Recent Results from Analyzing the Performance of Heuristic Search

Teresa M. Breyer and **Richard E. Korf**
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095
{tbreyer,korf}@cs.ucla.edu

Abstract

This paper extends existing analyses of the performance of heuristic search in several directions. First we show experimentally that, with minor modifications, an existing analysis of IDA* also applies to A*. Furthermore, we apply a related model to predict the performance of IDA*, using only the branching factor of the problem, the search depth, and the size of a pattern database, to the 15 puzzle. Finally, we extend this existing model to additive disjoint pattern databases. The reduction in the number of nodes expanded for IDA* using multiple additive pattern databases is the product of the reductions achieved by the individual databases. We experimentally verify this result using the 4-peg Towers of Hanoi problem. This is the first analysis of the performance of disjoint additive pattern databases.

Introduction

All heuristic search algorithms, including A* (Hart, Nilsson, & Raphael 1968), IDA* (Korf 1985), and Breadth-First Heuristic Search (BFHS) (Zhou & Hansen 2006), require a heuristic evaluation function. More accurate heuristic functions lead to fewer node expansions. For many problems, a heuristic evaluation function can be stored in a lookup table called a pattern database (PDB) (Culberson & Schaeffer 1998). An inverse relationship between the size of the PDB and the number of nodes expanded is well known.

For the 15 puzzle for example, the minimum number of moves needed to get a set of tiles, called the pattern tiles, to their goal positions is a lower bound on the total number of moves required to solve the puzzle, and thus an admissible heuristic function. For each possible configuration of pattern tiles and the blank, this value is stored in the PDB. If we only count moves of the pattern tiles when constructing the PDBs, we can use several disjoint groups of pattern tiles and sum the values from each of these individual PDBs to get an admissible heuristic function (Korf & Felner 2002). Such a set of PDBs is called a set of disjoint PDBs. To save memory, instead of storing one heuristic value for each position of the blank, it is common practice to only store the minimum over all positions of the blank for each configuration of the pattern tiles, i.e., the additive disjoint PDBs are in practice usually compressed by the position of the blank, making them inconsistent (Zahavi *et al.* 2007).

Similarly, for the Towers of Hanoi problem we store the number of moves needed to solve a problem consisting of only a subset of the discs in a PDB. Partitioning the discs into disjoint sets creates additive PDBs.

Overview

The first part of this paper analyzes the performance of IDA* and A*. It uses the number of nodes at each depth in the brute-force search tree or graph, the heuristic distribution, and the search depth to predict the performance of these algorithms. These analyses apply to heuristics computed at runtime as well as PDBs. In the case of PDBs, acquiring the heuristic distribution requires the complete PDB to be constructed. First, we experimentally test Korf, Reid, & Edelkamp's (2001) analysis of A* using the 15 puzzle. We use the actual number of nodes at each depth in the search graph, recently calculated by Korf & Schultze (2005), and the overall heuristic distribution. Second we show how to do better than using the overall heuristic distribution. Then we compare two different methods for determining the heuristic distribution of disjoint additive PDBs, sampling and convolution. This leads to the notion of *independent* PDBs.

The remainder of the paper applies only to IDA* with PDBs. The only variables used here are the size of the individual PDBs, the branching factor, and the search depth. We approximate the actual heuristic distribution without explicitly constructing the PDBs. We look at how well Korf's (2007) model applies to the 15 puzzle and explain in detail how this problem space differs from the model used. This model shows that the number of nodes expanded by IDA* using a PDB of size s is a fraction $\frac{1+\log_b s}{s}$ of the nodes expanded by a brute-force depth-first search, where b is the branching factor. Finally, we extend this theoretical model for a single PDB to disjoint and independent additive PDBs. In particular, we show that the number of nodes expanded by IDA* using the sum of two additive disjoint PDBs of size s_1 and s_2 is a fraction $\frac{1+\log_b s_1}{s_1} \cdot \frac{1+\log_b s_2}{s_2}$ of the nodes expanded by a brute-force depth-first search. This fraction is the product of the fractions of node expansions when using one database of size s_1 or s_2 as shown by Korf (2007). Furthermore, we relax the assumptions of the model and ex-

Depth	Theoretical	Problems	Experimental	Error
40	11,866	100,000	11,529	2.919%
41	22,605	100,000	22,340	1.188%
42	42,777	100,000	41,538	2.983%
43	80,392	100,000	79,335	1.332%
44	150,010	100,000	145,661	2.986%
45	277,875	100,000	273,903	1.450%
46	510,850	100,000	496,246	2.943%
47	931,867	100,000	917,729	1.540%
48	1,686,253	14,568	1,688,045	0.106%
49	3,026,192	6,137	2,927,342	3.377%
50	5,384,745	865	5,139,035	4.781%

Table 1: Nodes expanded by A* on 15 puzzle using Manhattan Distance

Depth	Theoretical	Experimental	Error
40	146	136	7.120%
41	282	273	3.473%
42	544	502	8.447%
43	1,043	994	4.908%
44	1,985	1,822	8.962%
45	3,756	3,551	5.784%
46	7,064	6,469	9.191%
47	13,199	12,421	6.263%
48	24,506	22,450	9.159%
49	45,204	42,448	6.493%
50	82,833	76,037	8.937%

Table 2: Nodes expanded by A* on 15 puzzle using 7-8 tile pattern database

perimentally verify a slightly more general result using the 4-peg Towers of Hanoi problem.

Analyses using Actual Heuristic Distribution

Time Complexity of IDA*

Korf, Reid, & Edelkamp (2001) analyzed the time complexity of IDA*. The number of nodes expanded by IDA* for a search to depth d is approximately

$$E(N, d, P) = \sum_{i=0}^{d} N_i P(d - i) \qquad (1)$$

where N_i is the number of nodes at depth i in the brute-force search tree, and P is the equilibrium distribution of the heuristic function. $P(x)$ is the probability that a node at a given depth chosen randomly and uniformly among all nodes at that depth has a heuristic estimate less than or equal to x, in the limit of large depth. The heuristic function is assumed to be admissible and consistent. This model very accurately predicts the actual number of nodes expanded by IDA*. For inconsistent heuristic functions, $E(N, d, P)$ only provides an upper bound (Zahavi *et al.* 2007).

Time Complexity of A*

Korf, Reid, & Edelkamp (2001) suggested that since A* expands each node exactly once, $E(N, d, P)$ also predicts the number of nodes expanded by A* when replacing the equilibrium distribution P by the overall heuristic distribution $\tilde{P}$. $\tilde{P}(x)$ is the probability that a randomly and uniformly chosen state has a heuristic value less than or equal to x. Furthermore, the number of nodes at depth i in the brute-force search tree N_i is replaced by the number of nodes $\tilde{N}_i$ in the problem space graph.

Experimental Results

For our first experiment we ran BFHS on the 15 puzzle with Manhattan distance. BFHS expands the same set of nodes as A*, but uses less memory. In all of our experiments we ignored the goal state and continued the search to the specified depth and our experimental results are averaged over a large number of random initial states, because all analyses in this paper hold on average. Usually, one wants to solve a set of problems, so the average performance is relevant. These analyses can also be used to choose the best from a set of heuristics. In equation (1) we substituted the exact $\tilde{N}_i$, which Korf & Schultze (2005) recently calculated for the 15 puzzle. For $\tilde{P}(x)$, we randomly sampled 10 billion states.

The experimental results can be seen in Table 1. The first column gives the search depth. The second column gives the node expansions predicted by the theoretical analysis. The third column gives the number of problem instances run. We randomly created 100,000 initial states with the blank in a corner, because the start state in Korf & Schultze's (2005) Breadth-First Search (BFS) had the blank in a corner as well, and the $\tilde{N}_i$ are accurate for all start states with the blank in a corner. The fourth column gives the average number of nodes expanded over those initial states. For depths 48 to 50 we ran out of main memory running 100,000 problems despite using BFHS instead of A*, so we stopped at the first problem that exceeded our main memory. The fifth column gives the relative error of our analysis. We also used a state of the art heuristic for the 15 puzzle, the sum of a 7 and an 8-tile disjoint additive PDB, which is inconsistent. Here BFHS is more efficient than A*, because A* can reopen closed nodes. Again we obtain $\tilde{P}$ by sampling over 10 billion states. Similar to Table 1, Table 2 shows the results over a set of 100,000 random initial states. The error is slightly larger than in Table 1, because $E(N, d, P)$ only gives an upper bound for inconsistent heuristics.

Both experiments show that the predicted number of node expansions of A* is very accurate. One can notice a systematic disparity between the errors at even and odd depths. A similar, but slightly smaller effect can be noticed in Korf, Reid, & Edelkamp's (2001) experiments on the 15 puzzle with Manhattan distance and IDA*. There is a simple explanation for this disparity. Manhattan distance and the 7-8 tile disjoint PDBs both use all tiles. In contrast to non-additive PDBs for example, where the heuristic estimate h would not change if a non pattern tile is moved, for Manhattan Distance and disjoint additive PDBs that use all tiles, each move either increases or decreases h by exactly one move. Since each move increases the search depth g by one, the f cost

Depth	Theoretical	Experimental	Error
40	11,669	11,529	1.218%
41	22,239	22,340	0.452%
42	42,095	41,538	1.341%
43	79,135	79,335	0.252%
44	147,705	145,661	1.403%
45	273,692	273,903	0.077%

Table 3: Nodes expanded by A* on 15 puzzle using Manhattan Distance, considering the position of the blank

either increases by two or remains the same. Consequently states with even solution depths always have even f cost, i.e. the sum of the search depth g and the heuristic estimate h will always be even. In our experiments states with the blank in the upper left and lower right corner have even solution depths, while those with the blank in the upper right and lower left corner have odd solution depths. For an initial state with even solution depth a search to even depth d expands the same number of nodes as a search to odd depth $d + 1$. Consequently, at even depths only the number of nodes expanded for initial states with even solution depths increases, and at odd depths the number of nodes expanded for initial states with odd solution depths increases. We get two sequences of numbers of node expansions and relative errors, one is based on states with even solution depths and one based on states with odd solution depths. If the relative error of the two sequences differs even by a miniscule amount, the weighted sum over these two sequences exhibits this disparity between the errors at even and odd depths.

This is the first time that a performance analysis of A* was verified experimentally. Holte & Hernádvölgyi (2004) use equation (1) to choose the best among a set of heuristic functions. They approximate N_i by $\tilde{b}^i$ where $\tilde{b}$ is the arithmetic average of the actual branching factor of all nodes expanded during the search. But this approximation did not generate accurate predictions of node expansions.

Refinement

To further refine the previous estimates, we make a new observation: At shallow levels in the 15 puzzle graph, states with the blank in the middle are more frequent than states with the blank in the corner because they have more neighboring states with operators leading to them. The overall fractions of states with the blank in the corner and states with the blank in the middle are equal however. Thus at deep depths, states with the blank in the corner have to become relatively more frequent than states with the blank in the middle. Korf, Reid, & Edelkamp (2001) showed that states with the blank in the middle, side, and center have different overall heuristic distributions. Consequently, the fractions of each of these three classes of states determine the heuristic distribution at each depth. We will see a similar effect in any search space that consists of different classes of states with different branching factors.

Depth	Sampling	Convolution	Experi-mental	Error
40	470	109	444	75.5%
41	1,001	233	986	76.4%
42	2,134	496	2,040	75.7%
43	4,547	1,057	4,487	76.4%
44	9,688	2,253	9,342	75.9%
45	20,641	4,802	20,397	76.5%
46	43,974	10,231	42,659	76.0%
47	93,684	21,797	92,579	76.5%
48	199,585	46,438	194,331	76.1%
49	425,198	98,932	419,795	76.4%
50	905,843	210,767	883,854	76.2%

Table 4: Nodes expanded by IDA* on 15 puzzle using the 7-8 tile pattern database (sampling vs. convolution)

Experimental Results

We reran Korf & Schultze (2005) BFS up to depth 45, keeping track of the number of states with the blank in a corner, side, and middle position and repeated our experiments on the 15 puzzle using BFHS and Manhattan distance using the same 100,000 random initial states. We used three different $\tilde{P}$, one for each type of state, calculated by sampling over 10 billion states in total. The small relative error when using the overall heuristic distribution is reduced by more than half by using this more accurate set of heuristic distributions. The results are given in Table 3, which is similar to Table 1, except for a smaller relative error. Again one can notice a systematic disparity between the errors at even and odd depths, which we explained above.

Sampling the Heuristic vs. Convolution

In the previous section we used the heuristic distribution function to analyze the performance of the search algorithm. Furthermore, sampling was used to obtain the heuristic distribution. When using a PDB, the exact heuristic distribution can be read directly from the database if each state in the PDB has the same number of pre-images in the original search space (Holte & Hernádvölgyi 2004). For additive PDBs however, we need the distribution of the sum of the individual values. The probability distribution of the sum of two independent random variables is the convolution of their distributions. The heuristic distributions of the disjoint additive PDBs for the sliding-tile puzzles are correlated however. In the case of the 7 and 8 tile disjoint additive PDBs for the 15 puzzle, the positions occupied by the 7 pattern tiles cannot be occupied by any of the 8 pattern tiles. The convolution however assumes that every state of the 7 tile pattern space (which is a projection of the original space) can be combined with every state of the 8 tile pattern space, to create a state in the original search space.

Experimental results for IDA* on the 15 puzzle with the 7-8 tile disjoint additive PDBs are shown in Table 4. The first column gives the search depth. The second column gives the node expansions predicted by equation (1) using sampling to determine the heuristic distribution. We randomly sampled 10 billion states and differentiated by the position of the blank. The third column gives the node expansions predicted when calculating the heuristic distribution using convolution instead of sampling. Again we differentiated by the position of the blank. The fourth column gives the average number of nodes expanded over 100,000 random initial states. The last column gives the relative error of the analysis using convolution compared to the experimental results in the fourth column.

In Table 4 the analysis using sampling overestimates the number of nodes expanded slightly. As mentioned earlier the 7-8 tile disjoint additive PDBs are inconsistent and for inconsistent heuristic functions equation (1) only gives an upper bound on node expansions. Mainly the table shows that the estimate using convolution underestimates the number of node expansions by approximately 75%, even though it is supposed to be an upper bound. This is because the convolution underestimates the probabilities of small heuristic values. A very small error in the probabilities of small heuristic values produces a large error in the predicted node expansions, because these probabilities are multiplied by the largest N_i. The probabilities of very large heuristic values are underestimated as well, but they only matter in low search depths, so they do not introduce significant error.

This observation motivates the following formal definition: A set of additive disjoint PDBs is *independent* if and only if the random variables they represent, or equivalently their heuristic distribution functions, are independent.

In other words, knowing the additive heuristic value of a state from one PDB does not change the probability of the heuristic estimates from the remaining disjoint PDBs. Even though the 7 and 8 tile PDBs are disjoint, i.e., each operator only modifies the values from one PDB, they are not independent. Examples of independent disjoint PDBs are the disjoint additive PDBs for the Towers of Hanoi problem. Once we know which peg a disc is located on, we also know its position on the peg, because discs are always ordered by size. A state description consists only of the peg that each disc is on. Each disc can be assigned to any peg, independent of the locations of all other discs. When we analyze the performance of disjoint additive PDBs later in this paper we will take advantage of this property of independence of the PDBs for the Towers of Hanoi problem.

Analysis Using Size of a Pattern Database

In the remainder of this paper we analyze the performance of IDA* using PDBs. The only variables used here are the size of the individual PDBs, the branching factor, and the search depth. In the previous section the distribution of the heuristic function was used. In case of PDBs this requires the entire PDB to be constructed. Even sampling random states to determine the heuristic distribution requires the complete PDB. In this section we use the size of the PDBs and the branching factor to approximate the heuristic distribution, which does not require constructing the complete PDB.

IDA* with a Single Pattern Database

Korf (2007) introduced a model for analyzing the performance of IDA* with a single PDB that builds on Korf, Reid, & Edelkamp's (2001) analysis. N_i, the number of nodes at depth i in the brute-force search tree, is approximated by b^i. Instead of the exact heuristic distribution P, the brute-force branching factor b and the size s of the PDB are used to approximate it. The PDB is constructed through a backward breadth-first search from the goal state. The forward and backward branching factors of the problem space are assumed to be equal and the graph is assumed to have a negligible number of cycles. In particular, it assumes that there is one node with heuristic value 0, b nodes with heuristic value 1, up to b^m nodes with maximum heuristic value m, such that $\sum_{i=0}^{m} b^i \geq s$. In other words this model only depends on b, s and the search depth d. The heuristic is also assumed to be consistent. The number of nodes expanded by IDA* for a search to depth d is approximately

$$E(b,d,s) \approx \frac{b^{d+1}}{b-1} \cdot \frac{\log_b s + 1}{s} \qquad (2)$$

This formula consists of the number of nodes expanded by a brute-force search to depth d, multiplied by a reduction fraction due to the heuristic.

Experimental Results

Korf (2007) showed experimentally that this analysis is accurate for Rubik's cube. Here we look at another problem space, the 15 puzzle.

In the 15 puzzle, b^d underestimates the number of nodes in the brute-force search tree at depth d for all three possible start positions of the blank. This is partly because during search we do not allow the reverse of the last move, which reduces the branching factor of each node, except for the initial state, by one. At shallow depths the asymptotic branching factor underestimates the actual branching factor, but eventually the asymptotic branching factor becomes very accurate. Consequently b^i underestimates the number of nodes at depth i in the brute-force search tree by a factor of c_i. If all states have the same branching factor, the branching factor converges to the asymptotic branching factor in just a few depths, such as in Rubik's cube. In the 15 puzzle convergence takes a bit longer. The branching factor converges to the asymptotic branching factor as the fractions of the different types of states converge to their equilibrium fractions. As the branching factor converges to the asymptotic branching factor b, c_i converges to c. Therefore, in our analysis we use a correction factor c which is numerically fitted so that cb^d accurately approximates the number of states at large depths d in the brute-force search tree. To be specific we calculated the weighted average of three different values for c, one for each type of state depending on the position of the blank, and get $c = 1.499$. Using one constant correction factor initially overestimates the number of nodes at very shallow depths, even though it is accurate for deep depths, but we mainly care about the number of nodes at deep depths for our analysis. Korf (2007) used this correction factor in his experiments on Rubik's cube as well.

In the first experiment we use a non-additive 7 tile PDB, meaning that moves of the 8 non-pattern tiles are counted in the database values. The numbers of node expansions predicted by Korf's (2007) model are given in Table 5. The first

Depth	$c \cdot E(b,d,s)$	$E(N,d,P)$	$c \cdot E(b,B,d,s)$
40	2,061,736	866,152	880,866
41	4,392,323	1,845,251	1,876,599
42	9,357,405	3,931,124	3,997,910
43	19,935,016	8,374,864	8,517,151
44	42,469,558	17,841,820	18,144,940
45	90,477,146	38,976,963	38,655,990
46	192,752,512	80,976,963	82,352,720
47	410,639,951	172,513,352	175,444,200
48	874,827,352	367,522,400	373,766,400
49	1,863,732,191	782,969,525	796,271,900
50	3,970,495,060	1,668,038,500	1,696,378,000

Table 5: Predicted node expansions by IDA* on 15 puzzle using non-additive 7-tile pattern database

Depth	$c \cdot E(b,d,s)$	$E(N,d,P)$	$c \cdot E(b,B,d,s)$
40	24,880K	144,485K	144,477K
41	53,004K	307,810K	307,793K
42	112,920K	655,759K	655,722K
43	240,565K	1,397,030K	1,396,951K
44	512,501K	2,976,232K	2,976,064K
45	1,091,831K	6,340,565K	6,340,207K
46	2,326,037K	13,507,940K	13,507,180K
47	4,955,389K	28,777,315K	28,775,690K
48	10,556,962K	61,307,190K	61,303,730K
49	22,490,552K	130,608,843K	130,601,500K
50	47,913,871K	278,249,050K	278,233,400K

Table 6: Predicted node expansions in thousands (K) by IDA* on 15 puzzle using additive 7-tile pattern database compressed by the position of the blank

column gives the search depth d. The second column gives the predicted number of node expansions $E(b,d,s)$ to depth d based on the size s of the PDB and the branching factor b multiplied by a correction factor $c = 1.499$ as explained above. The next column gives $E(N,d,P)$ the predicted number of node expansions using N_i and P. $E(N,d,P)$ is within a few percent of the experimental results. Ignore the last column for now. Clearly the model significantly overestimates the number of nodes expanded.

The underlying model assumes that the branching factors in the search tree and in the pattern space are equal, and that there are few cycles. But the pattern space graph of the 15 puzzle has fewer distinguishable nodes than the original search space, because 8! states of the original search space are projected onto the same pattern state. Nevertheless the operators are the same in both spaces. So we encounter a lot of duplicate states during our breadth-first search backwards from the goal state. Thus we cannot use the brute-force branching factor b of the original search tree to approximate the breadth-first search used to build the PDB. Instead we have to use the average branching factor B of the pattern space graph, which is smaller than b. There are B^i nodes with heuristic value i, not b^i. Consequently, Korf's (2007) model overestimates the number of nodes with low heuristic values, which also results in an overestimate of nodes expanded. This is reflected in Table 5.

For the next experiment we look at the 7 tile additive PDB. Only moves of pattern tiles are counted and we compress the PDB by the position of the blank.

Similarly to Table 5, Table 6 gives predicted numbers of node expansions. One can see that the model underestimates the number of nodes expanded by a significant amount.

For the 7 tile additive PDB one move in the pattern space is equivalent to a series of moves of non-pattern tiles followed by one move of a pattern tile in the original search space. Consequently, in the pattern space there are many more possible moves at a given state and the brute-force branching factor of the tree expansion of the pattern space graph becomes much larger than in the original space. On the other hand 9! states in the original space are projected onto one state in the pattern space, 8! because of the non-pattern tiles and another 9 because we compress by the position of the blank. It is not obvious whether the average branching factor B when searching the pattern space graph in a breadth-first search will be larger or smaller than the brute-force branching factor b of the original search space. The second column of Table 6 shows that the model underestimates the number of nodes expanded as well as the number of nodes with small heuristic values, which implies that despite the many duplicate nodes B is larger than b.

IDA* with a Single Pattern Database Revisited

The experiments above showed that we have to incorporate the effective branching factor B of the pattern space graph in the model. We assume B^i nodes with heuristic value i for $i \le M$, where M is the maximum heuristic value, and b^i nodes at depth i in the brute-force search tree. The probability that a random state has heuristic value x becomes $P(x) = \sum_{i=0}^{x} \frac{B^i}{s}$. We calculate the maximum heuristic value M the same way Korf (2007) did:

$$\sum_{i=0}^{M} \frac{B^i}{s} = 1 = \frac{B^{M+1}}{(B-1)s} \Rightarrow M = \log_B \frac{s(B-1)}{B}$$

The number of nodes expanded in a search to depth d can be calculated as

$$\begin{aligned} E(b,B,d,s) &= \sum_{i=0}^{d-M} b^i + \sum_{i=d-M+1}^{d} b^i P(d-i) \\ &= \frac{b^{d-M+1}}{b-1} + \sum_{i=d-M+1}^{d} b^i \frac{B^{d-i+1}}{(B-1)s} \end{aligned}$$

Determining the Effective Branching Factor

To validate this new estimate and to demonstrate that we can model the distribution of the heuristic function using a single exponential, we have to determine the effective branching factor B. The number of nodes expanded by IDA* in a search to depth d is estimated by $E(b,B,d,s)$ and multiplied by a correction factor $c = 1.499$ as explained above. Except for the effective branching factor of the pattern space graph B, all variables required are known, but there is no obvious way to determine B without constructing the complete

PDB. We calculate B from the actual distribution P as follows. We use a fixed depth d_0, N_i, and P to estimate the number of nodes expanded to depth d_0, $E(N, d_0, P)$. Setting $c \cdot E(b, B, d_0, s)$ equal to $E(N, d_0, P)$ leads to an equation with one unknown, B, and we obtain B by numerically solving the equation. Alternatively we could have run experiments to a small depth d_0 and used that number of node expansions instead of $E(N, d_0, P)$, or we could have used the arithmetic average of all the nodes expanded in the breadth-first search to construct the PDB, similar to how Holte & Hernádvölgyi (2004) approximated the average branching factor for A*. All three approaches require the complete PDB to be constructed though.

The goal here is to verify the model, i.e. to show that the heuristic distribution can be approximated using a single exponential, and to determine what minimum information is required to estimate the number of nodes expanded. Only the distribution for small heuristic values is important, because large heuristic values only matter at low search depths and only cause a small error in the number of node expansions. Conversely a small error in the estimation of small heuristic values can create a very large error in the number of node expansions. Future research focuses on calculating B cheaply, perhaps by partially constructing the PDB.

Experimental Results

We apply the above described technique to our two examples from above. We first ran experiments on the 15 puzzle with the non-additive 7 tile PDB and IDA*. We obtained $B = 1.9632$ numerically for depth $d_0 = 40$ as described above. B is smaller than $b = 2.1304$ as explained previously. The last column of Table 5 gives the estimated node expansions $E(b, B, d, s)$ multiplied by the correction factor $c = 1.499$. The predicted node expansions is within less than 1% error of $E(N, d, P)$, which is known to be accurate, for all depths.

Similarly we determined the effective branching factor of the compressed additive 7 tile PDB as $B = 2.5758$ for depth $d_0 = 40$. The last column of Table 6 gives the estimated node expansions $E(b, B, d, s)$ multiplied by the correction factor $c = 1.499$. It is within less than 1% of error of $E(N, d, P)$ for all depths. Despite the many cycles in the pattern space, the effective branching factor of the pattern space is larger than the branching factor of the original search space $b = 2.1304$ as expected.

Both examples show that the heuristic distribution can be approximated with a single exponential for low values of h.

IDA* with Additive Disjoint Pattern Databases

Here we extend Korf's (2007) theoretical analysis for IDA* to disjoint additive PDBs, i.e., we look at the sum of two independent additive PDBs of size s_1 and s_2 respectively. We assume $s_1 \leq s_2$. The forward and backward branching factors of the problem space are assumed to be b. $Pr[X_j = x]$ where $j = \{1, 2\}$, the probability that a random state has a heuristic estimate of x equals $\frac{b^x}{s_j}$. The heuristic distribution function of the individual PDBs are $Pr[X_j \leq x] = \sum_{i=0}^{x} \frac{b^i}{s_j}$ for $x \leq m_j$, where m_j is the maximum heuristic value of PDB j. We calculate m_j as follows:

$$\sum_{i=0}^{m_j} \frac{b^i}{s_j} = 1 = \frac{b^{m_j+1}}{(b-1)s_j} \Rightarrow m_j = \log_b \frac{s_j(b-1)}{b}$$

Cumulative Heuristic Distribution

We assume independence of the individual PDBs. Consequently the distribution of the sum of the two disjoint additive PDBs is the convolution of the distributions of the individual PDBs. A heuristic estimate x has two components, one from each PDB, x_1 and x_2. Equivalently the random variable X consists of the sum of X_1 and X_2. x_j always has to be less than or equal to m_j. Thus we have to look at different ranges for x. First we look at $P(x)$ for $x \leq m_1$:

$$\begin{aligned} P(x) &= Pr[X \leq x] = Pr[X_1 + X_2 \leq x] \\ &= \sum_{i=0}^{x} Pr[X_1 = i] (\sum_{j=0}^{x-i} Pr[X_2 = j]) \end{aligned}$$

For $m_1 < x \leq m_2$ we get:

$$P(x) = \sum_{i=0}^{m_1} Pr[X_1 = i] (\sum_{j=0}^{x-i} Pr[X_2 = j])$$

Finally we look at $P(x)$ for $m_1 \leq m_2 < x \leq m_1 + m_2$:

$$\begin{aligned} P(x) &= \sum_{i=0}^{x-m_2} Pr[X_1 = i] (\sum_{j=0}^{m_2} Pr[X_2 = j]) \\ &+ \sum_{i=x-m_2+1}^{m_1} Pr[X_1 = i] (\sum_{j=0}^{x-i} Pr[X_2 = j]) \end{aligned}$$

Substituting $Pr[X_j = i] = \frac{b^i}{s_j}$ and using simple algebraic transformations, the cumulative heuristic distribution function of the sum of two independent additive PDBs can be derived as:

$$P(x) = \begin{cases} \frac{b^{x+1}(x+1)}{(b-1)s_1 s_2} & \text{if } x \leq m_1 \leq m_2 \\ \frac{b^{x+1}(m_1+1)}{(b-1)s_1 s_2} & \text{if } m_1 < x \leq m_2 \\ \frac{b^{x-m_2+1}}{(b-1)s_1} + \frac{b^{x+1}(m_1+m_2-x)}{(b-1)s_1 s_2} & \text{if } m_1 \leq m_2 < x \\ & \text{and } x \leq m_1 + m_2 \\ 1 & \text{if } x > m_1 + m_2 \end{cases}$$

Number of Nodes Expanded

The number of nodes expanded by IDA* for a search to depth d using the sum of two additive PDBs of size s_1 and s_2 respectively can be predicted as:

$$\begin{aligned} E(b, d, s_1, s_2) &= \\ \sum_{i=0}^{d-(m_1+m_2)} N_i &+ \sum_{i=d-(m_1+m_2)+1}^{d-m_2} N_i P(d-i) + \\ \sum_{i=d-m_2+1}^{d-m_1} N_i & P(d-i) + \sum_{i=d-m_1+1}^{d} N_i P(d-i) \end{aligned}$$

	8 disc PDB			disjoint 8-8 disc PDB			Conjecture 8-8 disc PDB
Depth	$E(N,d,P)$	Experimental	Error	$E(N,d,P)$	Experimental	Error	$k_1k_2 \cdot \sum N_i$
16	142,350	139,850	1.79%	6	2	189.97%	6
17	536,154	526,089	1.91%	22	7	212.05%	22
18	2,019,383	1,979,783	2.00%	82	31	165.40%	82
19	7,605,845	6,797,384	11.89%	310	128	142.09%	310
20	28,646,710	25,763,875	11.19%	1,167	532	119.38%	1,167
21	107,895,100	97,582,631	10.57%	4,396	2,178	101.83%	4,396
22	406,376,300	369,362,820	10.02%	16,557	8,840	87.29%	16,557
23	1,530,576,000			62,359	35,579	75.27%	62,359
24	5,764,762,000			234,869	142,191	65.18%	234,869
25	21,712,400,000			884,612	564,685	56.66%	884,612
26	81,777,590,000			3,331,804	2,230,023	49.41%	3,331,804
27	308,007,100,000			12,548,910	8,762,686	43.21%	12,548,910

Table 7: Nodes expanded on 16 disc Towers Hanoi using IDA* with a single 8 disc and a disjoint 8-8 disc pattern database

b	2	3	4	5	10	15
$f_0(b)$	2	1.464	1.337	1.270	1.140	1.096
$f_1(b)$	1	1.131	1.126	1.111	1.065	1.046
$f_2(b)$	0	0.631	0.792	0.861	0.954	0.975

Table 8: Values of $f_0(b)$, $f_1(b)$, and $f_2(b)$

So far the formula also holds for dependent additive PDBs. From here on we require independence though. Plugging in the distributions and m_j from above, b^i for N_i, Pascal's second identity $\sum_{k=0}^{m} \binom{a+k}{k} = \binom{a+1+m}{m}$ and simple algebraic transformations yield:

$$\begin{aligned} E(b,d,s_1,s_2) &= \frac{b^{d+1}}{(b-1)s_1 s_2} \cdot \\ & \Bigg[\frac{b^2}{(b-1)^2} + \log_b \frac{b-1}{b}(\log_b \frac{b-1}{b})^2 \\ + & \frac{b}{b-1}\log_b \frac{b-1}{b} + \log_b s_1(\frac{b}{b-1} + \log_b \frac{b-1}{b}) \\ + & \log_b s_2(\log_b \frac{b-1}{b} + 1) + \log_b s_1 \log_b s_2\Bigg] \end{aligned}$$

Using $f_0(b)$, $f_1(b)$, and $f_2(b)$ as a shortcut for the coefficients of 1, $\log_b s_1$, and $\log_b s_2$ we get:

$$\begin{aligned} E(b,d,s_1,s_2) &= \frac{b^{d+1}}{(b-1)s_1 s_2} \cdot \\ & (f_0(b) + f_1(b)\log_b s_1 + f_2(b)\log_b s_2 + \log_b s_1 \log_b s_2) \end{aligned}$$

Table 8 shows that $f_0(b)$, $f_1(b)$, and $f_2(b)$ are very close to 1 for larger values of b. Thus, setting them to 1 yields:

$$E(b,d,s_1,s_2) \approx \frac{b^{d+1}}{(b-1)} \cdot \frac{1+\log_b s_1}{s_1} \cdot \frac{1+\log_b s_2}{s_2} \quad (3)$$

This shows that the nodes expanded by IDA* for a search to depth d are a fraction $\frac{1+\log_b s_1}{s_1} \cdot \frac{1+\log_b s_2}{s_2}$ of the nodes expanded by a brute-force search. This fraction is the product of the fractions of node expansions when using one database of size s_1 or s_2 from equation (2) as shown by Korf (2007).

Conjecture: Multiplicative Improvement

In most problem spaces the brute-force branching factor is not equal to the average branching factor in the pattern space graph. We make a conjecture for these relaxed assumptions: Given two consistent and independent disjoint additive PDBs, as well as the fractions k_1 and k_2, such that IDA* expands a fraction k_1 of the nodes expanded by a brute-force search when using PDB 1, and a fraction k_2 when using PDB 2, then IDA* expands a fraction k_1k_2 of the nodes expanded by a brute-force search when using the sum of the two PDBs.

Experimental Results

We used the Towers of Hanoi problem for our experiments. IDA* is not the algorithm of choice for this problem, because of a large number of duplicate nodes generated during a depth-first search. But it has the property that the disjoint additive PDBs are independent. We use the 4-peg 16 disc problem with a single 8 disc PDB as well as the sum of two disjoint additive 8 disc PDBs. The initial states are random configurations of the discs, and the goal state has all discs on the goal peg.

Table 7 gives experimental results. The first column gives the search depth d. The next 3 columns have results for a single 8 disc PDB generated by the 8 smaller discs. The first column in this block gives $E(N,d,P)$, the predicted number of nodes expanded using N_i and P. We approximated the equilibrium distribution P by the overall distribution from the PDB. The second column gives experimental results for the number of nodes expanded averaged over 100,000 random problems above the line, and 10,000 problems below the line. Because of the increasing number of nodes at each depth, we were not able to complete experiments for all depths. The third column gives the relative error, which shows that $E(N,d,P)$ is very accurate, and consequently P is very close to the overall distribution. The error appears to increase from depth 18 to 19, but this is because we reduced the number of problem instances run.

The second block uses the sum of two disjoint additive 8 disc PDBs, generated by the smaller and larger 8 discs. The first column in this block gives $E(N,d,P)$, where P is calculated as the convolution of the overall distributions of the two additive PDBs. Since any set of 8 different-sized discs generates the same PDB (Korf & Felner 2007) we used the same PDB for the smaller and for the larger 8 discs, and both have the same overall distributions. The second column gives experimental results averaged over 100,000 random problems for all depths. The third column gives the relative error. It seems that the experimental results for the sum of two 8 disc PDBs are very inaccurate, but the error keeps decreasing as d increases. One major source of error is introduced when approximating the equilibrium distributions by the overall distributions. Even though both additive PDBs have the same overall distributions, their equilibrium distributions differ. Experiments on smaller problem instances have shown, that the equilibrium distribution for the larger discs deviates more from the overall distribution than the one for the smaller discs. Also, the heuristic distribution for the larger discs converges much slower to the equilibrium distribution than the one for the smaller discs, because most moves move one of the smaller discs.

The last column gives the predicted number of nodes expanded using our conjecture, i.e. the nodes in the brute-force search tree times k_1k_2. The fractions k_1 and k_2 are calculated as $E(N,d,P)/\sum_{i=0}^{d} N_i$ or the nodes expanded by IDA* using one 8 disc PDB from the first block, divided by the number of nodes expanded by a brute-force search. The reduction fractions k_1 and k_2 are equal, i.e. $k_1 = k_2 = 4.0742 \cdot 10^{-5}$, because we used the same overall distribution for both PDBs. The conjecture predicts the same number of node expansions as predicted by $E(N,d,P)$ from the second block, where P is calculated using the convolution of the distributions of the two additive PDBs. We also calculated the number of node expansions predicted by the conjecture for a different number of discs and two PDBs of different size, and confirmed that it predicts the same number of node expansions as $E(N,d,P)$ using convolution.

Conclusions and Further Work

We have extended existing analyses of the performance of heuristic search. The first result was experimentally verifying that $E(\tilde{N},d,\tilde{P}) = \sum_{i=0}^{d} \tilde{N}_i \tilde{P}(d-i)$ for A* for the 15 puzzle. We showed that for A*, using the overall distribution $\tilde{P}$ is only approximate, because nodes with high branching factors are more likely to be visited at lower depths than nodes with low branching factors. This fact applies generally, not just for the sliding tile puzzle. We improve our prediction by using more accurate distributions for different classes of states, with the fraction of nodes of each class depending on the search depth. We demonstrated such a refinement for the 15 puzzle. These are the first results accurately predicting actual numbers of node expansions for A*.

Next we looked at how well Korf's (2007) model for IDA* applies to the 15 puzzle. We observed that the major discrepancy between the model and this problem is the difference in the brute-force branching factor of the problem tree and the branching factor of the pattern space graph.

A further contribution was extending Korf's (2007) model to disjoint additive PDBs. This model only uses the size s_j of two additive PDBs, the branching factor b of the problem space, and the search depth. It assumes that the forward and backward branching factors are uniform and equal, and that the disjoint PDBs are consistent and independent. For this model we showed that the number of nodes expanded by IDA* is a fraction $\frac{1+\log_b s_1}{s_1} \cdot \frac{1+\log_b s_2}{s_2}$ of the number of nodes expanded by a brute-force search. Finally we generalized this result, showing the reduction in the number of nodes expanded using multiple additive PDBs is the product of the reductions achieved by the individual databases. If we can predict the fractions k_j for individual PDBs, we can also predict the performance of the sum of disjoint additive PDBs. We experimentally verified this result using the Towers of Hanoi problem. This is the first result analyzing the performance of disjoint additive PDBs.

Currently we are working on bridging the gap in analyzing disjoint additive PDBs for real world problems, i.e., predicting the fractions k_j for a single PDB correctly. Furthermore we are trying to extend our analysis from two to n disjoint additive PDBs. We expect that this same analysis will extend to n PDBs. Also, we have been looking at the performance of A* using disjoint additive PDBs. Future work includes predicting the performance of A* without knowing the exact number of nodes at each depth.

Acknowledgment

This research was supported by NSF Grant IIS-0713178.

References

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

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* 4(2):100–107.

Holte, R. C., and Hernádvölgyi, I. T. 2004. Steps towards the automatic creation of search heuristics. Technical Report TR04-02, Computing Science Department, University of Alberta, Edmonton, Alberta.

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

Korf, R. E., and Felner, A. 2007. Recent progress in heuristic search: A case study of the four-peg towers of hanoi problem. In *IJCAI-07*, 2324–2329.

Korf, R. E., and Schultze, P. 2005. Large-scale parallel breadth-first search. In *AAAI-05*, 1380–1385.

Korf, R. E.; Reid, M.; and Edelkamp, S. 2001. Time complexity of iterative-deepening-A*. *Artif. Intell.* 129(1-2):199–218.

Korf, R. E. 1985. Iterative-deepening-A*: An optimal admissible tree search. In *IJCAI-85*, 1034–1036.

Korf, R. E. 2007. Analyzing the performance of pattern database heuristics. In *AAAI-07*, 1164–1170.

Zahavi, U.; Felner, A.; Schaeffer, J.; and Sturtevant, N. R. 2007. Inconsistent heuristics. In *AAAI-07*, 1211–1216.

Zhou, R., and Hansen, E. A. 2006. Breadth-first heuristic search. *Artif. Intell.* 170(4):385–408.