Yale University
Department of Computer Science

A Greedy Approximation Algorithm for the Partial-Order Search Problem

Philip Laird

YALEU/DCS/TR-547

June, 1987

Research funded in part by the National Science Foundation, IRI-8404226.

A Greedy Approximation Algorithm for the Partial-Order Search Problem

Philip Laird*

Yale University
Department of Computer Science
New Haven, CT. 06520

Abstract

The problem of devising an optimal strategy for finding elements of a partially ordered set has been previously shown to be NP-complete. Here we present a greedy approximation algorithm that runs in polynomial time and questions at most $O(\log n)$ times as many elements as the optimal. This bound is asymptotically tight for the algorithm: there are arbitrarily large instances for which the algorithm questions at least $H(n) - o(1/n)$ times as many elements as necessary, where $H(n)$ is the n'th harmonic number.

Introduction

The Partial-Order Search (POS) problem ([4]) is defined as follows:

> Given a finite, partially-ordered set $\langle \mathcal{E}, \geq \rangle$, an integer k such that $1 \leq k \leq |\mathcal{E}|$, and an oracle (to be described), to decide whether there is a strategy that identifies an arbitrary target element ν_0 of $\mathcal{E}$ by asking the oracle at most k questions.

Research funded in part by the National Science Foundation, IRI-8404226.
*Current address: NASA Ames Research Center, MS 244-17, Moffett Field, CA. 97035.

The oracle, when asked a question of the form ν? (with $e \in \mathcal{E}$), answers ">", "<", "=", "$\not\sim$" according to whether the target element is greater than, less than, equal to, or incomparable to ν.

This problem generalizes the familiar binary-search problem, for which the ordering is total. When the set being searched has n elements, the binary-search strategy identifies an arbitrary element in at most $\lfloor \log n \rfloor + 1$ queries. We would like to extend this result to an efficient algorithm for finding an optimal search strategy with an arbitrary partial order.

The POS problem has been shown to be NP-Complete with respect to polynomial-time reductions ([4]). Here we exhibit a "Greedy" algorithm that finds in polynomial time a correct strategy requiring $O(\log |\mathcal{E}|)$ times as many questions as the optimal strategy.

The Approximation Algorithm

A search strategy for a partially ordered set is conveniently represented by a decision tree whose internal nodes are questions and whose leaves are the elements identified by the questions on its ancestors. Such a tree is a valid search strategy if every element in the order appears on some leaf, and for each leaf element the questions occurring on the path from it to the root are sufficient to identify the element uniquely.

Let $N = |\mathcal{E}|$, which by assumption is finite. Let T be any decision tree (not necessarily optimal) for the POS problem; we may assume, without loss of generality, that T has exactly N leaf nodes. The height of T (maximum length of any path from the root to a leaf) is denoted $ht(T)$. For any positive integer i, let $H(i) = \sum_{j=1}^{i}(1/j)$.

Let ν be any node in $\mathcal{E}$. The *strength* of ν – written $S(\nu)$ – is the minimal number of nodes that may be eliminated by asking the oracle about ν. For example, suppose that ν has 4 nodes below (<) it, 6 above (>) it, and 5 incomparable ($\not\sim$) to it; then asking the oracle about ν will eliminate 12 nodes if the answer is "<", or 10 if ">", or 11 if "$\not\sim$", or 15 if "=", so $S(\nu) = 10$.

More generally, let $\mathcal{E}_1$ be a subset of $\mathcal{E}$ containing the target (unknown) element. The *strength of ν for $\mathcal{E}_1$* is the the minimum number of nodes in $\mathcal{E}_1$ that may be eliminated by asking the oracle about ν. This generalization is useful because the search algorithm may ask the oracle about an element ν, even if the algorithm can already deduce the fact that ν is not the target element, in order to find out whether the target is among the elements greater than, less than, or incomparable to ν. So if $\mathcal{E}_1$ is the set of nodes in $\mathcal{E}$ not yet ruled out by questions, the strength of $\nu \in \mathcal{E}$ will be measured according to how many elements of $\mathcal{E}_1$ it has the potential to eliminate.

To approximate the solution to an instance of POS, we construct a decision tree T_1 in the following "greedy" manner:

1. Initialize: $U = \mathcal{E}$ (the set of possible target nodes not yet eliminated by queries).
2. The recusive procedure *Greedy*(U) returns T_1.

The procedure *Greedy*(X) is as follows:

If X consists of a single element, return the tree consisting of a single leaf node for X. Otherwise,

1. For every node $\nu \in \mathcal{E}$, compute the strength $S(\nu)$ of ν for X. Let ν_m be a node of maximum strength.
2. Let $X_< = \{\nu' \in X \mid \nu' < \nu_m\}$. Compute $T_< = Greedy(X_<)$, the decision subtree for the nodes below ν_m.
3. Let $X_> = \{\nu' \in X \mid \nu' > \nu_m\}$. Compute $T_> = Greedy(X_>)$, the decision subtree for the nodes above ν_m.
4. Let $X_{\not<} = \{\nu' \in X \mid \nu' \not< \nu_m\}$. Compute $T_{\not<} = Greedy(X_{\not<})$, the decision subtree for the nodes incomparable to ν_m.
5. Return the decision tree whose root is the question ν_m? and whose subtrees for the answers "=", "<", ">", and "$\not<$" are, respectively, ν_m, $T_<$, $T_>$, and $T_{\not<}$.

Briefly, this procedure chooses a node which eliminates the largest number of remaining nodes in the worst case, and makes this node the root. It then recursively constructs the greedy decision tree for the remaining elements below, above, and incomparable to this root. The first step requires $O(N^2)$ time, and without any concerns for efficiency the entire procedure can be implemented in $O(N^3)$ time.

Upper Bound

The following theorem shows that the height of the greedy decision tree T_1 is greater than that of the optimal decision tree by a factor of at most $H(N)$. (Note that $H(N) \leq \ln N + 1$.) The proof uses techniques originally devised to obtain approximations to set cover problems ([2],[3]).

Theorem 1 Let T_1 be the decision tree returned by the Greedy approximation to a POS problem, and T_0 an optimal decision tree for the same problem instance. Then $ht(T_1) \leq ht(T_0) \cdot H(N)$, where $N = |\mathcal{E}|$.

PROOF: Let T be any decision tree for an instance of the POS problem, and let $\nu?$ be an internal node of T. Define $Elim(\nu, T)$ to be the minimum number of leaf nodes of T having $\nu?$ as an ancestor that are eliminated by the question "$\nu?$" in the tree – i.e., , the strength of ν relative to the leaves of the subtree of T rooted at ν. For a leaf node ν, $Elim(\nu, T)$ is defined to be 0.

We shall prove the following, stronger, statement: for *any* decision tree T_0, there exists a path $P = \langle \nu_1?, \nu_2?, \ldots, \nu_b?, \nu_b \rangle$ in T_0 from the root $\nu_1?$ to a leaf ν_b such that

$$ht(T_1) \leq \sum_{\nu \in P} H(Elim(\nu, T_0)). \tag{1}$$

The theorem then follows directly: when T_0 is optimal, and P is the promised path, we have

$$\begin{aligned} ht(T_1) &\leq \sum_{\nu \in P} H(Elim(\nu, T_0)) \\ &\leq |P|\, H(N) \\ &\leq ht(T_0)\, H(N). \end{aligned}$$

The proof is by induction on $K = \max\{Elim(\nu, T_0) \mid \nu \in T_0\}$. For the base case $K = 0$, $N = 1$ (there is only one node in the partial order) and $ht(T_1) = 0$. By letting P be the "path" of zero length in T_0, the condition (1) holds trivially.

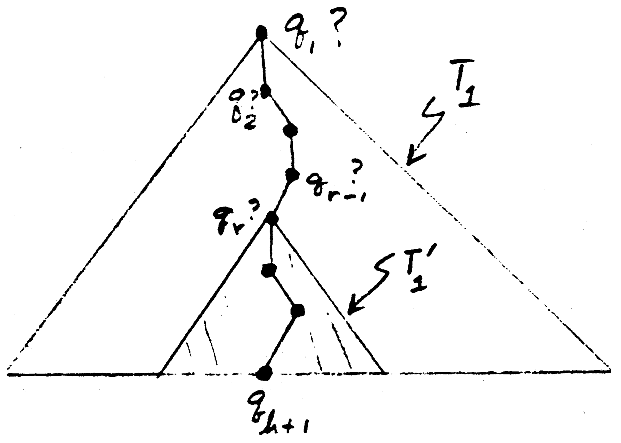


Figure 1: The path P_1.

For the induction step, we assume that (1) holds for $K < k$, and consider a tree T_0 such that $\max\{Elim(\nu, T_0) \mid \nu \in T_0\} = k$. Let $P_1 = \langle q_1?, q_2, \ldots, q_h?, q_{h+1} \rangle$ be a path of maximum length in T_1. (See Fig. 1.) Consider the sequence of integers $Elim(q_1, T_1), \ldots, Elim(q_h, T_1)$ along P_1. Because T_1 is a *greedy* decision tree, this sequence is non-increasing. Let r be the smallest integer such that

$$Elim(q_r, T_1) < k, \tag{2}$$

or h if no such integer exists. Let T_1' be the subtree of T_1 rooted at q_r. The strategy is to apply the inductive hypothesis to T_1'.

Note that

$$h = (r-1) + ht(T_1'). \tag{3}$$

By (2) we know that the total number of nodes,

$$\sum_{i=1}^{r-1} Elim(q_i, T_1),$$

eliminated by questions $q_1, \ldots, q_{r-1}$ is at least $k(r-1)$. Thus from (3), we have

$$h \le \frac{\sum_{i=1}^{r-1} Elim(q_i, T_1)}{k} + ht(T_1'). \tag{4}$$

Now we apply the inductive hypothesis to T_1', as follows. First, it is a greedy decision tree, and $Elim(\nu, T_1') < k$ for all $\nu \in T_1'$. Next, consider the decision tree T_0' obtained from T_0 by removing all leaves not in T_1' and recursively deleting all internal nodes with no descendents. (T_0' is T_0 reduced by the partial information contained in the answers to the questions $q_1, \ldots, q_{r-1}$.) Note that paths from the root to the remaining leaves in T_0 are not affected by this pruning, so $ht(T_0') = ht(T_0)$.

Next we observe that, for all internal nodes $\nu?$ in T_0', $Elim(\nu, T_0') < k$. To see this, suppose $Elim(\nu, T_0') > k - 1$ for some node $\nu \in T_0'$. Since T_0' and T_1' have the same set of leaves, T_1' (being greedy) would also be able to eliminate at least k leaves by placing the question $\nu?$ at its root. But this contradicts the fact that $Elim(\nu, T_1') < k$ for all nodes $\nu \in T_1'$.

Thus the inductive hypothesis applies to T_1' *vis à vis* T_0', and there exists a path P_0' in T_0' such that

$$\begin{aligned} ht(T_1') &\leq \sum_{\nu \in P_0'} H(Elim(\nu, T_0')) \\ &= \sum_{\nu \in P_0'} H(Elim(\nu, T_0)) - \sum_{\nu \in P_0'} [H(Elim(\nu, T_0)) - H(Elim(\nu, T_0'))]. \quad (5) \end{aligned}$$

Note that P_0' is also a path in T_0, and that $H(Elim(\nu, T_0)) \geq H(Elim(\nu, T_0'))$ for all $\nu \in P_0'$. And since $Elim(\nu, T_0) \leq k$ by hypothesis, we have:

$$\sum_{\nu \in P_0'} [H(Elim(\nu, T_0)) - H(Elim(\nu, T_0'))] \geq \frac{Elim(\nu, T_0) - Elim(\nu, T_0')}{k}.$$

Thus (5) becomes

$$ht(T_1') \leq \sum_{\nu \in P_0'} H(Elim(\nu, T_0)) - \frac{1}{k} \sum_{\nu \in P_0'} [Elim(\nu, T_0) - Elim(\nu, T_0')]. \quad (6)$$

But $\sum_{\nu \in P_0'} [Elim(\nu, T_0) - Elim(\nu, T_0')]$ is the number of leaves eliminated by the questions $q_1, \ldots, q_{r-1}$, which is the same as $\sum_{i=1}^{r-1} Elim(q_i, T_1)$. Thus (6) becomes:

$$ht(T_1') \leq \sum_{\nu \in P_0'} H(Elim(\nu, T_0)) - \frac{\sum_{i=1}^{r-1} Elim(q_i, T_1)}{k}.$$

Substituting this into (4), we obtain, finally,

$$h = ht(T_1) \leq \sum_{\nu \in P_0'} H(Elim(\nu, T_0)).$$

Thus P_0' is the required path in T_0. $\square$

Lower Bound

We now show that there exist partial orders with arbitrarily large cardinality N for which the height $ht(T_1)$ of the greedy decision tree is larger than the height $ht(T_0)$ of the optimal decision tree by a factor which can be made arbitrarily close to $H(N)$. Thus the ratio $ht(T_1)/ht(T_0) \sim \log N$ is asymptotically optimal for this algorithm.

As proof, we exhibit a sequence of partial orderings $\mathcal{E}_k$ $(k \geq 3)$ for which the greedy tree is of height $\Omega(N!H(N))$, and yet another decision tree can be found with height $\mathcal{O}(N!)$.

The poset $\mathcal{E}_k$ is the union of four disjoint subsets:

- A, with $k \cdot k!$ elements;
- U, with $k!H(k)$ elements;
- V, with $k!$ elements;
- X, with $k!(H(k)+1)-1$ elements.

To simplify the description, initially we ignore X altogether and assume that the target element is known to be among the members of A. Later we add the set X to the ordering and show that the decision trees for the entire partial order (including U, V, and X) increase in height only by an additive constant, for sufficiently large k.

The elements of A are the maximal elements in the ordering and are viewed as k collections of $k!$ nodes, which we name $A_1, \ldots, A_k$. The elements of A_i are labeled $a_{i1}, \ldots, a_{ik!}$. All elements of A are mutually incomparable.

The $k!$ elements of U are labeled $u_1, \ldots, u_{k!}$. In the ordering each u_i is below (i.e., $<$) k elements of the set A: u_i is below each of $a_{1i}, a_{2i}, \ldots, a_{ki}$.

The elements of V are viewed as k collections $V_1, \ldots, V_k$ of different sizes. V_1 has $k!$ members, V_2 has $\frac{1}{2}k!$ members, ..., and V_k has $(1/k)k!$ members, for a total of $k!H(k)$ elements. The elements of V_i (labeled $v_{i1}, v_{i2}, \ldots$) are below i elements of A_i: $v_{i1} < a_{i1}, a_{i2}, \ldots, a_{ii}$; $v_{i2} < a_{i,i+1}, a_{i,i+2}, \ldots, a_{i,2i}$; etc.

The sets A, U, and V are illustrated in Fig. 2 for $k = 3$.

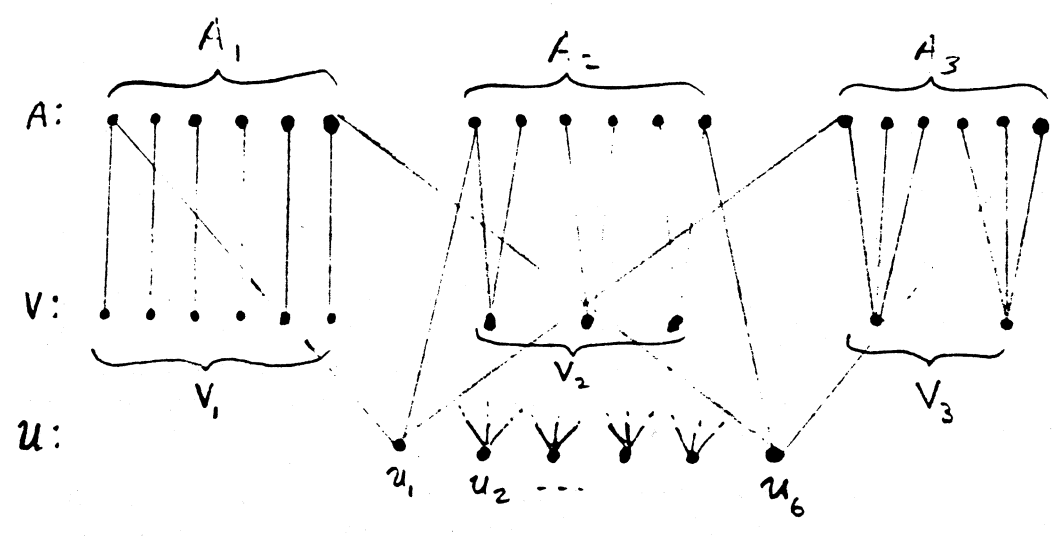


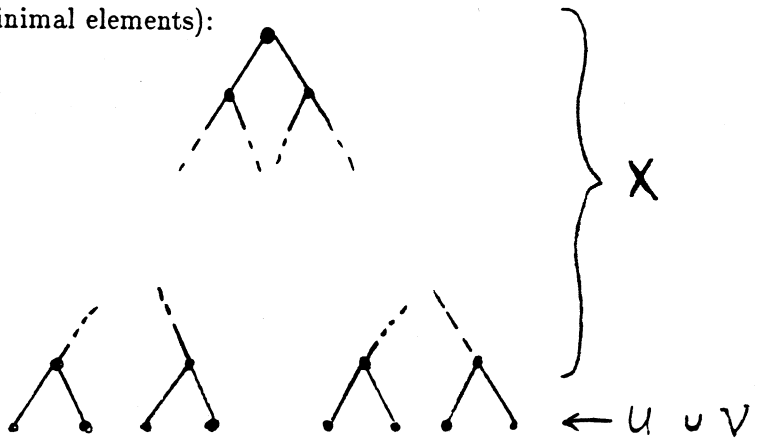
Figure 2: The ordering of sets A, U, and V.

Assume our decision tree need only identify elements of A. A greedy decision tree T_1 first asks about each of the $k!$ elements of V_k. An answer of ">" to any of these is followed by a decision subtree of height k to identify the target among the k elements of A_{ki}. If "$\not>$" is the answer to all of these, the elements of V_{k-1} are queried next, and so on. It is easy to check that this is a greedy tree, since a node of maximum strength for the uneliminated elements is always tested next. The resulting tree has height equal to $|V|$, or $k!H(k)$.

Another decision tree T_0 queries each of the elements of U in turn, up to $u_{k!-1}$. An answer of ">" to any of these is followed by a decision subtree of height k to identify the target among the k larger elements. If "$\not>$" is the answer to all of these, the target element must be among the k elements of A greater than $u_{k!}$, and a decision subtree of height k suffices to locate it. In all, the resulting tree has height $k! + k - 1$.

We note that $ht(T_1)/ht(T_0) = k!H(k)/(k! + k - 1)$, which approaches $H(k)$ for large k.

To complete the argument, we need to be able to identify the elements of U and V in the ordering. To this end, we introduce the set X which, together with U and V, form a tree-structured ordering for which the elements of U and V comprise the leaves (the minimal elements):



The elements of A are incomparable to those of X, and are above those of U and V as already described.

The reader can easily convince himself that the elements of $U \cup V \cup X$ can be identifed by a decision tree of height $\mathcal{O}(\log |U \cup V|) = \mathcal{O}(k \log k)$. If we modify both T_1 and T_0 by making the first (root) question a query about the maximum element of X, then the subtree corresponding to an answer of "$<$" is a tree of height $\mathcal{O}(k \log k)$ for identifying elements of U, V, and X, while the subtree for an answer of "$\not<$" is the tree T_1 or T_0 described above for identifying elements of A. Thus for sufficiently large k, the heights of the trees increase by one.

Open Questions

It should be possible to extend the POS problem to the ***Weighted*** POS problem, by assigning integer costs to each node of the partial order. In applications this cost would represent the difficulty in answering the query for that node. Accounting techniques akin to those of Chvatal ([2]) probably can be applied.

The most significant challenge is to discover why this problem and others have a reducibility and approximation structure so close to that of the set covering problem. Recently interest in set covering approximations has grown as a result of results in the theory of learnability (e.g., [1]). Evidently there is an interesting general combinatorial property underlying these problems that deserves to be identified.

Acknowledgement

I am grateful to Sandeep Bhatt for a helpful discussion this work.

References

[1] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. In *Proc. 18th Symposium on Theory of Computing*, pages 273–282, ACM, 1986. (To appear in *J.ACM*.

[2] V. Chvatal. A greedy heuristic for the set-covering problem. *Math. of Operations Research*, 4:233–235, 1979.

[3] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Sys. Sci.*, 9:256–278, 1974.

[4] P. D. Laird and L. B. Pitt. *Finding an optimal search strategy for a partial order is NP-complete.* Technical Report, Yale University Computer Science Dept., No. 493, 1986.