Probably Approximately Optimal Satisficing Strategies

Greiner, Russell

1996

Greiner, R & Orponen, P 1996, 'Probably Approximately Optimal Satisficing Strategies'
Artificial Intelligence, vol. 82, pp. 21-44. DOI: 10.1016/0004-3702(95)00010-0

http://hdl.handle.net/10138/243064
https://doi.org/10.1016/0004-3702(95)00010-0

Downloaded from Helda, University of Helsinki institutional repository.
This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.
Please cite the original version.
Probably Approximately Optimal Satisficing Strategies

Russell Greiner
Siemens Corporate Research
755 College Road, East
Princeton, NJ 08540

Pekka Orponen
Department of Computer Science
University of Helsinki, P. O. Box 26
FIN-00014 Helsinki, Finland

Abstract
A satisficing search problem consists of a set of probabilistic experiments to be performed in some order, seeking a satisfying configuration of successes and failures. The expected cost of the search depends both on the success probabilities of the individual experiments, and on the search strategy, which specifies the order in which the experiments are to be performed. A strategy that minimizes the expected cost is optimal. Earlier work has provided “optimizing functions” that compute optimal strategies for certain classes of search problems from the success probabilities of the individual experiments. We extend those results by providing a general model of such strategies, and an algorithm pao that identifies an approximately optimal strategy when the probability values are not known. The algorithm first estimates the relevant probabilities from a number of trials of each undetermined experiment, and then uses these estimates, and the proper optimizing function, to identify a strategy whose cost is, with high probability, close to optimal. We also show that if the search problem can be formulated as an and-or tree, then the pao algorithm can also “learn while doing”, i.e. gather the necessary statistics while performing the search.

1 Introduction
Consider the following situation: There are two reliable tests for deciding whether an individual has hepatitis; one involves a blood test and the other a liver biopsy. Assuming there can be false negatives but no false positives, there are two “strategies” a doctor can follow to obtain a diagnosis. Using strategy \( \Theta_1 = \{ \text{blood, liver} \} \), he would first perform the blood test and conclude the patient has hepatitis if that test is positive. If not, he would then

*Some of this work was performed while the authors were at the University of Toronto, supported respectively by an operating grant from the National Science and Engineering Research Council of Canada, and by the Academy of Finland. The authors thank Dale Schuurmans, Tom Hancock and the anonymous referees for their useful comments on earlier versions of this paper. Preliminary versions of parts of the work have appeared in the conference reports [10] and [19].
examine the patient's liver, and conclude his diagnosis based on the result of that biopsy. The doctor's other option, strategy $\Theta_2 = \langle \text{liver, blood} \rangle$, performs these tests in the other order — first the liver test and then, only if necessary, the blood test.

Which strategy is better? Our goal is a strategy that will perform well in practice. To quantify the measurement, we assume there is a distribution of patients that the doctor will be asked to evaluate. We can then define a strategy's expected cost as the average cost required to perform these tests, averaged over the distribution of anticipated patients. Assuming, for now, that these tests (blood and liver) have the same cost, strategy $\Theta_1$ is clearly better if the probability of a positive blood test ($p_B$) is larger than the probability of a positive liver test ($p_L$); otherwise, strategy $\Theta_2$ is preferable.

Earlier research on this decision making model has produced a number of "optimizing functions" that each identify a strategy optimal for a specific testing situation, given the success probability values of the relevant experiments [6, 21, 18, 22, 7]. A limitation of these techniques, however, is that the probability values are in practice typically not known a priori. This paper specifies the number of trials of each experiment that are required to obtain estimates of these probability values that are good enough to identify a nearly-optimal strategy, with high confidence. It also addresses the complexities of observing this many trials.

Section 2 below first generalizes from the doctor's situation to a general class of arbitrary "decision structures" and defines strategies, and optimal strategies, for these structures. Section 3 then specifies the pao algorithm, a general process that uses a set of observed trials of each experiment to identify a strategy whose cost is, with high probability, approximately optimal. The algorithm presumes the existence of an optimizing function for the class of search structures considered. When dealing with certain search structures, notably and-or trees, the pao algorithm can "learn while doing", i.e. gather the necessary statistics while solving relevant performance tasks.

An extended version of this paper, available as a technical report [11], discusses several variants and applications of the basic algorithm presented here.

2 Framework

2.1 Decision Structures

The doctor's task presented in Section 1 is a simple example of a satisficing search problem (term due to Simon and Kadane [21]), as his goal is to find a single satisfactory configuration of events: in this case, an informative combination of test results. Other examples of such problems include, e.g., performing a sequence of tests to decide whether a product specimen is satisfactory [6], screening employment candidates for a position [6], competing for prizes at a quiz show [6], mining for gold buried in treasure chests [21], and performing inference in simple expert systems [22, 10]. In general, such tasks may involve searching through general "decision structures", which can involve an arbitrary number of experiments, constrained by various precedence constraints.

And-Or Decision Trees: More general versions of this diagnostic task can be represented by and-or decision trees, such as $G_1$ in Figure 1. Here, the nodes $\{A, e_1, \ldots, e_6\}$ correspond
to experiments, and the arcs encode the precedence relationships — e.g., the doctor cannot perform experiment $e_3$ (test whether the patient’s blood reacts with a particular serum) until he has performed experiment $e_1$ (attempted to draw blood from the patient) and moreover, found that $e_1$ succeeds. The experiment associated with the $A$ node is formally degenerate, i.e. it is guaranteed to succeed. The set of arcs descending from a given node can be either disjunctive, or conjunctive (here indicated by a horizontal line, e.g., connecting the arcs from $e_2$ to $e_5$ and from $e_2$ to $e_6$). Hence, the graph in Figure 1 states that the patient has hepatitis iff the condition $[e_1 \land (e_3 \lor e_4)] \lor [e_2 \land e_5 \land e_6]$ on the experiments holds. The number near each arc designates the cost of traversing that arc — hence it costs 1 unit to reduce the top $e_0$ node to the $e_1$ subgoal (draw blood), and 2 more units to further reduce $e_1$ to $e_3$ (test the blood against serum $A$), and so forth. The incremental cost of performing each experiment is the sum of the costs of the additional arcs that must be traversed. (This cost specification means such trees cannot always be “collapsed” to simpler two-level trees.)

**Beyond And-Or Trees:** The general class of decision structures we shall consider is strictly more general than and-or trees. First, and-or trees can encode only simple formulae, which can include each experiment only once, and whose connections are only “and”s and “or”s. In general, we may want to express more complicated interrelationships of the experiments; e.g., the XOR of $m$ experiments, or “at least 3 of 5 specified experiments”. Second, and-or trees only permit relatively simple precedence relationships; in general, we may want to specify that an experiment can only be performed if some complicated boolean combination of other experiments has succeeded or failed. Third, and-or trees use a restricted form of cost-function, in which the incremental cost of performing experiment $e$ can depend only on which other experiments have been performed. In general, we may want the cost to depend also on whether $e$, and/or various prior experiments, have been successful. There are also situations which require yet more complicated ways of computing the incremental cost of performing a particular experiment; see the extended paper [11].

To accommodate these extensions, we define a more general class of “decision structures”. A decision structure can involve an arbitrary set of experiments $W = \{e_i\}_{i=1}^n$, with general precedence constraints that can prevent an experiment from being performed until after certain other specified experiments have been performed with the specified (success or failure) result. The overall test result (e.g., whether the patient has hepatitis) can correspond to an arbitrary boolean combination of the successes and failures of any subset of these experi-
ments, and the costs of performing a sequence of experiments can be given by an arbitrary non-decreasing function. This leads to the “decision structures” defined below.

Notation: Given two sequences, \( \sigma = \langle \sigma_1, \ldots, \sigma_n \rangle \) and \( \tau = \langle \tau_1, \ldots, \tau_m \rangle \), let \( \sigma \cdot \tau \) refer to the sequence formed by concatenating \( \sigma \) and \( \tau \) — i.e. \( \sigma \cdot \tau = \langle \sigma_1, \ldots, \sigma_n, \tau_1, \ldots, \tau_m \rangle \). The definition is extended to the case where \( \sigma \) or \( \tau \) are single elements in the obvious manner.

A sequence \( \sigma \) is a subsequence of sequence \( \tau \), denoted \( \sigma \subseteq \tau \), if \( \sigma_i = \tau_{h(i)} \) for all \( i = 1..n \), for some monotonically increasing function \( h \). The empty sequence \( \langle \rangle \) is trivially a subsequence of any sequence \( \sigma \).

**Definition 1 (Decision Structures)** A decision structure is a four-tuple \( G = (W, F, R, c) \) where

- \( W = \{ e_1, \ldots, e_n \} \) is a set of experiments.
- \( F \subseteq [W \times \{ + , - \}]^* \times W \) is a precedence relation that specifies which further experiments can be performed given the results of a previous sequence of experiments. (E.g., \( F(\langle \rangle, e_1) \) means that \( e_1 \) may be performed initially; and \( F(\langle e_1 \rangle \langle e_3 \rangle, e_2) \) means that experiment \( e_2 \) may be performed after \( e_1 \) has been performed successfully, and then \( e_3 \) has been performed but was unsuccessful.)

The following conditions use the notion of a legal labeled experiment sequence (abbreviated “lles”). This is a sequence of the form \( \langle \langle e_1 \pm_1 \rangle, \ldots, \langle e_k \pm_k \rangle \rangle \), where each \( e_i \in W \), each \( \pm_i \in \{ + , - \} \), and no \( e \in W \) appears more than once. Furthermore, the sequence must satisfy the precedence constraints specified by the \( F \) relation: a sequence \( \ell = \langle \langle e_1 \pm_1 \rangle, \ldots, \langle e_k \pm_k \rangle \rangle \) is a lles only if \( F(\langle e_1 \pm_1 \rangle, \ldots, \langle e_{m-1} \pm_{m-1} \rangle), e_m \) holds for all \( m = 1..k \). The collection of all such sequences is denoted \( \mathcal{LLES}(G) \).

- \( R: \mathcal{LLES}(G) \to \{ S, F, U \} \) is the result function that specifies whether a given legal labeled experiment sequence renders the overall test successful or not; i.e. \( R \) maps each lles to one of \( \{ S, F, U \} \) (for Success, Failure, and Undecided). We require \( R \) to be monotonic, in the sense that \( R(\sigma) = S \Rightarrow R(\sigma \cdot \tau) = S \) and \( R(\sigma) = F \Rightarrow R(\sigma \cdot \tau) = F \) whenever \( \sigma \) and \( \sigma \cdot \tau \) are lles.

- \( c: \mathcal{LLES}(G) \to \mathbb{R}_+^* \) is the cost function that maps each lles to its nonnegative real cost. It is required to be non-decreasing: \( c(\sigma \cdot \tau) \geq c(\sigma) \) whenever \( \sigma \) and \( \sigma \cdot \tau \) are lles.

We let \( DS \) refer to the class of all such decision structures.

To illustrate these definitions: the diagnostic tree of Figure 1 can be encoded as a structure \( G_1 = \{ \langle e_1, \ldots, e_6 \rangle, F_1, R_1, e_1 \} \), where e.g. \( e_3 \) corresponds to “a patient’s blood reacts with Serum A” and \( e_5 \) corresponds to “a patient’s liver has a positive cytological test”. The \( F_1 \) relation, encoding the precedence relationships, includes \( F_1(\langle \rangle, e_1) \) and \( F_1(\langle \rangle, e_2) \) to mean that \( e_1 \) or \( e_2 \) can be performed initially; and \( F_1(\langle e_1 \rangle), e_3 \) but not \( F_1(\langle e_1 \rangle), e_3 \) (resp., \( F_1(\langle e_1 \rangle), e_4 \) but not \( F_1(\langle e_1 \rangle), e_4 \) ) to indicate that \( e_3 \) (resp., \( e_4 \)) can be performed if and only if \( e_1 \) has already succeeded. The \( R_1 \) function includes \( R_1(\langle e_1 \rangle \langle e_3 \rangle) = S, R_1(\langle e_1 \rangle \langle e_4 \rangle) = S \), \( R_1(\langle e_2 \rangle \langle e_5 \rangle \langle e_6 \rangle) = S \) and so
forth, as well as \( R_1(\langle e_1 - \rangle \langle e_2 - \rangle) = \mathcal{F} \), and \( R_1(\langle e_1 - \rangle \langle e_2 + \rangle) = \mathcal{U} \), etc. The cost function \( c \) encodes the incremental cost of performing any sequence of experiments: for instance, 

\[
c_1(\langle e_1 + \rangle, \langle e_3 + \rangle, \langle e_4 - \rangle) = 1+2+1 = 4 \text{ and } c_1(\langle e_1 + \rangle \langle e_2 + \rangle \langle e_6 + \rangle) = 1+2+2+1 = 6.
\]

Notice that for standard graph-like decision structures, whenever an example becomes reachable, it stays so until performed, i.e. \( F(\sigma, e) \) implies \( F(\tau, e) \) whenever \( \sigma \) is a subsequence of \( \tau \) and \( \tau \) is a test that does not include experiment \( e \). For instance, both conditions \( F_1(\langle e_1 - \rangle, e_2) \) and \( F_1(\langle e_1 + \rangle, e_2) \) follow from the condition \( F_1(\langle \rangle, e_2) \). When there is a unique minimal reachability condition for every experiment, we say that the structure is “tree-like”. Formally, we define:

**Definition 2 (Tree-like Decision Structures)** A decision structure \( G = \langle W, F, R, c \rangle \) is tree-like if we can identify with each experiment \( e \in W \) a single minimal lles, denoted \( \text{path}(e) \), that encodes the necessary and sufficient conditions for reaching \( e \); i.e.

\[
\forall e \in W. \exists \text{path}(e) \in \mathcal{LLES}(G). \forall \sigma \in \mathcal{LLES}(G). \quad F(\sigma, e) \iff [\text{path}(e) \subseteq \sigma].
\]

The class of all tree-like decision structures is denoted TDS.

When the structure \( G \) represents an and-or decision tree, the lles \( \text{path}(e) \) corresponds to the unique path leading to the experiment \( e \) in the tree: e.g., as the path to \( e_4 \) in \( G_1 \) goes through \( e_0 \) and \( e_1 \), \( \text{path}(e_4) = \langle \langle e_0 + \rangle \langle e_1 + \rangle \rangle \).

### 2.2 Satisfying Search Strategies

A “search strategy” for a satisficing search problem specifies the order of traversal through the associated decision structure — in the sample application \( G_1 \) of Figure 1, it tells the doctor when to perform which tests to determine whether the patient has hepatitis.

A strategy can be represented as a binary tree; for example the tree shown on the right side of Figure 2 represents one possible strategy \( \Theta_1 \) for the decision structure \( G_1 \). Each internal node in the strategy tree is labeled with an experiment that corresponds to some node in the decision structure. The strategy specifies the sequence of experiments to be performed in any given situation. For example, the \( \Theta_1 \) strategy first performs the experiment \( e_1 \) associated with \( \Theta_1 \)'s root \( u_1 \). If \( e_1 \) succeeds, \( \Theta_1 \) then follows the -labeled arc to the strategy sub-tree rooted in the \( e_3 \)-labeled node, and performs \( e_3 \). If that test succeeds, \( \Theta_1 \) advances up to the S-labeled node, signifying that \( \Theta_1 \) terminates with success. Alternatively, if \( e_3 \) fails, \( \Theta_1 \) then follows the +labeled arc, descending to the tree rooted in the \( e_4 \)-labeled node, then performs \( e_4 \), and so forth. A general definition of this process is as follows:

**Definition 3 (Search Strategies)** A strategy for a decision structure \( G = \langle W, F, R, c \rangle \) is a node- and arc-labeled binary tree \( \Theta = \langle N, A, I_N, I_A \rangle \), where \( N \) is the set of nodes and \( A \subseteq N \times N \) is the set of arcs connecting nodes to their descendants. The node-labeling \( I_N \) maps each internal node \( n \in N \) in the tree to an experiment \( e \in W \), and each leaf node to
either $S$ or $F$. The arc-labeling $l_A$ maps each arc $a \in A$ to either $+$ or $-$. Each internal node must have exactly two descending arcs, one labeled $+$ and the other $-$.

A path $\pi$ in $\Theta$ is an alternating sequence of nodes and arcs leading from the root of the tree to a leaf — i.e., a sequence of the form $\pi = \langle n_1, a_{1,2}, n_2, a_{2,3}, \ldots, a_{k-1,k}, n_k \rangle$ where each $n_i \in N$, and each $a_{i,i+1} = \langle n_i, n_{i+1} \rangle \in A$. Each such path has an associated labeled experiment sequence $l(\pi) = \langle \langle l_N(n_1) \rangle A(a_{1,2}), \ldots, \langle l_N(n_{k-1}) \rangle A(a_{k-1,k}) \rangle$. For $\Theta$ to be a proper strategy for $G$, the following conditions must be fulfilled by each path $\pi = \langle n_1, \ldots, n_k \rangle$ in $\Theta$:

1. $l(\pi) \in \mathcal{LES}(G)$
   (i.e. $l(\pi)$ must be a legal labeled experiment sequence);

2. $l_N(n_k) = R(l(\pi)) \in \{S,F\}$
   (i.e. the label of the final node must be either “success” or “failure”); and

3. $R(l_{1,j}(\pi)) = U$ for all $j < k$, where $l_{1,j}(\pi) = \langle \langle l_N(n_1) \rangle A(a_{1,2}), \ldots, \langle l_N(n_{j}) \rangle A(a_{j,j+1}) \rangle$ are the first $j$ elements of $l(\pi)$
   (i.e. no proper prefix of a path can be conclusive).

We let $\text{path}(\Theta)$ refer to the set of all such proper paths in the strategy $\Theta$. We also let $\mathcal{SS}(G)$ refer to the set of all strategies defined for the decision structure $G$, and $\mathcal{SS}(\mathcal{DS}) = \{ \mathcal{SS}(G) | G \in \mathcal{DS} \}$ refer to the class of all strategies for all decision structures.

For an illustration of these notions, see the strategy tree $\Theta_1$ shown on the right side of Figure 2. There are 10 paths in $\Theta_1$, one corresponding to each leaf node (indicated by the letters $S$ and $F$ in the figure). For instance, one such path is $\pi_{16} = \langle u_1, a_{1,13}, u_{13}, a_{13,15}, u_{15}, a_{15,16}, u_{16} \rangle$, for which the corresponding lles is $l(\pi_{16}) = \langle \langle e_1 - \rangle, \langle e_{13} + \rangle, \langle e_{15} - \rangle \rangle$.
2.3 Optimal Strategies

We wish to identify the best strategy for traversing a given decision structure, i.e. the strategy whose expected cost is minimal. As this depends on the success probabilities of the individual experiments, different strategies will be optimal for different distributions. To state this more precisely, we define:

**Definition 4 (Expected Cost of a Strategy)** Let $\Theta$ be a strategy for the decision structure $G = (W, F, R, c)$, and $p: W \rightarrow [0, 1]$ a distribution function that maps each experiment to its success probability. The (expected) cost of strategy $\Theta$ relative to the distribution $p$, denoted $C_p(\Theta)$, is defined as the sum of the cost of each path in the strategy, weighted by its probability, i.e.

$$C_p(\Theta) = \sum_{\pi \in \text{path}(\Theta)} p(l(\pi)) \times c(l(\pi)).$$

Here the probability of a path $\pi$ is defined as $p(l(\pi)) = \prod_{(e_i, \pm_1) \in \pi} p^{\pm_1}(e_i)$, where $p^\pm(e)$ is $p(e)$ if $\pm$ equals $+$, and $1 - p(e)$ if $\pm$ equals $-$.

**Definition 5 (Optimizing Functions)** An optimizing function for a class of decision structures $\mathcal{D} \subseteq DS$ is a function $\text{OSS}$ that maps any decision structure $G = (W, F, R, c) \in \mathcal{D}$, together with a distribution $p \in [0, 1]^W$, to a strategy in $SS(G)$ whose cost is minimal. That is,

$$\forall G \in \mathcal{D}. \ \forall p \in [0, 1]^W. \ \forall \Theta \in SS(G). \ C_p(\text{OSS}(G, p)) \leq C_p(\Theta).$$

For brevity, we often denote by $\Theta_p$ the optimal strategy $\text{OSS}(G, p)$ provided by the optimizing function for a given distribution $p$.

While these definitions assume that the experiments are independent of each other, both these definitions and the theorems below could be extended to handle more complicated situations.

Since we are only dealing with finite decision structures, optimal strategies can always be found by exhaustive search. Of course, exhaustive search is in general impractical, and if we are dealing with decision structures with concise encodings, such as and-or trees, the optimal strategies may not even have polynomial-size representations.

Nevertheless, optimal strategies can be determined in polynomial time in many interesting special cases. Garey [6] provided an algorithm for finding the optimal search strategy when the constraints can be represented as a regular “or-tree” (i.e. no conjunctive subgoals and no multiple predecessors are allowed; cf. also [22]). Simon and Kadane [21] later extended this algorithm to deal with directed acyclic graphs in the special case where success at any intermediate node implies global success. (In dag’s where global success requires reaching a specified goal node, the problem is NP-hard [8].) It is currently not known whether optimal strategies can be found in polynomial time for and-or trees. Some partial results on this question exist: for instance, Natarajan [18] presents an efficient algorithm for finding optimal “depth-first” search strategies in this case, and Smith [22] provides an algorithm for finding optimal “serial strategies”. In the more general case of and-or dag’s, the problem is NP-hard even when all the success probabilities are 1 [20].
3 The pao Algorithm

Each of the abovementioned optimization algorithms assumes that the precise success probabilities of the experiments are known, which of course is not the case in most real-life situations. The best one can do then is to estimate these probabilities by observing a set of trials of the experiments, and then use these estimates to compute a near-optimal strategy. A potential pitfall in this approach, however, is that the strategies computed by any of the above algorithms are very sensitive to errors in the probability estimates: small changes in the estimates may lead to drastically different strategies. Fortunately, even though the choice of the actual strategy is very sensitive to estimation errors, the cost of the strategy obtained is not. This realization is one of the main contributions of this paper, as it means that we can use our estimates to obtain a near-optimal strategy.

Below, we describe an algorithm pao that can be used in conjunction with any optimizing function OSS. Section 3.1 first formally defines the task of finding approximately optimal strategies and outlines the algorithm. The following sections then discuss the technical issues in more detail. First, in Section 3.2 we compute the sample complexity of this task: how many samples of each experiment are needed to guarantee, with a high level of confidence, that a strategy based on the resulting estimates will be close to optimal.

Section 3.3 then addresses a second problem: guaranteeing that the pao algorithm will be able to obtain a sufficient number of samples of each experiment. The main complication arises from the precedence constraints. For example, in the context of our diagnostic example (Figure 1), the sample complexity analysis may suggest that the doctor needs to obtain 100 samples of the test “CytologicalTest(liver)”. This is impossible if he is never able to perform a biopsy on any patients; i.e. if the experiment $e_2$ in structure $G_1$ never succeeds. In Section 3.3 we provide a solution to this problem for general “tree-like” decision structures, but also observe that the task is intractable in general.

3.1 The pao Task

A pao problem instance consists of: a decision structure $G = (W, F, R, \epsilon) \in \mathcal{DS}$; a bound on the allowed excess $\epsilon \in \mathbb{R}^+$; and the required confidence $\delta \in (0,1]$. The algorithm also uses an oracle $O$ that produces samples drawn at random from some fixed but unknown distribution.

For each instance, the pao algorithm returns a strategy $\Theta_{pao} \in \mathcal{SS}(G)$, whose expected cost is, with high probability, close to optimal. Stated more precisely, let $\Theta_p = OSS(G, p)$ be the optimal strategy for a given true distribution $p$. Then with probability at least $1 - \delta$, the cost of the strategy $\Theta_{pao}$ is no more than $\epsilon$ higher than the cost of this optimal strategy, i.e.

$$Pr \left[ C_p(\Theta_{pao}) \leq C_p(\Theta_p) + \epsilon \right] \geq 1 - \delta .$$

We split the pao task into two subtasks: subroutine GS, which gathers the relevant statistics, and OSS, which uses those statistics to produce an appropriate strategy; see Figure 3.

The GS subroutine takes as input the decision structure $G$ and the parameters $\epsilon$ and $\delta$; it computes how many samples are required, and makes the specified number of calls to the
Algorithm pao($G; DS$, $\epsilon; \mathbb{R}^+$, $\delta; (0, 1]$)
\[
p \leftarrow \text{GS}(G, \epsilon, \delta)
\]

/* GS may call oracle $O$ a polynomial number of times */
\[
\hat{\Theta} \leftarrow \text{OSS}(G, \hat{p})
\]
Return $\hat{\Theta}$
End pao

Figure 3: Outline of the pao Algorithm

oracle $O$ (specified below) to obtain them. The subroutine produces a vector of probability estimates, $\hat{p} = \langle \hat{p}_1, \ldots, \hat{p}_n \rangle$, where each $\hat{p}_i$ is the estimate for the success probability of the $i^{th}$ experiment $e_i \in W$. (To simplify our description, we are assuming that we do not know a priori the success probabilities of any of the experiments. If we happen to know some of the values, we can simply use those values directly, and not bother with the estimation.) The pao algorithm then concludes by running an appropriate optimizing function OSS on these estimated probabilities $\hat{p}$, instead of the unknown true values. We concentrate here on the sample-gathering part of the pao algorithm, GS; for the OSS functions, we rely on the ones provided by earlier researchers.

3.2 Sample Complexity

We first analyze the sample complexity of the pao task in the simple case where we can always access to an oracle $O$ that will, upon request, produce a sample $\kappa_j$ from the population, together with its complete labeling $\mathcal{L}(\kappa_j) = \langle \ell_1^j, \ldots, \ell_n^j \rangle$, where $\ell_i^j$ is 1 if $\kappa_j$ passes experiment $e_i \in W$, and 0 otherwise. The GS routine performs a number $M$ (specified below) calls to this oracle, and returns a vector of probability estimates $\hat{p} = \langle \hat{p}_1, \ldots, \hat{p}_n \rangle$, where each $\hat{p}_i = \frac{1}{M} \sum_{j=1}^M \ell_i^j$; the OSS optimizing function then uses these values. We prove (Corollary 1 in the Appendix) that the cost of the strategy $\Theta_{pao} = \text{OSS}(G, \hat{p})$ is within $\epsilon$ of the optimal, with reliability at least $1 - \delta$, whenever

\[
M = \left\lceil 2 \left( \frac{nC}{\epsilon} \right)^2 \ln \frac{2n}{\delta} \right\rceil,
\]

where $C$ is the worst-case cost of performing any sequence of experiments in $W$.

In fact, we can improve on the constant $C$ somewhat. Let us denote by $D(\epsilon)$ the maximal cost of any concluding sequence of experiments beginning with experiment $e$. Formally:

**Definition 6** Let $G = \langle W, F, R, c \rangle$ be a decision structure. For each experiment $e \in W$ we define

\[
D(\epsilon) = \max\{ c(\alpha \cdot \langle e \pm \beta \rangle - c(\alpha) | \alpha \cdot \langle e \pm \beta \rangle \in \mathcal{LLES}(G) \}.
\]

(The $\pm$ above indicates that the max should range over both $+$ and $-$ values.)
We can then let $C = \max_{e \in W} \{ D(e) \}$ be the maximum remainder cost starting with any experiment $e \in W$. These $D(e)$ values are quite easy to compute when the underlying decision structure is an and-or tree: Here, $D(e) = C_{tot} - c(\text{path}(e))$, where $C_{tot}$ is the sum of the costs of all of the tree’s arcs and $c(\text{path}(e))$ is the cost of path $(e)$, the unique path in the and-or tree that leads from the root to experiment $e$; see Definition 2. (For instance, in the tree $G_1$ of Figure 2 we have $C_{tot} = 10$ and $c(\text{path}(e_3)) = 2$, so $D(e_3) = 10 - 2 = 8$.)

The sample complexity bound (1) is derived in the Appendix as Corollary 1 of a more general result that also takes into account the difficulty of labeling the samples (i.e., performing the experiments; see below). To very briefly outline the proof for this simple case: we first prove that after $M$ samples, we are at least $1 - \delta$ confident that each probability estimate $\hat{p}_i$ is within $\epsilon/2nC$ of the correct value $p_i$; we then show, based directly on the definition of the cost of a strategy and independent of which optimizing function is used, that this precision of the probability estimates suffices to guarantee that the cost of the obtained strategy is within $\epsilon$ of the optimal, i.e. that

$$\left[ \forall i \left| p_i - \hat{p}_i \right| \leq \frac{\epsilon}{2nC} \right] \Rightarrow \left| C_p(\Theta_p) - C_p(\Theta_p) \right| \leq \epsilon.$$

### 3.3 “Learning While Doing” in Tree-like Structures

The simple pao algorithm presented above assumes that the oracle $O$ produces a complete labeling for each sample, i.e. it returns a complete vector $L(\kappa) = \{\ell_1, \ldots, \ell_n\} \in \{0, 1\}^n$ on each query. In practical situations, however, such an oracle will typically not be available. Instead, the learning system must collect the statistics it needs (i.e. the individual component $\ell_i$ values of $L(\kappa)$) while watching a performance element perform its task, over a sufficiently large set of samples. In the context of our diagnostic example, the learning module would observe the doctor as he examines patients, recording how many of these patients pass the various tests. After gathering enough information, the learner would compute the approximately optimal strategy $\Theta_{pro}$, instruct the doctor to use this $\Theta_{pro}$ strategy, and terminate itself.\(^1\) We view this as a “learning while doing” protocol [16], as the overall system is performing useful work during the learning phase (here, examining patients).

From now on, we assume our oracle $O$, when queried, provides only an unlabeled sample (e.g., a patient $\kappa$), rather than the full labelings of that sample, $L(\kappa)$. In order to determine the value of any label $\ell_i$ on sample $\kappa$, the GS subroutine must then actually “reach” and perform experiment $e_i$ on $\kappa$.

Computing these $\ell_i$ values is problematic when there are intermediate experiments: For instance, in the case of our decision structure $G_1$, the doctor can not immediately determine whether a patient’s blood will react to serum $A$; he must first be able to draw blood from the patient. Hence, our learning system will be unable to estimate the probability of event “blood reacts to serum $A$” if the doctor is never able to extract blood (i.e. if $\Pr[\text{Draw blood}] = 0$).

\(^1\)We are still considering only “one-shot learning”, in which the learner sets the strategy only once, after the learning phase. We are not considering ways of modifying the strategy gradually over time to become incrementally better; but see [9]. Also, this issue differs from the “Exploration-Exploitation” trade-off discussed in the context of the Bandit problem (cf. [3, 17]) as we are not concerned with minimizing the cumulative cost of the learning and performance systems together, over an infinite sequence of samples.
Fortunately, there is a way around this problem. The critical observation is the following: Let $\rho(e_i)$ be the probability of “reaching” an experiment $e_i$ during the execution of a strategy. (This notion is defined formally below.) If $\rho(e_i)$ is very small, we will be unlikely to reach $e_i$ and hence to obtain samples of this experiment. However, the smaller the value of $\rho(e_i)$, the less sensitive the cost of the optimal strategy is to the value of the success probability $p(e_i)$, which means that we also need fewer samples of $e_i$. In the limit, if there is no chance of reaching $e_i$ (i.e. $\rho(e_i) = 0$), then we will also need no samples of it (i.e. OSS can produce an optimal strategy even if $|\hat{p}_i - p_i| = 1$).

**Definition 7** Let $G = \langle W, F, R, e \rangle \in \mathcal{DS}$ be a decision structure, and $p : W \rightarrow [0,1]$ a distribution function that maps each experiment to its success probability. For any strategy $\Theta = \langle N, A, l_N, l_A \rangle \in \mathcal{SS}(G)$, and any experiment $e \in W$, let $\rho(e, \Theta)$ be the probability that $\Theta$ will reach $e$, i.e.

$$\rho(e, \Theta) = \sum_{n: l_N(n) = e} p(l(\pi(n))),$$

where the sum is over nodes $n$ in the strategy $\Theta$ labeled with $e$, $\pi(n)$ is the path in $\Theta$ that leads to $n$, and the probability of this path $p(l(\pi(n)))$ is as defined above in Definition 4. Finally, let $\rho(e) = \max\{\rho(e, \Theta) | \Theta \in \mathcal{SS}(G)\}$.

The formula for $\rho(e)$ reduces to a particularly simple form when the decision structure $G$ is tree-like. In this case $\rho(e) = \prod_{i=1}^{k} p^{|\pm(i)|}$, where $\text{path}(e) = \langle \{f_{1, \pm 1}\}, \ldots, \{f_{k, \pm k}\} \rangle$ is the unique path that leads to $e$ in $G$. For instance, in the $G_1$ decision structure of Figure 1, we have $\text{path}(e_4) = \langle \{e_0, +\}, \{e_1, +\} \rangle$, and so $\rho(e_4) = p(e_0) \times p(e_1)$.

Now let $\Theta_p = \text{OSS}(G, p)$ be the actual optimal strategy based on the unknown correct probability vector $p = \langle p_1, \ldots, p_n \rangle$, and $\Theta_{\hat{p}} = \text{OSS}(G, \hat{p})$ be the strategy that our pao algorithm will produce, based on the estimates the GS subroutine has obtained, $\hat{p} = \langle \hat{p}_1, \ldots, \hat{p}_n \rangle$. We wish to bound the cost difference $C_p(\Theta_{\hat{p}}) - C_p(\Theta_p)$. The following lemma shows that, in place of obtaining precise estimates of the probabilities $p_i$, we need only ensure that the product $\rho(e_i) \times |p_i - \hat{p}_i|$ is small for each $e_i$.

**Lemma 1** Let $G = \langle W, F, R, e \rangle$ be a decision structure with $|W| = n$ experiments. Let $p = \langle p_1, \ldots, p_n \rangle$ be a vector of success probabilities for $W$, and $\hat{p} = \langle \hat{p}_1, \ldots, \hat{p}_n \rangle$ a vector of their estimates. Let the optimal search strategy for $G$ w.r.t. $p$ be $\Theta_p = \text{OSS}(G, p)$, and let $\Theta_{\hat{p}} = \text{OSS}(G, \hat{p})$ be the strategy based on the estimated probabilities. Then

$$C_p(\Theta_{\hat{p}}) - C_p(\Theta_p) \leq 2 \sum_{i=1}^{n} D(e_i) \times \rho(e_i) \times |p_i - \hat{p}_i| .$$

A further complication now arises from the fact that the $\rho(e_i)$ values actually depend on the unknown true distribution $p$. Fortunately, we can also approximate these values as we are obtaining the estimates of the $p_i$’s. In essence, we need only “aim for $e_i$” a certain number of times: each time we reach $e_i$, we improve our estimate of $p_i$ (i.e. reduce the $|\hat{p}_i - p_i|$)

\footnote{The Appendix contains the proofs for all lemmata, theorems and corollaries presented in the text.}
"error bars") and each time our path to $e_i$ is blocked, we can, with confidence, reduce the value of $\rho(e_i)$.

The rest of this subsection first shows how to estimate the products $\rho(e_i) \times |p_i - \hat{p}_i|$ in tree-like decision structures, then discusses the difficulties in computing near-optimal strategies in more general structures.

**Dealing with Tree-like Decision Structures:** Given an experiment $e$ in a tree-like decision structure $G$, recall that $\text{path}(e) = \{e_1 \pm_1, e_2 \pm_2, \ldots, e_k \pm_k\}$ is the unique minimal path that determines when $e$ can be performed. We say that a strategy $\Theta \in SS(G)$ is a direct strategy for $e$ if it contains this path as an initial segment, in the sense that the root of $\Theta$ is labeled with the experiment $e_1$, and its $\pm_1$-labeled arc (i.e. the $+$-labeled arc if $\pm_1$ equals $+$, and the $-$-labeled arc if $\pm_1$ equals $-$) descends to a node labeled with the experiment $e_2$, and the $\pm_2$-labeled arc from that node descends to a node labeled with $e_3$, and so on, down to a node labeled $e_k$, whose $\pm_k$-labeled arc leads to a node labeled with $e$. We denote the class of direct strategies for an experiment $e$ by $SS(e)$. As an example, the strategy $\Theta_1$ shown in Figure 2 goes directly to $e_1$ and hence $\Theta_1 \in SS(e_1)$; it also contains the direct route to $e_3$ as an initial segment, and hence $\Theta_1 \in SS(e_3)$. On the other hand, the strategy "digresses" to consider $e_3$ before $e_4$, and so $\Theta_2 \notin SS(e_4)$; similarly $\Theta_1 \notin SS(e_2)$ as $\Theta_1$ considers $e_1$ before $e_2$.

The GS algorithm shown in Figure 4 can deal with any tree-like decision structure $G$. The algorithm first identifies a direct strategy $\Theta_e \in SS(e)$ for each $e \in W$. (There can in general be many such strategies, performing different experiments outside their common initial path to $e$; this paper does not consider how to choose between the alternatives. Nor does it consider the cost of identifying any of these strategies, except to observe that for e.g. and-or trees they can be constructed quite efficiently, directly from the tree structure.) After selecting this set of strategies, GS associates three counters with each experiment, $\text{tot}(e_i)$, $\text{suc}(e_i)$ and $m(e_i)$, which will record, respectively, the number of times experiment $e_i$ has been performed, the number of times $e_i$ succeeded, and the number of attempts that remain to be performed. As it processes the instances, GS updates each of these counters by: incrementing $\text{tot}(e_i)$ each time GS performs experiment $e_i$; incrementing $\text{suc}(e_i)$ each time the experiment $e_i$ succeeds; and decrementing $m(e_i)$ each time GS has attempted to reach experiment $e_i$ either by performing $e_i$, or by using the strategy $\Theta_e$, but failing to reach $e_i$.

The remaining challenge is to identify when to use which strategy. (Clearly GS will not, in general, be able to observe enough trials of the different experiments if it uses the same strategy throughout.) On each sample, GS first identifies the needy experiments, i.e. those $e$'s for which $m(e) > 0$. If there are none, then GS has collected enough samples, and so can terminate, passing the obtained vector of estimates $\hat{p}$ to the OSS algorithm. Otherwise, GS selects one of the needy experiments $e$, and executes the associated strategy $\Theta_e$.

Notice that GS decrements at least one $m(e)$ counter on each sample, viz. the one associated with the experiment $e$ to which it is currently aiming. Hence, after at most $\sum_{e \in W} m_0(e)$ samples (where the $m_0(e)$ are the initial values of the counters), all of the $m(e)$ counters will be zero and GS will terminate; it therefore requires only a polynomial number of samples. (The algorithm may of course use far fewer samples, as most $\Theta_e$ strategies will reduce the $m(e_j)$ values for several different experiments $e_j$. GS can also be changed to decrease the counters of all experiments $e'$ that are deemed unreachable in the process of following $\Theta_e$.)
Algorithm GS( $G: TDS$, $e: \mathbb{R}^+, \delta: (0,1]$ )
ForEach $e \in W$ do
  Find some $\Theta_e \in SS(e)$
  $tot(e) \leftarrow 0$
  $suc(e) \leftarrow 0$
  $m(e) \leftarrow \begin{cases} 
  2 \left(1 + \frac{\epsilon}{n D(e)}\right) \left(\frac{n D(e)}{\epsilon}\right)^2 \ln \frac{4n}{\delta} & \text{if} \quad F(\langle \rangle, e) \\
  2 \left(\frac{n D(e)}{\epsilon}\right)^2 \ln \frac{2n}{\delta} & \text{if} \quad \neg F(\langle \rangle, e) 
  \end{cases}$
End ForEach
While $\exists e$ such that $m(e) > 0$ do
  Get sample $\kappa$ from oracle $\mathcal{O}$
  Execute strategy $\Theta_e$ on sample $\kappa$
  After performing each experiment $e_j$:
    $tot(e_j) \leftarrow tot(e_j) + 1$
    $m(e_j) \leftarrow m(e_j) - 1$
    If $e_j$ succeeds: $suc(e_j) \leftarrow suc(e_j) + 1$
    If $e_j$’s result (success or failure) means $e$ cannot be reached: $m(e) \leftarrow m(e) - 1$
End While
ForEach $e_i \in W$ do
  $\hat{p}(e_i) \leftarrow \begin{cases} 
  \frac{suc(e_i)}{tot(e_i)} & \text{if} \quad tot(e_i) > 0 \\
  \frac{1}{2} & \text{otherwise}
  \end{cases}$
End ForEach
Return $\hat{p} = \langle \hat{p}(e_1), \ldots, \hat{p}(e_n) \rangle$
End GS

Figure 4: A GS Algorithm for Tree-like Decision Structures
The following lemma characterizes the behavior of the algorithm:

**Lemma 2** Let \( G = (W, F, R, c) \) be a tree-like decision structure with \(|W| = n\) experiments, and let \( p = (p_1, \ldots, p_n) \) be a vector of success probabilities for the experiments. Furthermore, let \( \epsilon, \delta > 0 \) be any given constants, and let \( \hat{p} = (\hat{p}_1, \ldots, \hat{p}_n) \) be a vector of probability estimates computed by the GS algorithm of Figure 4. Then

\[
\forall e_i \in W: \Pr \left[ D(e_i) \times \rho(e_i) \times |p_i - \hat{p}_i| \geq \frac{\epsilon}{2n} \right] \leq \frac{\delta}{n}.
\]

(While our analysis uses the \( \rho(e_i) \) values, notice that the GS algorithm never actually computes them.) Combining the results of Lemmas 1 and 2, we obtain the following theorem:

**Theorem 1** Let \( G = (W, F, R, c) \) be a tree-like decision structure with \(|W| = n\) experiments, and let \( p = (p_1, \ldots, p_n) \) be a vector of success probabilities for the experiments. Furthermore, let \( \epsilon, \delta > 0 \) be any given constants, and let \( \Theta_{\text{pa}} = \text{pa}(G, e, \delta) \) be the strategy produced by the \text{pa}(\text{a}) algorithm using the GS subroutine of Figure 4. Then, with probability at least \( 1 - \delta \), \( C_p(\Theta_{\text{pa}}) - C_p(\Theta_p) \leq \epsilon \), where \( \Theta_p = \text{OSS}(G, p) \) is the optimal strategy for probability vector \( p \).

**Beyond Tree-like Decision Structures:** While the specific GS algorithm presented above applies only to tree-like decision structures, there can be other related algorithms that can learn strategies for other decision structures. The main challenge is in estimating \( \rho(e_i) \), as required to bound the product \( \rho(e_i) \times |p_i - \hat{p}_i| \), which is complicated by the fact that there can be many distinct ways of reaching an experiment in a general decision structure.

To address this task, recall from Definition 7 that \( \rho(e) \) is the maximum probability of reaching the experiment \( e \), where the maximum is taken over all possible strategies. We can always approximate this value by first estimating \( \rho(e, \Theta) \) for every possible strategy \( \Theta \), and then taking the maximum of these values: If each estimate \( \hat{\rho}(e, \Theta) \) is within \( \epsilon \) of \( \rho(e, \Theta) \) with probability at least \( 1 - \delta/|SS(G)| \), then the value \( \rho(e) = \max\{\hat{\rho}(e, \Theta) \mid \Theta \in SS(G)\} \) will be within \( \epsilon \) of \( \rho(e) = \max\{\rho(e, \Theta) \mid \Theta \in SS(G)\} \) with probability at least \( 1 - \delta \). Even though the number of strategies in \( SS(G) \) for a given decision structure \( G \) can be exponential in the size of \( G \), there can be ways of exploiting the structure of \( G \), and hence of \( SS(G) \), to limit the number of \( \rho(e, \Theta) \) values that need to be considered. From this point of view, the GS algorithm for tree-like structures is based on the observation that for any tree-like structure \( G \), the direct strategies \( \Theta_e \in SS(e) \) necessarily yield the largest values of \( \rho(e, \Theta_e) \) for any experiment \( e \). In fact, one can use the same GS algorithm whenever it is possible to identify each experiment \( e \) with a strategy \( \Theta_e \) for which \( \rho(e, \Theta_e) = \rho(e) \).

This is not always straightforward. The extended paper [11] includes an algorithm that uses dynamic programming techniques to sequentially estimate the probabilities of each “layer” of certain types of decision structures. Unfortunately, the following general result shows that the computational complexity of any such algorithm is likely to be exponential in the number of experiments.

**Theorem 2** Assume \( RP \neq \text{NP} \). (\( RP \) is the class of problems solvable by probabilistic polynomial time algorithms with one-way error, cf. [12].) Then there is no probabilistic polynomial time algorithm, and consequently no deterministic polynomial time algorithm that,
given a decision structure $G = (W, F, R, c)$, an experiment $e \in W$, a distribution function $p: W \rightarrow [0, 1]$, and parameters $\epsilon, \delta > 0$, can estimate the value $p(e)$ to within $\epsilon$ with probability at least $1 - \delta$.

4 Conclusion

The results presented in this paper have been motivated by, and extend, various other lines of research. The underlying objective of finding a provably good search strategy comes from the work on optimal satisficing search strategies [6, 21, 2, 18, 22]. Each of these earlier papers considered some specifically defined class of search structures and, moreover, required the user to supply precise success probability values for the experiments. Our work extends this body of research in three ways. First, we have defined a general framework of “decision structures” and “search strategies”, which encompasses and generalizes the models used before. Second, we have analyzed, in this very general setting, the sensitivity of optimal search strategies to errors in the probability estimates. Third, we have provided an efficient algorithm for finding good estimates of the probability values in the case of tree-like decision structures, and proved that (unless $RP = NP$) there can be no efficient algorithm for this task for general structures.

Our approach also resembles the work on speed-up learning (including both “explanation-based learning” [15, 5, 14] and “chunking” [13]), as it uses previous solutions to suggest a way of improving the speed of a performance system. Most speed-up learning systems, however, use only a single example to suggest an improvement; we extend those works by showing how to use a set of samples and by describing, furthermore, the exact number of samples required. Also, while most speed-up learning systems are based on purely heuristic considerations, we use mathematically sound techniques to guarantee that our new strategies will be close to optimal, with provably high probability.

Finally, this work derives many of its mathematical methods, as well as its title, from the field of “probably approximately correct learning” [23]. We hope to have enriched this field by providing an application of the PAC framework outside of its traditional setting of concept learning.
A  Proofs

This appendix contains the proofs of the results mentioned in the body of the paper.

Lemma 1 Let $G = \langle W, F, R, c \rangle$ be a decision structure with $|W| = n$ experiments. Let $p = \langle p_1, \ldots, p_n \rangle$ be a vector of success probabilities for $G$, and $\hat{p} = \langle \hat{p}_1, \ldots, \hat{p}_n \rangle$ a vector of their estimates. Let the optimal search strategy for $G$ w.r.t. $p$ be $\Theta_p = OSS(G, p)$, and let $\Theta_{\hat{p}} = OSS(G, \hat{p})$ be the strategy based on the estimated probabilities. Then

$$C_p(\Theta_{\hat{p}}) - C_p(\Theta_p) \leq 2 \sum_{i=1}^{n} D(e_i) \times \rho(e_i) \times |p_i - \hat{p}_i|.$$

Proof: Given the vectors $p$ and $\hat{p}$, let $p^{(i)}$ denote the vector $\langle \hat{p}_1, \ldots, \hat{p}_i, p_{i+1}, \ldots, p_n \rangle$, and as special cases, $p^{(0)} = p$ and $p^{(n)} = \hat{p}$. We shall prove below that for any strategy $\Theta$ for $G$, and for every $i = 0, \ldots, n - 1$,

$$|C_p^{(i)}(\Theta) - C_p^{(i+1)}(\Theta)| \leq D(e_i) \times \rho(e_i) \times |p_i - \hat{p}_i|,$$

which implies that

$$|C_p(\Theta) - C_p(\Theta)| \leq \sum_{i=1}^{n} |C_p^{(i)}(\Theta) - C_p^{(i+1)}(\Theta)| \leq \sum_{i=1}^{n} D(e_i) \times \rho(e_i) \times |\hat{p}_i - p_i|.$$

Applying this bound to the strategies $\Theta_p$ and $\Theta_{\hat{p}}$, and noting that by optimality $C_p(\Theta_{\hat{p}}) - C_p(\Theta_p) \leq 0$, then yields the desired result:

$$C_p(\Theta_{\hat{p}}) - C_p(\Theta_p) = \left[ C_p(\Theta_{\hat{p}}) - C_p(\Theta_p) \right] + \left[ C_p(\Theta_{\hat{p}}) - C_p(\Theta_p) \right] + \left[ C_p(\Theta_{\hat{p}}) - C_p(\Theta_p) \right]$$

$$\leq \left[ \sum_{i=1}^{n} D(e_i) \times \rho(e_i) \times |p_i - \hat{p}_i| \right] + 0 + \left[ \sum_{i=1}^{n} D(e_i) \times \rho(e_i) \times |\hat{p}_i - p_i| \right]$$

$$= 2 \sum_{i=1}^{n} D(e_i) \times \rho(e_i) \times |\hat{p}_i - p_i|.$$

In proving inequality (2), we shall make use of the following notation (cf. Figure 5): Given any node $u_j$ in a strategy $\Theta$, let $\pi_j$ denote the path leading from the root of $\Theta$ to $u_j$. For any such path $\pi_j$, we denote the associated cost $c( l(\pi_j) )$ briefly by $c(\pi_j)$. (Here, we have extended the $l(\cdot)$ function to partial sequences: $u_j$, the final entry in $\pi_j$, does not have to be a leaf node in the strategy tree.) We also extend the cost function to incomplete paths in the strategy tree by defining

$$c( \langle u_i, a_{i,i+1}, u_{i+1}, \ldots, u_k \rangle ) = c( \langle u_0, a_0, 1, u_1, \ldots, u_i, a_{i,i+1}, u_{i+1}, \ldots, u_k \rangle ) - c( \langle u_0, a_0, 1, u_1, \ldots, u_i \rangle ),$$

where $u_0$ is the root node of the strategy tree and $\langle u_0, a_0, 1, u_1, \ldots, a_{k-1,k}, u_k \rangle$ is any connected path through the tree; furthermore, for single nodes we define $c( u_i ) = c( \langle u_i \rangle )$. 


Let $u_j^+$ be the $+$-descendant of node $u_j$, and let $t(u_j)$ denote the set of leaf nodes below $u_j$. For each $u_l \in t(u_j)$, let $\pi_{jl}$ denote the path from $u_j$ to $u_l$, and if $u_l \in t(u_j^+)$, let $\pi_{jl}^+$ denote the path from node $u_j^+$ to $u_l$. Let $C_p(\Theta_j^+)$ denote the "expected cost of the $+$-subtree of $u_j$":

$$C_p(\Theta_j^+) = c(u_j^+) + \sum_{u_l \in t(u_j^+)} p(\pi_{jl}^+) c(\pi_{jl}^+).$$

Analogous definitions hold for $u_j^-$, $t(u_j^-)$, $p(\pi_{jl}^-)$, and $C_p(\Theta_j^-)$.

For a given experiment $e_i \in W$, let $N(e_i) = \{u_j\}_{j=1,...,k}$ be the set nodes in $\Theta$ labeled with $e_i$. Let $\Theta | e_i$ be the subtree within $\Theta$ consisting of the paths from the root of $\Theta$ through a node in $N(e_i)$ down to a leaf; and let $\Theta | \bar{e}_i$ be the subtree consisting of the other paths. Notice that $C_p(\Theta) = C_p(\Theta | e_i) + C_p(\Theta | \bar{e}_i)$. We may partition the paths in $\Theta^+$ according to which of the $u_j$ nodes each passes through (recall that in any strategy $\Theta$, an experiment $e$ can occur at most once on any path from the root to a leaf node), and use this representation to obtain a very simple formula expressing the influences of $e_i$'s success probability $p_i$ on the
function $C_p(\Theta|e_i)$ (and hence on $C_p(\Theta)$):

$$C_p(\Theta|e_i) = \sum_{\pi \in \text{path}(\Theta^+)} p(\pi) c(\pi) = \sum_{j=1}^{k} \sum_{u_i \in \{u_i^+\}} p(\pi_j) c(\pi_i)$$

$$= \sum_{j=1}^{k} \left[ \sum_{u_i \in \{u_i^+\}} p(\pi_j) c(\pi_i) + \sum_{u_i \in \{u_i^-\}} p(\pi_j) c(\pi_i) \right]$$

$$= \sum_{j=1}^{k} \left[ \sum_{u_i \in \{u_i^+\}} p(\pi_j) p_i P(\pi_{ji}^+) \left( c(\pi_j) + c(u_j^+) + c(\pi_{ji}^+) \right) \right]$$

$$+ \sum_{u_i \in \{u_i^-\}} p(\pi_j) \left( 1 - p_i \right) p(\pi_{ji}^-) \left( c(\pi_j) + c(u_j^-) + c(\pi_{ji}^-) \right)$$

$$= \sum_{j=1}^{k} p(\pi_j) \left[ p_i \sum_{u_i \in \{u_i^+\}} p(\pi_{ji}^+) \left( c(\pi_j) + c(u_j^+) + c(\pi_{ji}^+) \right) \right]$$

$$+ \left( 1 - p_i \right) \sum_{u_i \in \{u_i^-\}} p(\pi_{ji}^-) \left( c(\pi_j) + c(u_j^-) + c(\pi_{ji}^-) \right)$$

$$= \sum_{j=1}^{k} p(\pi_j) \left[ p_i \left( c(\pi_j) + c(u_j^+) \right) \sum_{u_i \in \{u_i^+\}} p(\pi_{ji}^+) + \sum_{j=1}^{k} p(\pi_{ji}^+) c(\pi_{ji}^+) \right]$$

$$+ \left( 1 - p_i \right) \left( c(\pi_j) + c(u_j^-) \right) \sum_{u_i \in \{u_i^-\}} p(\pi_{ji}^-) + \sum_{j=1}^{k} p(\pi_{ji}^-) c(\pi_{ji}^-)$$

$$= \sum_{j=1}^{k} p(\pi_j) \left[ p_i \left( c(\pi_j) + c(u_j^+) + \sum_{j=1}^{k} p(\pi_{ji}^+) c(\pi_{ji}^+) \right) \right]$$

$$+ \left( 1 - p_i \right) \left( c(\pi_j) + c(u_j^-) + \sum_{j=1}^{k} p(\pi_{ji}^-) c(\pi_{ji}^-) \right)$$

$$= \sum_{j=1}^{k} p(\pi_j) \left[ p_i (c(\pi_j) + C_p(\Theta_{ji}^+)) + (1 - p_i) (c(\pi_j) + C_p(\Theta_{ji}^-)) \right]$$

$$= \sum_{j=1}^{k} p(\pi_j) \left( c(\pi_j) + p_i C_p(\Theta_{ji}^+) + (1 - p_i) C_p(\Theta_{ji}^-) \right).$$
value. Using now the facts that \( C_p(\Theta | \vec{e}) = C_{\tilde{p}}(\Theta | \vec{e}) \) and \( C_p(\Theta^+ | \vec{e}) = C_{\tilde{p}}(\Theta^+ | \vec{e}) \), as none of the substrategies \( \Theta | \vec{e}, \Theta^+_j, j \neq i \), involve the experiment \( e_i \), we obtain the following bound:

\[
| C_p(\Theta) - C_{\tilde{p}}(\Theta) | = | (C_p(\Theta | e_i) + C_p(\Theta | \vec{e})) - (C_{\tilde{p}}(\Theta | e_i) + C_{\tilde{p}}(\Theta | \vec{e})) | \\
= | (C_p(\Theta | e_i) - C_{\tilde{p}}(\Theta | e_i)) + (C_p(\Theta | \vec{e}) - C_{\tilde{p}}(\Theta | \vec{e})) | \\
= \left| \sum_{j=1}^{k} p(\pi_j) \left( (p_i - \hat{p}_i)C_p(\Theta^+_j) + (\hat{p}_i - p_i)C_p(\Theta^-_j) \right) \right| + | 0 | \\
= \left| \sum_{j=1}^{k} p(\pi_j) (p_i - \hat{p}_i) \left( C_p(\Theta^+_j) - C_p(\Theta^-_j) \right) \right| \\
\leq \sum_{j=1}^{k} p(\pi_j) \times | p_i - \hat{p}_i | \times \max_{l(u_j) = e_i} \left| C_p(\Theta^+_j) - C_p(\Theta^-_j) \right| \\
\leq \rho(\epsilon) \times | p_i - \hat{p}_i | \times \max_{l(u_j) = e_i} \max \left\{ C_p(\Theta^+_j), C_p(\Theta^-_j) \right\}.
\]

The last line of the calculation uses the facts that \( C_p(\cdot) \geq 0 \) and that \( \sum_{j=1}^{k} p(\pi_j) = \rho(\epsilon, \Theta) \) is the probability that this strategy \( \Theta \) will reach \( e_i \), and hence is bounded by \( \rho(\epsilon) = \max \{ \rho(\epsilon, \Theta) \mid \Theta \in SS(G) \} \).

All that remains is to show that the value \( \max_{l(u_j) = e_i} \{ C_p(\Theta^+_j), C_p(\Theta^-_j) \} \) is bounded by \( D(\epsilon) \). To see this, consider any \( u_j \) such that \( l(u_j) = e_i \). Then

\[
C_p(\Theta^+_j) = c(u^+_j) + \sum_{u_t(u^+_j)} p(\pi^+_j) c(\pi^+_j) \\
\leq c(u^+_j) + \left( \max_{u_t(u^+_j)} c(\pi^+_j) \right) \sum_{u_t(u^+_j)} p(\pi^+_j) \\
= c(u^+_j) + \max_{u_t(u^+_j)} c(\pi^+_j) \\
= \max_{u_t(u^+_j)} c(\pi^+_j) \\
\leq D(\epsilon).
\]

Similarly, \( C_p(\Theta^-_j) \leq \max_{u_t(u^-_j)} c(\pi^-_j) \leq D(\epsilon) \). \( \Box \) (Lemma 1)

**Lemma 2** Let \( G = \langle W, F, R, c \rangle \) be a tree-like decision structure with \( |W| = n \) experiments, and let \( p = \langle p_1, \ldots, p_n \rangle \) be a vector of success probabilities. Furthermore, let \( \epsilon, \delta > 0 \) be any given constants, and let \( \hat{p} = \langle \hat{p}_1, \ldots, \hat{p}_n \rangle \) be a vector of probability estimates computed by the GS algorithm of Figure 4. Then

\[
\forall e_i \in W. \quad \Pr \left[ D(\epsilon) \times \rho(\epsilon) \times | p_i - \hat{p}_i | \geq \frac{\epsilon}{2n} \right] \leq \frac{\delta}{n} \tag{3}
\]

**Proof:** We use Hoeffding’s Inequality, which is a simple form of Chernoff bounds [4, 1]: Let \( \{ X_i \} \) be a set of independent, identically-distributed random Bernoulli variables, whose
common mean is \( \mu \). Let \( \bar{S}(M) = \frac{1}{M} \sum_{i=1}^{M} X_i \) be the sample mean after taking \( M \) samples. Then

\[
Pr[\bar{S}(M) > \mu + \lambda] < e^{-2M\lambda^2}, \quad Pr[\bar{S}(M) > \mu - \lambda] < e^{-2M\lambda^2}
\] (4)

To prove that inequality (3) holds for each experiment \( e_i \in W \), we consider two cases, depending on whether the experiment can be performed initially or not. If so (i.e. if \( F((), e_i) \) and consequently \( \rho(e_i) = 1 \)), then the GS algorithm will perform at least

\[
m(e_i) = \left[ 2 \left( \frac{n D(e_i)}{\epsilon} \right)^2 \ln \frac{2n}{\delta} \right]
\] (5)

trials of \( e_i \). As the samples are drawn at random from a fixed distribution, we can use inequality (4): after \( m(e_i) \) samples,

\[
Pr \left[ \rho(e_i) \times |\hat{p}_i - p_i| \geq \frac{\epsilon}{2n D(e_i)} \right] = Pr \left[ |\hat{p}_i - p_i| \geq \frac{\epsilon}{2n D(e_i)} \right] \leq 2 \exp \left( -2m(e_i) \left( \frac{\epsilon}{2n D(e_i)} \right)^2 \right) \leq \frac{\delta}{n}.
\]

Now consider an experiment \( e_i \) that is not immediately reachable (i.e., such that \( F((), e_i) \) does not hold). Here, the GS algorithm will attempt to reach \( e_i \) along a direct path a total of \( M \) times, where

\[
M \geq \left[ 2 \left( 1 + \frac{\epsilon}{n D(e_i)} \right) \left( \frac{n D(e_i)}{\epsilon} \right)^2 \ln \frac{4n}{\delta} \right].
\] (6)

Of these, GS reaches and performs \( e_i \) some number \( k \) times; and does not reach \( e_i \) the remaining \( M - k \) times. Denote by \( j(k) \) the value of the product \( \rho(e_i) \times |\hat{p}_i - p_i| \) assuming that GS succeeds exactly \( k \) times. It suffices to show that for any value of \( k \), \( j(k) \leq \frac{\epsilon}{2n D(e_i)} \) with probability at least \( 1 - \frac{\delta}{2n} \).

Using the fact that \( \hat{\rho}(e_i) = \frac{k}{M} \) is the estimated value of \( \rho(e_i) \) here (as GS has succeeded \( k \) times out of \( M \)) together with inequality (4), we know that with probability at least \( 1 - \frac{\delta}{2n} \),

\[
\rho(e_i) \leq \hat{\rho}(e_i) + \sqrt{\frac{1}{2M} \ln \frac{2n}{\delta}}.
\]

and with probability at least \( 1 - \frac{\delta}{2n} \),

\[
|\hat{p}_i - p_i| \leq \sqrt{\frac{1}{2k} \ln \frac{4n}{\delta}}.
\]

Of course, each of these terms tops off at 1 (since \( \rho(e_i) \leq 1 \) and \( |\hat{p}_i - p_i| \leq 1 \)). We therefore define

\[
g(k) = \min \left\{ 1, \frac{k}{M} + \sqrt{\frac{1}{2M} \ln \frac{2n}{\delta}} \right\} \times \min \left\{ 1, \sqrt{\frac{1}{2k} \ln \frac{4n}{\delta}} \right\}
\]

20
and observe that \( j(k) \leq g(k) \) with probability at least \((1 - \frac{\delta}{2^M})^2 \geq 1 - \frac{\delta^2}{n}\).

We now need to bound the largest possible value of \( g(k) \). First, note that \( g(k) \) can be bounded by

\[
g(k) \leq \begin{cases} 
\frac{k}{M} + \left( \frac{1}{2M} \ln \frac{2n}{\delta} \right) & \text{when } 0 \leq k \leq k_0, \\
\left( \frac{k}{M} + \left( \frac{1}{2M} \ln \frac{2n}{\delta} \right) \right) \times \sqrt{\frac{1}{2k} \ln \frac{4n}{\delta}} & \text{when } k_0 \leq k \leq M,
\end{cases}
\]

where \( k_0 = \frac{1}{2} \ln \frac{4n}{\delta} \). (The two expressions for \( g(k) \) have the same value at \( k = k_0 \).) As the first expression has a positive first derivative with respect to \( k \), it is largest at its largest allowed value of \( k \), viz. \( k_0 \). Using first and second derivatives, we see that the second expression is upwards concave on the interval \( k = [k_0..M] \), and thus its value is maximal at either \( k = k_0 \) or \( k = M \).

Hence, the largest value of \( g(k) \) is bound by the value of the second expression at either \( k = k_0 \) or \( k = M \), i.e. by the larger of the values

\[
g_{k_0} = \frac{1}{2M} \ln \frac{4n}{\delta} + \sqrt{\frac{1}{2M} \ln \frac{2n}{\delta}} \\
g_M = \frac{1}{2M} \ln \frac{4n}{\delta} + \sqrt{\frac{1}{2M} \ln \frac{2n}{\delta}} \times \sqrt{\frac{1}{2k} \ln \frac{4n}{\delta}}
\]

By inspection, both of these values, and hence all values of \( g(k) \), are below

\[
g_M = \frac{1}{2M} \ln \frac{4n}{\delta} + \sqrt{\frac{1}{2M} \ln \frac{2n}{\delta}} = \left( \frac{1}{2} + \sqrt{\frac{1}{2M} \ln \frac{4n}{\delta}} \right)^2 - 1.4.
\]

As \( \rho(e_i) \times |\hat{p}_i - p_i| = j(k) \leq g_M \), we need only find an \( M \) sufficiently large so that \( g_M \leq \frac{\epsilon}{2nD(e_i)} \). Solving \( g_M = \frac{\epsilon}{2nD(e_i)} \) for \( M \) yields

\[
M = 2 \left( \left( \frac{2\epsilon}{nD(e_i)} + 1 - 1 \right) \right)^2 \ln \frac{4n}{\delta}.
\]

To see that the \( m(e_i) \) value from equation (6) is larger than this \( M \) (and hence the corresponding \( g_M \) value will be yet smaller), just observe that \((\sqrt{2\gamma + 1} - 1)^2 \leq \frac{4\epsilon^2}{\gamma^2} \) holds for any \( \gamma > 0 \), so in particular for \( \gamma = \frac{\delta}{e_i} \). Hence, after taking at least \( m(e_i) \) samples, we can be confident that the product \( \rho(e_i) \times |\hat{p}_i - p_i| \) is sufficiently small, as desired.

Notice that in the typical situation where \( \epsilon \) is small relative to \( D(e) \), the \( m(e_i) \) value obtained here is only slightly larger than the value obtained from equation (5).

\( \square \) (Lemma 2)

**Theorem 1** Let \( G = (W, F, R, e) \) be a tree-like decision structure with \( |W| = n \) experiments, and let \( p = \langle p_1, \ldots, p_n \rangle \) be a vector of success probabilities for the experiments. Furthermore, let \( \epsilon, \delta > 0 \) be any given constants, and let \( \Theta_{pa_0} = pao(G, \epsilon, \delta) \) be the strategy produced by the pao algorithm using the GS subroutine of Figure 4. Then, with probability at least \( 1 - \delta \),
$C_p(\Theta_{pao}) - C_p(\Theta_p) \leq \varepsilon$, where $\Theta_p = OSS(G, p)$ is the optimal strategy for probability vector $p$.

**Proof:** Let $\hat{p} = (\hat{p}_1, \ldots, \hat{p}_n)$ be the vector of probability estimates produced by the GS subroutine. By Lemma 2 each of the products $D(e_i) \times \rho(e_i) \times |p_i - \hat{p}_i|$ is upper bounded by the value $\varepsilon/2n$ with probability at least $1 - \delta/n$. Hence, the probability that they are all less than $\varepsilon/2n$ is at least $1 - \delta$. The theorem’s claim follows from this by Lemma 1. □ (Corollary 1)

**Corollary 1** Let $\Theta_{pao} = pao(G, \varepsilon, \delta)$ be the result of the pao algorithm, where $G = \langle W, F, R, c \rangle$ is any decision structure in DS, and $\varepsilon, \delta > 0$ are given constants. Then, with probability at least $1 - \delta$, $C_p(\Theta_{pao}) - C_p(\Theta_p) \leq \varepsilon$ where $\Theta_p = OSS(G, p)$ is the optimal strategy, based on the correct probability vector $p$.

**Proof:** This result follows immediately from the proof of Theorem 1, using only the easy first part of Lemma 2, and the observation that $C \geq D(e_i)$ guarantees that the value $M$ in equation (1) is larger than the value $m(e_i)$ in equation (5). □ (Corollary 1)

**Theorem 2** Assume $RP \neq NP$. Then there is no probabilistic polynomial time algorithm that, given a decision structure $G = \langle W, F, R, c \rangle$, an experiment $e \in W$, a distribution function $p: W \rightarrow [0, 1]$, and parameters $\varepsilon, \delta > 0$, can estimate the value $\rho(\varepsilon)$ to within $\varepsilon$ with probability at least $1 - \delta$.

**Proof:** Assume to the contrary that such an algorithm exists for some fixed values of $\varepsilon, \delta > 0$; say $\varepsilon = \delta = 1/3$. We show that this algorithm could also be used to decide the satisfiability of boolean formulas (SAT) with reliability $1 - \delta$. As the SAT problem is NP-complete, it would follow by standard arguments that $RP = NP$.

Let $\varphi$ be a boolean formula over the variables $x_1, \ldots, x_n$. We show how to construct a corresponding decision structure $G_\varphi = \langle W_\varphi, F_\varphi, R_\varphi, c_\varphi \rangle$, such that the formula $\varphi$ has (resp. does not have) a satisfying assignment to its variables if and only if the value $\rho(g)$ for a specific experiment $g$ in $W_\varphi$ is 1 (resp. 0). We could thus decide the satisfiability of formula $\varphi$, with reliability $1 - \delta$, by running our hypothetical algorithm on structure $G_\varphi$ and experiment $g$, and checking whether the estimate it provides for $\rho(g)$ is greater than $1 - \varepsilon$ or less than $\varepsilon$.

The structure $G_\varphi$ has $2n + 1$ experiments $W_\varphi = \{e_1, \bar{e}_1, \ldots, e_n, \bar{e}_n, g\}$. The precedence relation $F_\varphi$ permits exactly one of $e_1$ or $\bar{e}_1$ to be performed initially, then exactly one of $e_2$ or $\bar{e}_2$, and so on; in general permitting exactly one of $e_k$ or $\bar{e}_k$ as the $k^{th}$ experiment. (That is, $F_\varphi(\alpha, e_{k+1})$ and $F_\varphi(\alpha, \bar{e}_{k+1})$ both hold iff $\alpha$ is of the form $\langle e_1, \ldots, \bar{e}_k, + \rangle$, where each $\bar{e}_i$ is either $e_i$ or $\bar{e}_i$.) Now each complete sequence of the $\bar{e}$-type experiments, $\alpha = \langle (\bar{e}_1, +), (\bar{e}_2, +), \ldots, (\bar{e}_n, +) \rangle$, can be identified with a truth assignment to the variables $x_1, \ldots, x_n$, whereby $x_i$ is true (resp. false) if and only if $\bar{e}_i = e_i$ (resp. $\bar{e}_i = \bar{e}_i$) in $\alpha$. The precedence relation $F$ finally specifies that experiment $g$ can be performed (i.e. $F_\varphi(\alpha, g)$ holds) if and only if this truth assignment satisfies the formula $\varphi$.

---

Footnote: We are indebted to Tom Hancock for suggesting this reduction.

---

22
Consider then the trivial probability distribution that assigns success probability 1 to all experiments, and recall that \( \rho(g) \) is the maximum probability of reaching experiment \( g \) using any strategy. Given the precedence constraints specified above, it is clear that there exists a strategy \( \Theta \) for reaching \( g \) if and only there exists a satisfying truth assignment for \( \varphi \), and any such strategy will have \( \rho(g, \Theta) = 1 \). Hence \( \rho(g) = 1 \) if and only if \( \varphi \) is satisfiable, and otherwise \( \rho(g) = 0 \). □ (Theorem 2)

References


