OPTIMAL ALLOCATION OF VERY LIMITED SEARCH RESOURCES

David Mutchler†

Naval Research Laboratory, Code 7591
Washington, D.C. 20375-5000

Abstract

This paper presents a probabilistic model for studying the question: ***given n search resources, where in the search tree should they be expended?*** Specifically, a least-cost root-to-leaf path is sought in a random tree. The tree is known to be binary and complete to depth N. Arc costs are independently set either to 1 (with probability p) or to 0 (with probability $1-p$). The cost of a leaf is the sum of the arc costs on the path from the root to that leaf. The searcher (scout) can learn n arc values. How should these scarce resources be dynamically allocated to minimize the ***average*** cost of the leaf selected?

A natural decision rule for the scout is to allocate resources to arcs that lie above leaves whose current expected cost is minimal. The bad-news theorem says that situations exist for which this rule is nonoptimal, no matter what the value of n. The good-news theorem counters this: for a large class of situations, the aforementioned rule is an optimal decision rule if $p \leq 0.5$ and within a constant of optimal if $p > 0.5$. This report discusses the lessons provided by these two theorems and presents the proof of the bad-news theorem.

I Informal description of the problem

Searching the state-space for an acceptable solution is a fundamental activity for many AI programs. Complete search of the state-space is typically infeasible. Instead, one relies on whatever heuristic information is available. Interesting questions then arise as to how much speed-up is obtained and at what price.

Many authors have evaluated the complexity of algorithms that invoke heuristic search [1, 3, 6, 7, 9, 10, 11]. A typical question asked is:

How fast can the algorithm find an optimal (nearly-optimal) (probably nearly-optimal) solution?

This paper focuses upon the inverse question:

Given n search resources, how good a solution can one obtain?

This inverse question is appropriate for real-time processes characterized by an insistence upon an answer (decision) after X seconds have passed. For example, a chess-playing program is limited by the external chess clock. A speech recognizer should maintain pace with the speaker. In these and other processes, search resources are ***very*** limited; even linear time may not be fast enough.

Heuristics are often said to offer "solutions which are good enough most of the time" [4, page 6]. The converse of this phrase implies that heuristics will, by definition, fail some of the time. Worst-case analysis is unilluminating—***any*** algorithm using the heuristic information will, on occasion, perform poorly. One is forced, reluctantly perhaps, to turn to probabilistic, average-case analysis. Karp and Pearl said it well [10]:

Since the ultimate test for the success of heuristic methods is that they work well "most of the time", and since probability theory is our principal formalism for quantifying concepts such as "most of the time", it is only natural that probabilistic models should provide a formal ground for evaluating the performance of heuristic methods quantitatively.

In agreement with this philosophy, this paper seeks the algorithm whose ***average*** result is best. It must be emphasized from the outset that any conclusions drawn from average-case analysis depend fundamentally on the underlying probability distribution assumed. The concluding section of this paper discusses whether the results of this paper do in fact apply to real-world algorithms.

II The formal model

This paper restricts its interest to a particular variety of heuristic search—finding a least-cost root-to-leaf path in a tree. The trees considered are binary trees complete to depth N. The arcs of the trees are assigned costs randomly and independently; each arc costs either 1 (with probability p) or 0 (with probability $1-p$). The cost of a leaf is the sum of the costs of the arcs on the path from the root to the leaf. This arc-sum method for assigning dependent leaf costs has been used by several researchers [2, 5, 10, 12, 13, 17].

The searcher (hereafter called the ***scout***) begins with exactly the foregoing information. The scout acquires additional information by ***expanding*** arcs, i.e., learning the ***actual*** cost (either 1 or 0) of the arc expanded. At each stage of the search, the scout can expand any arc on the frontier of the search (any arc whose parent has been expanded already). Backtracking incurs no extra penalty. Recall that this paper focuses upon ***limited resources.*** Model this by insisting that the scout halt after n arc expansions. The general then comes forward to select a leaf whose cost is (in general) a random variable. The general seeks a low-cost leaf, of course. The optimal decision-strategy for the general is easily seen. The interesting issue is how the scout should allocate the n arc expansions.

Time for an example. Let $p = 0.7$ and N (depth of tree) be four. Suppose the scout expands the left arc beneath the root and finds that its cost is 1. Suppose further that the scout began with two arc expansions available, hence has only one more to use. Of the three arcs on the frontier of the search, which should the scout expand for this final action? Please pause a moment to decide for yourself.

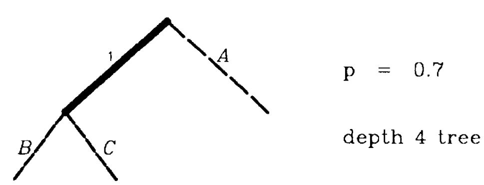

Figure 1. Which arc should the scout expand?

† This report describes work done in the Department of Computer Science at Duke University. It was supported in part by the Air Force Office of Scientific Research, Air Force Systems Command under Grant AFOSR 83-0205.

The natural answer seems to be arc A, perhaps because leaves beneath A have lower expected cost (given the current information) than those beneath the other frontier arcs. In fact, expanding arc A is *wrong*. The expected cost of the leaf selected by the general will, on average, be strictly lower if the scout expands arc B or C instead of arc A. The next section contains the generalized version of this startling result, and a reassuring theorem to counter it.

The scout is initially quite uncertain about the leaf costs. As the search proceeds, leaf costs beneath expanded arcs become less indeterminate. This accumulation of knowledge models what a "generic" heuristic rule might provide. This heuristic rule is better described as vague than as error-prone, as is fitting for a study of the *utilization* by various algorithms of heuristically acquired information. Contrast this with studies concerned with how the amount of error in the heuristic rule affects a given algorithm [6, 8, 9, 11, 14, 15, 16, 19].

This model differs from that used by Karp and Pearl [10] in only two aspects. First, they expand nodes (learning the costs of both arcs below the node) instead of arcs. This difference is not of consequence; more on this later. The second and more significant difference is the presence of a cutoff beyond which search cannot continue. This cutoff models the limitation on resources. The present model is the appropriate model if the search process must shut down and an answer must be given after a certain amount of time has elapsed, that is, after a certain number of arc expansions.

III Results

First consider the general's decision. For any frontier arc α, define the *zero-value of arc* α to be

$$\textit{sum of the costs of the arcs from the root to } \alpha$$
$$+ \; p \times \textit{(distance from } \alpha \textit{ to the bottom of the tree)}$$

For example, in the depth 4 tree of Figure 1, arcs B and C each have zero-value $1 + 3p$, while arc A has zero-value $4p$. The zero-value of an arc is the expected cost of each of the leaves beneath that arc based on current information. Since the general will received no further data, the optimal decision for the general is to select any leaf beneath the arc whose zero-value is smallest, breaking ties arbitrarily.

The scouting activity can be modeled as a finite-horizon Markov decision process [18]. The *score* of a scouting algorithm is the expected value of the smallest zero-value corresponding to a frontier arc after n arc expansions. An *optimal algorithm* is one that minimizes this score. Because zero-values are themselves expected costs, the score of a scouting algorithm is an expected expected cost. Note that an optimal algorithm does not usually discover the in-fact optimal path in the tree. An optimal algorithm finds a path whose *average* cost is no larger than the average cost of the path discovered by any other algorithm restricted to n arc expansions.

What are the optimal scouting algorithms? As discussed above, the general should choose any leaf below the arc whose zero-value is smallest. (The scout will assume that the general behaves optimally.) Perhaps the *same* policy should be used by the scout. Call this scouting algorithm—expand the arc whose zero-value is smallest—the *greedy policy*. The following theorem relates the bad news: the greedy policy is not optimal.

Bad-news theorem. No matter how many arc expansions remain, if p is large enough, there exist situations from which the greedy decision is *not* an optimal decision. Such situations can arise even if the dictates of the greedy policy have been followed from the beginning of search.

The bad-news theorem says that the scout should, under certain circumstances, apply search resources to an arc that is *not* currently the arc that looks "best", insofar as the final decision goes. Return to the example in Figure 1. The scout should expand arc B instead of arc A because, in that example, *information* about the second-best arc (B) is more valuable than *information* about the best arc (A).

This distinction between information and path-cost complicates the optimal scouting policy. A priori, one might have guessed that the two would coincide, i.e., that information is most valuable when gathered for the path whose expected cost is currently smallest. The bad-news theorem denies that these two measures are the same. The denial is strong—the measures may differ whether there be 5 or 500 arc expansions left.

Note the order of quantification in the bad-news theorem. The theorem says that for every n (number of arc expansions remaining), there exist values of p and situations from which the greedy decision is a mistake. The models considered in this paper are parameterized by p. Specifying p specifies the model. The bad-news theorem says: if you tell me how many arc expansions you have left, I will give you a specific model (value of p) and a specific state (subtree with its arc costs labeled) from which the greedy decision is wrong. What can be said for a *fixed* model, i.e., for fixed p? There the sun shines brightly. For a large class of situations, the greedy decision can be proved optimal.

Good-news theorem. Consider any state from which the scout cannot reach a leaf before exhausting the n remaining arc expansions. From any such state:
for $p \leq 0.5$, the greedy decision is an optimal decision;
for $0.5 < p \leq 0.618$, the greedy decision is optimal if $n \geq 2$;
for $0.618 < p \leq 0.682$, the greedy decision is optimal if $n \geq 3$.
(The numbers 0.618 and 0.682 are more accurately described as the solutions to the equations $p^2 = 1 - p$ and $p^3 = 1 - p$, respectively.)

So, if the scout is exploring a tree for which $p \leq 0.5$, the greedy policy can be used with complete confidence. If p is between 0.5 and 0.618, the good-news theorem tells the scout that *only* at the last arc expansion might the greedy policy err; for p between 0.618 and 0.682, only the last two arc expansions are suspect. Section VI discusses the gravity of the restriction that the scout be unable to reach a leaf (i.e., that search resources are *very* limited).

The proof of the good-news theorem contains three inductions on n, one for each of the three ranges of p listed. The basis case of each induction is the smallest value of n for which the greedy decision is guaranteed to be optimal: $n = 1$ for $p \leq 0.5$, $n = 2$ for $0.5 < p \leq 0.618$, and $n = 3$ for $0.618 < p \leq 0.682$. Interestingly, the *induction* step for the third range also applies to any $p \leq 0.99$. It fails for larger p only because the proof uses a numerical step for which arithmetic roundoff plays a role eventually. Unfortunately, the value of n for the basis case grows as p increases (see the conjecture below), so that the basis case involves increasingly complicated algebraic comparisons. This prevents the proof of the good-news theorem from extending beyond 0.682. There is no evidence that the basis case *fails* for larger values of p. Indeed, it would be strange—though of course possible—if the basis case were to hold for p up to 0.682 but fail for larger p, while the induction step provably works for p up to 0.99. The success of the induction step provides theoretical support for the following conjecture.

Conjecture. Consider any fixed model, i.e., any fixed value for p. Let M be the k for which $p^k = 1 - p$. That is, let $M = \log_p (1-p)$. If M or more arc expansions remain, the greedy decision is an optimal decision from *any* situation from which the scout cannot reach a leaf before exhausting the remaining arc expansions.

Corollary. Suppose the scout begins with no more arc expansions available than the depth of the tree. For any fixed p, the expected score of the greedy policy is within a constant of the expected score of the optimal scouting algorithm.

Computer simulation provides additional, albeit anecdotal, support for the conjecture. It is not hard to compute mechanically the optimal decision for any specific state, model (value of p), and small value of n. A wide variety of such runs yielded no exceptions to the conjecture. The restriction of the simulations to small values of n (≤ 10) is *not* particularly worrisome, because (as explained above) only the basis case needs to be verified.

IV Proof of the bad-news theorem

We prove the bad-news theorem for $n = 2$. The proof for larger n is analogous; see [12].

Choose p large enough that $p^2 > 1-p$. The troublemaking state is the example seen earlier. We show that expanding arc A is a nonoptimal decision if the scout has exactly two arc expansions to apply to the state shown in Figure 1. Suppose the contrary: suppose there is an optimal algorithm, call it algorithm OPT, that expands arc A from the state pictured. Here *optimal* means that the expected value of the leaf chosen by the general is minimized if the scout uses algorithm OPT. There are many optimal algorithms. Without loss of generality, algorithm OPT can be taken to be a deterministic algorithm.

Imagine that OPT expands arc A and finds that it has cost 1. OPT now has only one arc expansion left. It can expand any of the four frontier arcs—all have the same zero-value. Algorithm OPT, being deterministic, must fail to expand three of these frontier arcs. The expected value of OPT is the same no matter which three are skipped. Hence OPT can be chosen to skip the two arcs below arc A and (say) arc B, in the event that arc A has value 1. That is, OPT expands arc C if its previous expansion of arc A yielded a 1-arc.

We now define an algorithm called MIMIC that uses OPT as a subroutine. We will show that MIMIC performs better than OPT (on average), thus contradicting the assumption that OPT (and the greedy decision) are optimal. Algorithm MIMIC mimics OPT, but with arcs A and B reversed. OPT expands A from the state in Figure 1, so MIMIC expands B from that state. MIMIC continues its mimicry (still with A and B reversed) on its last arc expansion, as shown below. (The fat line indicates the arc to be expanded.)

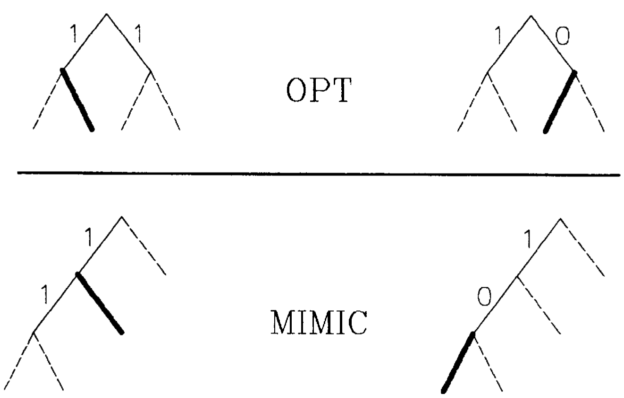

Figure 2. The final arc expansion by MIMIC and OPT

Return to the state pictured in Figure 1, from which 2 arc expansions remain. The expected score of algorithm OPT is the weighted average of the expected cost the general incurs when the scout uses OPT. This average is over all 2^2 trees τ that the scout might hand the general after 2 expansions from the pictured state. The four such trees possible after 2 expansions of the state in Figure 1 are shown in the top half of the following figure. (The highlighting therein becomes meaningful shortly.)

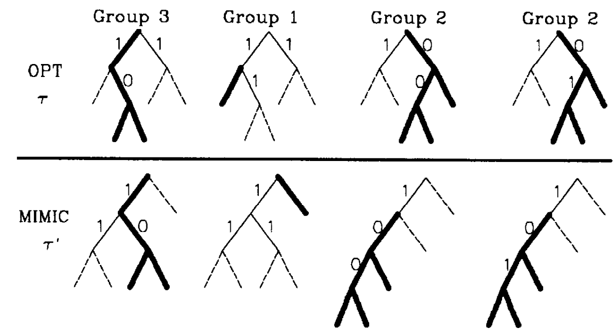

Figure 3. All the possible trees the general might be given

The expected score of MIMIC is likewise a weighted average, but over a ***different*** set of 2^2 trees, pictured in the bottom half of Figure 3. The action "exchange arc A and the subtree beneath it with arc B and its attached subtree" provides a 1-1 correspondence between these two sets of trees. (The correspondence is shown by vertical pairing in Figure 3.) Note that any tree τ and its corresponding tree τ' are equally likely. It follows that the difference between the expected score of OPT and the expected score of MIMIC is the difference between the score of OPT on τ and the score of MIMIC on the corresponding tree τ', averaged over all 2^2 trees τ that OPT might reveal to the general. Let us compute this difference as just described, but in three groups.

Group 1: consider any tree τ on which the general selects a leaf below arc B when the scout uses OPT on τ. The second tree in the top half of Figure 3 falls in this group. (Highlighted arc B is tied with two other frontier arcs for smallest zero-value in that tree. Let the general break ties in favor of arc B. This is as good a tie-breaking rule as any.) When the scout uses MIMIC on the corresponding tree τ', arc B and the subtree beneath it in τ also appear in τ', but from a better starting point. (Arc B is alone in the example in Figure 3; the visible subtree below it is null.) To be precise, algorithm MIMIC scores (on average) $1-p$ ***better*** on τ' than OPT does on τ, for any tree τ in Group 1.

Group 2: consider any tree τ on which the general selects a leaf below arc A when the scout uses OPT on τ. The third and fourth trees in the top half of Figure 3 fall in this group. When the scout uses MIMIC on τ', arc A and the subtree beneath it in τ also appear in τ', but from a worse starting point. (These subtrees are highlighted in Figure 3.) Algorithm MIMIC scores (on average) no worse than $1-p$ ***worse*** on τ' than OPT does on τ, for any tree τ in Group 2.

Group 3: consider any tree τ on which the general selects a leaf below neither arc A nor arc B when the scout uses OPT on τ. The first tree in the top half of Figure 3 falls in this group. The same frontier arc in τ beneath which the general chose a leaf also appears in τ'. (The common subtree is highlighted in Figure 3.) The expected score of the general (and MIMIC) is no worse on τ' than the expected score of the general (and OPT) on τ, for any tree τ in this group.

Conclude: if the collective likelihood of Group 1 exceeds that of Group 2, algorithm MIMIC performs (on average) better than algorithm OPT. A trite calculation shows this to be the case for the trees in Figure 3. The following discussion suggests how the proof works when $n > 2$.

What is the probability of Group 1? Return to the situation of Figure 1. In the event that both remaining arc expansions reveal only 1-arcs in the tree τ uncovered by OPT, arc B is tied for minimum zero-value at the conclusion of search. (Remember: arc B is *chosen* as an arc that OPT will *not* expand in just this case and the general breaks ties in favor of arc B.) Then

$$Pr\ (Group\ 1) \geq Pr\ (two\ 1\text{-}arcs) = p^2$$

What is the probability of Group 2? Remember that algorithm OPT—by design—will ignore the arcs below arc A if arc A has cost 1. The only hope for Group 2 is that arc A has cost 0. That is,

$$Pr\ (Group\ 2) \leq Pr\ (arc\ A\ has\ cost\ 0) = 1-p$$

By choice of model, $p^2 > 1-p$ so Group 1 is more likely than Group 2. This contradicts the assumption that OPT is optimal and shows that the greedy decision is not an optimal decision in this construction.

V Proof of the good-news theorem

The proof of the good news is long and involved. This section presents some of the more appetizing parts of the proof, to give its flavor. The reader's pardon is asked for the lack of rigor in the presentation that follows. See [12] for a careful exposition of all the details.

The devices used for $p \leq 0.5$ (where the greedy policy is optimal) are somewhat different from those used for $p > 0.5$ (where it is not). Within each half, further division is necessary as well. In each subcase, however, the proof is by induction on n, the number of arc expansions remaining. The basis cases compute the optimal policy explicitly.

Consider an arbitrary state z. Because of the assumption that the scout can no longer reach a leaf, the state is characterized by the zero-values of the frontier arcs. Let α denote the smallest of these zero-values. Let "arc α" denote the arc corresponding to α. Let P^β be an optimal algorithm. If this algorithm expands arc α from state z, the greedy decision (expand the arc whose zero-value is smallest) is an optimal decision, completing the induction step of the proof. Suppose the contrary: suppose this optimal algorithm expands some other arc whose zero-value is (say) $\beta \geq \alpha$; call this other arc β. (Hence the name of the algorithm.) Define P^α to be the policy that expands arc α and then proceeds optimally. The goal of the rest of this proof is to show that policy P^α performs (on average) as well as optimal policy P^β. Achieving this goal will demonstrate that expanding arc α is also an optimal decision, hence that the greedy decision is an optimal decision, hence (by induction) that the greedy policy is optimal.

First consider the case in which β exceeds α by at least p, i.e., $\beta \geq \alpha + p$. In this case, α will be the smallest zero-value after policy P^β expands arc β, no matter what the result of that expansion. By the induction hypothesis, policy P^β (being optimal) will continue by expanding the arc whose zero-value is smallest, namely, arc α. In other words, policy P^β is (in this case) equivalent to the policy that expands both α and β without regard to order. But such a policy is certainly no better than the more flexible policy P^α that expands α and then proceeds *optimally*. The goal described in the preceding paragraph has been achieved in this case.

Turn now to the case in which $\beta < \alpha + p$. The argument in this case operates by considering the performance of algorithms P^α and P^β as functions of β. It can be shown that:

a. The expected score the general achieves when the scout uses either of these algorithms is a continuous, piecewise-linear function of β.

b. At $\beta = \alpha$, the general achieves the same expected score by using either scouting algorithm.

c. For any scouting algorithm P, let the phrase *β wins by using policy P* be shorthand for the event *the general chooses a leaf below arc β when scouting policy P is used.* The slope of each linear segment in the graph of the general's expected score when the scout uses P^α is simply the probability that *β wins by using P^α*. A similar statement holds for algorithm P^β.

It follows that one way to show our goal (policy P^α performs as well as policy P^β) is to show that

$$\Pr\ (\ \beta\ wins\ by\ using\ P^\beta\ ) \geq \Pr\ (\ \beta\ wins\ by\ using\ P^\alpha\ ) \qquad (\dagger)$$

for *any* value of β such that $\alpha < \beta < \alpha + p$.

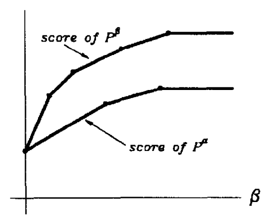

Figure 4. Piecewise linearity

The "meat" of the proof is devoted to showing the truth of inequality ($\dagger$). This is done by conditioning on the possible costs of arcs α and β and meticulously examining the four cases that result. The central theme is an application of the induction hypothesis: if "enough" 0-arcs lie on and beneath the arc that is first expanded, the policy (either P^α or P^β) never leaves the subtree beneath that arc; hence that first expanded arc *wins*. For example, if arc β has cost 0 and has an infinite path of 0-arcs directly beneath it, β *must* win when policy P^β is used. The classical theory of branching processes provides an easy formula for the probability that there is such an infinite path. New results for branching random walks developed for this proof give stronger approximations to the "win probabilities". These new results are of particular interest because *numerical* approximations are used to provide *analytic* bounds.

VI Discussion

Is the "limited resources" problem relevant to the real world? Is it reasonable that after n arc expansions, the search halts? This absolute cutoff is not typical in AI problem-solving programs. Only real-time processes might be so described. Nonetheless, I view this aspect of the model as a significant contribution to investigation of optimal, dynamic allocation of search resources. The cutoff clearly separates the *search* process from the *final-decision* process. Search gathers information for two purposes: to optimize some final decision, and to assist the accumulation of additional useful information. The present model, by design, accents this latter purpose.

One reasonable alternative is a staged search: the scout gains some information; the general makes some decision; then the process iterates, although still with some final cutoff. Such a model is appropriate if outside factors are involved: an unpredictable opponent, for instance, or events whose outcome is impervious to search but can be experienced as consequences of the general's decisions. A second alternative is to abandon the absolute cutoff. Allow the general to direct the scout to continue search, at some additional cost. The problem then becomes an optimal stopping process. Both of these alternatives are attractive. It is their analysis that appears forbidding.

Is our arc-sum model the right model for studying search with limited resources? Without doubt, the present model is simple-minded. Some of its simplicity has merit, capturing essence without detail. The restriction to binary tress with two-valued arcs falls into this class. On the other hand, the assignment of leaf costs by arc costs that are independent and identically distributed is artificial. Happily, the foregoing results are oblivious to some of the assumptions. Any value is permitted for the heuristic parameter p. The bad-news theorem can be shown to apply to any branching factor. Both the bad-news theorem and a weaker version of the good-news theorem apply to the model in which nodes are expanded instead of arcs [12].

Does the assumption that search resources be very *limited sabotage the substance of the good-news theorem?* From a practical standpoint, this restriction (that the scout be unable to reach a leaf) is a big winner. Without it, all sorts of rough-edged boundary problems are encountered. For example, a prejudice appears against expanding arcs at the bottom of the tree because such expansions cannot be followed-up. In addition to this practical justification, there is a heuristic argument that the restriction is of little effect. The argument goes like this. The search begins well away from leaves. Whether the tree has depth 50 or 5000 should have little effect while the search is rummaging around level 5 or so. Any reasonable algorithm (including the greedy policy) has a breadth-first character whenever 1-arcs are found. Conclude: the search typically will not reach a leaf. So long as this is the case, the analysis in this paper works.

Open questions: is the conjecture in section III true? Can similar results be obtained for generalizations of the present model? (In particular, what happens if one allows arc values other than 0 and 1? a random branching factor? a depth-dependent distribution for arc values?) Do the lessons of this study apply to other models of heuristic search? More to the point, do the lessons apply in practice? Is the greedy policy a good algorithm if the scout misestimates p?

What do the results in this study REALLY say? These results should not be taken as literal advice for finding a least-cost root-to-leaf path in a tree. The bad news and good news should be assimilated in a broader sense, as follows.

Bad news: intuition about heuristic search is not always right. The example at the beginning of this paper shows that one's intuitions can be firmly set, and firmly wrong. Our model and the bad-news theorem show that blind adherence to custom may prevent optimal use of search resources. In particular, there is a real difference between where best to gather information and how best to utilize it.

Good news: theoretical justification can be provided for the intuition that the best information is acquired from the path that currently looks best. As the bad-news theorem shows, this intuition fails when $p > 0.5$. But for $p \leq 0.5$, the intuition is sound; even for $p > 0.5$, the good-news theorem and accompanying conjecture show that the intuition provides a good approximation.

In sum, this study of heuristic search establishes that this intuition—*search* the path you currently judge *best*—can justifiably be labeled a heuristic. It sometimes fails, but on average provides a result close to optimal.

References

1. Bagchi, A. and A. Mahanti, "Search algorithms under different kinds of heuristics — A comparative study," *JACM* **30**(1) pp. 1-21 (January 1983).

2. Ballard, Bruce W., "The *-minimax search procedure for trees containing chance nodes," *Artificial Intelligence* **21** pp. 327-350 (1983).

3. DeWitt, H.K., "The theory of random graphs with applications to the probabilistic analysis of optimization algorithms," Ph.D. dissertation, Computer Science Dept., University of California, Los Angeles (1977).

4. Feigenbaum, E.A. and J. Feldman, *Computers and Thought*, McGraw-Hill Book Company, New York (1963).

5. Fuller, S.H., J.G. Gaschnig, and J.J. Gillogly, "An analysis of the alpha-beta pruning algorithm," Dept. of Computer Science Report, Carnegie-Mellon University, Pittsburgh, PA (1973).

6. Gaschnig, John, "Performance measurement and analysis of certain search algorithms," Ph.D. dissertation, Technical Report CMU-CS-79-124, Computer Science Dept., Carnegie-Mellon University (1979).

7. Golden, B. L. and M. Ball, "Shortest paths with Euclidean distances: An explanatory model," *Networks* **8**(4) pp. 297-314 (Winter 1978).

8. Harris, Larry R., "The heuristic search under conditions of error," *Artificial Intelligence* **5** pp. 217-234 (1974).

9. Huyn, Nam, Rina Dechter, and Judea Pearl, "Probabilistic analysis of the complexity of A*," *Artificial Intelligence* **15** pp. 241-254 (1980).

10. Karp, R.M. and J. Pearl, "Searching for an optimal path in a tree with random costs," *Artificial Intelligence* **21** pp. 99-116 (1983).

11. Munyer, J., "Some results on the complexity of heuristic search in graphs," Technical Report HP-76-2, Information Sciences Dept., University of California, Santa Cruz (1976).

12. Mutchler, David C., "Search with very limited resources," Ph.D. dissertation, Duke Technical Report CS-1986-10, Duke University Department of Computer Science (1986).

13. Newborn, M. M., "The efficiency of the alpha-beta search on trees with branch-dependent terminal node scores.," *Artificial Intelligence* **8** pp. 137-153 (1977).

14. Pearl, Judea, "Knowledge versus search: A quantitative analysis using A*," *Artificial Intelligence* **20** pp. 1-13 (1983).

15. Pohl, Ira, "First results on the effect of error in heuristic search," pp. 219-236 in *Machine Intelligence 5*, ed. Bernard Meltzer and Donald Mitchie,American Elsevier, New York (1970).

16. Pohl, Ira, "Practical and theoretical considerations in heuristic search algorithms," pp. 55-72 in *Machine Intelligence 8*, ed. E.W. Elcock and D. Mitchie,Wiley, New York (1970).

17. Reibman, Andrew L. and Bruce W. Ballard, "The performance of a non-minimax search strategy in games with imperfect players," Duke Technical Report CS-1983-17, Duke University Department of Computer Science (1983).

18. Ross, Sheldon M., *Introduction to Stochastic Dynamic Programming*, Academic Press, New York (1983).

19. VanderBrug, Gordon J., "Problem representations and formal properties of heuristic search," *Information Sciences* **11** pp. 279-307 (1976).