

An Average-Case Analysis of Branch-and-Bound with Applications: Summary of Results

Weixiong Zhang and Richard E. Korf*
 Computer Science Department
 University of California, Los Angeles
 Los Angeles, CA 90024
 Email: <last-name>@cs.ucla.edu

Abstract

Motivated by an anomaly in branch-and-bound (BnB) search, we analyze its average-case complexity. We first delineate exponential vs polynomial average-case complexities of BnB. When best-first BnB is of linear complexity, we show that depth-first BnB has polynomial complexity. For problems on which best-first BnB has exponential complexity, we obtain an expression for the heuristic branching factor. Next, we apply our analysis to explain an anomaly in lookahead search on sliding-tile puzzles, and to predict the existence of an average-case complexity transition of BnB on the Asymmetric Traveling Salesman Problem. Finally, by formulating IDA* as cost-bounded BnB, we show its average-case optimality, which also implies that RBFS is optimal on average.

Introduction

Consider a simple search problem. We define a *uniform random tree* as a tree of depth d , branching factor B , random edge costs independently and identically drawn from a non-negative discrete probability distribution, and node costs computed as the sum of the edge costs on the path from the root to the node. The problem is to find a frontier node at depth d of minimum cost. A uniform random tree is an abstract model of the state space of a discrete optimization problem, with a cost function that is monotonically non-decreasing along a path.

One of the most efficient linear-space algorithms for this problem is depth-first branch-and-bound (BnB) [Kumar 92]. Depth-first BnB starts with an upper bound of infinity on the minimum goal cost, and then searches the entire tree in a depth-first fashion. Whenever a node at depth d is reached whose cost is lower

*This research was supported by an NSF Presidential Young Investigator Award, No. IRI-8552925, to the second author and a grant from Rockwell International. Thanks to Joe Pemberton and Hui Zhou for many helpful discussions, and to Lars Hagen for pointing out an error in a draft.

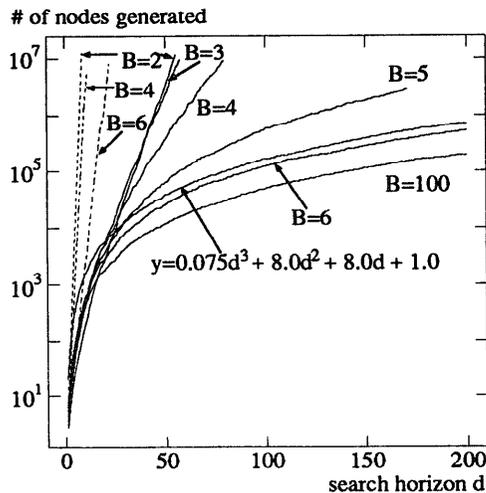


Figure 1: Anomaly of depth-first BnB.

than the best one found so far, the upper-bound is revised to the cost of this node. Whenever an internal node is encountered whose cost equals or exceeds the current upper bound, the subtree below it is pruned. The most important property of depth-first BnB is that it only requires space that is linear in the search depth, since only the nodes on the path from the root to the current node must be stored. One drawback of depth-first BnB is that it may expand nodes with costs greater than the minimum goal cost.

Best-first BnB [Kumar 92], or best-first search, always expands a node of least cost among all nodes which have been generated but not yet expanded, and as a result, never expands a node whose cost is greater than the minimum goal cost. To guarantee this, however, best-first BnB has to store all frontier nodes, making it useful only for small problems because of memory limitations.

Figure 1 shows the performance of depth-first BnB on uniform random trees with edge costs drawn independently and uniformly from $\{0, 1, 2, 3, 4\}$. The x -axis

is the search depth d , and the y -axis is the number of nodes generated. The straight dotted lines on the left show the complexity of brute-force search, which generates every node in the tree, or $(B^{d+1} - 1)/(B - 1) = O(B^d)$. The curved lines to the right represent the number of nodes generated by depth-first BnB, averaged over 1000 random trials. Figure 1 shows two effects of depth-first BnB. First, it is dramatically more efficient than brute-force search. Secondly, depth-first BnB displays the following counterintuitive anomaly. *For a given amount of computation, we can search deeper in the larger trees. Alternatively, for a given search depth, we can search the larger trees faster.*

We explore the reasons for this anomaly by analyzing the average-case complexity of BnB. We first discuss the behavior of the minimum goal cost. We then discuss the average-case complexity of BnB. As applications of our analysis, we explain this anomaly in lookahead search on sliding-tile puzzles [Korf 90], and predict the existence of an average-case complexity transition of BnB on Asymmetric Traveling Salesman Problems (ATSP) [Lawler *et al.* 85].

Iterative-deepening-A* (IDA*) [Korf 85] and recursive best-first search (RBFS) [Korf 92] are also linear-space search algorithms. By formulating IDA* as cost-bounded BnB, we show its average-case optimality. This leads to the corollary that RBFS is also optimal on average. Finally, we discuss the knowledge/search tradeoff. The proofs of theorems are omitted due to space limitations, but a full treatment of the results is in [Zhang & Korf 92].

Properties of Minimum Goal Cost

A uniform random tree models the state space of a discrete optimization problem. In general, a state space is a graph with cycles. Nevertheless, a tree is an appropriate model for analysis for two reasons. First, any depth-first search must explore a tree at the cost of generating some states more than once. Secondly, many operators used in practice for some optimization problems, such as the ATSP, partition the state space into mutually exclusive parts, so that the space visited is a tree.

If p_0 is the probability that an edge has zero cost, and B is the branching factor, then Bp_0 is the expected number of children of a node that have the same cost as their parent (same-cost children). When $Bp_0 < 1$, the expected number of same-cost children is less than one, and the minimum goal cost should increase with depth. On the other hand, when $Bp_0 > 1$, we should expect the minimum goal cost not to increase with depth, since in this case, we expect a minimum-cost child to have the same cost as its parent.

Lemma 1 [McDiarmid & Provan 91] *On a uniform random tree, asymptotically the minimum goal cost grows linearly with depth almost surely when $Bp_0 < 1$, and is asymptotically bounded above by a constant almost surely when $Bp_0 > 1$. □*

Average-Case Complexity

Given a frontier node value, to verify that it is of minimum cost requires examining all nodes that have costs less than the given value. When $Bp_0 < 1$, minimum-cost goal nodes are very rare, their cost grows linearly with depth, and the average number of nodes whose costs are less than the minimum goal cost is exponential. Thus, any search algorithm must expand an exponential number of nodes on average. The extreme case of this is that no two edges have the same cost, and thus $p_0 = 0$. On the other hand, when $Bp_0 > 1$, there are many nodes that have the same cost, and there are many minimum-cost goal nodes as well. In this case, best-first BnB, breaking ties in favor of deeper nodes, and depth-first BnB can easily find one minimum-cost goal node, and then prune off the subtrees of nodes whose costs are greater than or equal to the goal cost. The extreme case of this is that all edges have cost zero, *i.e.* $p_0 = 1$, and hence every leaf of the tree is a minimum-cost goal. Best-first BnB does not need to expand an exponential number of nodes in this case.

Theorem 1 [McDiarmid & Provan 91] *On a uniform random tree, the average-case complexity of best-first BnB is exponential in the search depth d , when $Bp_0 < 1$, and the average-case complexity of best-first BnB is linear in d , when $Bp_0 > 1$. □*

Theorem 1 also implies that depth-first BnB must generate an exponential number of nodes if $Bp_0 < 1$, because all nodes expanded by best-first BnB are examined by depth-first BnB as well, up to tie-breaking. What is the average-case complexity of depth-first BnB when $Bp_0 > 1$? In contrast to best-first BnB, depth-first BnB may expand nodes that have costs greater than the minimum goal cost. This makes the analysis of depth-first BnB more difficult. An important question is whether depth-first BnB has polynomial or exponential average-case complexity when $Bp_0 > 1$. In fact, depth-first BnB is polynomial when $Bp_0 > 1$.

Theorem 2 *When $Bp_0 > 1$, the average-case complexity of depth-first BnB on a uniform random tree is bounded above by a polynomial function, $O(d^{m+1})$, if edge costs are independently drawn from $\{0, 1, \dots, m\}$, regardless of the probability distribution. □*

For example, in Figure 1, $p_0 = 0.2$, thus $Bp_0 > 1$ when $B \geq 6$. By experiment, we found that the curves for $B \geq 6$ can be bounded above by the function $y = 0.075d^3 + 8.0d^2 + 8.0d + 1.0$, which is $O(d^3)$.

The above analysis shows that the average-case complexity of BnB experiences a dramatic transition, from exponential to polynomial, on discrete optimization problems, depending on the branching factor B and the cost function (p_0). When $Bp_0 < 1$, both best-first BnB and depth-first BnB take exponential average time. On the other hand, when $Bp_0 > 1$, best-first BnB and depth-first BnB run in linear and polynomial average time, respectively. Therefore, the average-case

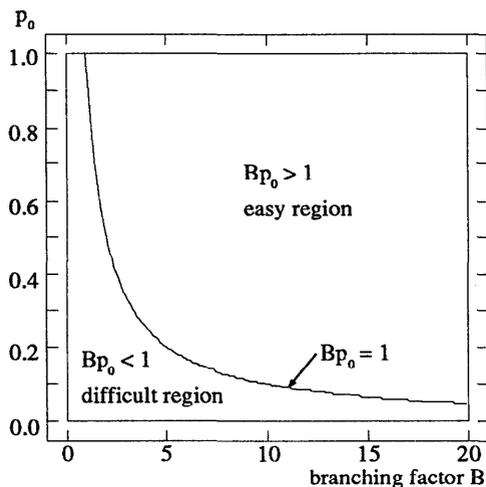


Figure 2: Difficult and easy regions for BnB.

complexity of BnB changes from exponential to polynomial as the expected number of same cost children, Bp_0 , changes from less than one to greater than one. Figure 2 illustrates these two complexity regions and the transition boundary.

For a uniform random tree with $Bp_0 < 1$, the *heuristic branching factor* of the tree measures the complexity of the problem modeled by the tree. The heuristic branching factor of a problem is the ratio of the average number of nodes with a given cost to the average number of nodes with the next smaller cost.

Theorem 3 When $Bp_0 < 1$ and edge costs are $0, 1, \dots, m$ with probabilities p_0, p_1, \dots, p_m , the asymptotic heuristic branching factor b of a uniform random tree is a solution greater than one to the equation

$$x^m - B \sum_{i=0}^m p_i x^{m-i} = 0. \quad \square \quad (1)$$

Although there exists no closed-form solution to this equation in general when $m \geq 5$, it can be solved numerically.

Applications

Anomaly of Lookahead Search

A sliding-tile puzzle consists of a $k \times k$ square frame holding $k^2 - 1$ movable square tiles, with one space left over (the blank). Any tiles which are horizontally or vertically adjacent to the blank may move into the blank position. Examples of sliding-tile puzzles include the 3×3 Eight Puzzle, the 4×4 Fifteen Puzzle, the 5×5 Twenty-four Puzzle, and the 10×10 Ninety-nine Puzzle. A common cost function for sliding-tile puzzles is $f(n) = g(n) + h(n)$, where $g(n)$ is the number of steps from the initial state to node n , and $h(n)$ is the Manhattan Distance [Pearl 84] from node n to the goal

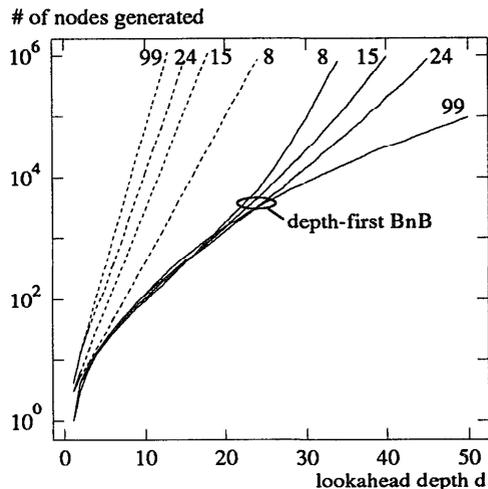


Figure 3: Anomaly of lookahead search [Korf 90].

state. The Manhattan Distance is computed by counting, for each tile not in its goal position, the number of moves along the grid it is away from its goal, and summing these values over all tiles, excluding the blank.

A lookahead search to depth k returns a node that is k moves away from the original node, and whose f value is a minimum over all nodes at depth k . The straight dotted lines on the left of Figure 3 represent the total number of nodes in the tree up to the given depth, for different size puzzles. The curved lines to the right represent the total number of nodes generated in a depth-first BnB to the given depth using Manhattan Distance (averaging over 1000 randomly generated initial states). This shows the same anomaly as in Figure 1, namely that *the larger problems can be searched faster to a given depth*.

The explanation of this anomaly on sliding-tile puzzles is the following. Moving a tile either increases its Manhattan Distance h by one, or decreases it by one. Since every move increases the g value by one, the cost function $f = g + h$ either increases by two or stays the same. Thus, this problem can be modeled by a uniform random tree where edge costs are either zero or two. The probability that the h value either increases by one or decreases by one by moving a tile is roughly one half initially, independent of the problem size. Thus, the probability that a child node has the same cost as its parent is roughly one half, *i.e.* $p_0 \approx 0.5$. On the other hand, the branching factor B increases with the size of a puzzle. For example, the branching factors of the Eight, Fifteen, Twenty-four, and Ninety-nine Puzzles are 1.732, 2.130, 2.368, and 2.790, respectively [Korf 90]. Therefore, a larger puzzle has a larger Bp_0 . According to the analysis, the asymptotic average-case complexity of fixed-depth lookahead search on a larger puzzle is smaller than that on a smaller puzzle.

This result does not imply that a larger sliding-tile puzzle is easier to solve, since the solution length for a larger puzzle is longer than that for a smaller one. Furthermore, the assumption that edge costs are independent is not valid for sliding-tile puzzles. For example, from experiments, p_0 is slightly less than 0.5 initially, but slowly decreases with search depth.

Complexity Transition of BnB on ATSP

Given n cities and an *asymmetric* matrix $(c_{i,j})$ that defines the cost between each pair of cities, the Asymmetric Traveling Salesman Problem (ATSP) is to find a minimum-cost tour visiting each city exactly once.

The most efficient approach known for optimally solving the ATSP is BnB using the assignment problem solution as a lower bound function [Balas & Toth 85]. The solution of an assignment problem (AP) [Martello & Toth 87], a relaxation of the ATSP, is either a single tour or a collection of disjoint subtours. The ATSP can be solved as follows. The AP of all n cities is first solved. If the solution is not a tour, then the problem is decomposed into subproblems by eliminating one of the subtours in the solution. A subproblem is one which has some *excluded arcs* that are forbidden from the solution, and some *included arcs* that must be present in the solution. Then a subproblem is selected and the above process is repeated until each subproblem is either solved, *i.e.* its AP solution is a tour, or all unsolved subproblems have costs greater than or equal to the cost of the best tour obtained so far. Many decomposition rules for the ATSP partition the state space into a tree [Balas & Toth 85]. Hence, the state space of the ATSP can be modeled by a uniform random tree, in which the root corresponds to the ATSP, a leaf node is a subproblem whose AP solution is a tour, and the cost of a node is the AP cost of the corresponding subproblem. The AP cost is monotonically non-decreasing, since the AP of a subproblem is a more constrained AP than the parent problem, and hence the cost of the child node is no less than the cost of the parent.

When the intercity distances are chosen from $\{0, 1, \dots, k\}$, the probability that two distances have the same value decreases as k increases. Similarly, the probability that two sets of edges have the same total distance decreases as k increases. Thus, when k is small with respect to the number of cities, the probability that the AP value of a child is equal to that of its parent is large, *i.e.* p_0 in the search tree is large. By our analysis, when k is large, p_0 is small, and more nodes need to be examined. On the other hand, when k is small, p_0 is large, $Bp_0 > 1$, and the ATSP is easy to solve. The above analysis implies the existence of a complexity transition of BnB on ATSP as k changes.

We verified this prediction by experiments using Carpaneto and Toth's decomposition rules [Carpaneto & Toth 80], and with intercity distances uniformly chosen from $\{0, 1, 2, \dots, k\}$. Figure 4 gives the results on

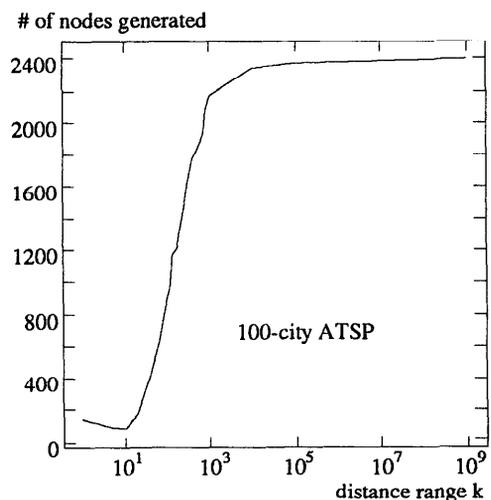


Figure 4: Complexity transition of BnB on ATSP.

100-city ATSPs using depth-first BnB, averaging over 1000 random trials, which clearly show a complexity transition. This transition has also been observed on 200-city and 300-city ATSPs as well.

Complexity transitions of BnB on ATSP are also reported in [Cheeseman *et al.* 91], where Little's algorithm [Little *et al.* 63] and the AP cost function were used. In their experiments, they observed that an ATSP is easy to solve when the number of same-distance edges is large or small, and the most difficult problem instances lie in between these regions. There is a discrepancy between the observations of [Cheeseman *et al.* 91] and our observations on ATSP. This is probably due to the fact that different algorithms are used, suggesting that such transitions are sensitive to the choice of algorithm.

Average-Case Optimality of IDA* and RBFS

Iterative-deepening-A* (IDA*) [Korf 85] is a linear-space search algorithm. Using a variable called the *threshold*, initially set to the cost of the root, IDA* performs a series of depth-first search iterations. In each iteration, IDA* expands all nodes with costs less than the threshold. If a goal is chosen for expansion, then IDA* terminates successfully. Otherwise, the threshold is updated to the minimum cost of nodes that were generated but not expanded on the last iteration, and a new iteration is begun.

We may treat IDA* as *cost-bounded depth-first BnB*, in the sense that it expands those nodes whose costs are less than the threshold, and expands them in depth-first order. A tree can be used to analyze its average-case performance.

On a tree, the worst-case of IDA* occurs when all node costs are unique, in which case only one new node

is expanded in each iteration [Patrick *et al.* 89]. The condition of unique node costs asymptotically requires that the number of bits used to represent the costs increase with each level of the tree, which may be unrealistic in practice.

When $Bp_0 > 1$ the minimum goal cost is a constant by Lemma 1. Therefore, IDA* terminates in a constant number of iterations. This means that the total number of nodes generated by IDA* is of the same order as the total number of nodes generated by best-first BnB, resulting in the optimality of IDA* in this case. The number of iterations of IDA*, however, is not a constant when $Bp_0 < 1$, since the minimum goal cost grows with the search depth. When $Bp_0 < 1$, it can be shown that the ratio of the number of nodes generated in one iteration to the number of nodes generated in the previous iteration is the same as the heuristic branching factor of the uniform random tree. Asymptotically, this ratio is the solution greater than one to equation (1). Consequently, we have the following result.

Theorem 4 *On a uniform random tree with $Bp_0 < 1$, IDA* is optimal on average. \square*

Similar results were independently obtained by Patrick [Patrick 91].

With a non-monotonic cost function, the cost of a child can be less than the cost of its parent, and IDA* no longer expands nodes in best-first order. In this case, recursive best-first search (RBFS) [Korf 92] expands new nodes in best-first order, and still uses memory linear in the search depth. It was shown that with a monotonic cost function, RBFS generates fewer nodes than IDA*, up to tie-breaking [Korf 92]. This, with Theorem 4, gives us the following corollary.

Corollary 1 *On a uniform random tree with $Bp_0 < 1$, RBFS is optimal on average. \square*

Discussion

The efficiency of BnB depends on the cost function used, which involves domain-specific knowledge. There is a tradeoff between knowledge and search complexity, in that a more accurate cost estimate prunes more nodes, resulting in less computation. The results of our analysis measure this tradeoff in an average-case setting. In particular, to make an optimization problem tractable, enough domain-specific knowledge is needed so that the expected number of children of a node that have the same heuristic evaluation as their parent is greater than one. We call this property *local consistency*.

Local consistency is a different characterization of a heuristic function than its error as an estimator. Local consistency is a property that can be determined locally, for example by randomly generating states and computing the average number of same-cost children. Determining the error in a heuristic estimator,

on the other hand, requires knowing the exact value being estimated, which is impossible in a difficult problem, since determining minimum-cost solutions is intractable. Thus, determining local consistency by random sampling gives us a practical means of predicting whether the complexity of a branch-and-bound search will be polynomial or exponential.

This prediction, however, is based on the assumption that edge costs are independent of each other. By Theorem 1, local consistency implies that the estimated cost of a node is within a constant of its actual cost on a uniform random tree with independent edge costs. In other words, local consistency implies constant absolute error of the heuristic estimator. Unfortunately, edge costs are often not independent in real problems. For example, in the sliding-tile puzzles, a zero-cost edge implies a tile moving toward its goal location. As a sequence of moves are made that bring tiles closer to their goal locations, it becomes increasingly likely that the next move will require moving a tile away from its goal location. On the other hand, the independence assumption may be reasonable for a relatively short fixed-depth search in practice.

Related Work

Dechter [Dechter 81], Smith [Smith 84], and Wah and Yu [Wah & Yu 85] analyzed the average-case complexities of best-first BnB and depth-first BnB using tree models, and concluded that they are exponential. Purdom [Purdom 83] characterized exponential and polynomial performance of backtracking, essentially a depth-first BnB on constraint-satisfaction problems. Stone and Sipala [Stone & Sipala 86] considered the relationship between the pruning of some branches and the average complexity of depth-first search with backtracking.

Karp and Pearl [Karp & Pearl 83] reported important results for best-first BnB on a uniform binary tree, with edge costs zero or one. McDiarmid and Provan [McDiarmid & Provan 91] extended Karp and Pearl's results by using a general tree model, which has arbitrary edge costs and variable branching factor. In fact, the properties of the minimum goal cost (Lemma 1) and the average-case complexities of best-first BnB (Theorem 1) discussed in this paper are based on McDiarmid and Provan's results. We further extended these results in two respects. First, when best-first BnB has linear average-case complexity, we showed that depth-first BnB has polynomial complexity. Second, for problems on which best-first BnB has exponential average-case complexity, we obtained an expression for its heuristic branching factor.

Huberman and Hogg [Huberman & Hogg 87], and Cheeseman *et al.* [Cheeseman *et al.* 91] argued that phase transitions are universal in intelligent systems and in most NP-hard problems.

The worst-case complexity of IDA* was first shown by Patrick *et al.* [Patrick *et al.* 89]. Vempaty *et al.*

[Vempaty *et al.* 91] compared depth-first BnB and IDA*. Recently, Mahanti *et al.* discussed performance of IDA* on trees and graphs [Mahanti *et al.* 92]. Our results on IDA* concludes the average-case optimality of IDA* on a uniform random tree, which also implies that RBFS is optimal on average. Similar results on IDA* were also obtained by Patrick [Patrick 91].

Conclusions

This paper contains the following results on branch-and-bound, one of the most efficient approaches for exact solutions to discrete optimization problems. Our analysis is based on a tree with uniform branching factor and random edge costs. We delineated exponential and polynomial complexities of BnB in an average-case setting. When best-first BnB has linear complexity, we showed that the complexity of depth-first BnB is polynomial. We further obtained an expression for the heuristic branching factor of problems on which best-first BnB has exponential complexity. The analysis uncovered the existence of a complexity transition of BnB on discrete optimization problems. Furthermore, the analysis explained an anomaly observed in look-ahead search with sliding-tile puzzles, and predicted the existence of an average-case complexity transition of BnB on ATSP, which was verified by experiments. In addition, our results showed a quantitative tradeoff between knowledge and search complexity, and provided a means of testing average-case search complexity. By formulating IDA* as cost-bounded BnB, we derived an expression for the ratio of the number of nodes generated in one iteration to the number of nodes generated in the previous iteration, and showed that IDA* is optimal on average. This implies the average-case optimality of RBFS in this model.

References

Balas, E. and P. Toth, 1985. "Branch and bound methods," *The Traveling Salesman Problem*, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (eds.) John Wiley and Sons, pp.361-401.

Carpaneto, G., and P. Toth, 1980. "Some new branching and bounding criteria for the asymmetric traveling salesman problem," *Management Science*, **26**:736-43.

Cheeseman, P., B. Kanefsky, W.M. Taylor, 1991. "Where the really hard problems are," *Proc. IJCAI-91*, Sydney, Australia, Aug. pp.331-7.

Dechter, A., 1981. "A probabilistic analysis of branch-and-bound search," Tech Rep. *UCLA-ENG-81-39*, School of Eng. and Applied Sci., UCLA, Oct.

Huberman, B.A., and T. Hogg, 1987. "Phase transitions in artificial intelligence systems," *Artificial Intelligence*, **33**:155-71.

Karp, R.M., and J. Pearl, 1983. "Searching for an optimal path in a tree with random cost," *Artificial Intelligence*, **21**:99-117.

Korf, R.E., 1985. "Depth-first iterative-deepening: An optimal admissible tree search," *Artificial Intelligence*, **27**:97-109.

Korf, R.E., 1990. "Real-time heuristic search," *Artificial Intelligence*, **42**:189-211.

Korf, R.E., 1992. "Linear-space best-first search: Summary of results," *Proc. AAAI-92*, San Jose, CA, July 12-17.

Kumar, V., 1992. "Search, Branch-and-bound," in *Encyclopedia of Artificial Intelligence*, 2nd Ed, S.C. Shapiro (ed.) Wiley-Interscience, pp.1468-72.

Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (eds.), 1985. *The Traveling Salesman Problem*, John Wiley and Sons.

Little, J.D.C., K.G. Murty, D.W.Sweeney, and C. Karel, 1963. "An algorithm for the traveling salesman problem," *Operations Research*, **11**:972-989.

Mahanti, A., S. Ghosh, D. S. Nau, A. K. Pal, and L. N. Kanal, 1992. "Performance of IDA* on Trees and Graphs," *Proc. AAAI-92*, San Jose, CA, July 12-17.

Martello, S., and P. Toth, 1987. "Linear assignment problems," *Annals of Discrete Mathematics*, **31**:259-82.

McDiarmid, C.J.H., and G.M.A. Provan, 1991. "An expected-cost analysis of backtracking and non-backtracking algorithms," *Proc. IJCAI-91*, Sydney, Australia, Aug. pp.172-7.

Patrick, B.G., 1991. Ph.D. dissertation, Computer Science Dept., McGill University, Canada.

Patrick, B.G., M. Almulla, and M.M. Newborn, 1989. "An upper bound on the complexity of iterative-deepening-A*," *Proceedings of the Symposium on Artificial Intelligence and Mathematics*, Ft. Lauderdale, Fla., Dec.

Pearl, J., 1984. *Heuristics*, Addison-Wesley, Reading, MA.

Purdom, P.W., 1983. "Search Rearrangement backtracking and polynomial average time," *Artificial Intelligence*, **21**:117-33.

Vempaty, N.R., V. Kumar, and R.E. Korf, 1991. "Depth-first vs best-first search," *proc. AAAI-91*, Anaheim, CA, July, pp.434-40.

Smith, D.R., 1984. "Random trees and the analysis of branch and bound procedures," *JACM*, **31**:163-88.

Stone, H.S., and P. Sipala, 1986. "The average complexity of depth-first search with backtracking and cutoff," *IBM J. Res. Develop.*, **30**:242-58.

Wah, B.W., and C.F. Yu, 1985. "Stochastic modeling of branch-and-bound algorithms with best-first search," *IEEE Trans. on Software Engineering*, **11**:922-34.

Zhang, W., and R. Korf, 1992. "An Average-Case Analysis of Branch-and-Bound with Applications," in preparation.