Analyzing the Performance of Pattern Database Heuristics

Richard E. Korf
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095
korf@cs.ucla.edu

Abstract

We introduce a model for predicting the performance of IDA* using pattern database heuristics, as a function of the branching factor of the problem, the solution depth, and the size of the pattern databases. While it is known that the larger the pattern database, the more efficient the search, we provide a quantitative analysis of this relationship. In particular, we show that for a single goal state, the number of nodes expanded by IDA* is a fraction of $(\log_b s + 1)/s$ of the nodes expanded by a brute-force search, where b is the branching factor, and s is the size of the pattern database. We also show that by taking the maximum of at least two pattern databases, the number of node expansions decreases linearly with s compared to a brute-force search. We compare our theoretical predictions with empirical performance data on Rubik's Cube. Our model is conservative, and overestimates the actual number of node expansions.

Introduction and Overview

Pattern Database Heuristics

Pattern databases are heuristic functions based on stored lookup tables (Culberson & Schaeffer 1998). For example, the movable pieces of a 3x3x3 Rubik's cube consist of eight *corner cubies*, with three faces each, and twelve *edge cubies*, with two faces each. There are only about 88 million different configurations of the corner cubies. Thus, we can build in memory a table, called a pattern database, that contains for each configuration of the corner cubies, the minimum number of moves required to move them all to their goal configuration. This table is built by performing a complete breadth-first search of the problem space backward from the goal state, considering only the corner cubies and ignoring the edge cubies. The first time that each new corner cubie configuration is reached, its search depth is stored in the table. Since the possible values stored in the table range from zero to eleven, we can store each entry in 4 bits, and the complete table requires only 42 megabytes of memory.

We can use this table to generate lower bounds in a heuristic search such as A* (Hart, Nilsson, & Raphael 1968) or IDA* (Korf 1985). For each state in the search, we look up the configuration of the corner cubies in the pattern database, and use the stored value as a heuristic evaluation for that state. This value is a lower bound, because it represents the minimum number of moves required to solve just the corner cubies, while both the corner and edge cubies must be solved to solve the entire puzzle.

Pattern databases are the most effective heuristics known for combinatorial puzzles such as Rubik's cube (Korf 1997), the sliding-tile puzzles (Korf & Felner 2002), and the Towers of Hanoi (Korf & Felner 2007). They have also been used for sequence alignment in computational biology (Hohwald, Thayer, & Korf 2003), vertex cover (Felner, Korf, & Hanan 2004), and planning problems (Edelkamp 2001).

Overview

In general, the larger the pattern database, the more accurate the heuristic, and the more efficient the search, in terms of node expansions. To date, however, there has not been an accurate quantitative analysis of this space-time tradeoff. The first question we address is given a single pattern database of a given size, how much can we expect it to reduce node expansions, relative to a brute-force search? The maximum of several different lower-bound heuristics is also a lower bound, and often a more accurate one. In the second part of this paper we analyze the quantitative effect on search efficiency of the maximum of several different pattern databases. We compare our theoretical predictions with empirical results on Rubik's Cube. Our model is conservative, and overestimates the actual number of node expansions.

Related Work

The first piece of related work is my initial informal analysis, the second is a set of experimental results, and the third is the theoretical foundation for analyzing the performance of heuristic functions that forms the basis of our analysis.

Informal Intuitive Analysis

In (Korf 1997), I used pattern databases to find the first optimal solutions to randomly scrambled Rubik's Cubes. I also addressed the space-time tradeoff described above, and hypothesized an inverse linear relation between the size of a pattern database and the number of search nodes expanded. For example, doubling the size of a pattern database would halve the number of node expansions. However, that was based on a set of informal intuitive arguments.

Experimental Results

(Holte & Hernadvolgyi 1999) subsequently empirically examined the relationship between node expansions and the size of single pattern databases on three different problems. They found general agreement with the analysis presented here, a preliminary version of which I had communicated to them. Their results are not directly comparable to this analysis, however, since they used the A* algorithm, while the analysis here applies primarily to IDA*.

Performance Analysis of Heuristic Functions

The first rigorous analysis of the performance of heuristic functions that provided precise predictions of numbers of node expansions, rather than asymptotic analyses, was published in (Korf & Reid 1998; Korf, Reid, & Edelkamp 2001). Since our analysis of pattern databases is based on that work, we briefly summarize the main result here.

According to our theory, to predict the number of nodes expanded by a heuristic search algorithm, we need three items of information. The first is the search depth d, which is normally the depth of an optimal solution. The second is the number of nodes in the brute-force problem space at each depth up to d, which we refer to as N_i, where i is the depth. For a depth-first algorithm such as IDA*, N_i can be approximated by b^i, where b is the brute-force branching factor.

The last piece of information is a characterization of the heuristic function. For this we use the heuristic distribution, which is the fraction of nodes that have a given heuristic value, for each possible heuristic value. An equivalent representation of the same information is the cumulative heuristic distribution $P(x)$, which we define as the probability that a randomly chosen state in the problem space has a heuristic value that is less than or equal to x. $P(x)$ is the number of states with heuristic value less than or equal to x, divided by the total number of states. For values of x greater than or equal to the maximum heuristic value, $P(x) = 1$. To be precise, the actual heuristic distribution needed is the distribution of heuristic values at a given depth, which we call the *equilibrium distribution*. For Rubik's Cube, these two distributions are the same, but for other problems they may differ. For simplicity, we use the overall heuristic distribution here.

Let $E(N, d, P)$ be the worst-case number of node expansions by an admissible heuristic search such as A* or IDA*, searching to depth d, in a problem space with N_i nodes at depth i, using a consistent heuristic function with cumulative heuristic distribution P. Then, the main result is:

$$E(N, d, P) = \sum_{i=0}^{d} N_i P(d - i)$$

in the limit of large d. The interested reader is referred to (Korf, Reid, & Edelkamp 2001) for the derivation of this result. An important corollary is that the number of nodes expanded by IDA* searching to depth d, divided by the number expanded to depth $d-1$, is the brute-force branching factor b. While the general result applies to arbitrary consistent cost functions with a minimum operator cost, for simplicity we described the result for the special case where all operators have unit cost, and the cost of a solution is its depth.

Even though we only get equality in the limit of large depth, in practice this formula accurately predicts the actual number of node expansions. On Rubik's Cube, the Eight Puzzle, and the Fifteen Puzzle, the formula predicts the actual number of node expansions by IDA* with an error of less than 1% at typical search depths of 10 to 50 moves, by averaging a large number of problem instances (Korf, Reid, & Edelkamp 2001). While this formula also applies to A*, for A* it is much more difficult to determine N_i, the number of nodes in a brute-force search to depth i, for large depths.

Single Pattern Database Heuristics

We begin by analyzing the performance of a single pattern database heuristic. Given a complete pattern database, its heuristic distribution can be read from it, and the above formula applied. Our challenge, however, is to predict the performance of a pattern database heuristic based only on its size, without constructing the entire database.

Modelling the Heuristic Distribution

The essential difficulty we face is modelling the heuristic distribution, based only on the size of the pattern database. We assume that there is only one goal state. A pattern database for a single goal state is constructed by starting with the goal state, and searching backward until all the pattern configurations are generated, storing with each one its depth in the search. We assume that the branching factor of this backward search is uniform and the same as the forward branching factor b of the original problem space. Thus, there can be only one state with heuristic value zero, at most b states with heuristic value one, at most b^2 states with heuristic value two, etc. We assume that the number of entries in the pattern database with heuristic value x is b^x, for all values of x up to the maximum heuristic value m. Therefore, the cumulative heuristic distribution function is

$$P(x) = \sum_{i=0}^{x} b^i / s$$

for $x < m$, and $P(x) = 1$ for $x \geq m$.

m is determined by the size of the pattern database s. We assign one entry heuristic value zero, b entries heuristic value one, b^2 entries heuristic value two etc, until the number of entries equals or exceeds s. Thus,

$$s = \sum_{i=0}^{m} b^i = \frac{b^{m+1} - 1}{b - 1} \approx \frac{b^{m+1}}{(b - 1)} \Rightarrow s(b - 1) = b^{m+1} \Rightarrow$$

$$\frac{s(b - 1)}{b} = b^m \Rightarrow m = \log_b \left(\frac{s(b - 1)}{b} \right)$$

Note that this is a conservative approximation, and will predict worse performance than will actually occur. Since a state is defined by the configuration of only the pattern elements, such as the corner cubies in Rubik's Cube, there will be many nodes for which the configuration of the pattern elements are identical. Thus, the pattern space is actually a graph, and the number of unique entries at depth x will be less than b^x. Since the total number of database entries is

fixed, fewer nodes at shallow depths must be made up for by more nodes at deeper depths, effectively shifting the heuristic distribution towards larger values, relative to our analytic model. Larger heuristic values result in more pruning, and fewer nodes expanded. Thus, our model will tend to overestimate the actual number of nodes expanded.

Predicting the Number of Node Expansions

Given the formula for $E(N, d, P)$, and an analytic model for $P(x)$, we need a model for N_i, the total number of nodes at depth i. We assume a tree with branching factor b, or $N_i = b^i$. Thus, we model $E(N, d, P)$ as $E(b, d, s)$, since the branching factor, the search depth, and the size of the pattern database are the only parameters to our model.

This will be reasonably accurate for a depth-first search such as IDA*, since it searches a tree expansion of the problem space graph. It may not be accurate for a graph-search algorithm that detects duplicate nodes, such as A*, since the branching factor of a graph eventually decreases with increasing depth, until all unique nodes have been generated. To apply this analysis to A*, we would need a model of the number of nodes in the problem space graph as a function of depth. We are not aware any such general models.

All that now stands between us and the number of node expansions by IDA* is some algebraic manipulation.

$$E(b, d, s) = \sum_{i=0}^{d} N_i P(d-i) = \sum_{i=0}^{d} b^i P(d-i)$$

We assume that the solution depth d is greater than the maximum heuristic value m. Since $P(x) = 1$ for $x \geq m$,

$$E(b, d, s) = \sum_{i=0}^{d-m} b^i + \sum_{i=d-m+1}^{d} b^i P(d-i)$$

Since

$$\sum_{i=0}^{x} b^i = \frac{b^{x+1} - 1}{b-1} \approx \frac{b^{x+1}}{b-1}$$

$$E(b, d, s) \approx \frac{b^{d-m+1}}{b-1} + \sum_{i=d-m+1}^{d} b^i P(d-i)$$

Since

$$P(x) = \sum_{i=0}^{x} b^i / s \approx \frac{b^{x+1}}{(b-1)s}$$

$$E(b, d, s) \approx \frac{b^{d-m+1}}{b-1} + \sum_{i=d-m+1}^{d} b^i \cdot \frac{b^{d-i+1}}{(b-1)s} =$$

$$\frac{b^{d-m+1}}{b-1} + \sum_{i=d-m+1}^{d} \frac{b^{d+1}}{(b-1)s} = \frac{b^{d-m+1}}{b-1} + \frac{m b^{d+1}}{(b-1)s} =$$

$$\frac{b^{d+1}}{b-1}\left(\frac{1}{b^m} + \frac{m}{s}\right)$$

Substituting

$$m = \log_b \left(\frac{s(b-1)}{b}\right) \text{ yields}$$

$$\frac{b^{d+1}}{b-1}\left(\frac{b}{s(b-1)} + \frac{1}{s}\left(\log_b \frac{s(b-1)}{b}\right)\right) =$$

$$\frac{b^{d+1}}{s(b-1)}\left(\frac{b}{b-1} + \log_b \frac{s(b-1)}{b}\right) =$$

$$\frac{b^{d+1}}{s(b-1)}\left(\frac{b}{b-1} + \log_b s + \log_b(b-1) - 1\right)$$

As shown by Table 1 below,

$$f(b) = \frac{b}{b-1} + \log_b(b-1) - 1$$

is very close to one for most values of b, and particularly for large values of b. Thus,

$$E(b, d, s) \approx \frac{b^{d+1}}{s(b-1)}(\log_b s + 1) = \frac{b^{d+1}}{b-1} \cdot \frac{\log_b s + 1}{s}$$

Written in this form, the number of node expansions is the product of the number of node expansions in a brute-force search to depth d, times a fraction representing the savings due to the pattern database heuristic. This shows that our informal analysis in (Korf 1997) was inaccurate, in that it ignored the multiplicative factor of $\log_b s + 1$.

b	2	3	4	5	6	7
$f(b)$	1.000	1.131	1.126	1.111	1.098	1.087

Table 1: Value of $f(b) = b/(b-1) + \log_b(b-1) - 1$

Comparison to Experimental Results

How well does our formula predict the actual number of node expansions in a real problem with a real pattern database? To address this question, we considered the standard 3x3x3 Rubik's Cube. We constructed four different pattern databases based on different subsets of movable cubies. These included all eight corner cubies, six corner cubies,[1] seven edge cubies, and six edge cubies. These databases are large enough to give reasonable performance, but small enough to fit in memory.

There are two main sources of error in our formula. The first is the difference between our model of the heuristic distribution, and the actual heuristic distribution of a real pattern database. Our model assumes there are b^i entries in the database with value i, where b is the branching factor. As explained above, this model is conservative, and will tend to overestimate the number of node expansions.

The second source of error is that our model assumes that the number of nodes in the problem space grows by a factor of b at every level. While this is true at large depths, at shallow depths the number of nodes is often greater. For example, the asymptotic branching factor of Rubik's Cube, counting a 180 degree rotation as a single move, is about 13.34847. The ratios between nodes at successive depths

[1] A pattern database for seven corner cubies is identical to one for eight corner cubies, because once the configuration of any seven corner cubies is fixed, the position and orientation of the last corner cubie is determined.

Database	Size s	T avg	A avg	$E(b, 12, s)$	$c \cdot E(b, 12, s)$	$E(N, 12, P)$	IDA*
6 corners	14,696,640	6.550	7.610	17,244,125	23,492,879	21,002,302	20,918,500
6 edges	42,577,920	6.845	7.573	6,320,185	8,610,430	8,045,477	8,079,408
8 corners	88,179,840	6.999	8.764	3,161,938	4,307,729	3,521,829	3,724,861
7 edges	510,935,040	7.827	8.508	591,606	805,986	676,561	670,231

Table 2: Results for Single Pattern Databases on Rubik's Cube to Depth 12

at the first few levels, however, is 18, 13.5, 13.333, 13.35, and 13.348. At large depths, the actual number of nodes is 1.36237 times the number predicted by the constant branching factor model. Thus, this model underestimates the number of nodes expanded in this problem. Since this constant is determined by the number of nodes in the first few levels of the tree, it is easy to determine for most problems.

Our results are shown in Table 2. Each row corresponds to a different pattern database, which is identified in the first column, ordered by increasing size. The second column gives the number of entries in the pattern database. The third column, labelled "T avg", shows the average pattern database value based on our theoretical model of its distribution, and the fourth column, labelled "A avg", shows the actual average database value. As expected, our model underestimates the average heuristic value in each case.

Each of the remaining columns shows actual or predicted node expansions for an IDA* iteration to a depth of 12 moves. 12 was chosen because it exceeds the maximum of all the heuristic values, but is small enough to run 1000 problem instances each. While less than the median of 18 moves for a randomly scrambled cube, it is sufficient for evaluating our model, since the ratio of the number of nodes expanded at successive depths is exactly b in our analysis, and almost exactly b experimentally (Korf, Reid, & Edelkamp 2001).

The fifth column, labelled $E(b, 12, s)$, is the number of node expansions predicted by our formula for $E(b, d, s)$, with $b = 13.34847$ and $d = 12$. The next column, labelled $c \cdot E(b, 12, s)$ is our prediction multiplied by $c = 1.36237$, to correct for the fact that a constant branching factor underestimates the number of nodes in the Rubik's Cube tree.

The column labelled $E(N, 12, P)$ predicts the number of node expansions based on N_i, the actual number of nodes in the problem space at each depth, and P, the actual heuristic distribution of the pattern databases. The column labelled "IDA*" shows the number of nodes expanded by IDA* using these pattern databases, averaged over 1000 cubes, each scrambled by 100 random moves from the goal state. For each row except the third, the last two column values will differ somewhat depending on the particular set of cubies chosen. The third row is based on all eight corner cubies, hence there is no choice of cubies. The values in the fifth and sixth columns don't change, since they are based only on the sizes of the pattern databases.

In each case, the number of nodes expanded decreases with larger pattern databases, as expected. The last analysis, $E(N, 12, P)$, based on the actual numbers of nodes in the problem-space tree and the actual heuristic distributions, accurately predicts the experimental results. This error could be reduced by running more problem instances, or iterations to greater depths. Our theoretical analysis based on a constant branching factor and the size of the pattern databases, when corrected by the actual number of nodes in the tree, $c \cdot E(b, 12, s)$, overestimates the actual number of node expansions by IDA*, as expected. Finally, the errors due to the constant branching factor assumption, and our model of the heuristic distributions, tend to cancel each other, so that our formula for $E(b, 12, s)$, predicts our experimental results fairly accurately. In particular, in each case the actual performance is bracketed between our formula based on a constant branching factor, $E(b, 12, s)$, and the formula corrected for the increased number of nodes, $c \cdot E(b, 12, s)$.

Multiple Pattern Databases

Given two or more admissible heuristic values, their maximum is admissible as well. It is well known that taking the maximum of several pattern database heuristics provides a more effective heuristic function than a single database whose size is the sum of the sizes of the multiple databases (Holte *et al.* 2004; 2006). Here we use our analytic model to quantify this effect.

Modelling the Heuristic Distribution

As with a single pattern database, the first step in our analysis is to model the heuristic distribution. Recall that the cumulative heuristic distribution $P(x)$ is the probability that a randomly chosen state has a heuristic value less than or equal to x. The maximum of several different heuristics will have a value less than or equal to x if and only if all the individual heuristic values are less than or equal to x. In order to make the analysis tractable, we assume that the different heuristic values are independent, and hence we model the cumulative heuristic distribution of the maximum of several pattern database heuristics as the product of the individual cumulative heuristic distributions. Let $P^k(x)$ be the cumulative heuristic distribution of the maximum of k pattern databases, all of the same size s. Then,

$$P^k(x) = \left(\sum_{i=0}^{x} b^i / s \right)^k = \left(\frac{b^{x+1} - 1}{(b-1)s} \right)^k \approx$$

$$\left(\frac{b^{x+1}}{(b-1)s} \right)^k = \frac{b^{kx+k}}{(b-1)^k s^k}$$

Predicting the Number of Node Expansions

We use $E(b, d, s, k)$ to denote the predicted number of node expansions for an IDA* iteration to depth d, of a problem space with branching factor b, and a heuristic which is the

maximum of k different pattern databases, each containing s entries. Thus, $E(b,d,s,k) =$

$$\sum_{i=0}^{d} b^i P^k(d-i) = \sum_{i=0}^{d-m} b^i + \sum_{i=d-m+1}^{d} b^i P^k(d-i) =$$

$$\frac{b^{d-m+1}-1}{b-1} + \sum_{i=d-m+1}^{d} \frac{b^i b^{k(d-i)+k}}{(b-1)^k s^k} \approx$$

$$\frac{b^{d-m+1}}{b-1} + \frac{1}{(b-1)^k s^k} \sum_{i=d-m+1}^{d} b^{k(d-i)+k+i} =$$

$$\frac{b^{d-m+1}}{b-1} + \frac{b^{kd+k}}{(b-1)^k s^k} \sum_{i=d-m+1}^{d} b^{-(k-1)i}$$

By rewriting the summation using the formula

$$\sum_{i=x}^{y} b^{-zi} = \frac{b^{(1-x)z} - b^{-yz}}{b^z - 1}$$

we get $E(b,d,s,k) =$

$$\frac{b^{d-m+1}}{b-1} + \frac{b^{kd+k}}{(b-1)^k s^k} \left(\frac{b^{(m-d)(k-1)} - b^{-d(k-1)}}{b^{k-1}-1} \right) =$$

$$\frac{b^{d-m+1}}{b-1} + \frac{b^{kd+k}}{(b-1)^k s^k} \left(\frac{b^{mk} b^{-m} b^{-dk} b^d - b^{-dk} b^d}{b^{k-1}-1} \right) =$$

$$\frac{b^{d-m+1}}{b-1} + \frac{b^{kd+k}}{(b-1)^k s^k} \left(\frac{b^{-dk+d}(b^{m(k-1)}-1)}{b^{k-1}-1} \right) =$$

$$\frac{b^{d-m+1}}{b-1} + \frac{b^{k+d}}{(b-1)^k s^k} \left(\frac{b^{m(k-1)}-1}{b^{k-1}-1} \right)$$

substituting $\log_b(s\frac{b-1}{b})$ for m gives us

$$\frac{b^{d+1}}{(b-1)s\left(\frac{b-1}{b}\right)} + \frac{b^{k+d}}{(b-1)^k s^k} \left(\frac{\left(s\frac{b-1}{b}\right)^{k-1} - 1}{b^{k-1}-1} \right) \approx$$

$$\frac{b^{d+1}}{(b-1)s\left(\frac{b-1}{b}\right)} + \frac{b^{k+d}}{(b-1)^k s^k} \left(\frac{\left(s\frac{b-1}{b}\right)^{k-1}}{b^{k-1}-1} \right) =$$

$$\frac{b^{d+2}}{(b-1)^2 s} + \frac{b^{k+d}}{(b-1)^k s^k} \left(\frac{s^{k-1}(b-1)^{k-1}}{b^{k-1}(b^{k-1}-1)} \right) =$$

$$\frac{b^{d+2}}{(b-1)^2 s} + \frac{b^{d+1}}{(b-1)s(b^{k-1}-1)} =$$

$$\frac{b^{d+1}}{s(b-1)} \left(\frac{b}{b-1} + \frac{1}{b^{k-1}-1} \right) \approx E(b,d,s,k)$$

Comparing this to our formula for one pattern database,

$$E(b,d,s) \approx \frac{b^{d+1}}{b-1} \cdot \frac{\log_b s + 1}{s} = \frac{b^{d+1}}{s(b-1)} \cdot (1 + \log_b s)$$

we see that for large b, taking the maximum of two heuristics essentially replaces $\log_b s$ with $1/(b^{k-1}-1)$, and is significantly more efficient than a single database. For example, with $b = 2$, and a one-gigabyte database, $\log_b s = 30$. For large b and/or large k, $1/(b^{k-1}-1)$ approaches zero, and for large b,

$$\frac{b}{b-1} + \frac{1}{b^{k-1}-1}$$

approaches one.

The above analysis assumes that all pattern databases are the same size. Since multiple pattern databases must share the same memory, an alternative assumption is that each of the k databases are of size s/k. Redoing our analysis with this assumption multiplies the number of node expansions by a factor of k, resulting in the formula:

$$E(b,d,s/k,k) \approx \frac{b^{d+1} \cdot k}{s(b-1)} \left(\frac{b}{b-1} + \frac{1}{b^{k-1}-1} \right)$$

Comparison to Experimental Results

How does our theoretical prediction compare to experimental results with real pattern database heuristics? Our model assumes multiple independent pattern databases with the same heuristic distribution, either of the same size s, or of size s/k for k databases. Finding an experimental setting with these properties is not easy. We chose four Rubik's Cube pattern databases based on six edge cubies each. Each such database contains $42,577,920$ entries. Any corner of a cube is the intersection of three edges, and for each corner there is a unique opposite corner with three different adjacent edges. Thus, for each corner and its diagonally opposite corner we get a unique set of six edge cubies. Since there are eight corners total, there are four such pairs of diagonally opposite corners, each generating a set of six edge cubies. Due to the symmetry of the cube, each of these sets is equivalent to the others, and hence has exactly the same heuristic distribution. Furthermore, each pair of six such edge cubies are equivalent to every other pair, and have the same combined heuristic distribution. This is also true of each triple of six such edge cubies. Unfortunately, these pattern databases are not independent. Any pair of six such edge cubies has two edge cubies in common. Even if their edge cubies were disjoint, they still wouldn't be independent, since once we know the locations of six edge cubies, we know that any others can't occupy the same positions.

Table 3 shows our results with these pattern databases. Each row corresponds to the maximum of a different number of pattern databases k, and the numbers represent actual or predicted node expansions. The second column, labelled $E(b,12,s,k)$, is the prediction from our formula above, assuming a constant asymptotic branching factor of 13.34847, a search depth of 12 moves, and the same size $s = 42,577,920$ for each of the databases. The third column, labelled $E(N,12,P^k,k)$, is the prediction based on N, the actual numbers of nodes in the tree at each depth, and a search depth of 12. For the single database in the first row, P is the exact cumulative heuristic distribution, and for the multiple databases, the cumulative heuristic distribution P^k is computed assuming independence of the heuristic values. The last column shows performance data for IDA*, averaged over 1000 randomly scrambled cubes searched to depth 12.

k	$E(b, 12, s, k)$	$E(N, 12, P^k, k)$	IDA*
1	6,320,185	8,045,477	8,079,408
2	944,070	179,824	225,960
3	882,858	79,536	99,064
4	878,615	53,361	66,596

Table 3: Results for the Maximum of Multiple Pattern Databases on Rubik's Cube to Depth 12

The first row of the table is for a single pattern database of six edge cubies. It corresponds to the second row of Table 2. In this case, our formula predicts a smaller number of node expansions than actually occur, due to the larger branching factor at shallow levels of the problem-space tree, compared to the assumption of a constant branching factor in our formula. As shown in the second row of Table 2, however, when we correct for this difference, we bracket the actual number of node expansions. Note that the prediction in the third column, based on the actual numbers of nodes in the tree and the actual heuristic distribution, very accurately predicts the actual number of node expansions in the last column, as we saw Table 2. The difference is due to noise in the average of 1000 problem instances.

The second row represents the maximum of two different pattern databases of six edge cubies each. The first thing to notice is that the number of node expansions predicted by our formula is a significantly less than for a single six-edge pattern database. This represents the loss of the factor of $\log_b s$, where s is the size of the database. Furthermore, the decrease in node expansions for the experimental data is over a factor of 35. In other words, taking the maximum of two different pattern databases provides an enormous reduction in node expansions.

The next thing to notice is that our formula in the second column significantly overestimates the number of node expansions observed experimentally, by about a factor of four. The reason is that our model of the heuristic distribution is conservative. It assumes that the heuristic values are as small as possible, subject to the branching factor. The actual distribution will tend to have larger heuristic values, resulting in more pruning. While this same effect exists with a single pattern database, the effect is much larger with multiple pattern databases, as we will explain below.

Next, when comparing the experimental data to $E(N, 12, P^k, k)$ in the third column, the prediction based on the actual number of nodes in the tree and the actual heuristic distribution, we find that the model underestimates the actual node expansions, by $179,824$ to $225,960$. This error is larger than that due to the limited number of problem instances, and represents a small but significant deviation. The reason for this is that the heuristic distribution for the maximum of two or more pattern database values is based on the assumption that the individual heuristic values are independent of one another. This is not the case here, since each pair of six edge-cubie databases share two edge cubies. If the values are positively correlated, then a small value from one database is less likely to be compensated for by a large value from the others, resulting in less pruning and more node expansions than predicted by the model.

The final thing to notice is that while our formula $E(b, 12, s, k)$ predicts a large improvement in going from one pattern database to two, it predicts relatively little further improvement for additional pattern databases. In fact, if we assume that the size of the pattern databases decrease with multiple databases, our formula for $E(b, 12, s/k, k)$ predicts that the maximum of three databases will result in more node expansions than the maximum of two databases. Our experimental data, however, shows further improvement as the number of pattern databases increases.

Discussion

One possible reason for this is as follows. Recall that the cumulative heuristic distribution function is a function from a heuristic value to a probability, which is the probability that the heuristic value of a random state is less than or equal to each particular heuristic. Thus, the range of the cumulative distribution function is from zero to one, and it is monotonically non-decreasing, reaching the value one only at the maximum heuristic value. Given two heuristic functions that are independent of each other, then the cumulative distribution of the maximum of the two heuristics is the product of the individual cumulative distribution functions. If the two original heuristics are identically distributed, then the cumulative distribution function of their maximum is the square of the original distribution function. Since the range of the cumulative distribution function is between zero and one, squaring such a function reduces its values everywhere, except where it equals one at its maximum heuristic value. The visual effect on the function is to delay its rise, and make its eventual rise to one steeper. If we continue taking higher powers, the cumulative distribution function approaches a step function at its maximum value.

Another view of this is that if we take the maximum value of multiple independent heuristic functions, the effect is to concentrate its probability mass on its maximum value. In other words, the maximum of multiple independent heuristics eventually converges to a constant function that always returns its maximum value. The rate at which this convergence occurs depends on the variance of the original function. The lower the variance, the faster the convergence.

In our theoretical model of a pattern database heuristic, the number of states with a given heuristic value grows exponentially with the heuristic value, until reaching its maximum value. Since Rubik's Cube has a relatively high branching factor, the exponential growth is quite rapid, and most of the heuristic values equal their maximum value. Furthermore, taking the maximum of several such heuristics very quickly concentrates almost all the probability mass on its maximum value. This is both predicted by our analytical formula and demonstrated by the values in the second column of Table 3, which show relatively little further improvement with the maximum of more than two such heuristics.

Because of duplicate nodes, however, a real pattern database of the same size distributes the same number of entries over a larger number of values, by shifting entries to larger values, creating a heuristic distribution with more variance and a larger maximum value. As a result, the max-

imum of several of these heuristics converges more slowly to the maximum value, and the maximum of more than two such heuristics continues to reduce the number of node expansions. Furthermore, the larger maximum value predicts that the number of node expansions will be significantly less than for the theoretical model of the heuristic distribution. Both of these effects are seen in Table 3.

Conclusions and Further Work

Conclusions

We have provided an analytic model to predict the number of node expansions by IDA*, using a pattern database heuristic for a single goal state. It is based on the brute-force branching factor b of the problem space, the search depth d, and the size s of the pattern database. Our model assumes that the backward branching factor of the problem space is uniform and the same as the forward branching factor. Unlike (Korf, Reid, & Edelkamp 2001), our model requires only the size of the database, and not its heuristic distribution, which requires generating the entire pattern database. In the case of a single pattern database, our model predicts that the number of node expansions will be a fraction of

$$\frac{\log_b s + 1}{s}$$

of the nodes expanded in a brute-force search. This prediction is supported by empirical results on Rubik's Cube, and in fact is conservative, overestimating the actual number of node expansions somewhat.

In the case of the maximum of two or more pattern database heuristics of the same size s, our model predicts that the number of node expansions will decrease linearly with s, assuming that the individual heuristics are independent of one another. If we assume that the sum of the sizes of the pattern databases is constant, then our theory predicts that the number of node expansions will decrease linearly with s/k, where k is the number of pattern databases. When compared to experimental results, our theoretical prediction for multiple pattern databases is too conservative, and significantly overestimates the actual number of node expansions in experiments with Rubik's Cube. One possible reason for this is that the maximum of multiple independent and identically distributed heuristics eventually converges to a constant which is the maximum heuristic value, and our theoretic model predicts a smaller maximum value than actually occurs in a real pattern database of the same size.

Further Work

Our current work is focussed in several directions. One is experimenting with other problem domains, such as the sliding-tile puzzles. Another is attempting to reconcile the difference between our theoretical predictions and our experimental results for the maximum of several different pattern databases. Finally, additive pattern database heuristics (Korf & Felner 2002; Felner, Korf, & Hanan 2004) are more effective for problems such as the sliding-tile puzzles and the Towers of Hanoi. In this case, rather than simply taking the maximum of different pattern database values, under certain conditions their values can be added together, resulting in a much more accurate heuristic, without sacrificing admissibility. We are in the process of extending our analysis to additive pattern databases as well.

References

Culberson, J., and Schaeffer, J. 1998. Pattern databases. *Computational Intelligence* 14(3):318–334.

Edelkamp, S. 2001. Planning with pattern databases. In *Proceedings of the European Conference on Planning (ECP)*, 13–25.

Felner, A.; Korf, R.; and Hanan, S. 2004. Additive pattern database heuristics. *Journal of Artificial Intelligence Research* 22:279–318.

Hart, P.; Nilsson, N.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* SSC-4(2):100–107.

Hohwald, H.; Thayer, I.; and Korf, R. 2003. Comparing best-first search and dynamic programming for optimal multiple sequence alignment. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, 1239–1245.

Holte, R., and Hernadvolgyi, I. 1999. A space-time tradeoff for memory-based heuristics. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99)*, 704–709.

Holte, R.; Newton, J.; Felner, A.; Meshulam, R.; and Furcy, D. 2004. Multiple pattern databases. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-2004)*, 122–131.

Holte, R.; Newton, J.; Felner, A.; Meshulam, R.; and Furcy, D. 2006. Maximimizing over multiple pattern databases speeds up heuristic search. *Artificial Intelligence* 170(16-17):1123–1136.

Korf, R., and Felner, A. 2002. Disjoint pattern database heuristics. *Artificial Intelligence* 134(1-2):9–22.

Korf, R., and Felner, A. 2007. Recent progress in heuristic search: A case study of the four-peg towers of hanoi problem. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2334–2329.

Korf, R., and Reid, M. 1998. Complexity analysis of admissible heuristic search. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, 299–304.

Korf, R.; Reid, M.; and Edelkamp, S. 2001. Time complexity of Iterative-Deepening-A*. *Artificial Intelligence* 129(1-2):199–218.

Korf, R. 1985. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence* 27(1):97–109.

Korf, R. 1997. Finding optimal solutions to Rubik's cube using pattern databases. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)*, 700–705.