

Iterative-Deepening-A*: An Optimal Admissible Tree Search¹

Richard E. Korf²

Department of Computer Science
Columbia University
New York, N.Y. 10027

Abstract

A serious limitation of the A* algorithm is that in practice it requires space which is exponential in the solution cost. We propose an iterative-deepening extension to A* which only requires linear space. The algorithm is admissible and expands the same number of nodes, asymptotically, as A* for exponential tree searches. This algorithm is the only known algorithm that is capable of finding optimal solutions to randomly generated instances of the Fifteen Puzzle within practical resource limits.

1. Introduction

Heuristic search is ubiquitous in artificial intelligence. The performance of most AI systems is dominated by the complexity of a search algorithm in their inner loops. The best known admissible algorithm, A*, has a serious space limitation, which can be overcome by applying an iterative-deepening technique. This article describes the application of iterative-deepening to A*, proves its optimality for exponential tree searches in terms of time, space, and solution cost, and demonstrates its practicality on the Fifteen Puzzle. A more comprehensive treatment of iterative-deepening, including its application to brute-force, bi-directional, and two-person game searches, can be found in [Korf 85].

The A* algorithm was first proposed by Hart, Nilsson, and Raphael [Hart 68]. Iterative-deepening has no doubt been rediscovered many times independently. The first use of the technique that is documented in the literature is in Slate and Atkin's Chess 4.5 program [Slate 77]. Pearl³ initially suggested the iterative-deepening extension of A*, and Berliner and Goetsch [Berliner 84] have implemented such an algorithm concurrently with this work. In addition, Stickel and Tyson [Stickel 85] have independently analyzed iterative-deepening for the special case of brute-force search.

The complexity of a search algorithm for a given problem will be expressed in terms of two parameters: the cost of a solution to the problem, and the effective branching factor of the algorithm. The *cost* (c) of a solution to a problem is the cost of the least expensive sequence of operators that map the initial state into a goal state. The *effective branching factor* (b) of a search algorithm is the ratio of the number of nodes that are generated at a given cost compared to those generated at the next smaller cost, and in general depends on the accuracy of the heuristic function. We will assume that the branching factor is constant throughout the problem space. The time complexity of a search algorithm in this model is simply the number of nodes that are generated, since we assume that the amount of time is proportional to the number of nodes generated. Similarly, since we assume that the amount of space required is proportional to the number of states that are stored, the space complexity of an algorithm in this model will be the number of states that must be stored.

Strictly speaking, the results in this paper hold only for trees. The reason is that the iterative-deepening algorithm is inherently depth-first and hence is constrained to explore all paths to a given cutoff. For graphs with cycles, the number of paths can be an exponential function of the number of nodes, and hence a depth-first approach would be dominated by a breadth-first algorithm. The performance of a depth-first algorithm on a graph is a function of the average number of edges that are incident to a given state rather than the number of nodes that are adjacent. For trees the edge and node branching factors are equal but can be quite different for some graphs.

This work is focused on searches which produce optimal solutions. We recognize that for most applications, optimal solutions are not required and that their price is often prohibitive. There are occasions, however, when optimal solutions are needed. For example, in assessing the quality of non-optimal solutions, it is often enlightening to compare them to optimal solutions for the same problem instances.

2. A Limitation of A*

The A* algorithm [Hart 68] works as follows: starting with the initial state, a node is expanded, an evaluation function is applied to its descendants, they are placed in a priority queue, and another node is chosen for expansion. The evaluation function of a node is the sum of the cost so far in reaching that node from the initial state (g), plus the heuristic estimate of the cost remaining to reach a goal state from the node (h). The next node expanded is always

¹This research was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0165, and by the National Science Foundation under grant IST-84-18879.

²Current address: Department of Computer Science, University of California, Los Angeles, Ca. 90024

³personal communication, 1984

a node of least cost among all those in the queue. To facilitate comparison with the iterative-deepening algorithm, we will assume that ties are broken by expanding the most recently generated node among all those of least cost. The algorithm terminates when a goal state is chosen for expansion.

The disadvantage of A^* is its space requirement. Every open node, i.e. those on the frontier of the search tree, must be stored since any one may eventually become the one with the least cost. Pearl has shown that if the heuristic used by A^* exhibits even constant relative error, then the number of nodes generated by the algorithm increases exponentially with solution cost [Pearl 84]. He also observes that heuristics with better accuracy almost never occur in practice. For example, most physical measurements are subject to constant relative error. Thus, we can conclude that in practice the number of nodes on the open list is an exponential function of solution cost and as a result is of the same asymptotic order as the total number of nodes generated.

As a practical matter, an A^* search of most problem spaces will exhaust the available memory long before an appreciable amount of time is used. The reason for this is that the typical ratio of memory to speed in modern computers is a million words of memory for each million instructions per second (MIPS) of processor speed. For example, if we can generate a million states per minute and require a word to store each state, memory will be exhausted in one minute.

3. Iterative-Deepening- A^*

The memory limitation of A^* can be overcome by applying iterative-deepening. Iterative-deepening- A^* (IDA*) works as follows: Starting from the initial state, perform a depth-first search, cutting off a branch when its total cost ($g+h$) exceeds a given threshold. Repeat the depth-first search with successively increasing thresholds until a goal state is selected for expansion. The threshold starts at the heuristic estimate of the cost of the initial state. After each iteration, the next threshold is always the minimum cost of all values that exceeded the previous threshold.

A well-known property of A^* is that it always finds a cheapest solution path if the heuristic is *admissible*, or in other words never overestimates the actual distance to the goal [Hart 68]. As we will see, this property also holds for iterative-deepening- A^* . In addition, since IDA* is a series of depth-first searches, at any given point only the stack of states corresponding to the current search path need be stored, and hence the space complexity of this algorithm is linear in the solution cost. Furthermore, we will show that for tree searches IDA* expands the same number of nodes, asymptotically, as A^* .

Theorem 1: Given an admissible heuristic with constant relative error, then heuristic depth-first iterative-deepening is optimal in terms of solution cost, time, and space, over the class of admissible best-first tree searches.

Proof: We first consider solution cost. Since the initial cost cutoff of IDA* is the heuristic estimate of the cost of the initial state, and the heuristic never overestimates cost, the cost of the shortest solution cannot be less than the initial cost cutoff. In addition,

since the cost cutoff for each succeeding iteration is the minimum value which exceeded the previous cutoff, and again the heuristic never overestimates, no paths can have a cost which lies in a gap between two successive cutoffs. Finally, consider an optimal solution path. Since the cost function never overestimates, no node on such a path can have a value exceeding the optimal cost. Therefore, this path will be generated by IDA* when the threshold equals the optimal cost. Furthermore, no greater cost solution will be found first because no nodes with a greater cost will be expanded on that iteration.

To determine the time used by IDA*, consider the final iteration, in other words the one which finds a solution. It must expand all paths from the initial state whose maximum values are less than the optimal solution cost, plus some number of nodes whose cost equals the optimal solution cost. If A^* employs the depth-first tie-breaking rule of "most recently generated", it will also expand these same nodes. Thus, the final iteration of IDA* expands the same set of nodes as A^* under this tie-breaking rule, and expands each one only once if the graph is a tree.

IDA* must also expand nodes during the previous iterations as well. As mentioned above, if the heuristic used by A^* exhibits constant relative error, then the number of nodes generated by the algorithm increases exponentially with cost [Pearl 84]. The branching factor b is determined by the accuracy of the heuristic function. We can use this fact to show that the previous iterations of IDA* do not affect the asymptotic order of the total number of nodes generated, as follows.

Let e be the smallest cost difference between two nodes of different cost, and let d be c/e , or the optimal solution cost expressed as a multiple of this smallest cost difference. In the worst case, the initial cost threshold would be zero and each successive threshold would increase by e . Under these conditions, d represents the "depth" of the solution or the iteration during which the solution is found. Since b was defined as the ratio between the number of nodes whose costs differ by e , the number of nodes at the frontier of iteration d is b^d . These nodes are expanded once during the final iteration of the search. The nodes at the frontier of iteration $d-1$ are expanded twice, once during iteration d , and once during iteration $d-1$. Similarly, the nodes at iteration $d-2$ are expanded three times, during iterations d , $d-1$, and $d-2$, etc. Thus, the total number of nodes expanded in a heuristic depth-first iterative-deepening search with d iterations is

$$\begin{array}{ll} b^d + 2b^{d-1} + 3b^{d-2} + \dots + db & \text{Factoring out } b^d, \\ b^d(1 + 2b^{-1} + 3b^{-2} + \dots + db^{1-d}) & \text{Letting } x = 1/b \\ b^d(1 + 2x^1 + 3x^2 + \dots + dx^{d-1}) & \text{This is less than} \\ b^d(1 + 2x^1 + 3x^2 + 4x^3 + \dots) & \text{which equals} \\ b^d(1-x)^{-2} & \text{for } abs(x) < 1 \end{array}$$

Since $(1-x)^{-2}$, or $(1-1/b)^{-2}$, is a constant that is independent of d , if $b > 1$ then the running time of depth-first iterative-deepening is $O(b^d)$.

Thus, IDA* expands the same number of nodes, asymptotically, as A^* for exponential tree searches. A recent result of Dechter and Pearl [Dechter 83] shows

that A^* is optimal, in terms of number of nodes generated, over the class of admissible best-first searches with *monotone* heuristics. A monotone heuristic function is one which never decreases along a path. This is not a serious restriction since given an admissible heuristic, we can trivially construct a monotone one by taking the maximum value along the path so far as the value of each succeeding node on the path [Mero 84]. Therefore, IDA^* is asymptotically optimal in terms of time over the class of admissible best-first tree searches.

Finally, we consider the space used by IDA^* . Since IDA^* at any point is engaged in a depth-first search, it need only store a stack of nodes which represents the branch of the tree it is expanding. Since it finds a solution of optimal cost, the maximum depth of this stack is d , and hence the maximum amount of space is $O(d)$.

To show that this is optimal, we note that any algorithm which uses $f(n)$ time must use at least $k \log f(n)$ space for some constant k [Hopcroft 79]. The reason is that the algorithm must proceed through $f(n)$ distinct states before looping or terminating, and hence must be able to store that many states. Since storing $f(n)$ states requires $\log f(n)$ bits, and $\log b^d$ is $d \log b$, any brute-force algorithm must use kd space, for some constant k . Q.E.D.

As mentioned above, most heuristics in practice exhibit at least constant relative error. Thus, we can conclude that heuristic depth-first iterative-deepening is asymptotically optimal for most best-first tree searches which occur in practice. An additional benefit of IDA^* over A^* is that it is simpler to implement since there are no open or closed lists to be managed. A simple recursion performs the depth-first search, and an outer loop handles the iterations.

As an empirical test of the practicality of this algorithm, both IDA^* and A^* were implemented for the Fifteen Puzzle. The implementations were in Pascal and were run on a DEC 2060. The heuristic function used for both was the admissible Manhattan distance heuristic: for each movable tile, the number of grid units between the current position of the tile and its goal position are computed, and these values are summed for all tiles. The two algorithms were tested against 100 randomly generated, solvable initial states. IDA^* solved all instances with a median time of 30 CPU minutes, generating over 1.5 million nodes per minute. The average solution length was 53 moves and the maximum was 66 moves. A^* solved none of the instances since it ran out of space after about 30,000 nodes were stored. As far as we know, this is the first algorithm to find optimal solutions to randomly generated instances of the Fifteen Puzzle within practical resource limits. An additional observation is that in experiments with the Eight Puzzle, even though IDA^* generated more nodes than A^* , it actually ran faster than A^* on the same problem instances, due to the reduced overhead per node. The actual data from these experiments are reported in [Korf 85].

4. Conclusions

The best known admissible search algorithm, A^* , is severely limited by its exponential space requirement. An iterative-deepening version of A^* uses only linear space, and is asymptotically optimal in terms of cost of solution, running time, and space required for exponential tree searches. Since almost all heuristic searches have exponential complexity, iterative-deepening- A^* is an optimal admissible tree search in practice. In addition, it was easier to implement and ran faster than A^* in our experiments.

Acknowledgments

I would like to acknowledge the helpful comments of Mike Lebowitz, Andy Mayer, and Mike Townsend who read an earlier draft of this paper, and several helpful discussions with Hans Berliner and Judea Pearl concerning this research. In addition, Andy Mayer implemented the A^* version that was compared with IDA^* .

References

- [Berliner 84] Berliner, Hans, and Gordon Goetsch. A quantitative study of search methods and the effect of constraint satisfaction. technical report CMU-CS-84-147, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa., July, 1984.
- [Dechter 83] Dechter, Rina, and Judea Pearl. The optimality of A^* revisited. Proceedings of the National Conference on Artificial Intelligence, Washington, D.C., August, 1983, pp. 95-99.
- [Hart 68] Hart, Peter E., Nils J. Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths." *IEEE Transactions on Systems Science and Cybernetics* SSC-4, 2 (1968).
- [Hopcroft 79] Hopcroft, John E., and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Mass., 1979.
- [Korf 85] Korf, Richard E. "Depth-first iterative-deepening: An optimal admissible tree search." *Artificial Intelligence to appear* (1985).
- [Mero 84] Mero, Laszlo. "A heuristic search algorithm with modifiable estimate." *Artificial Intelligence* 23 (1984).
- [Pearl 84] Pearl, Judea. *Heuristics*. Addison-Wesley, Reading, Mass., 1984.
- [Slate 77] Slate, David J., and Lawrence R. Atkin. CHESS 4.5 - The Northwestern University chess program. In Frey, Peter W., Ed., *Chess Skill in Man and Machine*, Springer-Verlag, New York, 1977.
- [Stickel 85] Stickel, Mark E., and W. Mabry Tyson. An analysis of consecutively bounded depth-first search with applications in automated deduction. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-85), Los Angeles, Ca., August, 1985.