal probability.

Squeaky-wheel optimization [Joslin and Clements, 1998] adapts during tree search, although it learns a variable ordering for use with a greedy constructive algorithm, rather than learning about the single tree that results from using an ordinary variable choice heuristic. The relative benefits of adapting the variable ordering as opposed to the value ordering seem unclear at present. Adaptive probing is slightly more general, as the squeaky-wheel method requires the user to specify a domain-specific analysis function for identifying variables that should receive increased priority during the next probe.

Adaptive tree probing is similar in spirit to iterative improvement algorithms such as adaptive multi-start [Boese *et al.*, 1994], PBIL [Baluja, 1997], and COMIT [Baluja and Davies, 1998] which explicitly try to represent promising regions in the search space and generate new solutions from that representation. For some problems, however, tree search is more natural and heuristic guidance is more easily expressed over extensions of a partial solution in a constructive algorithm than over changes to a complete solution. Adaptive probing gives one the freedom to pursue incomplete heuristic search in whichever space is most suitable for the problem. It is a promising area of future research to see how the two types of heuristic information might be combined.

The Greedy Random Adaptive Search Procedure (GRASP) of Feo and Resende [1995] is, in essence, heuristic-biased stochastic probing with improvement search on each leaf. Adaptive probing provides a principled, relatively parameter-free, way to perform the probing step. Similarly, aspects of Ant Colony Optimization algorithms [Dorigo and Gambardella, 1997], in which 'pheremone' accumulates to represent the information gathered by multiple search trials, can be seen as an approximation of adaptive probing.

Adaptive probing is also related to STAGE [Boyan and Moore, 1998], which attempts to predict promising starting points for hill-climbing given the values of user-specified problem-specific features. The discrepancy cost model requires less of the user, however, since the usual node-ordering function is used as the only problem-specific feature. The tree structure itself can be used to give the geometry for the search space model.

Although adaptive tree probing seems superficially like traditional reinforcement learning, since we are trying to find good actions to yield the best reward, important details differ. Here, we always start in the same state, choose several actions at once, and transition deterministically to a state we have probably never seen before to receive a reward. Rather than learning about sequences of actions through multiple states, our emphasis is on representing the possible action sets compactly to facilitate generalization about the reward of various sets of actions. We assume independence of actions, which collapses the breadth of the tree, and additivity of action costs, which allows learning from leaves. In essence, we generalize over both states and actions.

5 Possible Extensions

The particular adaptive probing algorithm we have evaluated is only one possible way to pursue this general approach. It would be interesting to try more restricted models, perhaps forcing action costs to be a smooth function of depth, for example. It may be worthwhile to distribute variance unequally among depths. Additional features besides depth might be helpful, perhaps characterizing the path taken so far.

The algorithm we have investigated here takes no prior experience into account. An initial bias in favor of the heuristic may be beneficial. Furthermore, it may also be possible to reuse the learned models across multiple problems in the same domain.

Adaptive probing can be used for goal search, as we saw with boolean satisfiability, as long as a maximum depth and a measure of progress are available. If a measure of leaf quality is not available, it may be possible to fit the model using many small instances of similar problems (or small versions of the current problem) that can be quickly solved and then to scale up the model to guide probing on the original problem.

6 Conclusions

It is a widely held intuition that tree search is only appropriate for complete searches, while local improvement search dominates in hard or poorly understood domains. Adaptive probing can overcome the strong assumptions that are built into systematic tree search procedures. By learning a model of the tree on-line and simultaneously using it to guide search, we have seen how incomplete heuristic search can be effective in a tree-structured search space. When the node-ordering heuristic is very accurate, a systematic discrepancy search algorithm may be more effective. But for problems with unknown character or domains that are less well-understood, the robustness of adaptive probing makes it superior. Its flexibility raises the possibility that, for difficult and messy problems, incomplete tree search may even be a viable alternative to local improvement algorithms.

Acknowledgments

Thanks to Stuart Shieber, Avi Pfeffer, Irvin Schick, Rocco Servedio, Jeff Enos, and the Harvard AI Research Group for their many helpful suggestions and comments. This work was supported in part by NSF grants CDA-94-01024 and IRI-9618848.

References

[Abramson, 1991] Bruce Abramson. *The Expected-Outcome Model of Two-Player Games*. Pitman, 1991.

[Baluja and Davies, 1998] Shumeet Baluja and Scott Davies. Fast probabilistic modeling for combinatorial optimization. In *Proceedings of AAAI-98*, 1998.

[Baluja, 1997] Shumeet Baluja. Genetic algorithms and explicit search statistics. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems 9*, 1997.

[Bedrax-Weiss, 1999] Tania Bedrax-Weiss. *Optimal Search Protocols*. PhD thesis, University of Oregon, Eugene, OR, August 1999.

[Boese *et al.*, 1994] Kenneth D. Boese, Andrew B. Kahng, and Sudhakar Muddu. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, 16:101–113, 1994.

[Boyan and Moore, 1998] Justin A. Boyan and Andrew W. Moore. Learning evaluation functions for global optimization and boolean satisfiability. In *Proceedings of AAAI-98*, 1998.

[Bresina, 1996] John L. Bresina. Heuristic-biased stochastic sampling. In *Proceedings of AAAI-96*, pp 271–278, 1996.

[Cesa-Bianchi *et al.*, 1996] Nicolò Cesa-Bianchi, Philip M. Long, and Manfred K. Warmuth. Worst-case quadratic loss bounds for on-line prediction of linear functions by gradient descent. *IEEE Transactions on Neural Networks*, 7(2):604–619, 1996.

[Crawford and Baker, 1994] James M. Crawford and Andrew B. Baker. Experimental results on the application of satisfiability algorithms to scheduling problems. In *Proceedings of AAAI-94*, pages 1092–1097, 1994.

[Dorigo and Gambardella, 1997] Marco Dorigo and Luca Maria Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[Feo and Resende, 1995] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[Gent and Walsh, 1996] Ian P. Gent and Toby Walsh. Phase transitions and annealed theories: Number partitioning as a case study. In *Proceedings of ECAI-96*, 1996.

[Hansson and Mayer, 1994] Othar Hansson and Andrew Mayer. Dts: A decision-theoretic scheduler for space telescope applications. In Monte Zweben and Mark S. Fox, editors, *Intelligent Scheduling*, chapter 13, pages 371–388. Morgan Kaufmann, San Francisco, 1994.

[Harvey and Ginsberg, 1995] William D. Harvey and Matthew L. Ginsberg. Limited discrepancy search. In *Proceedings of IJCAI-95*, pages 607–613, 1995.

[Johnson *et al.*, 1991] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon. Optimization by simulated annealing: An experimental evaluation; Part II, graph coloring and number partitioning. *Operations Research*, 39(3):378–406, May-June 1991.

[Joslin and Clements, 1998] David E. Joslin and David P. Clements. "Squeaky wheel" optimization. In *Proceedings of AAAI-98*, pages 340–346. MIT Press, 1998.

[Juillé and Pollack, 1998] Hughes Juillé and Jordan B. Pollack. A sampling-based heuristic for tree search applied to grammar induction. In *Proceedings of AAAI-98*, pages 776–783. MIT Press, 1998.

[Karmarkar and Karp, 1982] Narenda Karmarkar and Richard M. Karp. The differencing method of set partitioning. Technical Report 82/113, Berkeley, 1982.

[Karmarkar *et al.*, 1986] Narenda Karmarkar, Richard M. Karp, George S. Lueker, and Andrew M. Odlyzko. Probabilistic analysis of optimum partitioning. *Journal of Applied Probability*, 23:626–645, 1986.

[Korf, 1995] Richard E. Korf. From approximate to optimal solutions: A case study of number partitioning. In *Proceedings of IJCAI-95*, 1995.

[Korf, 1996] Richard E. Korf. Improved limited discrepancy search. In *Proceedings of AAAI-96*, pp 286–291, 1996.

[Meseguer, 1997] Pedro Meseguer. Interleaved depth-first search. In *Proceedings of IJCAI-97*, pp 1382–1387, 1997.

[Walsh, 1997] Toby Walsh. Depth-bounded discrepancy search. In *Proceedings of IJCAI-97*, 1997.