AN EXPECTED-COST ANALYSIS OF BACKTRACKING AND NON-BACKTRACKING ALGORITHMS

C.J.H. McDiarmid
Department of Statistics
University of Oxford
Oxford England 0X1 3TG
email: MCD@vax.oxford.ac.uk

G.M.A. Provan
Department of Computer and Information Science
University of Pennsylvania
Philadelphia PA 19104-6389 USA
email: provan@cis.upenn.edu

Abstract

Consider an infinite binary search tree in which the branches have independent random costs. Suppose that we must find an optimal (cheapest) or nearly optimal path from the root to a node at depth n. Karp and Pearl [1983] show that a bounded-lookahead backtracking algorithm A2 usually finds a nearly optimal path in linear expected time (when the costs take only the values 0 or 1). From this successful performance one might conclude that similar heuristics should be of more general use. But we find here equal success for a simpler *non-backtracking* bounded-lookahead algorithm, so the search model cannot support this conclusion. If, however, the search tree is generated by a branching process so that there is a possibility of nodes having no sons (or branches having prohibitive costs), then the non-backtracking algorithm is hopeless while the backtracking algorithm still performs very well. These results suggest the general guideline that backtracking becomes attractive when there is the possibility of "dead-ends" or prohibitively costly outcomes.

1 INTRODUCTION

Many algorithms considered in operations research, computer science and artificial intelligence may be represented as searches or partial searches through rooted trees. Such algorithms typically involve backtracking but try to minimize the time spent doing so (e.g. [Bitner and Reingold, 1975; Brown and P. W. Purdom, 1981; Brown and P. W. Purdom, 1982; Dechter, 1990; Haralick and Elliott, 1980; Karp, 1976; Knuth, 1975; Nudel, 1983; P. W. Purdom, 1983; Stone and Stone, 1986]). Indeed for some problems it may be best to avoid backtracking [de Kleer, 1984].

The paper extends work of [Karp and Pearl, 1983], and gives a probabilistic analysis of backtracking and non-backtracking search algorithms in certain trees with random branch costs. We thus cast some light on the question of when to backtrack: it seems that backtracking is valuable just for problems with "dead-ends" (or outcomes with prohibitively high costs).

Let us review briefly the model and results of Karp and Pearl [1983]. They consider an infinite search tree in which each node has exactly two sons. The branches have independent (0, I)-valued random costs X, with p = *P(X* = 0).[1] The problem is to find an optimal (cheapest) or nearly optimal path from the root to a node at depth *n.*

The problem changes nature depending on whether the expected number 2p of zero-cost branches leaving a node is > 1, = 1 or < 1. When 2p > 1 a simple uniform cost breadth-first search algorithm *AI* finds an optimal solution in expected time *O(n);* and when 2p = 1 this algorithm takes expected time $0(n^2)$. When *2p* < 1 any algorithm that is guaranteed to find a solution within a constant factor of optimal must take exponential expected time. However, in this case a "bounded-lookahead plus partial backtrack" algorithm *A2* usually finds a solution close to optimal in linear expected time. This successful performance of the backtracking algorithm *A2* for the difficult case when 2p < 1 seems to suggest that similar backtrack-based heuristics should be of more general use for attacking NP-hard problems.

This paper shows that a simple *non-backtracking* bounded-lookahead algorithm .43 performs as successfully as the backtracking algorithm A2, on the basis of this search model. Similar comments hold if we allow more general finite random costs on the branches.

However, there is a qualitative difference if we allow nodes to have no sons (or allow branches to have infinite costs) so that there are "dead-ends". We extend Karp and Pearl's work by considering search in random trees generated by a branching process, where the branches have independent random finite costs *X*. (This model includes the case of infinite costs—nodes would just produce fewer sons). In this extended model, let *m* be the mean number of sons of a node, let po be the probability that a node has no sons, and as before let p = *P(X* — 0).

Our results concerning algorithms *AI* and *A2* are natural extensions of Karp and Pearl's results. Thus the uniform cost algorithm *AI* finds an optimal solution in linear expected time if mp > 1 and in quadratic expected time if *mp* = 1. If *mp* < 1 then any algorithm with a constant performance guarantee must take exponential

[1] We have swapped *p* and 1 — *p* from the original paper

expected time, but the backtracking algorithm *A2* finds a nearly optimal solution in linear expected time.

However, the performance of the non-backtracking algorithm *A3* depends critically on the parameter po. Suppose that $mp < 1$, so that optimal search is hard. If Po = 0, so that as in the Karp and Pearl model there are no dead-ends, then algorithm *A3* usually finds a nearly optimal solution in linear expected time; that is, it performs as successfully as the backtracking algorithm *A2.* However, if po > 0 then algorithm *A2* usually fails to find a solution. Thus our model suggests that backtracking becomes attractive when there is the possibility of dead-ends.

In the next section we give details concerning the search model and the algorithms *Al,A2* and *AZ,* and then in section 3 we present our results. Section 4 briefly discusses the effect of noise on the algorithmic performance. In section 5 we make a few comments on proofs.

2 MODEL AND ALGORITHMS

We suppose that the search tree is the family tree of a branching process. For an introduction to the theory of such processes see for example [Harris, 1963; Athreya and Ney, 1972; Karp and Pearl, 1983]. Thus the search tree has a root node, at depth 0. Each node at depth n independently produces and is joined to a random number *Z* of sons at depth (n+ 1). We shall assume that the mean number m of sons produced satisfies $1 < m < \infty$. Thus the expected number of nodes at depth n is m^n and grows exponentially with *n.*

The Karp and Pearl model is the special case when each node always has exactly two sons. On the other hand our search model here is a special case of the more complicated model considered in [McDiarmid, 1990], namely an age-dependent branching process of Crump-Mode type [Crump and Mode, 1968]. For such a model the implications concerning backtracking are just the same.

Let *q* denote the extinction probability for the branching process, that is the probability that the search tree is finite. Since m > 1 it follows that $q < 1$. Let p_0 be the probability a node has no sons. Clearly $q > 0$ if and only if po > 0, and these conditions correspond to the existence of "dead-ends" in the search tree.

We suppose that the branches have independent non-negative random costs *X* with finite mean. A simple translation allows us to assume without loss of generality that small costs can occur; that is, for any $\delta > 0$ we have $P\{X < \delta) > 0$. The distinction between zero and non-zero costs turns out to be important. We let $p = P(X = 0) \geq 0$.

The cost of a path is the sum of its branch costs. We want to find an optimal (cheapest) or nearly optimal path from the root to a node at depth n, for large n. Let C* denote the random optimal cost of such a path, where $C^* = \infty$ if there is no such path. Thus $P(C_n = \infty) \to q$ as $n \to \infty$. The interesting case is when the search tree is infinite, and we shall usually condition on this happening, so that almost surely C_n is finite.

We shall discuss the performance of three algorithms, A_1, A_2 and A_3, the first two of which are taken from [Karp and Pearl, 1983]. Algorithm *A* is a uniform cost breadth-first search algorithm and will be analyzed for the cases $mp > 1$ and $mp = 1$, when there are many zero-cost branches and search is easy. Algorithm *A2* is a hybrid of local and global depth-first search strategies and will be analyzed for $mp < 1$. Algorithm *A3* consists of repeated local optimal searches, and will be analyzed also for $mp < 1$. Note that *A* is an exact algorithm, whereas *A2* and *A3* are approximation algorithms.

For each algorithm *Aj,* we let the random cost of the solution found be C^{*J} (= ∞ if no solution is found), and the random time taken be T_{AJ}. We measure time by the number of nodes of the search tree encountered.

The three algorithms are as follows:

Algorithm *Al:* At each step, expand the leftmost node among those frontiers nodes of minimum cost. The algorithm halts when it tries to expand a node at depth *n.* That node then corresponds to an optimal solution.

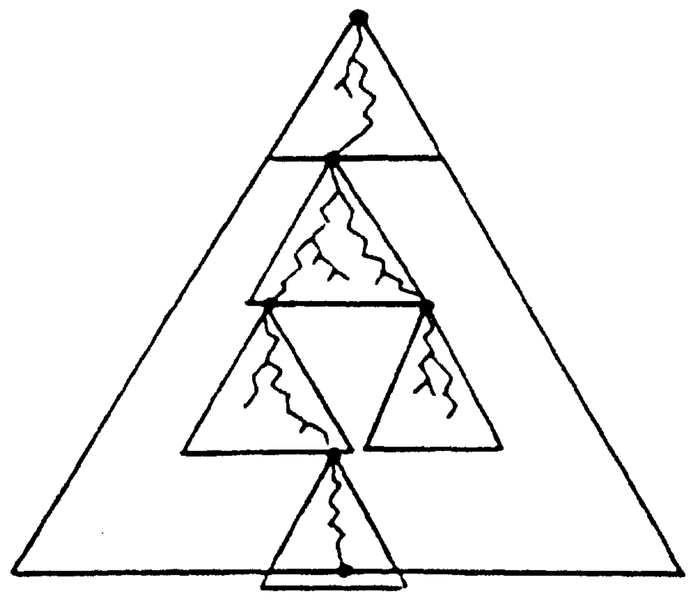

Figure 1: Operation of algorithm *A2:* The triangles represent local depth-first searches for (α, L)-sons.

Algorithm *A2:* Algorithm *A2* has three parameters: *d,* L, and a. An (α, L)-regular path is a path which consist of segments each of length L and cost at most *aL* (except that the last segment may have length < L). *A2* conducts a depth-first search to find an (α, L)-regular path from a depth *d* node to a depth *n* node. In other words, *A2* is a depth-first strategy which stops at regular intervals of L levels to appraise its progress. If the cost increase from the last appraisal is at most aL, the search continues; if that cost increase is above aL, the current node is irrevocably pruned, the program backtracks to a higher level, and the search resumes. If it succeeds in reaching depth *n,A2* returns the corresponding path as a solution: if it fails, the search is repeated from another depth *d* node. If all the nodes at depth *d* fail to root an (a,L)-regular path to a depth *n* node, *A2* terminates with failure.

Algorithm *A3:* The simple bounded-lookahead or "horizon" heuristic is a staged-search algorithm which avoids backtracking. It has one parameter L. Starting at the root it finds an optimal path to a node at depth L, makes that node the new starting point and repeats. If L is a constant clearly *A3* takes linear expected time.

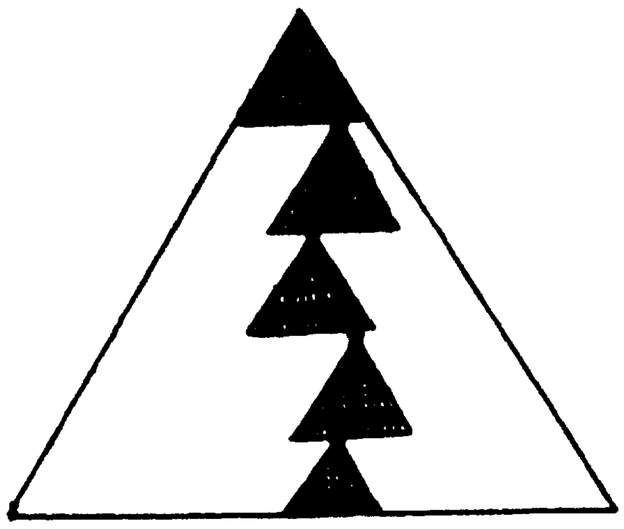

Figure 2: Operation of algorithm $A3$: The triangles represent local complete-enumeration searches for a least-cost path.

3 RESULTS

We summarize our results in six theorems. Theorem 1 concerns the region $mp > 1$, theorem 2 concerns $mp = 1$ and theorems $3 - 6$ concern $mp < 1$. When $mp \geq 1$, there are many zero costs, and the main distinction is between zero and non-zero costs.

Theorem 1 If $mp > 1$ then:
(a) conditional on non-extinction, the random variable

$$C^* = \lim_{n \to \infty} C_n^*$$

is finite almost surely, and indeed $E[C^*]$ is finite; and
(b) the time T_n^{A1} taken by algorithm $A1$ satisfies

$$E[T_n^{A1}] = O(n).$$

Thus, if the search tree is infinite, the optimal cost C_n^* remains bounded as $n \to \infty$, and algorithm $A1$ finds an optimal path in linear expected time.

By restricting ourselves to 0,1 costs and uniform r-ary trees (so that each node has r sons) we may obtain a tighter result than in part (a) above, namely

$$P(C_n^* > k) < \left[\frac{r(1-p)}{r-1}\right]^{r^{(k+1)}} \qquad (1)$$

The case $r = 2$ is a slight improvement on theorem 3.1 of Karp and Pearl [1983], and our proof (given below) is simpler and easier to generalise.

Next we consider the critical case $mp = 1$. It is convenient to make some simplifying assumptions on the typical random family size Z and cost X. We assume roughly that Z is not too big, and that the cost 0, which occurs with probability p, is "isolated", i.e. for some $\epsilon > 0$, $P(X < \epsilon) = 0$.

Theorem 2 Let $mp = 1$:
(a) If $E[Z^{2+\delta}] < \infty$ for some $\delta > 0$,

$$\begin{aligned} P(0 < X < 1) &= 0 \\ \text{and} \quad P(X = 1) &> 0, \end{aligned}$$

then, conditional on non-extinction

$$\frac{C_n^*}{\log_2 \log_2 n} \to 1 \text{ almost surely as } n \to \infty.$$

(b) If $E[Z^2] < \infty$ then the time T_n^{A1} taken by algorithm $A1$ satisfies

$$E[T_n^{A1}] = O(n^2).$$

Part (a) shows that if the optimal cost C_n^* is finite then it is usually close to $\log_2 \log_2 n$. This result is a special case of theorem 2 of [Bramson, 1978]: see also theorem 3.2 of [Karp and Pearl, 1983]. Part (b) states that the algorithm $A1$ finds an optimal solution in quadratic expected time.

Our main interest is in the case $mp < 1$, when we cannot quickly find optimal solutions and thus it is of interest to analyze heuristic approximation methods.

Theorem 3 If $mp < 1$, then any algorithm that is guaranteed to find a solution within a constant factor of optimal must take exponential average time.

Theorem 4 Let $mp < 1$. For $\alpha \geq 0$ let

$$\rho(\alpha) = \inf_{t \geq 0} E[\exp t(\alpha - X)].$$

Then there is a unique solution α^* to the equation $\rho(\alpha) = 1/m$; $\alpha^* > 0$ and conditional on non-extinction,

$$\frac{C_n^*}{n} \to \alpha^* \text{ as } n \to \infty$$

almost surely and in mean.

Thus if the optimal cost C_n^* is finite then it is usually close to $\alpha^* n$. For discussion concerning this result see [Hammersley, 1974; Kingman, 1975].

Theorem 5 Let $mp < 1$, and consider the random cost C_n^{A2} yielded by the "bounded lookahead plus partial backtrack" algorithm $A2$. For any $\epsilon > 0$ there are parameters d, α, L such that algorithm $A2$ runs in linear expected time, and almost surely

$$C_n^{A2} \leq (1 + \epsilon) C_n^* \quad \text{for } n \text{ sufficiently large.}$$

Thus algorithm $A2$ usually finds a nearly optimal solution (whether dead-ends can occur or not). This seems to be very successful performance, but in one sense it is not. For given any sensible parameters there will be a positive probability of failing to produce a solution (even when $p_0 = 0$ so that there are no dead-ends), and thus of course $E[C_n^{A2}] = \infty$.

However, returning "failure" (as in Karp and Pearl's algorithm $A2$) rather than a path of greater than near-optimal cost may possibly be too extreme. Failure of $A2$ to find a near-optimal solution can be easily avoided by adding a suitable "second phase" if the present algorithm fails. A possible second phase could be a depth-first search for a root-leaf path (ignoring costs).

Theorem 6 Let $mp < 1$, and consider the random cost C_n^{A3} yielded by the bounded lookahead but non-backtracking algorithm $A3$:

(a) If $p_0 = 0$ then for any $\epsilon > 0$ there is a (constant) parameter L such that the algorithm $A3$ runs in linear expected time; and almost surely

$$C_n^{A3} \leq (1+\epsilon)C_n^* \text{ for } n \text{ sufficiently large,}$$

and further

$$E[C_n^{A3}] \leq (1+\epsilon)C_n^* \text{ for } n \text{ sufficiently large.}$$

(b) If $p_0 > 0$, and if the lookahead parameter $L = o(n)$ (as is only reasonable) then almost surely

$$C_n^{A3} = \infty \text{ for } n \text{ sufficiently large.}$$

Part (a) above shows that if no dead-ends can occur then the simple non-backtracking heuristic $A3$ usually finds a nearly optimal solution and does so quickly. Part (b) show that $A3$ is hopeless if dead-ends can occur. Further, suppose that $L = O(\log n)$ so that each stage can be performed in polynomial average time. Then even if we consider a polynomial number of different starting points as with algorithm $A2$, still almost surely each search will fail if n is sufficiently large. This will follow from the proof of theorem 6.

4 EFFECTS OF READING ERRORS

In this section we discuss briefly the interesting effect of noise (i.e. reading errors) in the basic Karp and Pearl model. Suppose that the algorithm $A1$ may make occasional independent random reading errors. Thus, for the case of $(0,1)$-costs, there is a probability $\delta_0 \geq 0$ that a 0-cost is read as a 1, and a probability $\delta_1 \geq 0$ that a 1-cost is read as a 0; and so the probability that a 0 is read is:

$$\tilde{p} = p(1-\delta_0) + (1-p)\delta_1.$$

It is easy to see that if δ_0 and δ_1 are small then there will be a correspondingly small change in $\frac{1}{n}$ times the expected solution value obtained by an error-prone algorithm $A1$ (compared with the value obtained by an error-free algorithm $A1$).

However, near the critical value $p = \frac{1}{2}$ there may be a dramatic change in the expected running time. If p is just greater than $\frac{1}{2}$, small reading errors could make $\tilde{p} < \frac{1}{2}$.[2] Then although an error-free algorithm ($A1$) runs in linear expected time, an error-prone version takes exponential expected time. Conversely, if p is just less than $\frac{1}{2}$, small reading errors could make $\tilde{p} > \frac{1}{2}$. Then although an error-free algorithm $A1$ takes exponential expected time, an error-prone algorithm $A1$ runs in linear expected time. Thus although the optimal value is robust with respect to reading errors, the time taken by algorithm A1 to compute the optimal cost is certainly not robust when p is near $\frac{1}{2}$.

[2] Recall that p is the probability that a random cost equals 0.

5 COMMENT ON PROOFS

This section presents the following two proofs: (1) inequality 1, which has appeared before only in [Provan, 1985]; and (2) theorem 6, a proof of the performance of the simple bounded-lookahead algorithm $A3$—this theorem is not like anything from [Karp and Pearl, 1983], and it is also quick and easy to prove. Theorems 1-6 can be deduced from the corresponding results in [McDiarmid, 1990] by specialising to the model discussed here and using standard truncation arguments. The proofs of theorems 1—5 can follow roughly similar lines to the proofs of the corresponding results in Karp and Pearl [1983].

Proof of inequality 1: Consider a branching process in which the number of sons of an individual has the binomial distribution with parameters r and p. Let q be the extinction probability. Then

$$P(C_n^* > k) < q^{r^k},$$

since if $C_n^* > k$ then each of the r^k subtrees rooted at the nodes at depth k must fail to have an infinite path of zero-cost branches. We shall show that (for $rp > 1$) we have

$$q < x^r \qquad \text{where } x = \frac{r(1-p)}{(r-1)},$$

and then inequality 1 will follow. Using standard branching processes theory, q is the least positive root s of $f(s) = s$, where the generating function $f(s) = (1-p+ps)^r$. Since f is convex it suffices to demonstrate that

$$\begin{array}{lll} & f(x^r) & < x^r, \\ \text{that is} & 1-p+px^r & < x. \end{array}$$

But

$$x = 1 - p + \frac{x}{r},$$

and so this is equivalent to showing that

$$rpx^{(r-1)} < 1.$$

To do this, introduce

$$\begin{array}{lllll} & g(y) & = & (r-(r-1)y)y^{(r-1)} & \text{for } 0 \leq y \leq 1. \\ \text{But} & g(1) & = & 1, & \\ \text{and} & g'(y) & = & r(r-1)(1-y)y^{(r-2)} & \text{for } 0 < y < 1, \\ \text{so} & g(y) & < & 1 & \text{for } 0 < y < 1. \end{array}$$

Finally, put $y = x$ to obtain, as required,

$$1 > (r-(r-1)x)x^{(r-1)} = rpx^{(r-1)}. \square$$

Proof Of Theorem 6

(a) We have already noted that for any constant lookahead L the algorithm $A3$ runs in linear expected time. By theorem 5,

$$\frac{1}{n}E[C_n^*] \to \alpha^* > 0 \quad \text{as } n \to \infty.$$

Let $\epsilon > 0$ and choose L so that

$$\frac{1}{L}E[C_L^*] < (1+\epsilon)\alpha^*.$$

Now C_n^{A3} is at most the sum of $\lceil \frac{n}{L} \rceil$ independent random variables each distributed like C_L^*. Hence, by the strong law of large numbers, almost surely

$$\frac{C_n^{A3}}{n} < (1+2\epsilon)\alpha^* \quad \text{for } n \text{ sufficiently large.}$$

But again by theorem 4, almost surely:

$$\frac{C_n^*}{n} > (1-\epsilon)\alpha^* \quad \text{for } n \text{ sufficiently large,}$$

and part (a) of theorem 6 now follows.[3]

(b) To prove part (b) we need only observe that

$$\begin{aligned} P(C_n^* = \infty) &\geq 1-(1-p_0)^{n/L} \\ &\rightarrow 1 \quad \text{as } n \rightarrow \infty \text{ if } L = o(n). \square \end{aligned}$$

6 CONCLUSIONS

This paper has studied the performance of both backtracking and non-backtracking tree search algorithms in finding a least-cost root-leaf path in random trees with random non-negative costs. The investigations extend the work of Karp and Pearl in several ways, but in particular through the introduction of dead-ends. This analysis suggests the following conclusions:

1. When the possibility of "catastrophe" (dead ends or prohibitive costs) can be ignored then backtracking methods do not seem attractive, and a far simpler approach like that of the "horizon heuristic" .43 is preferable.
2. When catastrophe looms then a backtracking method like Karp and Pearl's bounded-lookahead plus partial backtrack algorithm A2 does seem an attractive option.

This conclusion lends some mathematical support to certain empirical studies which show that, under given conditions, backtracking algorithms do not perform as well as non-backtracking algorithms. Examples are the empirical analysis of [Dechter and Meiri, 1989; Haralick and Elliott, 1980] in binary constraint satisfaction problems, and the analysis of [de Kleer, 1984] in reason maintenance systems.

References

[Athreya and Ney, 1972] K.B. Athreya and P.E. Ney. *Branching Processes.* Springer-Verlag, Berlin, 1972.

[Bitner and Reingold, 1975] J. R. Bitner and E. M. Reingold. Backtrack programming techniques. *Communications of the ACM,* 18:651-655, 1975.

[Bramson, 1978] M. D. Bramson. Minimal displacement of branching random walk. *Z. Wahrsch. Varw. Gabiete,* 45:89-108, 1978.

[Brown and P. W. Purdom, 1981] C. A. Brown and P. W. Purdom, Jr. . An average case analysis of backtracking. *S1AM* J. *Computing,* 10 (3):583-593, 1981.

[Brown and P. W. Purdom, 1982] C A. Brown and P. W. Purdom, Jr.. An empirical comparison of backtracking algorithms. *IEEE Trans. PAMI,* 4, 1982.

[Crump and Mode, 1968] K.S. Crump and C.J. Mode. A general age-dependent branching process. *Journal of Mathematical Analysis and Applications,* 24, 1968.

[de Kleer, 1984] J. de Kleer. Choices without backtracking. *Proc. AAAI,* 79-85, 1984.

[Dechter, 1990] R. Dechter. Enhancement Schemes for Constraint Processing: Backjumping, Learning and Cutset Decomposition. *Artificial Intelligence,* 41(3):273-312, 1990.

[Dechter and Meiri, 1989] R. Dechter and I. Meiri. Experimental Evaluation of Preprocessing Techniques in Constraint Satisfaction. In *Proc. IJCAI,* 271—276, 1989.

[Hammersley, 1974] J. M. Hammersley. Postulates for subadditive processes. *Annals of Probability,* 2:652-680, 1974.

[Haralick and Elliott, 1980] R. M. Haralick and G. L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence,* 14:263-313, 1980.

[Harris, 1963] T. E. Harris. *The Theory of Branching Processes.* Springer-Verlag, Berlin, 1963.

[Karp, 1976] R. M. Karp. The probabilistic analysis of some combinatorial search algorithms. In J. F. Traub, editor, *Algorithms and Complexity,* pages 1-19, Academic Press, 1976.

[Karp and Pearl, 1983] R. M. Karp and J. Pearl. Searching for an optimal path in a tree with random costs. *Artificial Intelligence,* 21:99-116, 1983.

[Kingman, 1975] J.F.C. Kingman. The First Birth Problem for an Age-Dependent Branching Process. *Annals of Probability,* 3:790-801, 1975.

[Knuth, 1975] D. E. Knuth. Estimating the efficiency of backtrack programs. *Mathematics of Computation,* 29:121-136, 1975.

[McDiarmid, 1990] C.J.H. McDiarmid. Probabilistic Analysis of Tree Search. In G.R. Gummett and D.J .A. Welsh, editors, *Disorder in Physical Systems,* pages 249-260, Oxford Science Publications, 1990.

[Nudel, 1983] B. Nudel. Consistent labeling problems and their algorithms: expected complexities and

[3] It is also easy to prove a weaker version of this result using only an easy part of theorem 4 and the subadditivity of the sequence $E[C_n]$.

theory-based heuristics. *Artificial Intelligence,* 21:135-178, 1983.

[Provan, 1985] G. M. Provan. A Probabilistic Analysis of Search Algorithms in Uniform Trees. Mathematical Institute, University of Oxford, Unpublished D. Phil, qualifying dissertation, 1985.

[P. W. Purdom, 1983] Jr. P. W. Purdom. Search rearrangement backtracking and polynomial average time. *Artificial Intelligence,* 21:117-133, 1983.

[Stone and Stone, 1986] H. Stone and J. Stone. *Efficient Search Techniques: An Empirical Study of the N-Queen$ Problem.* Technical Report TRTC 12057 (#54343), IBM T.J. Watson Research Center, 1986.