UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-91-2216 CRHC-91-9	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois	6b. OFFICE SYMBOL (If applicable) N/A
7a. NAME OF MONITORING ORGANIZATION NASA Ames Research Ctr. National Science Foundation	
6c. ADDRESS (City, State, and ZIP Code) 1101 W. Springfield Avenue Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) Moffett Field CA 94035 Washington, DC 02236
8a. NAME OF FUNDING/SPONSORING ORGANIZATION see 7a	8b. OFFICE SYMBOL (If applicable)
9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER NCC20481 MIP88-10584	
8c. ADDRESS (City, State, and ZIP Code) 7b.	10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Mida*:An Ida* Search With Dynamic Control

12. PERSONAL AUTHOR(S)
Wah, Benjamin W.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM ____ TO ____	1991 April 5	13

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
automated reasoning, search algorithm threshold, heuristic values,

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

In this paper, we extend Korf's IDA* search method so it can determine the threshold used in each iteration of the search dynamically at run time. Current IDA* search stra- tegies are based on the assumption that the distribution of nodes in the search tree by their lower-bound values is exponen- tial, so a static strategy that extends the thresholds by a con- stant amount in each iteration will result in an exponential increase in the number of nodes expanded in successive itera- tions. We show that this distribution is not necessary exponen- tial for all problems, and that a dynamic strategy which deter- mines the thresholds, based on run-time information collected, will result in a lower search overhead. The objective function we use for evaluating IDA* searches is the ratio of the number of nodes evaluated by an IDA* search and that evaluated by a pure best-first search. We analyze the perfor- mance of IDA* searches for both continuous and discrete search problems and establish the optimal performance of an IDA* search. Finally, we develop strategies for determin- ing thresholds at run time and show performance results for the 15-puzzle, traveling-salesman, and knapsack problems.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

MIDA*: AN IDA* SEARCH WITH DYNAMIC CONTROL

Benjamin W. Wah

Center for Reliable and High Performance Computing
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1101 West Springfield Avenue
Urbana, IL 61801
wah@aquinas.csl.uiuc.edu

ABSTRACT

In this paper, we extend Korf's IDA* search method so it can determine the threshold used in each iteration of the search dynamically at run time. Current IDA* search strategies are based on the assumption that the distribution of nodes in the search tree by their lower-bound values is exponential, so a *static* strategy that extends the thresholds by a constant amount in each iteration will result in an exponential increase in the number of nodes expanded in successive iterations. We show that this distribution is not necessary exponential for all problems, and that a dynamic strategy which determines the thresholds, based on run-time information collected, will result in a lower search overhead. The objective function we use for evaluating IDA* searches is the ratio of the number of nodes evaluated by an IDA* search and that evaluated by a pure best-first search. We analyze the performance of IDA* searches for both continuous and discrete search problems and establish the optimal performance of an IDA* search. Finally, we develop strategies for determining thresholds at run time and show performance results for the 15-puzzle, traveling-salesman, and knapsack problems.

PRIMARY TOPIC: Automated Reasoning

SECONDARY TOPIC: Search Algorithms

Research was supported partially by National Aeronautics and Space Administration Contract NCC 2-481 and by National Science Foundation Grant MIP 88-10584.

1. INTRODUCTION

The evaluation of search algorithms is often limited resource constraints, such as the maximum space allowed. Traditional search strategies such as best-first and depth-first strategies do not work well. Various researchers have proposed search methods that cope with limited memory space: depth-m search [3], IDA* [4], MA* [1], MREC [6], and speculative search [2]. We focus in this paper on improving IDA* and MREC.

Korf in his seminal paper [4] proposed iterative-deepening A* or IDA*, a search strategy that can operate in a memory space linear in the size of the problem and that can approach asymptotically the behavior of A*. Like A*, it requires an admissible lower-bound function. It is a variant of *depth-first iterative deepening* (*DFID*): a series of distinct depth-first searches to progressively greater depths to mimic a breadth-first search. As originally described, IDA* initially sets an *incumbent* or *threshold* to the (lower-bound) value of the root node s, and searches depth-first from s, backtracking when it reaches nodes whose values exceed the threshold. Such a depth-first search is a *stage* or *iteration*. If a stage finds a solution, that solution is optimal; if not, IDA* alters the threshold, setting it to the smallest value borne by any of the leaves of the stage (its *periphery*). Then it does the next stage: a new depth-first search from the root, discarding all results of the previous stage apart from the new threshold.

Korf demonstrated the performance of IDA* using the 15-puzzle problem. Since the lower-bound value of a child node in the 15-puzzle problem either increases by 0 or 2, the threshold value in successive stages of an IDA* search always increases by 2. By noting that there is an exponential distribution of the number of nodes in the search tree with respect to their heuristic values, it is guaranteed that the last depth-first search, which finds the optimal solution, has an overhead which overwhelms the total overhead of all previous depth-first searches.

MREC [6] is essentially an IDA* that uses extra memory to save nodes near the root. So long as memory remains, it extends the stored search tree (or graph) by doing a series of depth-first stages from the root to nodes whose values just exceed an incumbent. The only differences from IDA* are that it does not re-expand nodes that are already in storage, and that it stores new nodes that are not. When memory is exhausted, it searches as before, but adds no more nodes to storage. Assuming that there is memory space for storing m nodes in the search tree, m nodes are generated in the first iteration to fill the memory. Successive iterations of MREC expand nodes similar to those expanded by IDA* except for nodes that lead to the m active nodes in memory. Note that IDA* is a special case of MREC with $m = 1$.

The extension of thresholds in successive iterations of an IDA* search is generally controlled by a *static* algorithm. Korf notes that IDA* search does not perform well on the traveling-salesman problem using the IDA* as presented originally, and suggests that performance can be improved by using thresholds sufficiently exceeding the value of the minimum leaf of the previous stage [5]. In general, a *dynamic* algorithm which determines the thresholds based on run-time information collected during the search is necessary. In the following, we discuss two problems associated with controlling and evaluating IDA* searches.

First, the distribution of the number of nodes expanded or generated in the search tree by their lower-bound values is assumed to be exponential in the original IDA* search studied by Korf [4]. Although this assumption is valid for many search problems, such as the 15-puzzle problems (which we verified by testing the distributions of the 15-puzzle problems studied by Korf), the density functions of a large number of finite optimization problems, such as the traveling-salesman and integer programming problems, are bell-shaped. Many nodes in the search tree have lower-bound values centered in the middle of the range. The latter assumption has been verified for the traveling-salesman, resource-constrained scheduling, knapsack, and production planning problems when the density of *all* nodes in the search space are plotted with respect to their lower-bound values. We found that the corresponding distributions rise initially in a form similar to an exponential distribution and level off when all nodes in the search space are expanded.

The assumption that the distributions are exponential is a *good* approximation for many optimization problems when we consider only nodes expanded before the optimal solution is found. This is true in the symmetric traveling-salesman and production-planning problems, which have many feasible solutions. An exponential distribution simplifies the determination of thresholds in the IDA* search because a constant increase in thresholds results in exponential increase in the number of nodes expanded.

For problems that we need to expand *all* nodes in the search space before termination, such as integer programming problems with no feasible solution, and for problems that have pruning due to dominance relations, such as knapsack problems, a non-exponential distribution is a better fit and should be used in deciding the thresholds applied. One difficulty exists when the distribution levels off eventually: it is impossible to determine, without knowing all the parameters of the distribution, when the distribution will level off.

In this paper, we address the above problem by selecting dynamically at run time the best distribution function to use in determining thresholds. The particular function used and its coefficients are unknown before the search begins, and such information is determined during the search based on information of nodes generated. Such an approach avoids the problem of starting with an assumed (and possibly incorrect) distribution and developing a method for extending thresholds without examining dynamic information collected during the search.

The second problem associated with using the conventional IDA* search is that only its asymptotic performance has been determined [4]. This performance behavior indicates that both IDA* and A* behaves identically (with a constant factor of difference) in the asymptotic case. In practice, for searches that terminate in finite time, it is more important to minimize the number of nodes generated beyond a pure A* search when the search terminates, assuming that the search is run in a fixed memory space. Let n_{A^*} (resp. n_{IDA^*}) be the number of nodes expanded by an A* search (resp. IDA* search) for solving a given problem instance. The objective in designing a good IDA* strategy is to

$$\text{minimize } I = \frac{n_{IDA^*}}{n_{A^*}} \tag{1.1}$$

This objective works under both finite and asymptotic conditions. Note that the number of nodes searched by an A* search can be found by discounting the nodes search beyond the optimal solution in the last iteration of an IDA* search.

In this paper we develop methods to determine the thresholds used in each iteration of the IDA* search, so that the resulting ratio defined in Eq. (1.1) can be reduced when the search terminates. *A more restricted objective is to find thresholds so that the numbers of nodes searched in successive iterations increase in a geometric fashion* with a constant ratio of increase r, where $r > 1$. This restricted objective results in a strategy which can be analyzed. We further assume that only admissible heuristic functions are used in the search.

2. IDA* SEARCH WITH CONTINUOUS LOWER BOUNDS

In this section, we analyze the performance of IDA* searches for problems with continuous lower-bound values. We assume that thresholds in the IDA* search can be chosen so that the numbers of nodes expanded in successive iterations always increase by a factor r. Let n_{t_1} be the number of nodes with lower-bound values less than or equal to threshold t_1 (for minimization problems), and n_{t_2} be the number of nodes with lower-bound values less than or equal to t_2, the next threshold chosen after t_1. Then

$$r = \frac{n_{t_2}}{n_{t_1}} \tag{2.1}$$

We find, for a given r, the performance of IDA*, and the optimal value of r that minimizes the increase in overhead as compared to an A* search (Eq. (1.1)).

In addition to the stack for carrying out depth-first searches, let m be the memory space available for storing active nodes. Let n_{A^*} be the number of nodes expanded by an A* algorithm, $n_{IDA^*}^{last,\,opt}$ be the number of nodes expanded by the IDA* search in the last iteration if the optimal solution were found, $n_{IDA^*}^{last,\,no\text{-}opt}$ be the number of nodes expanded by IDA* in the last iteration if *no* solution were found, and e_0 be the number of nodes expanded to generate m active nodes initially. Then the overshoot in the last iteration is characterized by q, which is defined as follows.

$$q = \frac{e_0 + n_{IDA^*}^{last,\,opt} - n_{A^*}}{e_0 + n_{IDA^*}^{last,\,no\text{-}opt} - n_{A^*}}, \qquad 0 \le q \le 1, \quad m \ge 1 . \tag{2.2}$$

The numerator in Eq. (2.2) represents the overshoot of IDA* in the last iteration, while the denominator represents the overshoot of IDA* in the last iteration assuming no solution were found. The denominator normalizes the numerator, so q is between 0 and 1.

Using the parameters defined, the objective I defined in Eq. (1.1) is a function of n_{A^*}, r, m, and q.

The value of r defined in Eq. (2.1) is important in controlling the performance of the IDA* search. If r is too small, then there will be too much repeated work in successive iterations; if r is too large, then there is overshoot in the last iteration. The best choice of r is to have a value such that the optimal solution is included in the first iteration. Such a choice is very difficult. A compromise is to choose a value of r between r_{min} and r_{max}.

Note that q is defined only for a given problem instance. A value of q close to zero means that there is little overshoot, while a value of q close to one means that the overshoot is large. The overhead due to overshoot is large if the threshold of the last IDA* iteration is extended too far beyond the value of the optimal solution. Although q is neither constant nor predictable before the search is completed, its statistical behavior for problem instances evaluated before is useful in selecting the r to be used in evaluating problem instances in the same class by IDA* search. If q is generally small, then choosing a large r results in a faster search, since the overhead due to overshoot is small.

Although it is impossible to achieve the ideal case in which r is chosen properly based on values of n_{A^*} and q, and successive iterations always increase by a factor r in the nodes expanded, the results derived in this section serve as benchmarks for comparing with the real performance of an IDA* search. Results in this section show the maximum and minimum increase in overhead as compared to an A^* search for all values of n_{A^*}; hence the knowledge of n_{A^*} before the search begins is unnecessary. During the evaluation of the IDA^* search, heuristic methods are applied to define thresholds so that successive iterations grow approximately by the factor of r prescribed. Heuristic methods for choosing thresholds are discussed in Section 4.

The following theorem relates n_{IDA^*}, the number of nodes expanded in an IDA^* search, to n_{A^*}, q, r, and m. The theorem shows the increase in the number of nodes expanded by an IDA^* search with respect to an A^* search.

Theorem 2.1. Assume that we can expand by a factor of r nodes in successive iterations of an IDA^* search. For given n_{A^*}, r, q, and m, the number of completed IDA^* iterations, not counting the last iteration, is

$$k = \left\lfloor \log_r \left[\frac{(n_{A^*} - 1)}{m} \right] \right\rfloor \tag{2.3}$$

n_{IDA^*}, the number of nodes expanded in IDA^*, is

$$n_{IDA^*} = n_{A^*} + \frac{m\,r\,(r^k - 1)}{r-1} + q\left[m\,r^{k+1} - n_{A^*} + \frac{(m-1)}{r-1} \right] \tag{2.4}$$

Proof. The proof is shown in four parts.

(a) We first find the number of nodes expanded so that there are m active nodes in memory. Assume that a best-first search is applied until m nodes are generated and that the branching factor of the best-first search is r.[1] We are interested in e_0, the number of nodes expanded in the search tree to obtain m active nodes. Since the root is assumed to exist initially, and each nonterminal node is expanded into r nodes, the following equation must be satisfied.

$$e_0 \cdot r = e_0 + m - 1 \quad \rightarrow \quad e_0 = \frac{m-1}{r-1} \tag{2.5}$$

(b) Next, we prove that k is as defined in Eq. (2.3). Starting with m nodes initially, and assuming k completed iterations of the IDA^* search (not counting the last), the k-th completed iteration must have expanded $m\,r^k$ nodes. Therefore, n_{A^*} must satisfy the following formula.

1. The assumption that the branching factor is r is a simplification here. In reality, the branching factor is r_0, which is different from r. There is negligible error introduced since m is generally much smaller than n_{A^*}.

$$m\,r^k \le (n_{A^*} - 1) < m\,r^{k+1} \tag{2.6}$$

Simplifying Eq. (2.6) results in Eq. (2.3).

(c) We find e_1, the total number of nodes expanded in the k completed iterations. Starting with m active nodes, the i-th, $1 \le i \le k$, iteration expands $m\,r^i$ nodes. Hence, e_1 is evaluated as

$$e_1 = \sum_{i=1}^{k} m\,r^i = \frac{m\,r\,(r^k - 1)}{r-1} \tag{2.7}$$

(d) Last, we find $n_{\mathrm{IDA}^*}^{\text{last, opt}}$, the number of nodes expanded in the last iteration, from Eq. (2.2). Since $n_{\mathrm{IDA}^*}^{\text{last, no-opt}} = m\,r^{k+1}$, Eq. (2.2) can be simplified as

$$n_{\mathrm{IDA}^*}^{\text{last, opt}} = (n_{A^*} - e_0)(1-q) + q\,m\,r^{k+1} . \tag{2.8}$$

Eq. (2.4) is proved by summing and simplifying the results found in Eq's (2.5), (2.7), and (2.8). □

To illustrate Eq's (2.4) and (2.5), suppose $m=17$, $r=3$, $q=0.3$, and $n_{A^*}=20008$. e_0 as computed by Eq. (2.6) is 8. Eq. (2.4) shows that $k=6$: starting with 17 nodes, the number of nodes expanded in the 6 completed iterations of an IDA* search are 51, 153, 459, 1,377, 4,131, and 12,393, which sum to 18,564 and can be verified by Eq. (2.7). The number of nodes expanded in the last iteration is computed by Eq. (2.8) and is 25,154. Hence, the total number of nodes expanded in the IDA* search is 43,726, which is 2.19 times more than what are expanded in an A* search.

To illustrate the results of the theorem, we plot in Figure 2.1 the overheads normalized with respect to n_{A^*} for $m=1$, (a) $r=3$, $q=0.5$ and (b) $r=10$, $q=0.05$. We observe (a) that for small values of r, the average normalized overhead and frequency of variations tend to be higher than those for large values for r, (b) that the normalized overhead is the lowest when the optimal solution lies exactly on the threshold of the last IDA* iteration, and (c) that larger values of r should be chosen when q is small.

Given that r is between $r_{\min}$ and $r_{\max}$, the next theorem shows the optimal value of r that will minimize the normalized overhead.

Theorem 2.2. For $r_{\min} \le r \le r_{\max}$, r_{opt}, the optimal value of r that minimizes $n_{\mathrm{IDA}^*}/n_{A^*}$ is

$$r_{opt} = \begin{cases} \exp\left[\dfrac{\log_e\left[\dfrac{n_{A^*}}{m}\right]}{\left\lceil \log_{r_{\max}}\left[\dfrac{n_{A^*}}{m}\right]\right\rceil}\right] & \left[\dfrac{n_{A^*}}{m}\right] > r_{\max} \\ \dfrac{n_{A^*}}{m} & \left[\dfrac{n_{A^*}}{m}\right] \le r_{\max} \end{cases} \tag{2.9}$$

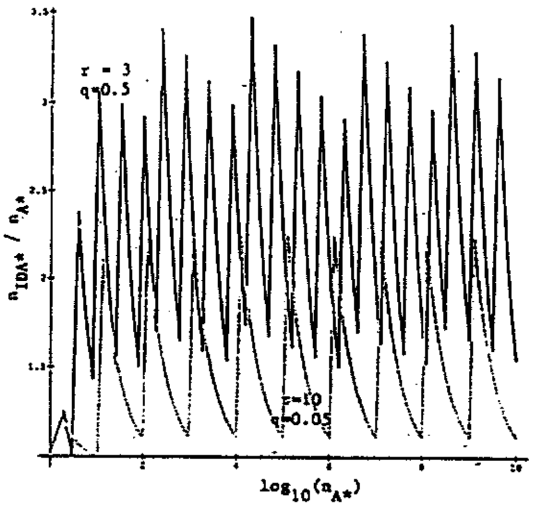

Figure 2.2. Normalized Overheads for $m = 1$ (Theorem 2.1).

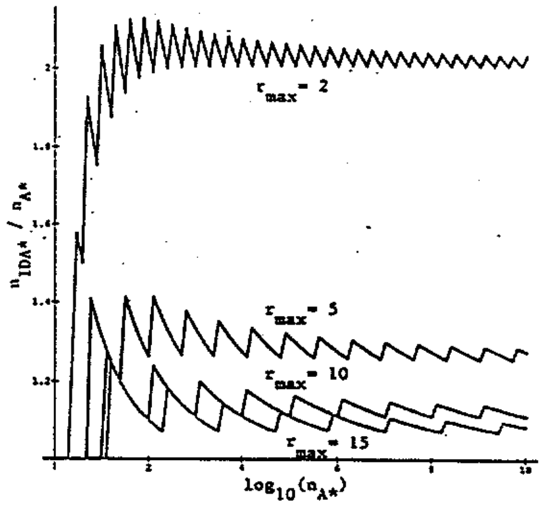

Figure 2.3. Minimum Normalized Overheads based on r_{opt} ($r_{\min} = 1$) (Theorem 2.2).

The increase in overhead as compared to an A* search is

$$\left.\frac{n_{IDA^*}}{n_{A^*}}\right|_{r=r_{opt}} = \frac{r_{opt}}{r_{opt}-1}\left[1-\frac{m}{n_{A^*}}\right] \tag{2.10}$$

Proof. The second part of Eq. (2.9) is straightforward because if the optimal solution can be generated in the first iteration of the IDA* search, then the optimal r to be used is n_{A^*}/m. In the remainder of this proof, we assume that $(n_{A^*}/m) > r_{\max}$.

For r between $r_{\min}$ and $r_{\max}$, there is a set $\{r_1, r_2, ..., r_Q\}$, $r_{\max} \geq r_1 > r_2 > \cdots > r_Q \geq r_{\min}$, such that

$$\frac{n_{A^*}}{m} = r_i^{k_i+1}, \qquad \text{where } k_i \text{ is integer.} \tag{2.11}$$

That is, r_i represents the ratio used such that the IDA* search terminates in $(k_i + 1)$ iterations and that the optimal solution lies *on the threshold* of the last iteration. Note that

$$k_1 = \left\lceil \log_{r_{\max}} \frac{n_{A^*}}{m} \right\rceil - 1\,; \quad k_Q = \left\lceil \log_{r_{\min}} \frac{n_{A^*}}{m} \right\rceil - 1\,; \quad k_{i+1} = k_i + 1, \quad 1 \leq i < Q \tag{2.12}$$

For example, for $r_{\min} = 3$, $r_{\max} = 10$, $n_{A^*} = 624$, $m = 1$, then $k_1 = 3$, $r_1 = 8.545$,

$k_2 = 4$, $r_2 = 4.998$, $k_3 = k_\varrho = 5$, and $r_3 = r_\varrho = 3.623$.

Let $T(n_{A^*}, m, r, q)$ (or in short, $T(r)$) be the overhead of the IDA* search. The proof has to be shown in three parts: (a) $T(r_1) < T(r_i)$ for $r_i < r_1$ and $1 < i \le \varrho + 1$, (b) $T(\hat{r_i}) > T(r_i)$ for $r_{i+1} < \hat{r_i} < r_i$ and $1 \le i < \varrho$, and (c) $T(r_0) > T(r_1)$ for $r_1 < r_0 < r_{\max}$. In each case, we can establish that $T(r_1)$ is the minimum by comparing corresponding terms in e_1 (Eq. 2.7). Due to the complexity of the proof and the space limitation, the proof is omitted here.

Substituting the value of k_1 in Eq. (2.12) into Eq. (2.11) to find r_1, Eq. (2.9) is proved. Eq. (2.10) is proved by finding n_{IDA^*} and assuming k_1 iterations. □

To illustrate Theorem 2.2, the normalized overheads using r_{opt} for various $r_{\max}$ are plotted in Figure 2.2. We have the following observations about the optimal overheads. (a) The optimal normalized overhead is close to one as $r_{\max}$ increases. (b) The frequency of variation is higher for smaller $r_{\max}$. Note that the curves are not smooth because r_{opt} is different for different n_{A^*}.

3. IDA* SEARCH WITH DISCRETE LOWER BOUNDS

For problems with discrete lower bounds, we assume that nodes in the search tree are arranged in *levels*: nodes with the same lower-bound values are on the same level. We are interested in finding the number of nodes expanded rather than those generated because nodes in a level of the search tree is either completely expanded or not expanded in an iteration, while nodes in a level may not be completely generated in the given iteration. By knowing the growth of nodes in successive levels, it is straightforward to estimate the total number of nodes expanded in successive iterations.

There is a distinct difference between the discrete and the continuous cases when a solution is found one level below the threshold of the last iteration. The search can terminate immediately at this point because the solution found must be optimal, and all nodes with solutions less than the optimal solution have been expanded in the last iteration. This phenomenon does not exist when lower bounds are continuous values. On the other hand, when a solution is found at a level other than the level immediately below the threshold of the previous iteration, the search cannot be terminated because it is possible for a better solution to be found in an intermediate level in between.

In controlling the IDA* search, it is easy to show that a new iteration should extend the threshold by at least one level. In choosing the threshold of the next iteration, the distribution of nodes in levels searched in previous iterations can be used so that the number of nodes expanded in the next iteration is at least r times more than that expanded in the previous iteration. Assuming that the threshold of an iteration is set at discrete levels, the problem is in finding how many levels to ''skip'' from the current threshold in defining the threshold of the next iteration.

The following symbols are defined in our analysis before we prove Theorem 3.1, which shows the number of nodes expanded in the IDA* search.

R: Ratio of the number of nodes in one level to that in the next level;

m: $= R^{\nu+1}$, where ν is a real number representing the equivalent number of levels evaluated completely in generating the m initial nodes;

$\ell+\nu$: level number where the optimal solution lies;

p: fraction of nodes in level $\ell+\nu$ that have to be expanded before the optimal solution is found;

s: number of levels skipped in defining the threshold of the next iteration of IDA* search so that the number of nodes expanded in the next iteration is increased by a factor r (defined in Eq. (2.1)).

We show the following theorem without proof (due to space limitation). The theorem shows an *estimate* of the number of nodes expanded in finding the optimal solution using an IDA* search. In this model we assume that nodes in successive levels grow by a constant ratio R, that the behavior of growth is similar for nodes with lower bounds larger than the optimal solution, and that no other optimal solution exist besides the unique one found. The advantage of using the number of nodes expanded is that we do not need to know the branching degree of the search tree.

Theorem 3.1.. n_{IDA^*}, the number of nodes expanded to find the optimal solution in the discrete case is obtained by summing d_1, d_2, d_3, and d_4, where

(a) d_1 is the total number of nodes expanded up to and including level $\ell+\nu$,

$$d_1 = \frac{R^{\nu+\ell+1} - 1}{R - 1}; \tag{3.1}$$

(b) d_2 is the number of nodes not evaluated in the last iteration due to pruning,

$$d_2 = \begin{cases} -(1-p)\,\dfrac{R^{\ell+\nu+1} - R^{\nu+1}}{R-1} & (\ell+1) \bmod s = 0 \\ -(1-p)\,R^{\ell+\nu} & (\ell+1) \bmod s > 0 \end{cases} \tag{3.2}$$

(c) d_3 is the number of overshoot nodes expanded beyond level $\ell+\nu$ in the final iteration,

$$d_3 = \begin{cases} 0 & \ell \bmod s = 0 \\ p\,R^{\ell+\nu+1}\left[\dfrac{R^{s-\ell \bmod s} - 1}{R-1}\right] & \ell \bmod s > 0 \end{cases} \tag{3.3}$$

(d) d_4 is the number of nodes evaluated in intermediate iterations,

$$d_4 = \begin{cases} \dfrac{R^{v+(t+1)s+1} - (t+1)R^{v+s+1} + tR^{v+1}}{(R-1)\,(R^s-1)} & t \geq 1 \\ 0 & t = 0 \end{cases} \tag{3.4}$$

$$\text{where} \quad t = \begin{cases} \dfrac{\ell}{s} - 1 & \ell \bmod s = 0 \\ \left\lfloor \dfrac{\ell}{s} \right\rfloor & \ell \bmod s > 0 \end{cases} \tag{3.5}$$

We illustrate the theorem by applying it on the 15-puzzle problems that Korf studied [4], which fit the assumptions stated in Theorem 3.1. In the 15-puzzle problems, the numbers of nodes in successive levels grow in approximately a geometric fashion. For problem 1 of the 15-puzzle problems tested by Korf, $\ell=9$, and the numbers of nodes at the various levels are 110 (LB=41), 662 (LB=43), 4,825 (LB=45), 31,772 (LB=47), 194,458 (LB=49), 1,178,160 (LB=51), 7,025,558 (LB=53), 412,162,086 (LB=55), and 237,913,457 (LB=57). The optimal solution exists in the level with LB=57 after expanding 66,281,653 nodes; hence $p=0.28$. With the above values, the average increase in nodes in successive levels is $R=6.21$. Since LB starts at 41 with 110 nodes, we assume that $m=17.71$ so that when these nodes are expanded, 110 nodes will be generated in level 1. The corresponding v is $(\log_{6.21} 110)-1=1.57$. From the above data, we can compute the number of nodes expanded in a pure best-first search as 115,879,285. With $s=1$, the actual IDA* search expands 140,036,821 nodes. Our model defined in Theorem 3.1 estimates that $d_1=287{,}893{,}653$, $d_2=-207{,}283{,}415$, $d_3=0$, and $d_4=55{,}257{,}706$, resulting in a total of 135,867,944 and an error of 3%. With $s=2$, our model estimates that d_1 and d_2 are the same as the case of $s=1$, and $d_3=419{,}979{,}261$ and $d_4=47{,}593{,}730$. The total number of nodes that we estimate to be expanded when $s=2$ is 548,183,229.

Table 3.1 shows a summary of the simulation results and predictions on the first 50 15-puzzle problems studied by Korf [4]. Results in Table 3.1 with s=1 is obtained by simulations, while results with larger s are predicted using Theorem 3.1. Our results predict that Korf's original IDA* algorithm (with s=1) is the most efficient algorithm to use on the average, although it is possible that evaluations with $s>1$ could result in a smaller overhead. This happens when the optimal solution lies exactly on the threshold of the last iteration.

Table 3.1. Summary of results using different thresholds for the first 50 15-puzzle problems studied by Korf [4].

Statistics	*Levels*	R	p	n_{IDA^*}/n_{A^*}				
				s=1	s=2	s=3	s=4	s=5
minimum	6.00	4.23	0.01	1.12	1.02	1.00	1.00	1.00
maximum	12.00	44.62	0.98	1.31	29.52	1095.45	28759.49	41102.47
average	8.80	10.22	0.34	1.20	3.88	39.93	915.48	1130.31
std. dev.	1.54	8.24	0.32	0.03	5.14	157.22	4396.06	5865.01

Table 4.1. Summary of results for dynamic selection of thresholds for 20 random symmetric TSPs and knapsack problems

Statistics	n_{A^*}	q	n_{IDA^*}/n_{A^*}				
			r=2	r=3	r=4	r=5	r=6
Statistics of 20 random 22-city symmetric TS problems							
minimum	4055.00	0.00	1.28	1.28	1.28	1.28	1.28
maximum	2848047.00	0.38	3.14	2.74	2.71	2.30	2.50
average	290288.85	0.09	2.61	2.03	1.92	1.73	1.78
std. dev.	637829.81	0.14	0.43	0.37	0.38	0.35	0.28
Statistics of 20 random 110-object KS problems							
minimum	124749.00	0.00	1.40	1.00	1.00	1.00	1.00
maximum	526267.00	0.02	2.50	2.31	2.29	2.22	2.02
average	251668.00	0.00	1.97	1.72	1.63	1.55	1.51
std. dev.	99684.65	0.00	0.32	0.35	0.34	0.38	0.28

4. DYNAMIC PREDICTION OF THRESHOLDS

Predicting the threshold dynamically is achieved by regressing on the (partial) distributions obtained at run time using a polynomial fit (first-order or second-order) or an exponential fit. The particular equation selected is based on the one that gives the smallest mean square error. Simulation results are shown in Table 4.1. r is the desired growth ratio that we wish to achieve, although the ratios of growth achieved usually deviate from the desired one. Our results show (a) that an exponential fit is better in symmetric TSPs, (b) that the second-order polynomial fit is better for knapsack problems (due to the extensive pruning by dominance relations), and (c) that r=5 (resp. r=6) gives the minimum average increase over A^* for *TSPs* (resp. knapsack problems).

ACKNOWLEDGEMENTS

The author would like to acknowledge the help of Mr. Lon-Chan Chu, who developed WISE, an integrated search environment, and implemented the search algorithms for the traveling salesman and knapsack problems studied in this paper. The author is also indebted to Mr. Mark Gooley, who implemented the search algorithm for the 15-puzzle problems, and to Mr. Arthur Ieumwananonthachai, who implemented the IDA* search in WISE.

REFERENCES

[1] P. P. Chakrabarti, S. Ghose, A. Acharya, and S. C. de Sarkar, "Heuristic Search in Restricted Memory," *Artificial Intelligence*, no. 41, pp. 197-221, 1989.

[2] M. M. Gooley and B. W. Wah, "Speculative Search: An Efficient Search Algorithm for Limited Memory," *Proc. Int'l Workshop on Tools for Artificial Intelligence*, pp. 194-200, IEEE, Nov. 1990.

[3] T. Ibaraki, "*m*-depth search in branch-and-bound algorithms," *Int'l. J. of Computer and Information Sciences*, vol. 7, no. 4, pp. 315-373, Plenum Press, 1978.

[4] R. E. Korf, "Depth-First Iterative Deepening: An Optimal Admissible Tree Search," *Artificial Intelligence*, vol. 27, pp. 97-109, North-Holland, 1985.

[5] R. E. Korf, "Optimal Path Finding Algorithms," in *Search in Artificial Intelligence*, ed. V. Kumar, pp. 223-267, Springer-Verlag, New York, 1988.

[6] A. K. Sen and A. Bagchi, "Fast Recursive Formulations for Best-Fist Search That Allow Controlled Use of Memory," *Proc. Int'l Joint Conf. on Artificial Intelligence*, pp. 297-302, IJCAI, Inc., Detroit, MI, Aug. 1989.