An Average-case Analysis of Graph Search

Anup K. Sen
Indian Institute of Management Calcutta
Joka, D. H. Road, Post Box 16757
Calcutta 700 027, INDIA
sen@iimcal.ac.in

Amitava Bagchi ♣
School of Management
University of Texas at Dallas
Richardson, Texas 75083
abagchi@utdallas.edu

Weixiong Zhang ♠
Computer Science Department
Washington University
St. Louis, Missouri 63130
zhang@cs.wustl.edu

Abstract

Many problems in real-world applications require searching graphs. Understanding the performance of search algorithms has been one of the eminent tasks of heuristic search research. Despite the importance of graph search algorithms, the research of analyzing their performance is limited, and most work on search algorithm analysis has been focused on tree search algorithms. One of the major obstacles to analyzing graph search is that no single graph is an appropriate representative of graph search problems. In this paper, we propose one possible approach to analyzing graph search: Analyzing the performance of graph search algorithms on a representative graph of a cluster of problems. We specifically consider job-sequencing problems in which a set of jobs must be sequenced on a machine such that a penalty function is minimized. We analyze the performance of A* graph search algorithm on an abstract model that closely represents job sequencing problems. It is an extension to a model widely used previously for analyzing tree search. One of the main results of our analysis is the existence of a gap of computational cost between two classes of job sequencing problems, one with exponential and the other with polynomial complexity. We provide experimental results showing that real job sequencing problems indeed have a huge difference on computational costs under different conditions.

Introduction and Overview

Graph search has been shown in many cases to be more effective and efficient than tree search. There are real-world applications where tree search is simply not feasible. For example, sequence alignment, an important problem in computational biology that can be formulated as a shortest-path problem in a grid, is only amenable to graph search algorithms [Korf and Zhang 2000]. There are also real problems that can be solved more efficiently by graph search algorithms. For instance, it was shown in [Sen and Bagchi 1996] that when the evaluation function is non-order-preserving ([Pearl 1984], pp. 100-102), graph search for job sequencing problems significantly outperforms tree search in terms of running time. Moreover, a graph search usually uses much less memory than a tree search [Sen, Bagchi and Ramaswamy 1996], making many large problems solvable on our current machines.

Despite its importance in understanding, characterizing and solving difficult problems, the performance analysis of graph search algorithms is almost an untouched topic. This sharply contrasts to a large amount of effort and literature devoted to the topic of performance analysis of tree search algorithms [Huyn, Dechter and Pearl 1980, Pearl 1984, Bagchi and Sen 1988, Davis 1990, Chenoweth and Davis 1991, Zhang and Korf 1995, Korf, Reid and Edelkamp 2001]. To further advance the state-of-the-art on heuristic search, especially on performance analysis, it is desirable to extend our current research to the performance analysis of graph search.

One major difficulty that has crippled the research on the performance analysis of graph search algorithms is perhaps that no single graph is an authenticated representative of various real search problems. Therefore, general results on the performance of graph search seem to be out of reach, which to some extent explains and reflects the state-of-the-art on performance analysis of graph search. On the other end of the spectrum of possibilities to performance analysis, we may consider each individual problem that we encounter. There are numerous important graph search problems. To solve many of them and try to generalize the results may be a tedious and difficult task.

In this research, we consider an alternative to the performance analysis of graph search. We take a middle ground between a "general" graph search problem and a single problem, i.e., we consider a representative model of a set of related problems. We hope that the results will not only shed lights on individual class of problems, but also can be combined relatively easily to provide a deep understanding of graph search problems and algorithms.

In this paper, we are particularly interested in a class of job sequencing problem, an important topic in Computer Science and Operations Research. Job sequencing and scheduling problems appear in many real applications in manufacturing and production systems as well as in information-processing environments [Pinedo 1995]. We consider the class of problems in which N jobs must be so sequenced on a machine that a penalty function on job completion time is minimized. The penalty function may be

♣ On leave from Indian Institute of Management Calcutta.

♠ Funded in part by NSF Grants IIS-0196057 and IET-0111386, and in part by DARPA Cooperative Agreements F30602-00-2-0531 and F33615-01-C-1897.

in various other different forms, such as to minimize the mean job lateness and/or earliness, weighted sum of non-linear functions of completion times, etc.

Our analytic model of job sequencing problems is a graph that defines a partial ordering of subsets of a set of N elements under the set inclusion property. In this graph, there are 2^N nodes; the set of N elements is the root node at level 0 and the empty set is the goal node at level N. Thus it is a directed acyclic graph (DAG) with one goal node and allows multiple solution paths.

To make analysis feasible, following [Pearl 1984] it is assumed that the normalized errors of heuristic estimates of non-goal nodes are independent, identically distributed (i.i.d.) random variables. Using this abstract model, we analyze the expected complexity of A* graph search algorithm, measured by the number of node expansions [Pearl 1984]. We choose A* because it is optimal in terms of the number of node expansions among all algorithms that use the same heuristic information [Dechter and Pearl 1985]. Therefore, the results reflect the expected complexity of searching the abstract model as well as the underlying problems represented by the model.

We present two main theoretical results in this paper. First, we show that under certain weak conditions the expected number of distinct nodes expanded by A* increases exponentially with the number of jobs N for large N. This result matches the previous experimental results on the single machine job sequencing applications [Sen, Bagchi and Ramaswamy 1996]. Second, we identify cases of interest where the expected number of node expansions of A* is polynomial in N for large N. These two classes of complexity indicate that the expected complexity of A* graph algorithm on job sequencing problems has two phases, one exponential and the other polynomial, showing a huge gap similar to a phenomenon of phase transitions. Indeed, our experimental results on single machine job sequencing problems support our theoretical analysis. Specifically, we summarize the previous results for the exponential case, and provide new test results on the polynomial case.

The paper is organized as follows. The basic concepts and the analytic graph model are introduced in the next section. We then analyze the expected complexity of A* using the model. The proofs to the theorems are included in the Appendix. Then we present our experimental results. Concluding remarks are given at the end.

Basic Concepts

A *search graph* (or *network*) G is a finite directed graph with nodes n, n', n_1, n_2, ... The search always begins at the *start* (or *root*) node s, and ends at the *goal* node r. Each directed arc (n_1,n_2) in G has a finite *arc cost* $c(n_1,n_2) > 0$. A path is a sequence of directed arcs. A *solution path* is a path that begins at the start node s and ends at the goal node r. The cost c(P) of a path P is the sum of the costs of the arcs that make up the path. The objective of a search algorithm like A* is to find a solution path of minimum cost in G. To find such a solution path, A* uses a nonnegative *heuristic estimate* h(n) associated with each nongoal node n in G; h(n) can be viewed as an estimate of h*(n), which is the cost of a path of least cost from n to the goal node.

Let g*(n) be the cost of a path of least cost from the start node to node n, and let f*(n) = g*(n)+h*(n). Then f*(n) can be viewed as the cost of a solution path of least cost constrained to pass through node n. During an execution of A*, we use g(n) to represent the cost of the path of least cost currently known from s to n. So g(n) can be viewed as the current estimate of g*(n), and f(n) = g(n)+h(n) as the current estimate of f*(n). As is customary, f*(r) denotes the cost of a minimum cost solution path in G.

Our networks are directed acyclic graphs. In such graphs, introducing more than one goal node adds no extra generality because there are many paths from the root to the goal node. When A* is run on such a network, a node may reenter OPEN from CLOSED; as a result, a node may get expanded more than once. Let Z_d and Z_t denote, respectively, the number of distinct nodes expanded by A* and the total number of node expansions made by A* when run on a given network G. Our primary goal is to determine the expected values $E(Z_d)$ and $E(Z_t)$.

In order to assign a probability distribution on the heuristic estimates of nongoal nodes in G in a meaningful way, we adopt the notion of a *normalizing function* [Pearl 1984, pp. 184]. A normalizing function $\Phi(.)$ is a total function with the set of nonnegative real numbers as domain and the set of real numbers ≥ 1 as range. It has the following properties:

(i) $\Phi(0) = 1$;
(ii) $\Phi(x)$ is nondecreasing in x;
(iii) $\Phi(x)$ is unbounded, i.e. the range of Φ has no finite upper bound.

We allow Φ to take one of three functional forms, viz. identity, less-than-linear and logarithmic:

$\Phi(x) = \max\{1, x\}$	identity
$\Phi(x) = \max\{1, x^\delta\}$ for some δ, $0 < \delta < 1$,	less-than-linear
$\Phi(x) = \max\{1, \ln x\}$	logarithmic

The *normalized error* at a nongoal node n is $X(n)=(h(n)-h^*(n))/\Phi(h^*(n))$. We assume that for all nongoal nodes in G, the normalized errors X(n) are i.i.d. random variables. The normalizing function Φ determines the accuracy of the heuristic estimate function. When Φ is the identity function, the magnitude of the error h(n)-h*(n) is proportional to h*(n). The purpose of allowing other functions, such as logarithmic ones for example, is to enable us to study the consequences of limiting the error h(n)-h*(n) to lower order values, implying greater accuracy of heuristic estimates. We use X in place of X(n), as the X(n)'s are identically distributed. Let $F_X(x) = \text{Prob}\{X \leq x\}$ be the cumulative probability distribution function of X, which is nondecreasing in x. We allow $F_X(x)$ to have discontinuities; these must be left-discontinuities, since $F_X(x)$ by definition is right-continuous. We do not assume any specific functional form for $F_X(x)$.

A heuristic function h is *admissible* if for every nongoal node n in the network G, $h(n) \leq h^*(n)$. Otherwise h is *inadmissible*. If $F_X(x) = 1$ for $x \geq 0$, then all nongoal nodes have admissible heuristic estimates with probability 1. Let $a_1 = \text{lub}\{x \mid F_X(x) = 0\}$ and $a_2 = \text{glb}\{x \mid F_X(x) = 1\}$. For $x < a_1$,

$F_X(x)$ is identically 0, while for $x \geq a_2$, $F_X(x)$ is identically 1. The heuristic estimate function is admissible if $a_2 \leq 0$; it is *purely inadmissible* if $a_1 > 0$. Note that when $\Phi(x)$ is the identity function, we must have $a_1 \geq -1$. From now onwards, whenever a normalizing function is under discussion, we assume that the corresponding a_1 and a_2 are finite.

Minimum Penalty Job Sequencing

Let S_N be the set {1, 2, …, N}. The subsets of S_N under the partial ordering relation induced by set inclusion form a directed acyclic graph G_{js} of 2^N nodes. In this graph S_N is the root node at level 0; and the empty set {} is the goal node at level N. The immediate successors of a node n are the various subsets of S_N obtained from n by removing one element. Thus a node corresponding to a subset of k elements has k immediate successors. G_{js} for N = 4 is shown in Figure 1. Such a graph has the following characteristics:

i) A node with i elements is at level N-i.
ii) There are i! distinct paths to a node at level i.
iii) There are ${}^NC_i$ nodes at level i.
iv) The total number of paths starting at the root and going up to level i is ${}^NC_i.(i!)$.

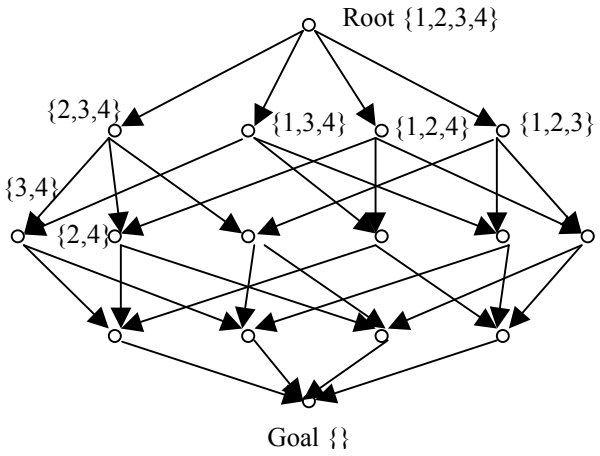

Figure 1: Analytic model G_{js} for N=4

G_{js} is representative of the type of search graphs that arise in certain single machine job-sequencing problems. These search graphs were searched using A* or TCBB [Kaindl et al 1995]. In the analysis below, we restrict ourselves to the A* algorithm due to its optimality in terms of node expansions.

Suppose that jobs J_i with processing times $A_i > 0$, $1 \leq i \leq N$, have been submitted to a one-machine job shop at time 0. The jobs are to be processed on the given machine one at a time. In this analysis we assume that jobs have no setup times. Let the processing of job J_i be completed at time T_i.

Penalty functions $H_i(.)$ are supplied such that the penalty associated with completing job J_i at time T_i is $H_i(T_i) > 0$. H_i is nondecreasing, and in general nonlinear. The jobs must be sequenced on the machine in such a way that the total penalty $F = \Sigma\{H_i(T_i)|\ 1 \leq i \leq N\}$ is minimized. Nodes in G_{js} correspond to (unordered) subsets of jobs that remain to be processed; the root node corresponds to the set of all N jobs, and the goal node to the empty set of jobs. Arc costs are assigned as follows. Suppose there is an arc from node n_1 to node n_2, and suppose job J_i is present in the subset of jobs associated with n_1 but absent from the subset of jobs at n_2; then $c(n_1,n_2) = H_i(T_i)$. Here T_i is the time at which the processing of job J_i is completed; its value does not depend on the order in which jobs prior to job J_i are processed. Since setup times are ignored in this model, arc costs are *order preserving* [Pearl 1984].

We now assign a probability distribution on the heuristic estimates of nodes. To compute the number of nodes expanded, we impose some restrictions on the job processing times and the penalty functions. We first suppose that $1 \leq A_i \leq k$, $1 \leq i \leq N$, for some constant $k > 1$. Thus we do not permit jobs to have arbitrarily large processing times. We then assume that there is a positive integer constant β such that $H_i(x)$ is $o(x^\beta)$, $1 \leq i \leq N$. This means that each penalty function is polynomial in its argument, with the highest power in the polynomial being less than β. These are both reasonable assumptions to make about the processing of jobs in single machine job sequencing problems. Similar assumptions were made in [Sen, Bagchi and Ramaswamy 1996]. Under these circumstances, when the normalizing function is the identity function and the heuristic is not purely inadmissible, the expected number of distinct nodes expanded turns out to be exponential in N for large N.

Theorem 1: Suppose that for $1 \leq i \leq N$, $H_i(x)$ is $o(x^\beta)$ where β is a positive integer constant, and $1 \leq A_i \leq k$ for some constant $k > 1$. If $\Phi(x)$ is the identity function and $a_1 < 0$, then $E(Z_d)$, the expected number of distinct nodes of G_{js} that get expanded, is exponential in N for large N.

Proof: See Appendix. ♦

This theorem is quite general and covers a wide variety of situations that arise in minimum-penalty sequencing (see quadratic penalty problems and other forms in [Sen and Bagchi 1996]). For example:

i) There is really no need to assume a constant upper bound on the job processing times. In the proof, we may assume that there exists a (small) positive integer constant ε such that A_i is $o(N^\varepsilon)$, $1 \leq i \leq N$.
ii) The proof also applies to normalizing functions that are less-than-linear i.e. to $\Phi(x) = x^\delta$, $0 < \delta < 1$.

Experimental results for the above case have been described in Section 4.

What happens if the normalizing function is logarithmic? Nothing specific can be said in general as described in the following example.

Example 1: Let all arcs in G_{js} have unit cost.

i) Suppose the heuristic estimate function h is perfect. Then $E(Z_t)$ can be linear to exponential in N depending on how ties are resolved.
ii) Suppose that the heuristic estimates are not perfect. Suppose $a_1 < 0$, i.e. heuristic estimates of nodes have a nonzero probability of being admissible. Then for any nongoal node n, $f(n) < N$ with probability $p > 0$, so that $E(Z_d) = E(Z_t) \geq 1 + p\ {}^NC_1 + p^2\ {}^NC_2 + \ldots + p^N$ which is exponential in N for large N. This holds even when the normalizing function is logarithmic. ♦

There exist cases of polynomial complexity when the normalization function is logarithmic in nature. In such cases, we don't need the restriction of the constant upper bound on arc costs; instead, we put restrictions on the number of outgoing arcs emanating from a node with arc costs lying within a give upper bound. This assumption covers the cases where the arcs below a node may have varying cost structure. Our assumptions are as follows:

- We restrict ourselves to graphs G_{js} with a cost function defined on its arcs. In such graphs, not too many outgoing arcs at a node have arc costs lying within a given upper bound. Thus, we may assume that for each nongoal node in G_{js}, and for each y, $y > 0$ and integer, there are at most $\min(N-i, \lfloor \theta(y) \rfloor)$ outgoing arcs from node n with arc cost $\leq y$, where i is the level of node n in G_{js} and θ: { 1,2,…,N } → R be a given non-decreasing (total) function with the positive real numbers as domain as well as range. Further, for every nongoal node, there is at least one outgoing arc of arc cost $\leq k$ for some given constant $k \geq 1$ independent of N. Let's call such job sequencing search graphs as *θ-restricted*.
- In addition, we assume that the processing times of jobs are distinct. Then S_N represents the set of jobs with distinct processing times and therefore the jobs in S_N can be viewed as ordered in increasing order of processing times. The costs of the outgoing arcs would then have the same relative order as the processing times of jobs that have been scheduled from the subset. The outgoing arc corresponding to the scheduling of the job with the smallest processing time from a node n would have the lowest cost, and so on. We call such job sequencing graphs *C-ordered*. Thus, for the penalty functions considered in ([Kaindl et al 1995, Sen, Bagchi and Ramaswamy 1996]), the graphs are C-ordered so long as the processing times of jobs are distinct.

Theorem 2: Let $\theta(y) = y^{\beta}$ for integer $y > 0$ and $0 < \beta \leq 1$, and let (G_{js}, C) be θ-restricted and C-ordered. If $\Phi(x)$ is the logarithmic function and $a_1 < 0$, then $E(Z_t)$ is polynomial in N for large N.

Proof: See Appendix. ♦

The following theorem specifies the sufficient condition to be imposed on θ-restricted graphs for which the total number of nodes expanded by A* will always be polynomial in N for large N.

Theorem 3: Let y_0 be a given positive integer independent of N. Let

$$\theta(y) = y^{\beta} \quad \text{for } y \leq y_0,\ \beta > 0$$
$$= y_0^{\beta} \quad \text{for } y > y_0.$$

Let (G_{js}, C) be θ-restricted and the graph be C-ordered. Then regardless what $\Phi(x)$ is, $E(Z_t)$ is polynomial in N for large N.

Proof: See Appendix. ♦

We now present our experimental findings in the next section.

Experimental results

We carried out a number of experiments on a single machine job sequencing problem in which the penalty function for a job is proportional to the square of its completion time. Jobs have processing times but no setup times. All jobs are submitted at time t=0. The objective is to find a sequence of jobs so that the sum of the penalties is minimized [Townsend 1978]. Penalty coefficients (proportionality constants) were taken as integers and were generated randomly in the range 1 to 9 from a uniform distribution. Processing times were also integers and were generated randomly in the range 1 to 99 from a uniform distribution. For a given number of jobs, 100 random instances were generated and each of these instances were solved using A*. The results averaged over these 100 runs are presented in the tables.

Exponential Complexity

We first solved the random instances using A*. The heuristic estimate at a node was computed as suggested in [Townsend 1978], which is known to be consistent [Pearl 1984]. Table 1 below presents the average number of nodes generated and expanded by A* on problems of different sizes. No node is expanded more than once since heuristic is consistent. These results[1] were also presented in [Sen, Bagchi and Ramaswamy 1996]. The nodes generated and expanded are exponential in job size.

Table 1: Performance of A* using consistent heuristic

Job	Nodes Generated		Nodes Expanded	
	Mean	Std Dev	Mean	Std Dev
6	23.99	4.18	5.96	1.54
8	53.06	17.54	10.88	4.52
10	104.05	38.56	17.38	7.25
12	191.78	82.12	26.63	13.06
14	372.51	198.53	44.49	26.28
16	688.61	399.27	71.53	46.30
18	1299.37	837.37	119.70	86.13
20	2327.02	1523.12	191.52	137.97

Next, we experimented assuming that normalizing function is the identity function (Theorem 1). For each of the problem instance,

i) we used the quadratic penalty job sequencing problem where $H_i(T_i)=C_i T_i^2$, that is, $\beta=3$ in Theorem 1;
ii) processing times $A_i <= 99$ for all i.
iii) $\Phi(x)$ is the identity function.
iv) heuristic estimates (h) are admissible.
v) $h = h^* - \wp h^*$ since $\Phi(x)$ is identity function, and $\wp$ is a random number satisfying uniform distribution

The results are given in Table 2. Since heuristic estimate became inconsistent, same node was repeatedly expanded along different paths to a node. As a result, the number of nodes expanded may be more than the number of nodes generated. The results clearly indicate that the performance

[1] Relatively, a little less number of generated and expanded nodes was reported in [Sen, Bagchi and Ramaswamy 1996] since a pruning rule was applied.

of A* with admissible heuristic deteriorates very fast with N when normalizing function is an identity function.

Table 2: A* performance with linear error

Job	Nodes Generated		Nodes Expanded	
	Mean	Std Dev	Mean	Std Dev
6	61.22	4.69	60.55	18.31
8	249.24	22.27	317.28	87.12
10	1001.67	96.65	1606.49	443.89
12	4029.92	400.12	7906.96	1886.63
14	16156.33	1623.79	39349.97	10288.72
16	64752.36	6529.43	185991.06	40743.79

Polynomial Complexity

Next we assumed that the normalizing function is logarithmic in nature (Theorem 2). For each problem instance of the quadratic penalty problem,

i) we generated C-ordered graphs (distinct processing times) which by the nature of the problem are also θ-restricted;
ii) below every node, there are at most N arcs with arc costs < 10 x $(100\ N)^2$;
iii) heuristic estimates (h) are admissible;
iv) $h = h^* - \wp \log(h^*)$ since Φ(x) is logarithmic function and $\wp$ is a random number satisfying uniform distribution.

Table 3: A* performance with logarithmic error

Job	Nodes Generated		Nodes Expanded	
	Mean	Std Dev	Mean	Std Dev
6	21.00	0.00	5.00	0.00
8	36.00	0.00	7.00	0.00
10	55.00	0.00	9.00	0.00
12	78.00	0.00	11.00	0.00
14	105.62	2.49	13.06	0.24
16	136.51	2.52	15.12	0.91

We also experimented with $h = h^* - \log(\kappa h^*)$ where κ is a large constant, the objective being to have a relatively larger logarithmic error. The result obtained is similar, as shown for $\kappa=10^6$ in Table 4.

Table 4: A* performance with logarithmic error, $\kappa=10^6$

Job	Nodes Generated		Nodes Expanded	
	Mean	Std Dev	Mean	Std Dev
6	21.04	0.40	5.01	0.10
8	36.12	0.84	7.02	0.14
10	55.16	1.13	9.02	0.14
12	78.09	0.90	11.01	0.10
14	105.84	2.89	13.08	0.27
16	136.92	3.89	15.15	0.94

With logarithmic error, the node generations and node expansions reduce drastically, and appear polynomial in problem size. Since the error is assumed to be logarithmic, the heuristic estimates are close to perfect heuristic and hence, with distinct arc costs below a node, the performance of algorithm A* becomes polynomial in problem size.

Conclusion and Future Directions

In this paper, we proposed a method to extend the analysis of the average-case performance of A* from tree search problems to graph search problems. The topic has importance because many practical problems can be solved more efficiently using graph search algorithms and a better understanding of their performance may help to characterize the features of real graph search problems. Our main contribution consists of a set of average-case complexity results of A* graph search algorithm on an analytic model that captures the main features of various job sequencing problems. Both our analytical and experimental results show that the expected complexity of job sequencing problems may change from exponential to polynomial with the accuracy of heuristic functions used. In other words, the expected complexity exhibits an exponential to polynomial transition when the heuristic function becomes accurate.

We expect that the approach we proposed and the results we obtained here can be generalized in a number of directions. The first direction is to use a model similar to the incremental random trees [Karp and Pearl 1983, Zhang and Korf 1995]. The second possibility is to directly compare the expected complexity of graph and tree search algorithm on graph problems. The questions to be answered along this direction include the expected savings on the number of nodes explored that a graph search algorithm, such as A*, can provide. Such analysis will help decide which algorithm to use for real applications in practice.

Appendix

Proof of Theorem 1: We renumber jobs if needed and assume that there is a minimal cost solution path in G of cost $M \geq N$, such that if we move along it from root to goal, jobs get scheduled from the set {1,2, ...,N} in the sequence 1 2 ... N. Choose $0 < \delta < 1/(\beta+1)$, and consider the nongoal nodes in G_{js} for which all the missing jobs, corresponding to the jobs already completed, belong to the set {1,2,...,V}, where $V = N^{\delta}$. There are 2^V-1 such nodes excluding the root, and for any such node n, $g(n) < V(Vk)^{\beta}$ and $h^*(n) > M - V(Vk)^{\beta}$.

Let n' be the node for which the missing elements are exactly 1,2,...,V. Then $h^*(n') < M$, since n' lies on the minimum cost solution path, and the cost of the path to n' from any predecessor n of n' cannot exceed $V(Vk)^{\beta}$. It follows that $M - V(Vk)^{\beta} < h^*(n) < M + V(Vk)^{\beta}$. Let $a' = a_1/2 < 0$, so that $p = F_X(a') > 0$. Then $h(n) \leq h^*(n)+a'\Phi(h^*(n))$ with probability p, so that f(n) < M with probability p, provided $g(n)+h^*(n) < M - a'\Phi(h^*(n))$ with probability p. As $a' < 0$ and Φ is the identity function, it suffices to show $M + 2V(Vk)^{\beta} < M - a'M + a'V(Vk)^{\beta}$, which always holds for large N because V is of smaller order than N. The above conditions are true for n' or any ancestor of n'. Thus if node n is at level i, it gets expanded with probability p^i. Therefore, $E(Z_d) \geq 1+pV+p^2\ {}^{V}C_2+...+p^V = (1+p)^V$ which is exponential in N for large N. ♦

Proof of Theorem 2: We first show that $E(Z_d)$ is polynomial in N. To do this, we need to find out the total number of

nodes n with $g^*(n) + h(n) \le f^*(r) + a_2 \ln (f^*(r))$. If node n is at level i, expressing h(n) in terms of h*(n) as $h(n) \ge h^*(n) + a_1 \ln(h^*(n))$, the condition becomes $g^*(n) \le h^*(r) - h^*(n) + a_2 \ln (h^*(r)) - a_1 \ln(h^*(n))$ or $g^*(n) < k\, i + k_0 \ln (kN)$ where k and k_0 are constants, k_0 given by

$$k_0 = \begin{cases} a_2 + |a_1| & \text{if } a_1 < 0 \\ a_2 & \text{if } a_1 \ge 0. \end{cases}$$

Since G is θ-restricted, outgoing arcs at a node have costs bounded below by 1, $2^{1/\beta}$, $3^{1/\beta}$, …, respectively. In computing upper bounds on the number of expanded nodes, these lower bounds can be viewed as the exact costs of the outgoing arcs. We have to find out, for a node n at level i, how many paths from the root to node n of length i have arc costs summing up to at most $k\, i + k_0 \ln (kN)$.

Assuming G_{js} is C-ordered. Consider the subset of jobs corresponding to a node n at level i. Let $J_1, J_2, \ldots, J_i$ be the jobs of the subset which are in S_N but are missing from node n. Suppose $J_1 < J_2 < \ldots < J_i$. There are i! ways of scheduling the i jobs leading to i! paths to node n from the root. The scheduling of the jobs in the sequence $J_1, J_2, \ldots, J_i$ will determine the path of least cost from the root to node n. The arc costs in such a path from the root to node n will form a nondecreasing sequence. Thus, to find an upper bound on $E(Z_d)$, we have to count the number of nondecreasing sequences of length i which sum up to at most $k\, i + k_0 \ln (kN)$ such that each element in the sequence has values 1, $2^{1/\beta}$, $3^{1/\beta}$, ... , since $1/\beta \ge 1$, $k'^{1/\beta} \ge k'$ for any integer $k' > 1$. So to get an upper bound, it is enough to find out the total number of partitions of a positive integer k" and then sum over all $k" \le k_0 \ln(kN)$ and over all levels i. Using the Hardy-Ramanujan asymptotic formula [Andrews 1976, pp. 70, 97] for the number of partitions of an integer, we get

$E(Z_d) \le [k\, N\, (k_0 \ln (kN))\, A_1\, e^{A2 \sqrt{(ko \ln (kN))}}]/[\, k_0 \ln (kN)\,]$ where A_1, A_2 are positive real constants. Thus $E(Z_d) \le A_1\, N\, e^{A2 \sqrt{(ko (\ln kN))}}$ which is polynomial in N for large N.

To get an upper bound on $E(Z_t)$, we have to find out the number of paths from the root to a node n of distinct cost, since n can get expanded along each of these paths in the worst case. Since the outgoing arc costs at a node may be taken as 1, $2^{1/\beta}$, $3^{1/\beta}$, …, the total number of paths of distinct cost to a node at a level $i \le N$ will be polynomial in N. Hence $E(Z_t)$ is also polynomial in N for large N. ♦

Proof of Theorem 3: Let $k_0 = y_0^{\beta}$, a constant. Every node at level i, $1 \le i \le N$, will have at most $\min(k_0, N\text{-}i)$ outgoing arcs having arc costs ≤ y for any $y \ge y_0$; other outgoing arcs can be viewed as having infinite cost. The number of nodes at level i with finite g*-value is $\le {}^{ko+i-1}C_i$ since the job sequencing graph is C-ordered. So the total number of nodes in the graph with finite g*-value is $\le \Sigma\{\, {}^{ko+i-1}C_i \mid 1 \le i \le N \,\} + 1 \le {}^{ko+N}C_N$ which is polynomial in N for large N. As the total number of paths of distinct cost to a node at a level $i \le N$ will be polynomial in N, $E(Z_t)$ is polynomial in N as well. ♦

References

Andrews, George E. 1976. The Theory of Partitions, *Encyclopedia of Mathematics and Its Applications*, vol. 2, Ed. Gian Galo Rota.: Addison –Wesley.

Bagchi, A., and Sen, Anup K. 1988. Average-case analysis of heuristic search in tree-like networks. *Search in Artificial Intelligence* (Ed L N Kanal and V Kumar), Springer-Verlag, pp. 131-165.

Chenoweth, Stephen V., and Davis, H W. 1991. High performance A* search using rapidly growing heuristics. In *Proc* International Joint Conference on Artificial Intelligence (*IJCAI-91*), Sydney, Australia: Morgan Kaufman Publishers, pp. 198-203.

Davis, H. W. 1990. Cost-error relationships in A* tree-searching. *JACM* **37**(2):195-199.

Dechter, R. and Pearl, J. 1985. Generalized best-first search strategies and the optimality of A*. *JACM* **32**: 505-536.

Huyn, N.; Dechter, R.; and Pearl, J. 1980. Probabilistic analysis of the complexity of A*. *Artificial Intelligence* **15**: 241-254.

Kaindl, H.; Kainz, G.; Leeb, A.; and Smetana, H. 1995. How to Use Limited Memory in Heuristic Search. In *Proc. Fourteenth International Joint Conference on Artficial Intelligence (IJCAI-95)*. San Francisco, CA: Morgan Kaufman Publishers, pp. 236—242.

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

Korf, R. E.; Reid, M.; and Edelkamp, S. 2001. Time complexity of iterative-deepening-A*. *Artificial Intelligence* **129**:199-218.

Korf, R. E., and Zhang W. 2000. Divide-and-Conquer Frontier Search Applied to Optimal Sequence Alignment, In *Proc AAAI-2000*, Austin, TX: Morgan Kaufman Publishers, pp. 910-916.

Pearl, J. 1984. Heuristics: Intelligent Search Strategies for Computer Problem Solving.: Addison- Wesley

Pinedo, M. 1995, *Scheduling: Theory, Algorithms and Systems*. Prentice Hall.

Sen, Anup K., Bagchi, A.; and Ramaswamy, R. 1996. Searching graphs with A*: applications to job sequencing. *IEEE Trans. Syst., Man Cybern. Part A Syst, Humans* **26**:168-173.

Sen, Anup K. and Bagchi, A. 1996. Graph search methods for non-order-preserving evaluation functions: Applications to job sequencing problems. *Artificial Intelligence* **86**(1):43-73.

Townsend, W. 1978. The single machine problem with quadratic penalty function of completion times: A branch and bound solution. *Management Science* **24**(5):530-534.

Zhang, W. and Korf, R. E. 1995. Performance of linear-space search algorithms. *Artificial Intelligence* **79**:241-292.