HEURISTIC SEARCH THEORY: SURVEY OF RECENT RESULTS

Judea Pearl
Cognitive Systems Laboratory
Departments of Computer Science and Engineering Systems
University of California, Los Angeles
Los Angeles, California 90024

ABSTRACT

This paper summarizes recent analytical Investigations of the mathematical properties of heuristics and their Influence on the performance of common search techniques. The results are reported without proofs, together with discussions of motivations and Interpretations.

Highlights include the following: relations between the precision of the heuristic estimates and the average complexity of the search, comparisons of the average complexities of A* and BACKTRACKING, procedures for comparing and combining non-admissible heuristic functions, the influence of the weight u> [In f ■ (l-u>)g ♦ <*>h] on the complexity of A*, determination of the branching factors of alpha-beta and SSS*, and the effects of successor ordering on the complexity of alpha-beta and of search depth on the quality of decisions.

1.0 THE MEAN COMPLEXITY OF ADMISSIBLE BEST-nftST ALflORITHMS

1.1 Introduction

Research in this area has focused on unraveling the relation between the accuracy of the heuristic estimates and the complexity of the search which they control. We Imagine the following probabilistic search space: a uniform m-ary tree T has a unique goal state G at depth N, at an unknown location. The best-first algorithm A* [1] searches for the goal state G using the evaluation function:

f(n) »'g(n) + h(n)

where g(n) is the depth of node n and h(n) is a heuristic estimate of h*(n), the distance or number of branches from n to G. The estimates h(n) are assumed to be random variables ranging over [0, h*(n)], characterized by distribution functions ^{F}h(n)(^{x}) ^{s} *K^M* * x] which may vary over the nodes in trie tree. We further assume that each Fk/$_0$\(x) depends only on the distance between n and G (I.e., on h*(n)) and not on the heuristics assigned to neighboring nodes. Our task is to characterize the Influence of the distributions F^Wx) on E(Z), the expected number of nodes expanded by A*, for large values of N.

The first analysis of the effect of errors or inaccuracies on the performance of A* was conducted by Pohl [2]; the topic has since been pursued by Munyer [3], Vanderbrug [4], Pohl [5], and Gaschnig [6]. The basic motivation for these studies has been the following enigma: when A* employs a perfectly informed heuristic (h - h*), it is propelled directly toward the goal without ever getting sidetracked, spending only N computational steps, whereas at the other extreme, when no heuristic at all is available (h = 0), the search becomes an exhaustive, breadth first one, yielding an exponentially growing complexity. Between these two extremes lies an unknown accuracy-complexity dependency containing the answers to important design questions. Would the added computation invested in improving the accuracy of a given heuristic pay for itself in reduced search complexity? Can some heuristics beat the "exponential explosion" that beset breadth-first search? When 1s the large storage-space required by A* justified in view of lower storage procedures such as BACKTRACKING?

Some initial answers to these questions were obtained by Pohl [5] and Gaschnig [6]. For Instance, they found that if the relative error [h*(n)-h(n)]/h*(n) remains constant, then the search complexity is exponential, but that when the absolute error h*(n)-h(n) is constant, the search complexity 1s linear. These results, however, were derived for a worst case model where it Is assumed that a clever adversary distributes the errors in such a way that A* exhibits Its poorest performance. Probabilistic extensions of these analyses are justified for two main reasons. First, worst case results are often too pessimistic for describing the typical behavior of an algorithm over a large class of problems. Second, it is often hard to guarantee precise bounds on the magnitude of errors produced by a given heuristic, whereas probabilistic characterization of these magnitudes might be more natural.

1.2 When 1s One Heuristic Better Than Another For Admissible Best-First Algorithms?

If one heuristic consistently provides a more accurate estimate of A*, it ought to be preferred. This 1s Indeed the essence of a theorem by Gelperin [7], who showed that if for each node of the search

This research was supported In part by the National Science Foundation under Grants MCS 78-07468 and MCS 78-18924.

graph $h_1(n) < h_2(n)$, and if both are admissible, then every node expanded by A_2^* is also expanded by A_1^*, that is, $h_2(n)$ is to be preferred. However, we seldom possess sufficient a priori knowledge to guarantee that the inequality $h_1(n) < h_2(n)$ holds for every node in the graph. Even when the improved accuracy of h_2 is a product of invoking more sophisticated computation procedures than h_1, the improvement is seldom guaranteed to take place at every node of the problem space. Generally, when $h(n)$ is made more accurate for some nodes, it may become less accurate for others. It is natural to ask, then, whether a statement of preference can be made in the case where the inequality $h_1 < h_2$ is known only to be a reasonably probable but occasionally violated event. The formalization and affirmation of such a statement is expressed in the following theorem (Huyn, Dechter, & Pearl [8]).

Definition: Given two random variables X_1 and X_2, we say that X_2 is stochastically greater than X_1 (denoted by $X_1 \circledS X_2$) iff for every x:

$$P(X_2 > x) \geq P(X_1 > x)$$

Definition: Let A_1^* and A_2^* employ the admissible heuristic functions h_1 and h_2, respectively. A_2^* is said to be stochastically more informed than A_1^* iff $h_1(n) \circledS h_2(n)$, $\forall n \in T$. Similarly, A_2^* is said to be stochastically more efficient than A_1^* iff $Z_2 \circledS Z_1$, where Z_1 and Z_2 are the number of nodes expanded by A_1^* and A_2^*, respectively.

Theorem 1: For any error distribution if A_2^* is stochastically more informed than A_1^*, then A_2^* is stochastically more efficient than A_1^*.

Theorem 1 establishes a partial-order relation on admissible heuristic functions characterized by probability distributions. A different order holds for non-admissible heuristics (see Section 2.3). Note that stochastic superiority ($Z_2 \circledS Z_1$) subsumes superiority in the mean ($E(Z_2) \leq E(Z_1)$).

1.3 The Mean Complexity of A* Under Distance-Dependent Errors

The quality of the heuristic estimates h(n) often improves with proximity to the goal state. This fact can be modeled by assuming that the typical magnitude of the errors h*(n)-h(n) increases proportionally to the distance h*(n), or that for all nodes in the tree, the relative errors $Y(n) = [h^*(n)-h(n)]/h^*(n)$ are likely to be bounded away from zero.

Theorem 2: If, for *eyery* node in the tree, the probability that the relative error exceeds some fixed positive quantity e is greater than 1/m, then the average complexity of A* is exponential in N (Huyn et al., [8]).

Theorem 2 Implies that in order to avoid an exponential growth of E(Z), the magnitude of the typical errors 1n the tree should Increase slower than linearly with the distance from the goal. Theorem 2 extends the results of Gaschnlg [6] from the worst case to the average case analysis. Gaschnlg has shown that 1f at all nodes off the solution path the relative errors stay at their maximal value above some fixed constant, then the complexity of A* 1s exponential. Evidently, the exponential explosion 1s not eliminated by diffusing the errors smoothly over a continuous Interval.

This result delineates the spectrum of the precision-complexity exchange by two points. On one extreme we have Pohl [5] and Gaschnlg's [6] results stating that 1f the absolute errors h*(n)-h(n) are bounded by a fixed quantity, then A* is guaranteed a linear complexity. On the other extreme, Theorem 2 states that if these errors h*(n)-h(n) grow linearly with h*(n), A* exhibits an exponential connplexity. The following result quantifies the precision-complexity exchange over the Interior of its spectrum, thus determining how accurate the estimates must be in order to guarantee a polynomial complexity.

Definition: A heuristic estimate $h(\cdot)$ is said to induce a typical error of order $\phi(n)$ if for all nodes in an m-ary tree there exist two fixed positive quantities ε and δ, and a normalizing function $\phi(\cdot)$, such that:

$$P\left[\frac{h^*(n)-h(n)}{\phi[h^*(n)]} > \varepsilon\right] > \frac{1}{m} \tag{1}$$

and simultaneously:

$$P\left[\frac{h^*(n)-h(n)}{\phi[h^*(n)]} \leq \delta\right] < \frac{1}{m} \tag{2}$$

The first condition guarantees that the ϕ-normalized errors $[h^*(n)-h(n)]/\phi[h^*(n)]$ remain bounded away from zero with sufficiently high probability. The second condition insists that the ϕ-normalized error is likely to remain finite even when the distance to the goal increases indefinitely.

The proportional errors treated in Theorem 2 above are represented by the special case $\phi(N) = N$, whereas the absolute errors analyzed by Pohl correspond to $\phi(N) =$ constant.

Theorem 3: If the typical error induced by $h(\cdot)$ is of order $\phi(N)$ and $\lim_{N\to\infty} \phi(N)/N < \infty$, then the mean complexity of A* is given by:

$$E(Z) = G(N) \exp\,(c\phi(N)[1 + o(1)]) \tag{3}$$

where c is a constant and G(N) is at most $O(N^2)$. Moreover, if only condition 1 (or 2) is known to hold, then the expression in (3) constitutes a lower or upper bound, respectively (Pearl, [9]).

The exponential relationship exhibited in Theorem 3 implies that the precision-complexity exchange for A* is fairly "inelastic," requiring that highly precise heuristics be devised to contain the search complexity within a reasonable growth rate. For example, Theorem 3 implies that if the typical distance-estimation error increases faster than logarithmically in the actual distance to the goal, then the mean complexity of A* grows faster than N^k for any finite k regardless of the shapes of the distribution functions. Thus, a nee essary and sufficient condition for maintaining a polynomial search complexity 1s that A* be guided by heuristics with logarithmic precision, that Is, 4»(n) « 0(log N).

1.4 Comparison to BACKTRACKING and the Effect of Multiple Goals

The BACKTRACKING search strategy [10] also employs an element of the best-first principle but the "best," lowest f, is chosen only among the set of newly generated nodes, not among all the nodes encountered in the past. Consequently, once BACKTRACKING decides to expand a node not lying on a solution path, the entire tree beneath that node must be irrevocably expanded down to a given depth bound before another path is tried. Thus, assuming that node ordering in BACKTRACKING is governed by the same heuristic function $f(n) = g(n) + h(n)$ which guides A*, the complexity Z_B depends on how often each node n on the solution path is assigned an estimate f(n) higher than its off-course siblings. Assuming further that the ϕ-normalized errors $Y = [h^*-h]/\phi(h^*)$ are identically distributed over all nodes, that $P(Y=0) = \beta$ and $\lim_{N\to\infty} \phi(N)/N < \infty$, one obtains [9]:

$$E(Z_B) \geq \frac{1}{2}(1-\beta^2)\, m^{N-1}\, [1 + O(\frac{1}{\phi(N)})] \qquad (4)$$

implying that BACKTRACKING's complexity retains its exponential character in spite of any improvement in heuristic precision. Whereas the parameters $F_Y(\cdot)$, $\phi(\cdot)$, characterizing the informedness of $h(\cdot)$, have a direct impact on the growth-rate of the complexity of A* (see, for example, Theorem 3) they have only a tenuous influence upon the complexity of BACKTRACKING strategies via the multiplication constant $(1-\beta^2)$. Consequently, if an admissible distance-estimating heuristic is available, then A* is a more effective instrument for converting this information into appreciable savings in search time.

One may wonder whether A* retains this effectiveness when the search tree contains multiple goal states. Clearly the presence of sub-optimal solutions should cause some off-track nodes to obtain deceptively low h and would, therefore, increase the number of nodes expanded. An extreme model whereby all nodes at depth N+1 are goal nodes while only one goal exists at depth N will give an idea of the magnitude of this effect. Under this condition the expected complexity of A* becomes [9]:

$$E(Z) = G(N)\, [m(1-\beta)]^N\, e^{-\alpha\psi(N)} \qquad (5)$$

where $\psi(N) = \sum_{k=0}^{N} 1/\phi(k)$ and G(N) is at most $O(N^2)$. Hence, A* would retain its exponential complexity regardless of the character of the precision function ϕ; its growth rate, however, still would be affected by ϕ. For example, when ϕ = const. (i.e., bounded absolute errors), E(Z) becomes $O[(m')^N]$ where $m' = mP(h^*-h > 1)$; for $\phi(N) = N$ we obtain $E(Z) = G(N)\, N^{-\alpha}\, [m(1-\beta)]^N$. Thus, although the presence of multiple solutions may significantly impair the ability of A* to benefit from Improved precision, the complexity of A* remains more sensitive to error reduction than that of BACKTRACKING.

BACKTRACKING control strategies, on the other hand, have several advantages over A*. They are typically simpler to implement and, more significantly, require much less storage. Whereas A* stores all expanded nodes in either OPEN or CLOSED lists, BACKTRACKING need only store nodes along a single path, and therefore has a storage complexity linear in N. Equation (4), though, reveals the price paid for this storage economy—an exponential time-complexity coupled with ineffective utilization of heuristic knowledge. Mixed search strategies combining the storage economy of BACKTRACKING with the time savings of A* warrant empirical and theoretical investigations.

2.0 THE COMPLEXITY OF NON-ADMISSIBLE HEURISTICS

2.1 Introduction

Admissible search strategies are cursed with two basic defects: they spend a disproportionate amount of time investigating aV[equally meritorious alternative solutions, and they limit the selection of heuristic functions to only those which never over-estimate the optimal completion cost. The literature on heuristic search contains many examples of non-admissible heuristics which empirically out-perform any known admissible heuristics and yet *yery* frequently discover the optimal path.

The aims of theoretical investigations in this area are to answer several basic and pressing questions which until now have been treated only by lengthy simulations 1n a few, specific domains. When are non-admissible heuristics safe from a catastrophic over-computation? How often are they likely to miss the optimal solution? When is one heuristic better than another? Is admissibility a virtue in cases where just any solution will do? How should one "debias" a given admissible heuristic? How should one aggregate the estimates provided by several heuristic functions?

The results reported in the following sections provide answers to some of these questions and pave the way to resolve others.

2.2 Conditions for Node Expansion

The analysis of mean run time of admissible best-first algorithms is facilitated by the simplicity of the condition for expansion. We know that *eyery* node n in OPEN whose evaluation function satisfies f(n) < f*(root) must eventually be expanded and conversely, that *eyery* node satisfying f(n) > f*(root) will not be expanded. In our standard model of an m-ary tree with one goal at depth N, these conditions amount to deciding whether f(n) 1s larger or smaller than N [8].

The expansion condition for non-admissible heuristics is complicated by the fact that $f(n)$ may exceed N at several places along the solution path. This leads to the more general expansion condition for nodes in OPEN:

$$f(n) < \max_{0 \le i < j} f(n_i^S) \tag{6}$$

where n_0^S n_1^S n_2^S ... n_N^S are nodes along the solution path and n_j^S is the deepest common ancestor of G and n (see Figure 1):

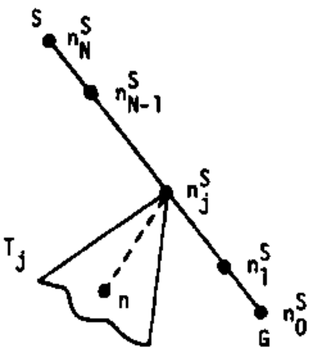


Figure 1

The right-hand side of (6) is a random variable L_j whose distribution determines the probability of expanding any node in subtree T_j. The probabilistic analysis of complexity is facilitated by the following theorem [11] which permits us to replace L_j with a deterministic quantity.

Theorem 4: Let the ϕ-normalized errors $Y = [h-h^*]/\phi(h^*)$ along the solution path be independent random variables all having a non-zero over-estimation probability so that $P[Y(n_i^S) < y'] > 0$ for some $y' < 0$ and all i. For sufficiently large j, the random variable $L_j = \max_{0 \le i < j} f(n_i^S)$ is almost sure to exceed the quantity $N - \phi(j)y'$.

To put it bluntly, if anything can go wrong on the solution path, it almost surely will. Theorem 4 implies that if y_0 is the maximal common over-estimation error for all nodes along the solution path (i.e., $y_0 = \inf\{y' | P[Y(n_i^S) < y'] > 0 \ \forall i\}$), then we are safe in assuming that this maximal over-estimation indeed took place along the solution path and in substituting it in the expansion condition, which becomes:

$$f(n) < N - \phi(j)y_0 \tag{7}$$

Theorem 4 makes this condition, normally invoked in worst case analysis, applicable to average case analysis as well.

2.3 When is One Heuristic Better Than Another if Over-Estimations are Possible?

If h_1 and h_2 are both admissible then the inequality $h_1 < h_2$ also implies that h_2 is a closer approximation to h^* and would naturally lead to a more efficient search. This simple criterion of choosing the highest cost estimates breaks down when over-estimations are encountered. Over-estimations reduce the search complexity when they occur at off-track nodes and increase it when they take place along the solution path. The exact effect of over-estimations on the mean run time of A* depends on a delicate balance between these two forces.

The following set of results are derived under the assumption that the relative errors $Y(n) = [h^*(n)-h(n)]/h^*(n)$ are independent and identically distributed random variables. Letting $H = h/h^*$, it is convenient to characterize the errors by the distribution function $F_H(x) = P([h/h^*] \le x)$. We will say that h_1 is more efficient than h_2 if the use of h_1 results in a mean complexity lower or equal to that of h_2.

Definition: A random variable X is said to have an upper support r iff $X \le r$ and $P(X \le x) < 1$ for all $x < r$.

Theorem 5: Let $h_1(\cdot)$ and $h_2(\cdot)$ be two heuristic functions such that $H_1 = h_1/h^*$ and $H_2 = h_2/h^*$ possess the same upper support r. If $F_{H_1}(x) \ge F_{H_2}(x)$ for all $x \le r$, then H_2 is more efficient than H_1.

Theorem 5 generalizes the criterion of Theorem 1 to the non-admissible case simply by stating that the heuristic whose density is more concentrated around its upper support is the one to be preferred. It is a reasonable criterion in the admissible case where such concentration also means increased accuracy. It is somewhat surprising, though, in the non-admissible case where concentration around the upper support and accuracy may not go hand in hand. In Figure 2, for example, h_1 is a more accurate heuristic since it assigns a greater weight to the neighborhood of the correct estimate $h/h^* = 1$. Still, Theorem 5 proclaims h_2 as the more efficient heuristic.

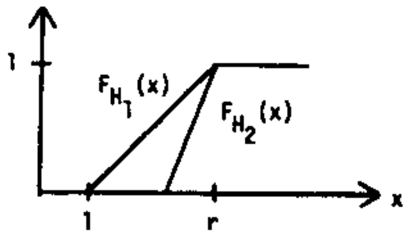


Figure 2

The next theorem establishes a preference between heuristics with different upper supports.

Theorem 6: Given two heuristic functions $h_1(\cdot)$ and $h_2(\cdot)$ such that H_1 and H_2 have the upper supports r_1 and r_2 $(1 \le r_1 < r_2)$, respectively, if $F_{H_1}(x[r_1/r_2]) \le F_{H_2}(x)$ for all $x < r_2$, then h_1 is strictly more efficient than h_2.

Figure 3 below illustrates this preference criterion graphically. The two dotted distribution functions represent heuristics inferior to h_1 because they lie above its stretched distribution $F_{H_1}([r_1/r_2]x)$. Note that h_2 can be stochastically greater than h_1 and still be inferior to it as illustrated by the lower dotted curve. Simply stated, Theorem 6 implies that H_2 is inferior to H_1 if it is stochastically smaller than or equal to $(r_2/r_1)H_1$. Thus, once a heuristic h possesses an upper support which is greater than unity, all attempts to further boost its value by using $f = g + \alpha h$ and $\alpha > 1$ will only degrade its search performance. This has a far-reaching consequence for "debiasing" techniques. When a heuristic h is known to underestimate h^* consistently, that is, when $r < 1$, then the weighted combination $f = (1-\omega)g + \omega h$ $(\omega > 1/2)$ may be used [12] to "debias" f and compensate for h's underestimation. Theorem 6 states that ω should not be made too large; the mean complexity monotonically increases with ω once ω is greater than $1/1+r$.

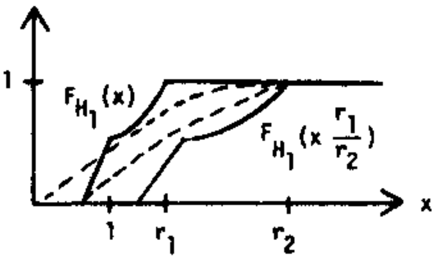


Figure 3

The quantity $1/1+r$, of course, is the highest value of ω which renders f admissible and, clearly, maintaining ω below that limit is necessary if one wishes to guarantee that A* will not miss the optimal solution. However, the preceding results demonstrate that admissibility constraints are also beneficial in cases where only one solution exists and performance is judged solely by the effort required to discover it. Moreover, since admissible heuristics always improve by increasing the weight of h (Theorem 1), we conclude that the admissibility limit $\omega = 1/1+r$ is optimal.

Figure 4 demonstrates the effect of the weight ω on the average complexity $E(Z_\omega)$ of A* for three cases: $r = 1/2$ (h admissible with 50% underestimation bias), $r = 1$ (h admissible and unbiased), and $r = 2$ (h non-admissible with 200% overestimation bias). The ordinate represents the growth rate $\log_m E(Z_\omega)$ normalized by that of exhaustive search $\omega = 0$. The three curves were computed with the assumption that the relative errors are distributed by the truncated exponential distribution:

$$P(h(n) \le x\, h^*(n)) = \begin{cases} 0 & x < 0 \\ e^{-\alpha(r-x)} & 0 \le x \le r \\ 1 & r \le x \end{cases}$$

Variations in the underlying distribution will only change the amplitudes of these curves but not their essential characteristics. The optimality of the admissibility limit $\omega = 1/1+r$ is clearly demonstrated by the sharp dips of the curves at this point. Note that when ω is made too high the average complexity of A* may exceed that of an exhaustive search and may become unbounded for $\omega = 1$. This is to be expected since reliance on h alone may cause A* to explore depths greater than N.

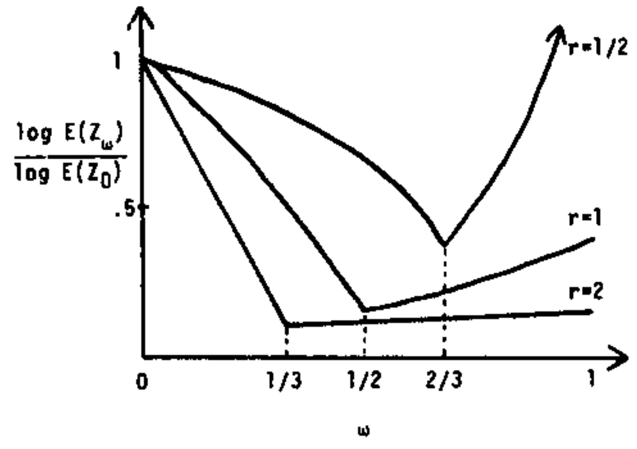


Figure 4

These results lead to a very simple procedure of combining information from several heuristic sources, admissible as well as non-admissible. Given the set of estimates $h_1, h_2, \ldots h_k$ having upper supports $r_1, r_2, \ldots r_k$, respectively, the combination rule:

$$h = \max(h_1/r_1, h_2/r_2, \ldots h_k/r_k)$$

would yield a more efficient estimate than any of its constituents. The division by the upper support removes the bias from each constituent and renders it admissible, with $r = 1$; the "max" operator further improves the estimate by rendering it larger than each of its constituents, while also maintaining $r = 1$.

Needless to state, these manipulations only diminish the growth rate of the mean search effort but cannot turn its complexity from exponential to polynomial. The exponential complexity implied by Theorem 2 remains valid for non-admissible heuristics as long as the condition:

$$P[h(n) \le h^*(n)\ (r-\epsilon)] > 1/m$$

is satisfied by all nodes for some positive constant ϵ.

3.0 GAME-SEARCHING

3.1 Introduction

Game-sea,ching differs from shortest-path search in two essential ways. First, the solution desired is not a simple sequence of moves, a path, but a strategy, a subtree, which specifies the player's best response to *every* conceivable move of the opponent. Second, the quality of a given strategy Is determined solely by the properties of the terminal nodes it contains, not by the cost of the paths leading to these nodes.

The theoretical results reported 1n this paper concern three main issues: establishing absolute limits on the complexity of game-searching procedures, comparing performances of known procedures under various conditions, and evaluating the quality of a decision as a function of the search effort.

The model most frequently used for evaluating the performance of game-searching methods consists of a uniform tree of depth d (d even) and degree n, where the terminal positions are assigned random values independently drawn from a common distribution F. We shall refer to such a tree as a (d, n, F)-tree. The expected number of terminal nodes examined during the search and its branching factor have become standard criteria for the complexity of the search method.

<u>Definition</u>: Let A be a deterministic algorithm which searches a (d, n, F)-tree to determine the minimax value of its root, and let IA(d, n, F) denote the expected number of terminal positions examined by A. The quantity:

$$\mathscr{R}_A(n, F) = \lim_{d\to\infty} [I_A(d, n, F)]^{1/d}$$

is called the <u>branching factor</u> corresponding to the algorithm A.

The results reported in this paper are direct consequences of a somewhat surprising convergence property of tall minimax trees [13]:

<u>Theorem 7</u>: The root value of a (d, n, F)-tree with a continuous, strictly increasing terminal distribution F converges as $d \to \infty$ (in probability) to a unique predetermined value v^* satisfying $F(v^*) = 1-\xi_n$, where ξ_n is the solution of $x^n+x-1 = 0$.

3.2 <u>The Branching Factor of Alpha-Beta</u>

The Alpha-Beta (a-e) pruning algorithm is the most commonly used procedure in game-playing applications. Yet although the exponential growth of game-tree searching is slowed significantly by that algorithm, quantitative analyses of its effectiveness have been frustrated for over a decade. One concern has been to determine whether the a-e algorithm is optimal over other game-searching procedures.

Slagle and Dixon (1969) showed that the number of terminal nodes examined by a-e must be at least $_n$Ld/2j + nld/2 l . i but may, in the worst case, reach the entire set of n° terminal nodes [14]. The analyses of expected performance using uniform trees with random terminal values had begun with Fuller, Gaschnlg, and Gillogly [15] who obtained formulas by which the average number of terminal examinations, $N_{n\ d}$, can be computed. Unfortunately, the formula would not facilitate asymptotic analysis; simulation studies led to the estimate $\mathscr{R}_{\alpha-\beta} \approx (n)^{.72}$.

Knuth and Moore [16] analyzed a less powerful but simpler version of the α-β procedure by ignoring deep cut-offs. They showed that the branching factor of this simplied model is O(n/log n) and speculated that the inclusion of deep cut-offs would not alter this behavior substantially. A more recent study by Baudet [17] confirmed this conjecture by deriving an integral formula for $N_{n,d}$, (deep cut-offs included), from which the branching factor can be estimated. In particular, Baudet shows that $\mathscr{R}_{\alpha-\beta}$ is bounded by $\xi_n/1-\xi_n \le \mathscr{R}_{\alpha-\beta} \le M_n^{1/2}$, where ξ_n is the positive root of $x^n+x-1 = 0$ and M_n is the maximal value of the polynomial $P(x) = [(1-x^n)/(1-x)] \times [1-(1-x^n)^n]/x^n$ in the range $0 \le x \le 1$. Pearl [13] has shown that both $\xi_n/1-\xi_n$ lower bounds the branching factor of every directional game-searching algorithm, and that an algorithm exists (called SCOUT) which actually achieves this bound. Thus, the enigma of whether α-β is optimal remained contingent upon determining the exact magnitude of $\mathscr{R}_{\alpha-\beta}$ within the range delineated by Baudet.

This enigma has been resolved [18] with the aid of Theorem 7 and the fact that if the terminal values of a (d, n, F)-tree are drawn from a characteristic distribution $\phi(x)$, then the distribution of every node in the tree would be identical in shape to $\phi(x)$ save for a scale factor depending on the node's level [19]. The result is summarized by the following theorem (Pearl [18]).

<u>Theorem 8</u>: The branching factor of the α-β procedure for a continuous-valued uniform tree of degree n is given by:

$$\mathscr{R}_{\alpha-\beta} = \frac{\xi_n}{1-\xi_n} \tag{8}$$

where ξ_n is the positive root of the equation $x^n+x-1 = 0$.

The asymptotic behavior of $\mathscr{R}_{\alpha-\beta}$ is O(n/log n), as predicted by Knuth's analysis [16]. However, for moderate values of n ($n \le 1000$) $\xi_n/1-\xi_n$ is fitted much better by the formula $(.925)n^{.747}$ (see Figure 4 of reference [13]), which vindicates the simulation results of Fuller et al. [15]. This approximation offers a more meaningful appreciation of the pruning power of the α-β algorithm. Roughly speaking, a fraction of only $(.925)n^{.747}/n \approx n^{-1/4}$ of the legal moves will be explored by α-β. Alternatively, for a given search time allotment, the α-β pruning allows the search depth to be increased by a factor log n/log $\mathscr{R}_{\alpha-\beta} \approx 4/3$ over that of an exhaustive minimax search.

The establishment of the precise value of $\mathscr{R}_{\alpha-\beta}$ for continuous-valued trees, together with a previous result that $\mathscr{R}_{\alpha-\beta} = n^{1/2}$ for almost all discrete-valued trees [13], completes the characterization of the asymptotic behavior of α-β. Moreover, the fact that $\xi_n/1-\xi_n$ lower-bounds the branching factor of any directional algorithm [13] renders α-β asymtotically optimal over this class of algorithms. However, the global optimality of α-β has remained an unresolved issue until very recently. Naturally, the focus of attention has turned to non-directional algorithms, raising the question whether any such algorithm exists which exhibits a branching factor lower than $\xi_n/1-\xi_n$.

3.3 A Minimax Algorithm Better Than Alpha-Beta? Yes and No

Stockman [20] has Introduced a non-directional algorithm called SSS* which consistently examined fewer nodes than a-£. Hopes were then raised that the superiority of Stockman's algorithm reflected an Improved branching factor over that of a-B.

A simple heuristic argument exists which refutes these hopes and Indicates that SSS* and aB$ possess Identical branching factors. It Is based on the fact that when the terminal nodes are assigned only two values (say 1 and 0), SSS* becomes directional (Identical to a-B). Now, since the search of continuous games must be harder than the search of any b1valued games of the same structure, and since directional algorithms for b1valued games may require [13] a branching factor of Cn/^-Cn* we conclude that SSS* would also exhibit a branching factor of at least $\xi_n/1-\xi_n$.

The weakness of this argument lies in the possibility that SSS* belongs to the rare class of algorithms whose performances Improve with randomization, thus finding 1t easier to search continuous-valued games. This weakness has motivated the evaluation of ^?s$s* by direct methods using techniques similar to those of Baudet [17] and Pearl [18]. This evaluation has been completed recently [21] with a definite confirmation of the relation:

$$\mathscr{R}_{SSS^*} = \mathscr{R}_{\alpha-\beta} = \frac{\xi_n}{1-\xi_n} .$$

Thus, the superiority of SSS* over a-e 1s not reflected in their growth rates; the two algorithms can be regarded as asymptotically equivalent.

The possible existence of some other algorithm with a branching factor superior to that of α-β has also been eliminated by a more recent result of Tarsi [22]. Considering a standard bivalued game-tree in which the terminal nodes are assigned the values 1 and 0 with the probabilities ξ_n and $1-\xi_n$, respectively, Tarsi's result states that <u>any</u> algorithm which solves such a game-tree must, on the average, examine at least $(\xi_n/1-\xi_n)^d$ terminal positions. At the same time the task of solving any bivalued game-tree is equivalent to the task of verifying an inequality proposition regarding the minimax value of a continuous-valued game-tree [13] of identical structure, and, consequently, the former cannot be more complex than the latter. Thus, the quantity $(\xi_n/1-\xi_n)^d$ should also lower-bound the expected number of nodes examined by any algorithm searching a continuous-valued game-tree. This, together with (8), establishes the asymptotic optimality of α-β over all game-searching algorithms, directional as well as non-directional.

3.4 When Is Successor Ordering Beneficial?

The analyses presented in the preceding two sections assumed that the order in which a-B selects nodes for expansion 1s completely arbitrary, say from left to right. In practice, the successors of each expanded node are first ordered according to their static evaluation function. Successors of MAX nodes are then expanded in a descending order of their evaluation function (their highest first) and those of MIN nodes 1n ascending order (their lowest first). It is normally assumed that such preordering increases substantially the number of cut-offs induced and results in a lower branching factor.

Clearly, when the static evaluation function 1s well Informed, correlating with the actual node value, such ordering will induce all possible cut-offs yielding *31* « n^/2. On the other hand, when the static evaluation is uninformed, the ordering is superfluous and yields <3f » Un)/0~tn) ^{as} *^{n} random ordering. The analysis of this section attempts to quantify the relation between the informedness of the static evaluation function and the branching factor induced by successor ordering.

For simplicity, we treat bivalued n-ary trees with probability P " £$_n$ that ^{a} terminal node obtains the value 1. The information quality of the static evaluation function V can be characterized by two distribution functions:

$$\begin{cases} F_W(x) = P[V \le x \text{ given that a node is actually a WIN}] \\ F_L(x) = P[V \le x \text{ given that a node is actually a LOSS}] \end{cases} \tag{9}$$

If we treat x as a variable parameter, then F_W can be regarded as a function of F_L, or $F_W = g(F_L)$. g(z) is a monotonic, weakly increasing function of z between the points (0,0) and (1,1). Totally uninformed V will be represented by g(z) = z, while "noiseless" V will be characterized by:

$$g(z) = \begin{cases} 0 & z < 1 \\ 1 & z = 1 \end{cases} .$$

The parameter which directly controls the branching factor is given by the integral:

$$S = \frac{1-\xi_n}{\xi_n} \; n \int_{z=0}^{1} [\xi_n - g(z)(1-\xi_n)]^{n-1} \, dz$$

<u>Theorem 9</u>: The branching factor of α-β with successor ordering over a (d, n, ξ_n)-game tree is given by:

$$\mathscr{R}(S) = [n + \frac{A(S)}{2} (1 + \sqrt{1 + 4n/A^2(S)})]^{1/2}$$

where:

$$A(S) = \frac{\xi_n}{1-\xi_n} S - \frac{1-\xi_n}{\xi_n} n$$

Numerical computations of $\mathscr{R}(S)$ show that, for all n, $\mathscr{R}(S)$ is almost linear with S, gradually increasing from its minimal value $\mathscr{R} = n^{1/2}$ to its maximal value $\mathscr{R}(S=1) = (\xi_n)/(1-\xi_n)$.

3.5 When is Look-Ahead Beneficial?

The basic rationale behind all game-searching methodologies is the belief that look-ahead followed by minimaxing improves the quality of decisions or, 1n other words, that the "back-up" evaluation function has a greater discrimination power to distinguish between good and bad moves. Although no theoretical model has supported this belief, it has become entrenched 1n the practice of game-playing and its foundation rarely challenged.

Two heuristic arguments are usually advanced in support of look-ahead. The first invokes the notion of visibility, claiming that since the fate of the game is more apparent near its end, nodes at deeper levels of the game-tree will be more accurately evaluated and choices based on such evaluation should be more reliable. The second alludes to the fact that whereas the static evaluation is computed on the basis of the properties of a single game position, the back-up value integrates the features of all the nodes lying on the search frontier, and so should be more informed. This latter argument essentially takes after a filtering model; the more samples, the less noise.

A recent work of Nau [23] demonstrated that the filtering argument is utterly fallacious. In a large class of game-trees, reaching deeper consistently degrades the quality of a decision. This phenomenon, which Nau termed pathologicall, is not confined to the special game-trees considered by Nau but can be shown to be a common occurrence in the ensemble of games defined by the standard probabilistic model of (h, n, P)-trees.

Assume that the evaluation function V computed at each terminal node of a (h, n, P*s$_n$)-tree reflects the likelihood of that node being a WIN position. The quality of such an estimate can be characterized by the pair of distribution functions defined in (9). Given the pair Fy(x) and FL(X), one can compute the two conditional distributions of V_d, the minimax value of the root node:

$$F_W^d(x) = P[V_d \le x \text{ given that the root is WIN}]$$

$$F_L^d(x) = P[V_d \le x \text{ given that the root is LOSS}]$$

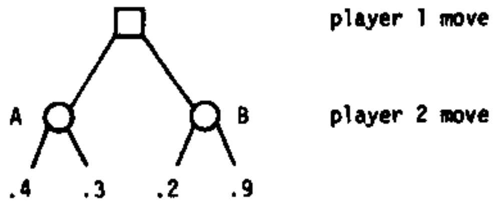
The analysis shows that, for every starting pair (F_W, F_L), the back-up pair (F_W^d, F_L^d) satisfies $\lim_{d\to\infty} (F_W^d - F_L^d) = 0$. This implies that as the search depth increases, the minimax value of the root node possesses the same statistics regardless of whether the root is in fact a WIN or a LOSS. Thus, the ability to discriminate between a WIN and a LOSS situation deteriorates by the minimax back-up procedure.

To understand why the filtering argument fails in the case of game-trees, consider the task of estimating the value of an arbitrary function £(xi» X£. ... x$_n$) on the basis of the estimates xj, x$_2$, ... x$_n$ of its arguments. Knowing x1 ... x$_n$ can Improve the estimation of y if we integrate them according to strict statistical rules, as by forming the conditional expectation of y given X1 ... x$_n$. Instead, the minimax procedure amounts to taking y(xj, £2, ... x$_n$) as an estimate for y— It computes the minimax value of the estimators rather than estimates the minimax.

This point can be further illustrated by the following game-tree. Assume that the terminal values signify the probability that player 1 can force a WIN from these positions. Which move should be selected, towards A or towards B? The minimax evaluation procedure would assign to node A the value .3, to node B the value .2, and so would lead player 1 to prefer A over B. On the other hand, if one computes the probabilities that nodes A or B are WIN (for player 1) on obtains:

$$P(A \text{ is WIN}) = P(\text{all A's successors are WIN}) = .3 \times .4 = .12$$

$$P(B \text{ in WIN}) = P(\text{all B's successors are WIN}) = .9 \times .2 = .18$$



Clearly, B is to be preferred. It is obvious from this example that if one wants to maximize the chances of choosing WIN positions, the minimax rule is inadequate and should be replaced by product-propagation rules, *nPj* for MIN nodes and l-n(l-Pi) for MAX nodes. If these propagation rules 1 are used, the phenomenon of pathology should be reduced, if not eliminated.

However, the example also facilitates a line of defense 1n favor of the minimax rule. In practical game-playing, one 1s not concerned with maintaining a WINNING position, winning regardless of what the opponent does, but simply with beating a fallible opponent. Such an opponent, if he shares our assessment of the terminal values, can be predicted to choose the left move from position B and the right move from position A. Therefore, on the basis of such a prediction, we have:

P(player 1 wins from A) » .3

P(player 1 wins from B) ■ .2

which agrees with the minimax calculation. In summary, although the minimax rule is inadequate when playing against an omnipotent opponent, tacitly it contains a realistic model of a fallible opponent who shares the knowledge and limitations of player 1, and therefore should be effective against such opponents. This, perhaps, may account for the fact that pathology 1s not observed in practical game-playing programs; the quality of play usually Improves with search depth.

Another reason for the absence of pathology 1n practical game-playing could be that the evaluation functions become more accurate toward the end of the game, possessing Increased visibility then, an effect not included in the preceding analysis. Nau [24] has recently demonstrated that pathological behavior occurs 1n a specific game despite an Increase in evaluation function accuracy. Determining the rate of Improved accuracy necessary for combating pathology remains an open theoretical problem.

REFERENCES

[1] Hart, P., N. NIlsson, and B. Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths." IEEE Trans, on Systems Science and Cybernetics, Vol. SSC-4, No. 2. July 1968, pp. 100-107.

[2] Pohl, I. "First Results on the Effect of Error in Heuristic Search." In B. Meltzer and D. M1ch1e (Eds.), Machine Intelligence 5. Edinburgh: Edinburgh University Press, 1970.

[3] Munyer, J. "Some Results on the Complexity of Heuristic Search 1n Graphs." Technical Report HP-76-2, Information Sciences Department, University of California, Santa Cruz, September 1976.

[4] Vanderbrug, G. "Problem Representation and Formal Properties of Heuristic Search." Information Sciences, Vol. 11, No. 4, 1976.

[5] Pohl, I. "Practical and Theoretical Considerations In Heuristic Search Algorithms." In E. Elcock and D. M1ch1e (Eds.), Machine Intelligence 8, Chichester, England: Ellis Horwood

[6] Gaschnig, J. "Performance Measurement and Analysis of Certain Search Algorithms." Ph.D. Dissertation, Department of Computer Science, Carnegie-Mellon University, 1979.

[7] Gelperln, D. "On the OptImallty of A*." Artificial Intelligence, Vol. 8, No. 1, 1977, pp. 69-76.

[8] Huyn, N., R. Dechter, and J. Pearl. "Probabilistic Analysis of the Complexity of A*." Artificial Intelligence, Vol. 15, No. 3, 1980, pp. 241-254.

[9] Pearl, J. "The Utility of Precision 1n Search Heuristics." UCLA-ENG-CSL-8065, Cognitive Systems Laboratory, University of California, Los Angeles, October 1980.

[10] NIlsson, N. Principles of Artificial Intelligence. Palo Alto, CA: Tioga Publishing Company, 1980.

[11] Pearl, J. "The Complexity of Non-Adm1ss1ble Search Heuristics." In preparation.

[12] Pohl, I. "Heuristic Search Viewed as Path Finding In a Graph." Artificial Intelligence, Vol. 1, 1970, pp. 193-5ET

[13] Pearl, J. "Asymptotic Properties of MInImax Trees and Game-Searching Procedures." Artificial Intelligence, Vol. 14, No. 2, 1980, pp. 113-138.

[14] Slagle, J. R., and J. K. Dixon. "Experiments with Some Programs that Search Game Trees." Journal of the ACM, Vol. 2, 1969, pp. 189-207.

[15] Fuller, S. H., J. G. Gaschnig, and J. J. GIllogly. "An Analysis of the Alpha-Beta Pruning Algorithm." Department of Computer Science Report, Carnegie-Mellon University, 1973.

[16] Knuth, 0. E., and R. N. Moore. "An Analysis of Alpha-Beta Pruning." Artificial Intelligence, Vol. 6, 1975, pp. 293-326.

[17] Baudet, G. M. "On the Branching Factor of the Alpha-Beta Pruning Algorithm." Artificial Intelligence, Vol. 10, 1978, pp. 173-199.

[18] Pearl, J. "The Solution for the Branching Factor of the Alpha-Beta Pruning Algorithm." UCLA-ENG-CSL-8019, Cognitive Systems Laboratory, University of California, Los Angeles, April 1980. To be published 1n Communications of the ACM.

[19] Pearl, J. "A Space-Efficient 0n-L1ne Method of Computing Quantile Estimates." UCLA-ENG-CSL-8018, Cognitive Systems Laboratory, University of California, Los Angeles, December 1980. To be published in Journal of Algorithms.

[20] Stockman, G. "A Minimax Algorithm Better Than Alpha-Beta?" Artificial Intelligence, Vol. 12, 1979, pp. 179-IW:

[21] Rolzen, I., and J. Pearl. "A MInimax Algorithm Better Than Alpha-Beta?: Yes and No." In preparation.

[22] Tarsi, M. "Optimal Searching of Some Game Trees." UCLA-ENG-CSL-8108, Cognitive Systems Laboratory, University of California, Los Angeles, May 1981.

[23] Nau, D. S. "Pathology on Game Trees: A Summary of Results." Proceedings of the first National Conference"^ Artificial Intelligence, August 1M6, pp. 162-164.

[24] Nau, D. S. "Pearl's Game 1s Pathological." Technical Report TR-999, Computer Science Department, University of Maryland, January 1981.