Prediction of Regular Search Tree Growth by Spectral Analysis

Stefan Edelkamp
Institut für Informatik
Georges-Köhler-Allee, Gebäude 51
79110 Freiburg
eMail: edelkamp@informatik.uni-freiburg.de

April 9, 2001

Abstract

The analysis of the time complexity of the IDA* algorithm has shown that predicting the growth of the search tree essentially relies on only two criteria: The number of nodes in the brute-force search tree and the equilibrium distribution of heuristic estimate. Since the latter can be approximated by random sampling we accurately predict the number of nodes of the brute-force search tree for large depth by analyzing the spectrum of the problem graph or one of its factorization.

We further derive that the asymptotic brute-force branching factor is in fact the spectral radius of the problem graph and exemplify our considerations in the domain of the (n^2-1)-Puzzle.

1 Introduction

Heuristic search is essential to AI, since it allows to traverse very large problem spaces with a considerably small number of node expansions. Nevertheless, storing this number of nodes in main memory, as required in the A* algorithm [7], often exceeds main memory resources. This is bypassed in an iterative deepening version of A*, IDA* [9], that searches the tree expansion instead of the original state graph. IDA* applies bounded depth first traversals with an increasing threshold for the same merit function as in A*. The tree expansion may have several duplicate nodes such that low memory consumption is counterbalanced with a considerably high overhead in time.

Fortunately, due to simple search tree pruning rules and expressive heuristic estimates to direct the search process, duplicates in regular search spaces are rare such that IDA* has been very successfully applied to solve solitaire games like the (n^2-1) Puzzle [11] and Rubik's Cube [10].

Korf, Reid and Edelkamp [13] have analyzed the IDA* algorithm to predict the search performance of IDA* in the number of node expansions for a specific problem. The main result is that assuming consistency of the integral heuristic estimate in the limit of large c the expected total number of node expansions with cost threshold c in one iteration of IDA* is equal to

$$\sum_{d=0}^{c} n^{(d)} P(c-d),$$

where $n^{(d)}$ is the number of nodes in the brute-force search tree with depth d and P is the equilibrium distribution defined as the probability distribution of heuristic values in the limit of large depth. More precisely, $P(h)$ is the probability that a randomly and uniformly chosen node in a given depth has a heuristic value less or equal to h. In practice the equilibrium distribution for admissible heuristic functions will be approximated by random sampling [12]; a representative sample of the problem space is drawn and classified according to the integral heuristic evaluation function. The value $n^{(d)}$ for large depths d without necessarily exploring the search tree, can be approximated with the asymptotic brute-force branching factor; the number of nodes at one depth divided by the number of the next shallower depth, in the limit as the depth goes to infinity. Moreover, in some domain we observe anomalies in the limiting behavior of the asymptotic branching factor, in the (n^2-1)-Puzzle and odd values of n it alternates between two different values [4].

The observation that a consistent heuristic estimate h affects the relative depth to a goal instead of the branching itself is backed by the fact that IDA*'s exploration is equivalent to undirected iterative deepening exploration in a re-weighted problem graph with costs $1 + h(v) - h(u)$ for all edges (u, v). The new merit of a node u_j on path $p = (s = u_1, \ldots, u_t = t)$ equals $\sum_{i=1}^{j} 1 + h(u_{i+1}) - h(u_i)$ and telescopes to the old merit minus $h(s)$. Therefore, the heuristic is best understood as a bonus to the search depth. Moreover, since we have only altered edge weights it is not surprising that for bounded heuristic estimates and large depth the asymptotic heuristic branching factor equals the asymptotic brute-force branching factor.

Our main result in this paper is that in undirected problem graphs the value of the number of nodes in depth d of the brute-force search can be computed effectively by analyzing the spectrum of the adjacency matrix for the problem graph. The analysis requires some results of linear algebra and an algorithm of applied mathematics. Since the problem graph is considered to be large for regular search spaces we show how to factorize the problem graph through an equivalence relation of same branching behavior. We take the (n^2-1)-Puzzle as the running example, discuss the generality of the results from various points of view: other problem domains, general, especially undirected graph structures, predecessor pruning, and the equilibrium distribution. Finally we give concluding remarks and shed light on future research options.

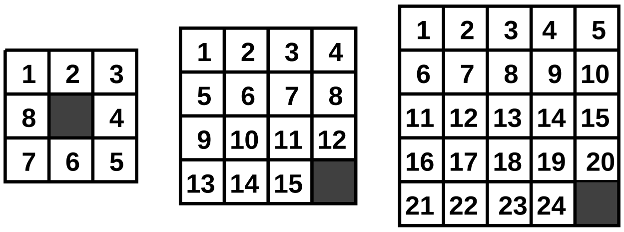

Figure 1: The Eight-, Fifteen and Twenty-Four-Puzzle.

2 Partitioning the Search Space

The (n^2-1)-Puzzle is a sliding tile toy problem. It consists of (n^2-1) numbered tiles that can be slit into a single empty position, called the blank. The goal is to rearrange the tiles such that a certain goal position is reached. Figure 2 depicts possible end configurations of well-known instances to the (n^2-1)-Puzzle: For $n = 3$ we get the Eight-Puzzle, for $n = 4$ the Fifteen-Puzzle and for $n = 5$ the Twenty-Four-Puzzle is met. The state spaces for these problems grow exponentially. The exact number of reachable states (independent of the initial one) is $(n^2)!/2$ which resolves to approximately 10^5 states for the Eight-Puzzle, 10^{13} states in the Fifteen-Puzzle and 10^{25} states in the Twenty-Four-Puzzle.

We partition the search space S according to k different classes $S_1, \ldots, S_k$ grouping states with the same branching behavior. In other words we are given an equivalence relation that partitions the state space into equivalence classes: Two states are equivalent if their long term branching behavior coincides. All states in one equivalence class S_i, $i \in \{1, \ldots, k\}$ necessarily have the same node branching factor, defined as the number of children a node has in the brute-force search tree.

For the example in the (n^2-1)-Puzzle a partition is given by the following relation: Two states are equivalent if the blank is at the same absolute position. Obviously the subtrees of such nodes are isomorphic, since the branching behavior of equivalent states has to be the same. A further reduction of the search tree is established by partitioning the search space with respect to symmetry. For the (n^2-1) Puzzle we establish three branching types: corner or c-nodes with node branching factor 2, side or s-nodes with node branching factor 3, and middle or m-nodes with node branching factor 4. However, does the long time node branching behavior depend on these node types only? In the Eight- and Fifteen-Puzzle this is the case, since for symmetry reasons all c, s and m nodes generate the same subtree structure. For the Twenty-Four-Puzzle, however, the search tree of two side or two middle states may differ. For this case we need six classes with a blank at position 1,2,3,7,8, and 13 according to the tile labeling

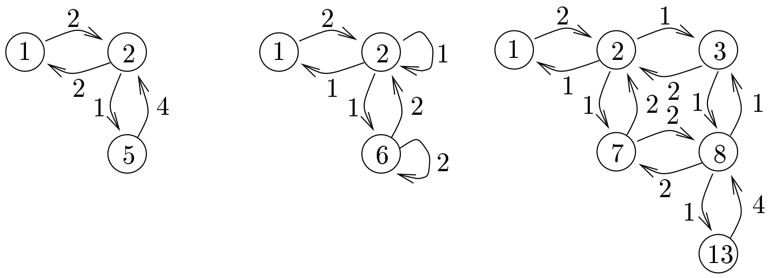

Figure 2: Equivalence Graph for the Eight-, Fifteen- and Twenty-Four-Puzzle.

in Figure 2. In the general case the number of different node branching classes in the (n^2-1) Puzzle is

$$\sum_{i=0}^{\lceil n/2 \rceil} i = \binom{\lceil n/2 \rceil}{2} = \lceil n/2 \rceil (\lceil n/2 \rceil - 1)/2.$$

This still compares well to a partition according to the n^2 equivalent classes in the first factorization (savings of a factor of about eight) and of course to the $(n^2)!/2$ states in the overall search space (exponential savings).

3 Equivalence Graph Structure

Utilizing this partition technique we assign the weighted *equivalence graph* $\overline{G} = (\overline{V}, \overline{E}, w)$. An edge e from class $S_i \in \overline{V}$ to $S_j \in \overline{V}$ with weight $w(e)$ is drawn, if a state in S_i leads to w states in class S_j. Obviously, the sum of all outgoing edges equals the node branching factor. Let $A_{\overline{G}}$ be the adjacency matrix with respect to the *equivalence graph* $\overline{G}$. Since exploration in G and $\overline{G}$ span the same search-tree structure the search tree growth will be the same.

A generator matrix P for the population of nodes according to the given equivalence relation is defined by $P = A_{\overline{G}}^T$. More precisely, $P_{j,i} = l$ if a node of type i in a given level leads to l nodes of type j in the next level. We immediately infer that $N^{(d)} = PN^{(d-1)}$ with $N^{(d)}$ being the vector of nodes in depth d of the search tree. If $||\cdot||_1$ denotes the vector norm $||x|| = |x_1| + |x_2| + \ldots + |x_k|$ then the number of nodes $n^{(d)}$ in depth d is equal to $||N^{(d)}||_1$.

The asymptotic branching factor b (if it exists) is defined as the limit of $n^{(d)}/n^{(d-1)}$ for increasing d and equals the weighted product of the node frequencies $b = \sum_{i=1}^k b_i f_i$, where f_i is the fraction of nodes of class i with respect to the total number of nodes. As we will see we can determine the branching factor analytically without actually determining node frequency values.

The first observation is that in case of convergence the asymptotic branching factor is not only met in the the overall search tree expansion but in every equivalence class. Since all frequencies of nodes converge we have that $b = \lim_{d\to\infty} N_i^{(d)}/N_i^{(d-1)}$, with $N_i^{(d)}$ being the number of nodes of class i in depth d, $i \in \{1,\ldots,k\}$. In other words if the ratio of the cardinality of one equivalence class and the overall search space size settles and the search space size grows with factor b, then the equivalence class size itself grows with factor b.

We represent the fractions f_i as a distribution vector F. We first assume that this vector converges in the limit of large depth. The considerations for an analytical solution to the branching factor problem result in the equations $bF = FP$, where b is the asymptotic branching factor. In addition, we have the equation that the node frequencies sum to one. The underlying mathematical issue is an eigenvalue problem. Transforming $bF = PF$ leads to $0 = (P - bI)F$ for the identity matrix I. The solutions for b are the roots of the characteristic equation $\det(P - bI) = 0$ where det is the determinant of the matrix. Since $det(P - bI) = det(P^T - bI)$ the transposition of the equivalence graph matrix $A_{\overline{G}}$ preserves the value of b. In case of the Eight-Puzzle $det(P - bI)$ equals

$$\det\begin{pmatrix} 0-b & 2 & 0 \\ 2 & 0-b & 1 \\ 0 & 4 & 0-b \end{pmatrix} = 0.$$

This equation is equivalent to $b(4-b^2)+4b = 0$ yielding the following three solutions $-\sqrt{8} = -2.828427124$, 0, $\sqrt{8} = 2.828427124$. Experimental results show that the branching factor alternates every two depth values between 3 and $8/3 = 2.666666666$. Since $\sqrt{8}$ is the geometric average of 3 and $8/3$ the value $\sqrt{8}$ is the proper choice for the asymptotic branching factor b of the brute-force search tree.

For the case of the Fifteen-Puzzle we have to calculate

$$\det\begin{pmatrix} 0-b & 2 & 0 \\ 1 & 1-b & 1 \\ 0 & 2 & 2-b \end{pmatrix} = 0,$$

which simplifies to $(1-b)(b-2)b+4b-4 = 0$. The solution to this equation are 1, $1+\sqrt{5} = 3.236067978$, and $1-\sqrt{5} = -1.236067978$. The value $1+\sqrt{5}$ matches experimental data for the asymptotic branching factor.

For the Twenty-Four-Puzzle we have to solve

$$\det\begin{pmatrix} 0-b & 2 & 0 & 0 & 0 & 0 \\ 1 & 0-b & 1 & 1 & 0 & 0 \\ 0 & 2 & 0-b & 0 & 1 & 0 \\ 0 & 2 & 0 & 0-b & 2 & 0 \\ 0 & 0 & 1 & 2 & 0-b & 1 \\ 0 & 0 & 0 & 0 & 4 & 0-b \end{pmatrix} = 0.$$

The six eigenvalues are 0, 0, $\sqrt{3} = 1.732050808$, $-\sqrt{3} = -1.732050808$, $\sqrt{12} = 3.464101616$, and $-\sqrt{12} = -3.464101616$. Experiments show that for large depth the branching factor oscillates and that the geometric average is 3.464101616.

We conclude that the asymptotic branching factor in the example problems is the *largest* eigenvalue of the adjacency matrix for the equivalence graph and that we observe anomalies if the largest eigenvalues has a negative counterpart of the same absolute value.

In the following we will analyze the structure of the eigenvalue problem to show, why this is the case.

4 Exact Prediction of the Search Tree Size

The equation $N^{(d)} = PN^{(d-1)}$ can be unrolled to $N^{(d)} = P^d N^{(0)}$. We briefly sketch how to compute P^d for large d. Linear algebra states that P is *diagonizable*, if there exists a invertible matrix C and a diagonal matrix Q with $P = CQC^{-1}$. This simplifies the calculation of P^d, since we have $P^d = CQ^dC^{-1}$ (the remaining terms $C^{-1}C$ cancel). By the diagonal shape of Q the value of Q^d is obtained by taking the matrix elements $q_{i,i}$ to the power of d. These elements are called eigenvalues. This connection is not surprising, since in case of convergence of the vector of node frequencies F we have seen that the branching factor itself is a solution to the eigenvalue problem $PF = bF$. We conclude that in case of *diagonizability* we can exactly predict the number of nodes in depth d. The vector of equivalent class sizes by determining the set of eigenvalues of P.

In the example of the Eight-Puzzle the eigenvectors for the eigenvalues $-\sqrt{8}$, 0, and $\sqrt{8}$ are $(2, -\sqrt{8}, 1)^T$, $(-2, 0, 1)^T$, and $(2, \sqrt{8}, 1)^T$, respectively. Therefore, the base-transformation matrix C is given by

$$C = \begin{pmatrix} 2 & -2 & 2 \\ -\sqrt{8} & 0 & \sqrt{8} \\ 1 & 1 & 1 \end{pmatrix}$$

with the following inverse

$$C^{-1} = \frac{1}{16} \begin{pmatrix} 2 & -\sqrt{8} & 4 \\ -2 & 0 & 8 \\ 2 & \sqrt{8} & 4 \end{pmatrix}.$$

With $Q = diag(-\sqrt{8}, 0, \sqrt{8})$ we have $C^{-1}C = I$ and $C^{-1}PC = Q$. Therefore, calculating $N^{(d)} = P^d N^{(0)}$ for $d \geq 1$ corresponds to $N^{(d)} = CQ^dC^{-1}N^{(0)}$, where $Q^d = diag((-\sqrt{8})^d, 0, (\sqrt{8})^d)$. Hence, $N^{(d)}$ equals to

$$\frac{1}{16} \begin{pmatrix} 2 & -2 & 2 \\ -\sqrt{8} & 0 & \sqrt{8} \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} (-\sqrt{8})^d & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & (\sqrt{8})^d \end{pmatrix} \begin{pmatrix} 2 & -\sqrt{8} & 4 \\ -2 & 0 & 8 \\ 2 & \sqrt{8} & 4 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

which resolves to

$$N^{(d)} = \frac{1}{16}\left(4\sqrt{8}^{d}((-1)^{d}+1), 2\sqrt{8}^{d+1}((-1)^{d+1}+1), 2\sqrt{8}^{d}((-1)^{d}+1)\right)^{T}.$$

The exact formula for $N^{(d)}$ and small values of d validates the observed search tree growth: $N^{(1)} = (0,2,0)^T$, $N^{(2)} = (4,0,2)^T$, $N^{(3)} = (0,16,0)^T$, $N^{(4)} = (32,0,16)^T$, etc.

The closed form for $N^{(d)}$ explicitly states that the asymptotic branching factor for the Eight Puzzle is $\sqrt{8}$. Moreover, the odd-even effect for branching in that puzzle is established by the factor $(-1)^d + 1$, which cancels for an odd value of d. Nevertheless, solving the characteristic equation and establishing the basis of eigenvectors by hand is tedious. Fortunately, the application of symbolic mathematical tools such as Maple and Mathematica help to perform the calculations in larger systems.

For the Fifteen-Puzzle the base transformation matrix C and its inverse C^{-1} are

$$C = \begin{pmatrix} 1 & -1 & 1 \\ 1-\sqrt{5} & -1 & 1+\sqrt{5} \\ 3/2-1/2\sqrt{5} & 1 & 3/2+1/2\sqrt{5} \end{pmatrix}$$

and

$$C^{-1} = \begin{pmatrix} 1/50\,(5+3\sqrt{5})\sqrt{5} & -1/50\,(5+\sqrt{5})\sqrt{5} & 1/5 \\ -2/5 & -1/5 & 2/5 \\ 1/50\,(-5+3\sqrt{5})\sqrt{5} & -1/50\,(-5+\sqrt{5})\sqrt{5} & 1/5 \end{pmatrix}.$$

The vector of node counts is

$$N^{(d)} = \begin{pmatrix} 1/50\,(1-\sqrt{5})^{d}(5+3\sqrt{5})\sqrt{5} + 2/5 + \\ 1/50\,(1+\sqrt{5})^{d}(-5+3\sqrt{5})\sqrt{5} \\[1ex] 1/50\,(1-\sqrt{5})(1-\sqrt{5})^{d}(5+3\sqrt{5})\sqrt{5} + 2/5 + \\ 1/50\,(1+\sqrt{5})(1+\sqrt{5})^{d}(-5+3\sqrt{5})\sqrt{5} \\[1ex] 1/50\,(3/2-1/2\sqrt{5})(1-\sqrt{5})^{d}(5+3\sqrt{5})\sqrt{5} - 2/5 + \\ 1/50\,(3/2+1/2\sqrt{5})(1+\sqrt{5})^{d}(-5+3\sqrt{5})\sqrt{5} \end{pmatrix}$$

such that the exact total number of nodes in depth d is

$$1/50\left(7/2-3/2\sqrt{5}\right)\left(1-\sqrt{5}\right)^{d}\left(5+3\sqrt{5}\right)\sqrt{5} + 2/5 +$$
$$1/50\left(7/2+3/2\sqrt{5}\right)\left(1+\sqrt{5}\right)^{d}\left(-5+3\sqrt{5}\right)\sqrt{5}$$

The numbers of corner nodes (1,0,2,2,10,26,90,...), the number of side nodes (0,2,2,10,26,90,282,...) and the number of middle nodes (0,0,6,22,70,230,...) grow as expected. The largest eigenvalue $1+\sqrt{5}$ determines the growth of the search tree in the limit for large d.

In the Twenty-Four-Puzzle the value $N^{(d)}$ equals

$$\left(\begin{array}{c}
1/36\,\left(-2\sqrt{3}\right)^d + 2/9\,\left(-\sqrt{3}\right)^d + 2/9\,\left(\sqrt{3}\right)^d + 1/36\,\left(2\sqrt{3}\right)^d \\
-1/18\sqrt{3}\left(-2\sqrt{3}\right)^d - 2/9\sqrt{3}\left(-\sqrt{3}\right)^d + 2/9\sqrt{3}\left(\sqrt{3}\right)^d + 1/18\sqrt{3}\left(2\sqrt{3}\right)^d \\
1/18\,\left(-2\sqrt{3}\right)^d + 1/9\,\left(-\sqrt{3}\right)^d + 1/9\,\left(\sqrt{3}\right)^d + 1/18\,\left(2\sqrt{3}\right)^d \\
1/12\,\left(-2\sqrt{3}\right)^d + 1/12\,\left(2\sqrt{3}\right)^d \\
-1/18\sqrt{3}\left(-2\sqrt{3}\right)^d + 1/9\sqrt{3}\left(-\sqrt{3}\right)^d - 1/9\sqrt{3}\left(\sqrt{3}\right)^d + 1/18\sqrt{3}\left(2\sqrt{3}\right)^d \\
1/36\,\left(-2\sqrt{3}\right)^d - 1/9\,\left(-\sqrt{3}\right)^d - 1/9\,\left(\sqrt{3}\right)^d + 1/36\,\left(2\sqrt{3}\right)^d
\end{array}\right)$$

for the following total of nodes in depth d

$$n^{(d)} = 1/36\,\left(7-4\sqrt{3}\right)\left(-2\sqrt{3}\right)^d + 1/9\,\left(2-\sqrt{3}\right)\left(-\sqrt{3}\right)^d +$$

$$1/9\,\left(2+\sqrt{3}\right)\left(\sqrt{3}\right)^d + 1/36\,\left(7+4\sqrt{3}\right)\left(2\sqrt{3}\right)^d.$$

The value for small d validates that the total number of nodes increases as expected (2,6,18,60,198,684,...). Once again the vector of the largest absolute value determines search tree growth.

If the size of the system is large, the exact value of $N^{(d)}$ has to be approximated. One option to bypass the intense calculations for determinants of large matrices and roots of high-degree polynomials is to calculate the asymptotic branching factor b. The number of nodes in the brute-force search tree is then approximated by $n^{(d)} \approx b^d$.

5 Approximate Prediction of the Search Tree Size

The matrix denotation for calculating the population of nodes according to the given equivalence relation implies $N^{(d)} = PN^{(d-1)}$, with $N^{(d)}$ being the vector of equivalent class sizes. The asymptotic branching factor b is given by the limit of $||N^{(d)}|_1|/||N^{(d-1)}||_1$ which equals $N_i^{(d)}/N_i^{(d-1)}$ in any component $i \in \{1,\ldots,k\}$. Evaluating $N_i^{(d)}/N_i^{(d-1)}$ for increasing depth d is exactly what is considered in the algorithm of van Mises for approximating the largest eigenvalue (in absolute terms) of P. The algorithm is also referred to as the *power iteration* method.

As a precondition to the algorithm again it is required that P is diagonizable. This implies that we have n different eigenvalues $\lambda_1, \ldots, \lambda_n$ and each eigenvalue λ_i with multiplicity of α_i has α_i linear independent eigenvectors. Without loss of generality we assume that the eigenvalues are given in decreasing order $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_k|$. The algorithm further requires that the start vector $N^{(0)}$ has a representation in the basis of eigenvectors in which no coefficient according to λ_1 is trivial.

We distinguish the following two situations: $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_k|$ and $|\lambda_1| = |\lambda_2| > \ldots \geq |\lambda_k|$. In the first case we obtain that (independent of the choice of $j \in \{1, \ldots, k\}$) the value of $\lim_{d \to \infty} N_j^{(d)}/N_j^{(d-1)}$ equals $|\lambda_1|$. For the second case one can similarity establish that $\lim_{d \to \infty} N_j^{(d)}/N_j^{(d-2)}$ is in fact λ_1^2. The cases $|\lambda_1| = \ldots = |\lambda_l| > \ldots \geq |\lambda_k|$ for $l > 2$ are dealt with similarity. The outcome of the algorithm and therefore the limit in the number of nodes in layers with difference l is $|\lambda_1|^l$, such that once more the geometric average turns out to be $|\lambda_1|$.

We indicate the proof of the first case only. Diagonizability implies an basis of eigenvectors $b_1 \ldots, b_k$. Due to $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$ the quotient of $|\lambda_i/\lambda_1|^d$ converges to zero for large values of d. If the initial vector $N^{(0)}$ with respect to the eigenbasis is given as $x_1 b_1 + x_2 b_2 + \ldots + x_k b_k$ applying P^d yields $x_1 P^d b_1 + x_2 P^d b_2 + \ldots + x_k P^d b_k$ by linearity of P, which further reduces to $x_1 b_1 \lambda_1^d + \lambda_2^d x_2 b_2 + \ldots + \lambda_n^d x_k b_k$ by the definition of eigenvalues and eigenvectors. The term $x_1 b_1 \lambda_1^d$ will dominate the sum for increasing values of d. Factorizing λ_1^d and λ_1^{d-1} in the quotient of $N_j^{(d)}/N_j^{(d-1)}$ results into an equation of the form $x_1 b_1 \lambda_1 + R$ where $\lim_{d \to \infty} R$ is bounded by a constant, since except of the leading term $x_1 b_1 \lambda_1$ both numerator and denominator only involve expressions of the form $O(|\lambda_i/\lambda_1|^d)$.

Therefore, to find the asymptotic branching factor analytically, it suffices to determine the set of eigenvalues of P and to take the largest one. This corresponds to the results of the asymptotic branching factors in the (n^2-1)-Puzzles.

In the Eight-Puzzle the ratio $N_1^{(d)}/N_1^{(d-2)}$ is equal to 8 for $d > 3$ and, therefore, approximates λ_1^2. The value $n^{(d)}/(1+\sqrt{5})^d$ alternates between 3/4 and $1/\sqrt{2}$. Hence, $(1+\sqrt{5})^d$ approximates the search tree growth.

For the Fifteen-Puzzle $N_1^{(d)}/N_1^{(d-1)}$ equals $1, 3, 13/5, 45/13, 47/15, 461/141$, 1485/461,4813/1485, 15565/4813, 50381/15565, 163021/50381, 527565/163021 = 3.236178161, etc. for increasing depth d; a sequence approximating $1+\sqrt{5} =$ 3.236067978. Moreover, the ratio of $n^{(d)}$ and $(1+\sqrt{5})^d$ quickly converges to $1/50\ \left(7/2 + 3/2\sqrt{5}\right)\left(-5 + 3\sqrt{5}\right)\sqrt{5} = .5236067984$.

In the Twenty-Four-Puzzle the ratio $N_1^{(d)}/N_1^{(d-2)}$ converges to 12 starting with the sequence 6, 9, 11, 129/11, 513/43, 683/57, 8193/683, 32769/2731, 43691/3641 = 11.99972535, etc. The quotient $n^{(d)}/(1+\sqrt{5})^d$ for larger depth alternates in between .3888888889 and .3849001795 and is therefore bounded by a small constant.

If n is even – as in the Fifteen-Puzzle – the largest eigenvalue is unique and if n is odd – as in the Eight- and in the Twenty-Four-Puzzle – we find two eigenvalues with the same absolute value verifying that every two depths the node sizes will asymptotically increase by a the square of these value.

6 Generalizing the Approach

Iterating the algorithm with $||N^{(d)}||_1/||N^{(d-1)}||_1$ instead of $N_j^{(d)}/N_j^{(d-1)}$ shows that the convergence according to G and $\overline{G}$ are equivalent. This is important, since other graph properties may alter, e.g. symmetry of A_G is not inherited to $A_{\overline{G}}$. Therefore, we concentrate on diagonizability results of A_G which are easier to obtain. The Theorem of Schur states that symmetric matrices are indeed diagonizable. Moreover, the eigenvalues are real and the matrix to perform the base transformation has the eigenvectors in its columns.

For the (n^2-1)-Puzzle we are done. Since G is undirected, A_G is indeed symmetric. In the spectrum of A_G power iteration either obtains a unique branching factor $b = |\lambda_1|$ or a branching factor of λ_1^2 for every two iterations. Therefore, the branching factor is the *spectral radius* $\rho = |\lambda_1|$.

6.1 Other Problem Spaces

Since the search tree is often exponentially larger than the problem graph we have reduced the prediction of the search tree growth to the spectral analysis of the explicit adjacency representation of the graph. As long this graph is available accurate and approximate predictions for the brute-force and subsequently for the heuristic search tree growth can be computed.

However the calculations for large implicitly given graphs are involved such that reduction of the analysis to a smaller structure is desirable. For the (n^2-1)-Puzzle we proposed a compression to a few branching classes. The application of equivalence class reduction to exactly predict the search tree growth relies on the regular structure of the problem space. This technique is available as long as the same branching behavior for different states in the exploration process is given.

For Rubik's Cube without predecessor elimination $N^{(d)}$ equals 18^d since all nodes in the search tree have a complete 18-ary subtree. With predecessor elimination the node branching factor reduces to 15, since for each of the three rotations single clockwise, double clockwise, counterclockwise there is a remaining of five of sides (front, back, right, left, up and down) that are available. If we further restrict rotation of opposite sides to exactly one order we get the transition matrix $\begin{pmatrix} 6 & 6 \\ 9 & 6 \end{pmatrix}$, where the first class is the set of primary nodes with branching factor 15, and the second class is the class of secondary nodes with branching factor 12. The eigenvalues are $6+3\sqrt{6}$ and $6-3\sqrt{6}$ and the value $n^{(d)}$ equals $1/2\,\left(6+3\sqrt{6}\right)^d + 1/2\,\left(6-3\sqrt{6}\right)^d$. For small values of d experimental data as given in [10] matches this analytical study. The observed asymptotic branching factor is $6+3\sqrt{6} = 13.34846923$ as expected.

6.2 Non-Diagonizability

Since we assumed diagonizability, the eigenspaces $L(\lambda_i)$ according to the values λ_i have full rank α_i. In general this is not true. Not all matrices are diagonalizable. In this case the best thing one can do is to transform the matrix into *Jordan Form* which has blocks on the diagonal, each block being $r \times r$, with the eigenvalue on the diagonal, 1's above the diagonal and 0's everywhere else. More precisely, a matrix A has *Jordan Form* J for an invertible matrix T, if $J = T^{-1}AT$ consists of so-called Jordan-blocks $J_1, \ldots, J_m$. One Jordan-block has an eigenvalue on the main diagonal and ones on the diagonal above. Therefore, T gives a basis of eigenvectors and so-called *main vectors*. Each Jordan-block J_l of dimension j_l corresponds to one eigenvector t_1 and $j_l - 1$ main vectors $t_2, \ldots, t_{j_l}$ with $(A - \lambda_i I)t_0 = 0$ and $(A - \lambda_i I)t_m = t_{m-1}$, $m = 2, \ldots, j_l$. Using the Jordan basis one can devise $P^d N^{(0)}$ similar to the case above.

Fortunately, we know that the branching factor is a positive real value since the iteration process is real. Therefore, we may perform all calculation to predict the search tree growth with complex numbers. For this case the characteristic polynomial factorizes.

6.3 Start Vector

The second subtlety arises even if the matrix is diagonalizable. We are interested in determining the behavior of $P^d N^{(0)}$ for large d, where P is an $n \times n$ matrix and $N^{(0)}$ is an $n \times 1$ vector. Suppose that P is diagonalizable, which means that there is a basis of eigenvectors. Hence, $N^{(0)}$ can be written as a sum of eigenvectors: $N^{(0)} = v_1 + v_2 + v_3 + \ldots + v_n$ where v_i is an eigenvector with eigenvalue λ_i. It follows that $P^d N^{(0)} = \lambda_1^d v_1 + \lambda_2^d v_2 + \lambda_3^d v_3 + \ldots + \lambda_n^d v_n$ So the term with the largest corresponding eigenvalue will dominate for large d, *provided that eigenvector is non-zero.* It may happen that the initial vector v has component of zero in the eigenspace of the largest eigenvalue. In general, the algorithm finds the largest eigenvalue in which the corresponding component is non-zero.

Fortunately, this observation is more theoretical in nature. In the iteration process this case is very rarely fulfilled. As stated in several books on scientific computing rounding errors will soon or later lead to non-zero components. Moreover, to determine the asymptotic branching factor we have several initial states to choose from such that at least one has to yield non-zero coefficients.

6.4 Pruning

When incorporating pruning to exploration symmetry of the underlying graph structure is affected. Once again we consider the Eight-Puzzle. The adjacency matrix A_G^{pred} for predecessor elimination now consists of four classes: cs, sc, mc and cm, where the class ij indicates that the predecessor of a j-node in the search tree is an i node.

$$A_G^{pred} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 \end{pmatrix}$$

In this case we cannot directly infer diagonizability. However, the branching factor and the search tree growth can be calculated analytically and the iteration process eventually converges.

The set of (complex) eigenvalues is $i\sqrt{2}$, $-i\sqrt{2}$, $\sqrt{3}$, and $-\sqrt{3}$. Therefore, the asymptotic branching factor is $\sqrt{3}$. The vector $N^{(d)}$ is equal to

$$\begin{pmatrix} 1/5\left(i\sqrt{2}\right)^d + 1/5\left(-i\sqrt{2}\right)^d + 3/10\left(\sqrt{3}\right)^d + 3/10\left(-\sqrt{3}\right)^d \\ -1/10\, i\sqrt{2}\left(i\sqrt{2}\right)^d + 1/10\, i\sqrt{2}\left(-i\sqrt{2}\right)^d + 1/10\sqrt{3}\left(\sqrt{3}\right)^d - 1/10\sqrt{3}\left(-\sqrt{3}\right)^d \\ 3/20\, i\sqrt{2}\left(i\sqrt{2}\right)^d - 3/20\, i\sqrt{2}\left(-i\sqrt{2}\right)^d + 1/10\sqrt{3}\left(\sqrt{3}\right)^d - 1/10\sqrt{3}\left(-\sqrt{3}\right)^d \\ -1/10\left(i\sqrt{2}\right)^d - 1/10\left(-i\sqrt{2}\right)^d + 1/10\left(\sqrt{3}\right)^d + 1/10\left(-\sqrt{3}\right)^d \end{pmatrix}.$$

Finally, the total number of nodes in depth d is

$$n^{(d)} = 1/5\left(1/2 + 1/4\, i\sqrt{2}\right)\left(i\sqrt{2}\right)^d + 1/5\left(1/2 - 1/4\, i\sqrt{2}\right)\left(-i\sqrt{2}\right)^d + 1/10\left(4 + 2\sqrt{3}\right)\left(\sqrt{3}\right)^d + 1/10\left(4 - 2\sqrt{3}\right)\left(-\sqrt{3}\right)^d.$$

For small values of d the value $n^{(d)}$ equals 1, 2, 4, 8, 10, 20, 34, 68, 94, 188 etc.

6.5 Computing the Equilibrium Distribution

Exact calculation of the brute-force search tree raises the question if the other source of uncertainty, namely the heuristic equilibrium distribution, can also be eliminated.

As said, the equilibrium distribution of the estimate can be obtained by random sampling. However, in some cases of regular search trees exact values can be produced. If the estimate is given with respect to a pattern database storing pairs of the form (estimated value, state pattern) than by analyzing the pattern database a histogram of heuristic values can established: We determine the number of states that satisfy a pattern with a total to be computed for each integral heuristic value in a predefined range. For consistent heuristics this range will be bounded by the heuristic estimate of the start state and the optimal solution length.

For algorithmically computed heuristics like the Manhattan Distance in the $(n^2 - 1)$-Puzzle accurate predication is more difficult. However, there are some available options to cope with that problem, for example utilizing a symbolic representation of the heuristic estimated as a boolean formula H encoding pairs of the form (estimate,state). Moreover, applying binary decision diagrams (BDDs [1]) as the unique representation for H is advantageous, since BDDs support counting the number of represented states in time linear to their representation size. Usually, in practice this size turns out to be exponentially smaller that the cardinality of the represented set. Therefore, if we encode the estimate advance to the state variables the number of states represented in the BDD according to a given heuristic estimate can be computed efficiently. In the experimental section of [6] the authors show that H can be constructed effectively and further be used in a symbolic A*-like traversal of the search tree.

It has to be denoted that in some domains like propositional AI-planning heuristic pattern data bases have been computed symbolically [3] such that the approach to determine the equilibrium distribution by symbolic representation and counting is also available in domains, where the total number of occurring states with respect to a given pattern cannot be computed precisely.

7 Previous Work

This paper extends the work of Edelkamp and Korf [4] that already derived the asymptotic branching factors of the sliding-tile puzzles and Rubik's Cube. However, their approach lack sufficient convergence conditions. We established the criterion of diagonizability of the adjacency graph matrix of the problem graph that emerges of the algorithm of van Mises and showed that this criterion is fulfilled in undirected graphs by the Theorem of Schur. The $(n^2 - 1)$-Puzzles and Rubik's Cube are chosen to illustrate the techniques, since they are inherently difficult to solve and often considered in case studies.

The set of recurrence relations in [4] also showed that the numbers of nodes at various depths can be calculated in time linear to the product of node classes and depth of search by numerically iterating the recurrence relations. In contrast to this finding the current paper contributes how to compute a closed form for the number of nodes.

Last but not least, the given mathematical formalization of equivalence classification, diagonalization and power iteration builds a bridge for more powerful results in applying known mathematical theorems. Generality to different problem spaces is given, since this approach applies to any problem graph with a diagonizable matrix and probably to more than that.

8 Conclusion

In the paper have improved the prediction the number of node expansions in IDA* by an exact derivation of the number of nodes in the brute-force search tree. We have resolved the question of convergence to explain anomalies in of the asymptotic branching factor. The asymptotic branching factor is the spectral radius of the successor generation matrix and can be computed with the power iteration method. The approach extends to further regular problem spaces and can cope simple pruning rules.

The main result is that diagonizability is granted in undirected problem graphs, such that exact and approximate calculation of the brute-force search-tree are mathematically sound.

We summarize that given the adjacency matrix P of an undirected graph by analyzing $N_i^{(d)}/N_i^{(d-1)}$ and $N_i^{(d)}/N_i^{(d-k)}$, $k > 1$, of the equation $N^{(d)} = N^{(d-1)}P$ gives the (mean) asymptotic branching factor. This is in fact the algorithm of van Mises to determine the largest eigenvalue of P for which applicability we have to test if P is diagonizable.

Since for practical problems in which IDA* applies it is very unlikely that the entire graph structure can be kept in main memory, the approach helps only if some reduction of the branching behavior with respect to equivalence classes can be obtained. Therefore the analysis is limited to the case where the successor generator matrix can be build. However it closes a gap in literature to accurately predict search tree growth in closed form and to compute the branching factor both analytically and numerically without relying on strong experimental assumption on the convergence.

Extending the work to problem domains like Sokoban or chess with their non-regular search trees is challenging. It is difficult to derive an accurate prediction, since the branching behavior of the tree reveals almost all state facets. Therefore, a more complicated search model has to be devised to derive exact or approximate search tree in these or other domains. As Andreas Junghanns has coined in his Ph.D. dissertation [8], the impact of the search tree node prediction formula $\sum_{d=0}^{c} n^{(d)} P(c-d)$ for these domain is not clear. At the very far end there is the error-free prediction for the IDA* search tree for various kind of heuristics and problem domain.

All analysis given in this or precursury papers on search tree prediction do not include the application of transposition tables, in which visited states together with their best encountered state merits (path length plus heuristic estimate) are kept. This in fact is also a challenge for analysts. One possibility is the prediction of the search tree growth of IDA* with respect to bit-state hashing, which turns out to be an improvement to transposition tables at least in some domains [5]. For this model of partial search some mathematical considerations have been made [2]. Transferring the result to directed search will also be one of the core research topics in future.

Acknowledgments I thank Richard Korf and Michael Reid for intense and helpful discussions on the topic of this paper.

References

[1] R. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, 24(3):142–170, 1992.

[2] J. Eckerle and T. Lais. Limits and possibilities of sequential hashing with supertrace. In *Formal Description Techniques (FORTE XI), Protocol Specification, Testing, and verification (PSTV XVIII)*, Lecture Notes in Computer Science. Springer, 1998.

[3] S. Edelkamp. Directed symbolic exploration and its application to AI-planning. In *AAAI-Spring Symposium on Model-based Validation of Intelligence*, 2001. To Appear.

[4] S. Edelkamp and R. E. Korf. The branching factor of regular search spaces. In *National Conference on Artificial Intelligence (AAAI)*, 1998. 299–304.

[5] S. Edelkamp, A. L. Lafuente, and S. Leue. Protocol verification with heuristic search. In *AAAI-Spring Symposium on Model-based Validation of Intelligence*, pages 75–83, 2001.

[6] S. Edelkamp and F. Reffel. OBDDs in heuristic search. In *German Conference on Artificial Intelligence*, Lecture Notes in Computer Science, pages 81–92. Springer, 1998.

[7] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for heuristic determination of minimum path cost. *IEEE Transaction on SSC*, 4:100, 1968.

[8] A. Junghanns. *Pushing the Limits: New Developments in Single-Agent Search*. PhD thesis, University of Alberta, 1999.

[9] R. E. Korf. Linear-space best-first search. *Artificial Intelligence*, 62(1):41–78, 1993.

[10] R. E. Korf. Finding optimal solutions to Rubik's Cube using pattern databases. In *AAAI*, pages 700–705, 1997.

[11] R. E. Korf and A. Felner. Disjoint pattern database heuristics. *Artificial Intelligence*, 2001. To appear.

[12] R. E. Korf and M. Reid. Complexity analysis of admissible heuristic search. In *National Conference on Artificial Intelligence (AAAI)*, 1998. 305–310.

[13] R. E. Korf, M. Reid, and S. Edelkamp. Time complexity of Iterative-Deepening-A*. *Artificial Intelligence*, 2001. To appear.