Using Prior Knowledge with Adaptive Probing

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

Abstract

When searching a tree to find the best leaf, complete search methods such as depth-first search and depth-bounded discrepancy search use a fixed deterministic order that may or may not be appropriate for the tree at hand. Adaptive probing is a recently-proposed stochastic method that attempts to adjust its sampling on-line to focus on areas of the tree that seem to contain good solutions. While effective on a variety of trees, adaptive probing wastes time learning basic features of the problem that are built into other algorithms, such as the fact that the heuristic is often helpful. In this paper, we investigate two simple methods for adding such prior knowledge to adaptive probing. The first simply reuses the model learned during a previous run on a similar problem. The second uses a heuristically biased policy at the start of the search, gradually deferring to learned information in later iterations. Empirical results on two different representations of number partitioning confirm that these methods can allow adaptive probing to search efficiently from the very start of a run. However, reusing previous models seems to more frequently preserve the ability of the algorithm to adapt to the search space.

Introduction

In many search problems of practical interest, time limitations prohibit an exhaustive search of the tree to find the best leaf. The best strategy one can adopt in such a situation is to visit those leaves first that have the greatest chance of being optimal. Complete search methods, such as depth-first search and depth-bounded discrepancy search (Walsh 1997), follow fixed deterministic search orders as they backtrack and explore alternative solutions. In most problems, a heuristic function is used to rank the children of each node according to the expected quality of their descendants. If this function tends to be more accurate at the bottom of the tree than at the top, then the search order of depth-first search will be exactly wrong—the decisions for which the heuristic was most accurate will be revisited first. In general, the fixed ordering of a systematic algorithm embodies a set of assumptions that may or may not match the tree at hand. In effect, the algorithms embody an assumption of certainty while any real world application has at least two elements of uncertainty: it can be unclear which set of assumptions best models a general problem class and it can be unclear how closely any given instance adheres to its class's typical characteristics.

Adaptive probing (Ruml 2001) is a stochastic sampling method that attempts to adjust to the tree on-line, rather than using a prespecified ordering. The algorithm iteratively probes from the root to a leaf. Starting without preconceptions about whether or not to follow the heuristic, the observed leaf costs are used to incrementally build a model of the tree. In a manner reminiscent of reinforcement learning, the estimates are simultaneously used during probing to choose which child to expand. This approach allows the algorithm to adjust to the particular tree it finds itself in, learning when to trust the heuristic and when not to.

Ruml (2001) presents results showing that adaptive probing can successfully adapt to a wide variety of trees and that it is competitive with systematic algorithms except when the heuristic is very accurate. When the heuristic is very often correct, adaptive probing suffers the overhead of having to discover that fact from scratch. In this paper, we investigate two methods for remedying this liability. Both are based on the idea of adding our prior knowledge as an element of the search. The first approach is simply to re-use a model that was built during a previous run on a similar problem. This avoids having to specify an initial bias manually, although it requires identifying classes of similar problems. The second is to use a pre-specified probing policy initially, but slowly discount its influence in favor of the learned model. We will empirically evaluate the effectiveness of these approaches on two different search spaces derived from the combinatorial optimization problem of number partitioning. We will see that the simpler method is more robust, while the combination of policies provides a less reliable advantage. But first, we will briefly review the basic adaptive probing algorithm itself.

Adaptive Probing

Following Ruml (2001), we will use an additive cost model for adaptive probing. We will postulate costs for choosing a preferred or non-preferred child at each depth and then assume that the cost of a leaf is the sum of the costs along its path from the root. Each cost is assumed to be drawn from a normal distribution with some fixed mean and variance. This variance is assumed to be the same for all costs and the mean cost of expanding the preferred or non-preferred child is assumed to be the same at each particular depth in the tree. For example, the cost of choosing the non-preferred child at depth 23 is assumed to be the same at all nodes at depth 23. For a tree of depth d and branching factor b, this model will have db parameters for the costs, plus one for the variance.

To estimate these parameters, we use the observed leaf costs. If $a_j(i)$ is the cost of taking child 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:

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

We use the perceptron learning algorithm to update the $a_j(i)$ according to $\eta(l_k - \hat{l}_k)/d$, where the learning rate η is set to 0.2. If we assume the costs at each level are independent, then the variance can be estimated from the observed variance in leaf costs.[1] The only complication is that the observed variance will be magnified according to the difference in the means at each level—if they differ, we will see variance even if each cost has none. Happily, this correction term is easy to compute (Ruml 2001).

To use the model, we select each action at a node according to the probability that it leads to solutions will lower costs (ie, that it has the lower mean). Given that we know the estimated means and variance for each action and we know how many times we have selected each of them, this can be computed using a standard statistical test for the difference of two sample means. This test is sensitive both to the difference between the costs of the actions and to the number of times each one has been tried. For a given level of variance, an action either has to have a much lower mean cost or we have to have tried it many times before we will give it high probability of being actually lower. To prevent the algorithm from eventually converging to a single path, the probability of choosing any action is clamped at $0.05^{1/d}$, which ensures at least one deviation on 95% of probes.

By giving the non-preferred children relatively high costs at shallow depths and costs at deep depths that are comparable to the preferred children, this model should be able to express behavior similar to depth-first search. Similarly, a preference for the heuristic at deep depths but not a shallow ones would mimic depth-bounded discrepancy search (Walsh 1997), and a moderate preference at all levels should produce behavior somewhat akin to limited discrepancy search (Harvey & Ginsberg 1995; Korf 1996).

Reusing Learned Models

With db parameters and a single observed leaf cost per probe, adaptive probing will take $O(db)$ probes to estimate the costs of choosing each child. While the ability to adjust to any possible configurations of costs is admirable, it is unlikely that the heuristically preferred child is actually significantly worse than the others. We would like to avoid having to spend the time to learn this, while still maintaining the flexibility to change the model if evidence suggests we have encountered one of these rare situations. Perhaps the simplest way of avoiding a prolonged initial learning period is to begin with an estimated model. More specifically, we can use the costs estimated from a previous run on a similar problem, while setting the variance to ∞ and the recorded number of counts for each action to 0. This should improve the accuracy of our estimated costs, speeding our identification of the preferred child as useful, while still allowing the algorithm plenty of latitude to explore and revise the costs as it gradually becomes confident in its estimates and focuses the search.

[1]We ignore, at our peril, the fact that the probing will tend to converge to high-quality solutions, thereby distorting our estimates.

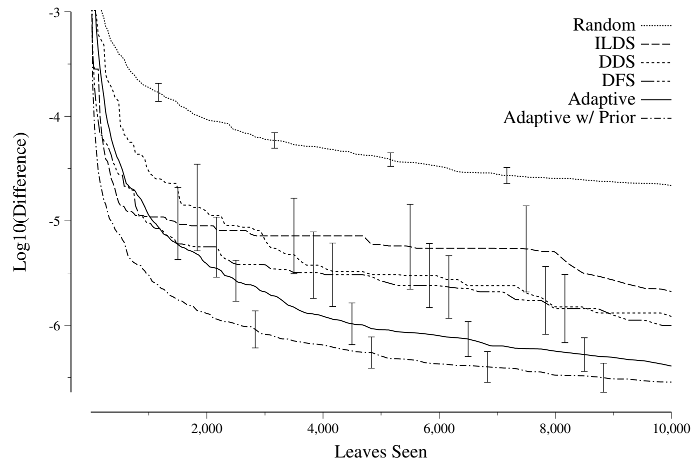

Figure 1: Searching the greedy representation of number partitioning. Error bars indicate 95% confidence intervals around the mean over 20 instances, each with 128 44-digit numbers.

We will test this method in two different search trees derived from the combinatorial optimization problem of 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. Common Lisp, which provides arbitrary precision integer arithmetic, was used to implement the algorithms. All results are normalized as if the original numbers were between 0 and 1. To better approximate a normal distribution, the logarithm of the partition difference was used as the leaf cost.

Greedy Number Partitioning

The first type of search tree we will examine is derived from a straightforward greedy encoding of the problem. The numbers are sorted in descending order and then each decision places the largest remaining number in a partition, preferring the smaller partition. Figure 1 compares the performance of adaptive tree probing with depth-first search (DFS), improved limited discrepancy search (ILDS), depth-

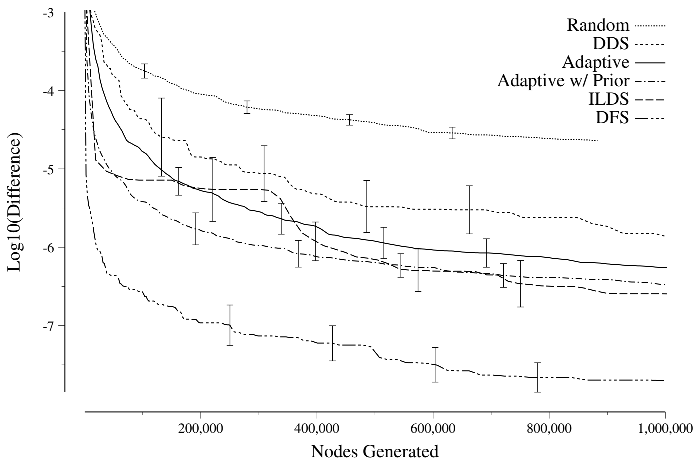

Figure 2: Performance on the greedy representation as a function of nodes generated.

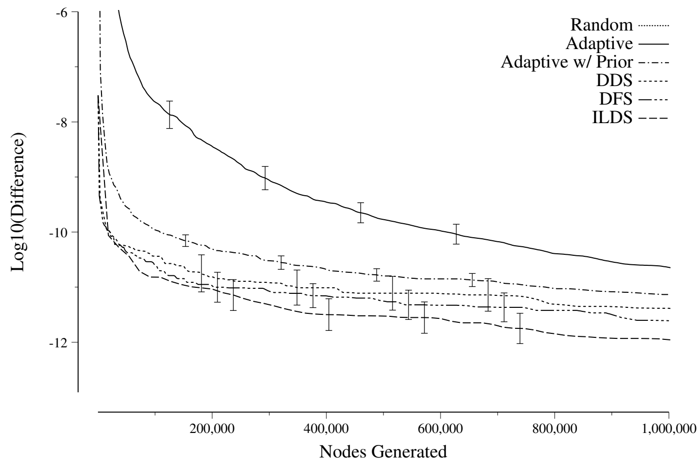

Figure 3: Searching the CKK representation of number partitioning problems.

bounded discrepancy search (DDS), and completely random tree probing. Adaptive probing was run twice–the first time with its model's costs initialized to zero and the second time with the costs that were estimated by a previous run on a different problem instance. To provide a comparison of the algorithms' search orders, the horizontal axis represents the number of leaves seen.

The figure shows that adaptive probing, in addition to learning to explore a profitable part of the search space, benefits from the prior knowledge. While *tabula rasa* adaptive probing needs to see approximately 1,500 leaves before overtaking the systematic algorithms, the estimates transferred from the previous problem lead the algorithm directly to good solutions.

Although Figure 1 shows that adaptive probing with transferred knowledge quickly learns a good search order, it ignores the overhead that is inherent in restarting at the root with each probe. Figure 2 corrects for this factor, showing performance as a function of the number of nodes (both internal and leaves). Both adaptive probing and DDS suffer the maximum possible overhead compared to DFS, which generates roughly one internal node per leaf. This is reflected in the figure, as DFS finds superior solutions when given the same number of node generations. Although prior knowledge provides a benefit, it is not enough to overcome the overhead of adaptive probing. Adaptive probing is competitive with ILDS, although results using only 64 numbers (which appear in Figure 4) indicate that adaptive probing surpasses ILDS as problem size grows.

CKK Number Partitioning

A more sophisticated representation for number partitioning, called CKK, 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, as the heuristic tends to be extremely accurate.

Figure 3 presents the performance of the search algorithms as a function of the number of nodes generated. Random probing would appear off the top of the plot. ILDS performs best in this domain, indicating that the heuristic tends to be inaccurate with equal probability at all depths. Ordinary adaptive probing takes a long time to learn that the heuristic is usually accurate everywhere, although it looks as if it may eventually approach the systematic algorithms' performance. When imbued with prior knowledge, adaptive probing quickly approaches DDS (which suffers similar node generation overhead). The benefit of using prior knowledge seems to be greater in this search space than in the greedy one, even though it is the harder one for plain adaptive probing. When the knowledge is harder to acquire, receiving it in advance represents a greater savings.

Blending Search Policies

While reusing an old model is easy and seems remarkably effective, it is only possible if one has the luxury of previous experience with a similar problem. If the previous problem has a very different distribution of leaf costs, the initial bias can be counter-productive. Another method for taking advantage of our *a priori* expectation that the heuristic is beneficial is to behave at first according to that belief, while continuing to learn a fresh model of the tree. We can then gradually reduce the frequency with which we make our decisions according to the prejudiced policy and begin to rely more heavily on our experience in the tree at hand. While this method applies even in the absence of experience with similar problems, it requires a prior judgment on how quickly to make the switch. This is essentially the same problem as deciding how much to trust the initial bias.

In the experiments reported below, we used a multiplicative discounting policy. At the first probe, we use the prior

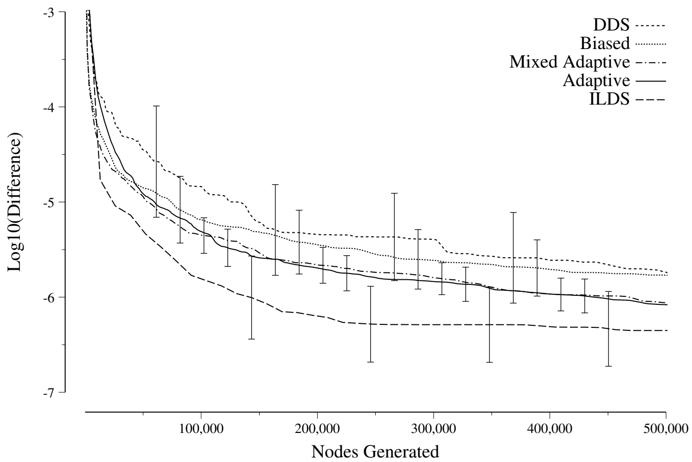

Figure 4: Searching the greedy representation of number partitioning instances, each with 64 25-digit numbers.

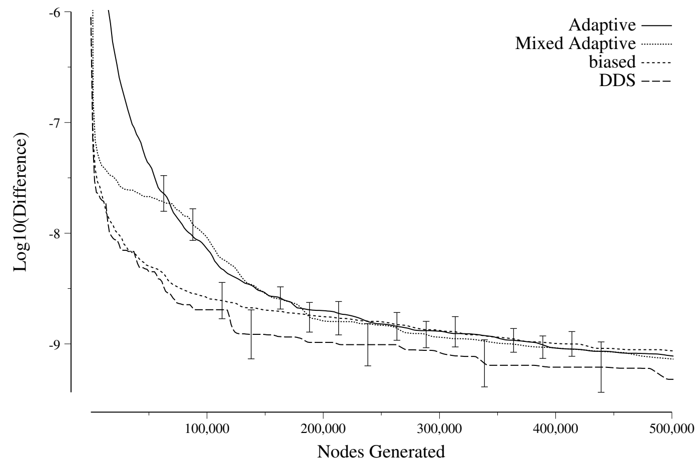

Figure 5: Searching the CKK representation of number partitioning.

bias with probability 1. After every probe, this probability is multiplied by a constant such that, after $15d$ iterations, we are as likely to use the current estimated model as we are to use the prior bias.[2] In effect, this creates a changing blend of the initial policy and the current model. For an initial policy, we use an algorithm which selects the preferred child with the maximum probability that would be allowed under adaptive probing (recall that we clamped the probability of any child as a safeguard against complete convergence of the algorithm).

Evaluation

The empirical performance of blending policies was mixed. Figure 4 shows algorithm performance using the greedy representation of number partitioning instances with 64 numbers. Besides plain adaptive probing and the blended policy, we also show the performance of a biased probing algorithm that just uses the initial policy of the blended algorithm. This biased probing algorithm performs on par with DDS in the greedy search space, but seems to be a little worse than plain adaptive probing. The blended adaptive probing algorithm seems equivalent to the plain. On larger problems, however, the blended policy was eventually surpassed by plain adaptive probing. This might be caused by the very skewed distribution of examples on which the model is trained.

In the CKK search space, policy blending seemed to work reasonably well. Figure 5 shows the performance of the algorithms on 64-number problems. The blended algorithm follows the biased probing algorithm at first, then switches over to mimic the adaptive one. Unfortunately, the good performance of the initial biased policy seems to provide little benefit to the model learned by the adaptive component. Figure 6 shows performance on larger problems. Here, the model learned by the blended algorithm seems to have benefitted from its initial experience, although the algorithm still suffers a significant stagnant period during the transition in which little improvement is seen. Use of a shorter or longer blending time seemed to result in worse performance in preliminary experiments on small problems. Using an abrupt changeover rather than a gradual blending also led to the learning of a poor model.

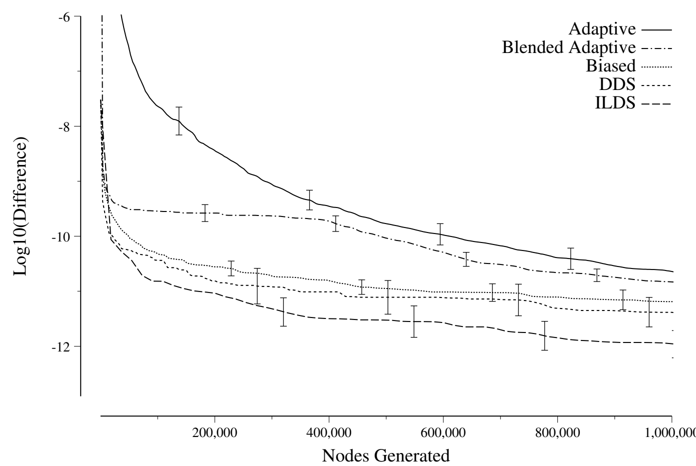

Figure 6: Searching the CKK representation of instances with 128 numbers.

Related Work

Much work has been done on using learned models to guide heuristic optimization (Boese, Kahng, & Muddu 1994; Baluja 1997; Baluja & Davies 1998) although few authors have explicitly investigated integrating prior knowledge or transferring problem-solving experience between problems. The X-STAGE algorithm (Boyan & Moore 2000) performs a similar task in the context of improvement search instead of tree search. The STAGE algorithm learns a model during search that predicts when an initial solution will yield good results with hill-climbing (or a similar algorithm). This model is used to intelligently restart after the hill-climbing has reached a local maximum by switching temporarily to hill-climbing according to the model's prediction of a solu-

[2] This constant is $0.5^{1/(15d)}$.

tion's potential as a starting place (which might be different from its quality). In X-STAGE, several initial training problems are run and a separate model is learned on each using STAGE. These models are then used to solve a new problem by having each model vote on whether or not to accept a proposed modification to the starting solution. This avoids worrying about having to scale a model to appropriate values for use on a new problem, but does not allow any adaptation to the new problem instance.

Horvitz *et al.* (2001) use runs on training data to learn a model of running time, and then use this model to derive a restart policy for a randomized backtracking search. This high-level wrapper approach could be used on top of the approach we pursue here, which focuses on the structure of the search space rather than the black-box behavior of a solver.

Conclusions

Systematic tree search algorithms fail to exhibit a fundamental trait of intelligent behavior: they do not adapt to their surroundings. Purely adaptive algorithms, on the other hand, are too general to exhibit high performance in the commonly encountered search spaces in which the heuristic may be assumed to be helpful. We investigated two methods for using prior knowledge with an adaptive probing algorithm. The simplest one, merely reusing the action costs estimated on a similar problem, seemed to perform the best. An attempt to blend an *a priori* policy with the learning algorithm gave some improvement, but seems prone to leading to ineffective learning. One way to circumvent these difficulties would be to allow a biased adaptive probing algorithm to recognize when its training data is unrepresentative and initiate further exploration on its own. This is an important direction for future work.

Acknowledgments

Many thanks to Stuart Shieber and the Harvard AI Research Group for helpful suggestions and comments relating to this work.

References

Baluja, S., and Davies, S. 1998. Fast probabilistic modeling for combinatorial optimization. In *Proceedings of AAAI-98*.

Baluja, S. 1997. Genetic algorithms and explicit search statistics. In Mozer, M. C.; Jordan, M. I.; and Petsche, T., eds., *Advances in Neural Information Processing Systems 9*.

Boese, K. D.; Kahng, A. B.; and Muddu, S. 1994. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters* 16:101–113.

Boyan, J. A., and Moore, A. W. 2000. Learning evaluation functions to improve optimization by local search. *Journal of Machine Learning Research* 1:77–112.

Harvey, W. D., and Ginsberg, M. L. 1995. Limited discrepancy search. In *Proceedings of IJCAI-95*, 607–613. Morgan Kaufmann.

Horvitz, E.; Ruan, Y.; Gomes, C.; Kautz, H.; Selman, B.; and Chickering, M. 2001. A bayesian approach to tackling hard computational problems. In *Proceedings of UAI-01*.

Johnson, D. S.; Aragon, C. R.; McGeoch, L. A.; and Schevon, C. 1991. Optimization by simulated annealing: An experimental evaluation; Part II, graph coloring and number partitioning. *Operations Research* 39(3):378–406.

Karmarkar, N., and Karp, R. M. 1982. The differencing method of set partitioning. Technical Report UCB/CSD 82/113, Computer Science Division, University of California, Berkeley.

Karmarkar, N.; Karp, R. M.; Lueker, G. S.; and Odlyzko, A. M. 1986. Probabilistic analysis of optimum partitioning. *Journal of Applied Probability* 23:626–645.

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

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

Ruml, W. 2001. Incomplete tree search using adaptive probing. In *Proceedings of IJCAI-01*, 235–241.

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