Independent Additive Heuristics Reduce Search Multiplicatively

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 analyzes the performance of IDA* using additive heuristics. We show that the reduction in the number of nodes expanded using multiple independent additive heuristics is the product of the reductions achieved by the individual heuristics. First, we formally state and prove this result on unit edge-cost undirected graphs with a uniform branching factor. Then, we empirically verify it on a model of the 4-peg Towers of Hanoi problem. We also run experiments on the multiple sequence alignment problem showing more general applicability to non-unit edge-cost directed graphs. Then, we extend an existing model to predict the performance of IDA* with a single pattern database to independent additive disjoint pattern databases. This is the first analysis of the performance of independent additive heuristics.

Introduction

All heuristic search algorithms, including IDA* (Korf 1985), use a heuristic evaluation function h to prune nodes. $h(n)$ estimates the lowest cost to get from node n to a goal state. If h never overestimates this cost, it is *admissible*. If $h(n) \leq k(n,m) + h(m)$ for all states n and m, where $k(n,m)$ is the cost of a shortest path from n to m, h is *consistent*.

For many problems, a heuristic evaluation function can be precomputed and stored in a lookup table called a *pattern database* (PDB) (Culberson and Schaeffer 1998). For example, for the Towers of Hanoi problem we choose a subset of the discs, the *pattern discs*, and ignore the positions of all other discs. For each possible configuration of the pattern discs we store the minimum number of moves required to solve this smaller Towers of Hanoi problem in the PDB. In general, a state is defined by an assignment of values to *state variables* and a *pattern* is a projection of a state from the original problem space onto the pattern space. In case of unit-cost operators, PDBs are constructed through a backward breadth-first search from the goal pattern in the pattern space. A perfect hash function maps each pattern to one entry in the PDB where the depth at which it is first generated is stored. This is exactly the minimum number of moves required to reach the projection of the goal state, the *goal pattern*, in the pattern space. During search we get the heuristic estimate of a state by projecting it on the pattern space and then using the perfect hash function to retrieve the pattern's entry in the PDB.

Under certain conditions it is possible to sum values from several PDBs without overestimating the solution cost (Korf and Felner 2002). For the Towers of Hanoi problem we can partition the discs into disjoint sets and construct a PDB for each of these sets. In general, if there is a way to partition all state variables into disjoint sets of pattern variables so that each operator only changes variables from one set, we can add the resulting heuristic estimates without loosing admissibility. We call the resulting PDBs *additive* and such a set of PDBs *disjoint*. In general, we call heuristics that can be added to get a better admissible heuristic *additive heuristics*.

A heuristic function can be characterized by its distribution. Given a population of states, the heuristic distribution is the probability that a randomly and uniformly selected state has a heuristic value less than or equal to x. The distribution over all states in the problem space graph is called the *overall distribution*. In case of PDBs, this distribution can be read directly from the PDB, if each pattern has the same number of pre-images in the original problem space (Holte and Hernádvölgyi 2004). For IDA* we use the *equilibrium distribution*, which is the heuristic distribution over all states at depth i in the brute-force search tree in the limit of large i. These two distribution are not always the same.

Overview

First, we review an important result on the performance of IDA* from Korf, Reid, and Edelkamp (2001). Secondly, we define the property of independence for heuristics. Then, we present the core result of this paper: The reduction in the number of nodes expanded by IDA* using multiple independent additive heuristics is the product of the reductions achieved by the individual heuristics. We formally state and prove this result on unit edge-cost undirected graphs with a uniform branching factor. We empirically verify it using a model of the 4-peg Towers of Hanoi problem. Next, we run experiments on the multiple sequence alignment problem showing more general applicability to non-unit edge-cost directed graphs. Then, we extend Korf's (2007) model to predict the performance of IDA* with a single PDB to independent disjoint additive PDBs. Finally, we show that our result does not extend immediately to dependent heuristics.

Time Complexity of IDA*

Korf, Reid, and 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. The heuristic function is assumed to be admissible and consistent. This model very accurately predicts the actual number of nodes expanded by IDA*.

IDA* with Independent Consistent Additive Heuristic Estimates

Independent Heuristics

Here we point out a property of heuristics that has not been previously used. As already mentioned, a heuristic function can be characterized by its distribution.

Definition 1. *Two heuristics h_1 and h_2 are independent if their distributions P_1 and P_2 are independent. In other words, knowing $h_1(x)$ tells us nothing about $h_2(x)$.*

Two equilibrium distributions are independent if and only if the overall distributions are independent. Thus, we just use the term distribution here, but the definition could be stated using either. The heuristic distribution of the sum of two independent heuristics is the convolution of their distributions:

$$(P_1 * P_2)(x) = \sum_{x_1=0}^{x} Pr[X_1 = x_1]\left(\sum_{x_2=0}^{x-x1} Pr[X_2 = x_2]\right)$$

where $Pr[X_1 = x_1]$ is the probability that a random state has h-cost x_1 according to heuristic h_1. The same definition holds for h_2. Examples of independent heuristics are the disjoint additive PDBs for the Towers of Hanoi problem. Each disc can be assigned to any peg, independent of the locations of all other discs. Once we know which peg each disc is located on, we also know its position on the peg because discs are always ordered by size. Consequently, knowing the value from one PDB tells us nothing about the value from the second PDB. Additive heuristics are also used for the sliding-tile puzzles. The number of moves required to get a set of tiles, the *pattern tiles*, in their goal positions is an admissible heuristic. Unlike in the Towers of Hanoi problem, non-pattern tiles are present, but indistinguishable. If we only count moves of pattern tiles we can use disjoint sets of pattern tiles to generate disjoint additive PDBs. For the 15 puzzle two disjoint sets of 7 and 8 tiles are often used. The positions occupied by the 7 pattern tiles cannot be occupied by the 8 pattern tiles. Thus, the resulting PDBs are not independent even though the sets of pattern tiles are disjoint.

Unit Edge-Cost Undirected Graphs with Uniform Branching Factor

Here we analyze the performance of independent additive heuristics, given certain properties of the problem space graph and the heuristic functions. First we look at the graph:

Assumption 1. *Let the problem space graph be a tree with unit edge costs and uniform brute-force branching factor b.*

Thus, the brute-force search tree grows exponentially with depth. In particular, there are b^i nodes at depth i.

Secondly we look at properties of the heuristic functions:

Assumption 2. *Let h_1 and h_2 be two integer valued heuristics based on the exact cost of optimal solutions of relaxed problems (Pearl 1984). Let h_1 and h_2 be additive and independent with equilibrium distributions P_1 and P_2, respectively. Furthermore, in a search to cost d, IDA* expands a fraction k_1 of the nodes expanded by a brute-force search when using h_1 and a fraction k_2 when using h_2, in the limit of large cost d.*

Heuristics based on the exact cost of optimal solutions of a relaxed problem are guaranteed to be admissible and consistent. We use h_1 and h_2 being based on solutions of a relaxed problem to show that the sum of heuristics h_1 and h_2 is consistent. Furthermore, Korf, Reid, and Edelkamp (2001) showed that fractions k_1 and k_2 always exist. Here we only assume that these fractions have values k_1 and k_2.

Finally, here is our result given the above assumptions:

Theorem 1. *Given a graph and h_1 and h_2 such that assumptions 1 and 2 hold, in the limit of large d, IDA* expands a fraction k_1k_2 of the nodes expanded by a brute-force search when using the sum of h_1 and h_2 in a search to cost d.*

Proof Outline. The actual proof requires two pages, so we only give an outline here. First, for a consistent heuristic, the set of nodes expanded in a search to cost d is exactly the set of nodes with f-cost less than or equal to d.

Korf, Reid, and Edelkamp (2001) showed that if the brute-force search tree grows exponentially with branching factor b, the ratio of the number of nodes expanded in a heuristic search to cost d, divided by the nodes expanded in a search to cost $d-1$ is also b. We define F_i^1 as the set of nodes with $f^1(n) = g(n) + h_1(n) = i$. Thus, these sets F_i^1 also grow exponentially by a factor b as i increases.

We calculate the heuristic distribution of h_2 over F_i^1. Since h_1 and h_2 are independent, knowing a node's h_1-cost does not tell us anything about its h_2-cost, so the distribution of h_2 over F_i^1 only depends on the distribution of g-costs over F_i^1. We show that the heuristic distribution of h_2 over F_i^1 converges to the equilibrium heuristic distribution P_2.

Next, we assume we search a tree which has the same structure as our brute-force search tree. Instead of its depth, however, we assign each node its f_1-cost. Then, we search this tree using IDA* with h_2 as the heuristic. Now in (1), $N_i = F_i^1$, all states with f_1-cost i, and the equilibrium heuristic distribution P is the heuristic distribution over the sets F_i^1 as i goes to infinity. In the limit of large depth we can replace N_i with b^i and we have shown that P converges to the equilibrium heuristic distribution P_2. Using this we can show that in a search to cost d, IDA* expands a fraction k_2 of the nodes with f_1-cost less than or equal to d.

Putting it all together, IDA* expands a fraction k_1k_2 of the nodes expanded by a brute-force search when using the sum of heuristics h_1 and h_2 in a search to cost d. □

	8-disc PDB	disjoint 8-8-disc PDB	
Depth	$E(b,c,d,P)$	$E(b,c,d,P*P)$	$k^2 \cdot \sum cb^i$
21	107,895,112	4,396	4,396
22	406,376,304	16,557	16,557
23	1,530,576,037	62,359	62,359
24	5,764,762,069	234,869	234,869
25	21,712,400,515	884,612	884,612
26	81,777,586,286	3,331,804	3,331,804
27	308,007,102,618	12,548,907	12,548,907
28	1,160,077,586,286	47,264,203	47,264,204
29	4,369,317,604,426	178,015,897	178,015,898

Table 1: Predicted Number of Nodes Expanded on our Model of the 16-Disc Towers of Hanoi Problem using IDA*

Depth	$E(b,c,d,P*P)$	Experimental	Error
21	4,396	2,178	101.83%
22	16,557	8,840	87.29%
23	62,359	35,579	75.27%
24	234,869	142,191	65.18%
25	884,612	564,685	56.66%
26	3,331,804	2,230,023	49.40%
27	12,548,907	8,762,686	43.21%
28	47,264,203	34,278,183	37.88%
29	178,015,897	133,549,960	33.30%

Table 2: Predicted and Actual Number of Nodes Expanded on the 16-Disc Towers of Hanoi Problem using IDA*

Experimental Results

We used the 4-peg Towers of Hanoi problem for experiments because its disjoint additive PDBs are independent. The classic problem has only three pegs, with a simple recursive optimal solution. For the 4-peg problem a recursive strategy has been proposed as well, but, absent a proof, search is the only way to verify optimality of this solution (Frame 1941; Stewart 1941). IDA* generates a large number of duplicate nodes and thus is not the algorithm of choice for this problem, but our result is for IDA*, and so we nevertheless ran IDA*. We used 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.

We create a model of the Towers of Hanoi problem matching assumptions 1 and 2: First, the actual problem does not have a uniform branching factor. A state's branching factor depends on the number of occupied pegs, but there are many more different types of states with different numbers of children, grandchildren, or great-grandchildren etc. We define two states to be of the same type if and only if they generate the same brute-force search tree below them, apart from labels of nodes and edges. In the Towers of Hanoi problem, two states are of the same type if they differ only by a permutation of the pegs because any series of moves with the pegs permutated has to leave the same number of pegs occupied in both states. Thus, each type has at most 4! states.

In our model, we assume a uniform branching factor equal to the asymptotic branching factor b, which is the number of nodes expanded in a depth-first search to depth d divided by the number of nodes expanded in a depth-first search to depth $d-1$, in the limit of large depths d. We do not allow moving the same disc twice, but we did not apply any operator ordering. In this problem, using b underestimates the number of nodes in the brute-force search tree. At shallow depths, b underestimates the actual branching factor, but eventually b becomes very accurate. In particular, b^i underestimates the number of nodes at depth i in the brute-force search tree by some factor c_i. As the ratio of numbers of states at consecutive depths converges to b, c_i converges to a constant c. We numerically fit c so that cb^d accurately approximates the number of states at large depths d.

Secondly, we approximate the equilibrium distribution by the overall distribution. Computing the equilibrium distribution requires computing the overall heuristic distribution for each type of state as well as the equilibrium fraction of nodes belonging to each type. Since there are so many types of states, an exact derivation becomes computationally unfeasible even for small problems.

Table 1 has our theoretical predictions. The first column gives the search depth d. The second column has results for a single 8-disc PDB generated by the 8 smallest discs. $E(b,c,d,P)$ gives the predicted number of nodes expanded where $N_i = cb^i$ and P is the overall heuristic distribution. The last two columns use the sum of two disjoint additive 8-disc PDBs, generated by the smallest and largest 8 discs. Here $E(b,c,d,P*P)$ gives the predicted number of nodes expanded where $P*P$ is the convolution of the overall distributions of the two additive PDBs. Since any set of 8 different-sized discs generates the same PDB (Korf and Felner 2007), we used only one PDB and thus both heuristics have the same overall distribution P. The last column gives the predicted number of nodes expanded using our theorem, the nodes in the brute-force search tree times k^2. The fraction k is calculated as $E(b,c,d,P)/\sum_{i=0}^{d} cb^i$ or the predicted number of nodes expanded by IDA* using one 8-disc PDB from the second column, divided by the number of nodes expanded by a brute-force search. The reduction fraction $k = 4.0742 \cdot 10^{-5}$ is the same for both PDBs. Apart from a small precision error, the theorem predicts the same number of nodes expanded as $E(b,c,d,P*P)$ from the third column. This shows that our theorem holds for this model.

Table 2 compares our theoretical predictions from our model to empirical results on the actual problem. The first column gives the search depth d. The second column gives $E(b,c,d,P*P)$ from Table 1. The third column gives experimental results averaged over $100,000$ random initial states. The optimal solution depth to move all discs from one peg to another is 161, but the average solution depth for random initial states is about 125. We ignore the goal state and continue search until the specified depth. The fourth column gives the relative error of our prediction compared to experimental results. The initially large error keeps decreasing monotonically as d increases, and the depths d of our experiments are still far from the average solution depth 125. One major source of error is introduced when approximating the equilibrium distributions by the overall distributions. Even

Depth	Experimental	$E(N, d, P_c)$	Error	$E(N, d, P_c^{(i)})$	Error
10	81	127	57.4%	81	0.0%
11	298	477	60.0%	298	0.0%
12	1,100	1,788	62.6%	1,100	0.0%
13	4,063	6,703	65.0%	4,063	0.0%
14	15,031	25,122	67.2%	15,031	0.0%
15	55,672	94,152	69.1%	55,672	0.0%

Table 3: Number of Nodes Expanded by IDA* on the 8-Disc Towers of Hanoi Problem using Two Disjoint 4 Disc PDBs

though both additive PDBs have the same overall distributions, their equilibrium distributions differ. Furthermore, the branching factor in the original problem space converges very slowly towards the asymptotic branching factor b because the larger discs are moved very infrequently. Thus, cb^i still overestimates the number of states at relevant depths i.

To verify that these are the only sources of error we ran experiments on a smaller 4-peg Towers of Hanoi problem with 8 discs and two PDBs generated by the smallest and largest 4 discs. Again, both heuristics have the same overall distributions, but different equilibrium distributions. Since this is a very small problem, for each depth, we can determine the exact fraction of states belonging to each type of state and the overall distributions for each type of state and then calculate the exact heuristic distributions $P_1^{(i)}$ and $P_2^{(i)}$ for all depths i as the weighted sum of the overall distributions. This shows that the equilibrium probabilities of low heuristic values are higher when using the smallest discs as pattern discs than when using the largest discs, and even higher in the overall distribution. Also, the heuristic distribution for the largest discs converges much more slowly to the equilibrium distribution than the one for the smallest discs because most moves move one of the smaller discs.

Table 3 has experimental results. The first column gives the search depth. The second column gives the number of nodes expanded averaged over all possible $65,536$ configurations of the discs as initial states. The third column gives the predicted number of nodes expanded using $E(N, d, P_c)$ from (1), where N_i is the exact number of states at depth i and $P_c = P_1 * P_2$ is the convolution of the overall distributions. The fourth column gives the relative error compared to experimental results. The fifth column uses the convolution of the exact heuristic distributions $P_c^{(i)} = P_1^{(i)} * P_2^{(i)}$ for each depth to predict the number of nodes expanded. The last column gives the relative error of this prediction.

One can notice that using the overall distributions P_1 and P_2 introduces a significant error. The second column ruled out the source of error that comes from approximating N_i by cb^i. Using cb^i would have overestimated the number of nodes expanded even further. The prediction using $P_1^{(i)}$, $P_2^{(i)}$ and N_i in the fifth column accurately predicts the number of nodes expanded, which shows that we have identified all the sources of error in Table 2.

Non-Unit Edge-Cost Directed Graphs with Non-Uniform Branching Factor

To show the generality of the multiplicative effect of independent additive heuristics on IDA*, we ran experiments on the multiple sequence alignment problem, an important problem in computational biology (Korf and Zhang 2000; Hohwald, Thayer, and Korf 2003). DNA sequences are represented by strings consisting of the characters `A`, `T`, `C`, and `G`. Protein sequences are represented by strings chosen from an alphabet of 20 amino acids. Alignment is achieved by inserting gaps in the sequences so that similar regions align when the sequences are compared. Here is an example of a three-way DNA sequence alignment (Thayer 2003):

```
AGTTA-
AGCT-G
-GACAG
```

A cost function is used to evaluate the quality of an alignment. The cost of an alignment of two sequences is the sum of the alignment costs of the characters in each column. The simplest cost function assigns a penalty of zero if the two characters are the same or both are gaps, one if they are different and two if there is a gap in one of the two sequences. The alignment of the first two sequences from the multiple sequence alignment above has cost $1 + 2 + 2 = 5$.

The cost of a multiple sequence alignment is the sum of all pairwise alignment costs, and is called the *sum-of-pairs* cost. In our example the alignment of the second and third and of the first and third sequence each has cost 6. Thus, this alignment of all three sequences has cost $5 + 6 + 6 = 17$. Based on this cost function, it is actually not an optimal alignment. The optimal alignment has cost 10 with no gaps inserted in any of the three sequences.

An easy to compute admissible heuristic is the *pairwise heuristic*. It takes advantage of the fact that the cost of an optimal pairwise subalignment is always less than or equal to the pairwise cost in a sum-of-pairs multiple sequence alignment. Therefore, the sum of all optimal pairwise subalignment costs is an admissible heuristic function. In our example, an optimal alignment of the first and second sequence would have no gaps and thus cost two. These two sequences also contribute a cost of two to the sum-of-pairs cost of the optimal alignment, which has no gaps either.

The optimal pairwise subalignment costs are not always independent. For example given three sequences, if we know that in the first column the first and the second sequence have the same character and the second and the third sequence have the same character, then the first and third sequence must also have the same character. But our experiments show that they are not highly correlated, and thus they can be treated as independent additive heuristics.

During search, alignments are created incrementally. The start state is an alignment of length zero, and all leaf nodes of the brute-force search tree are full alignments of the sequences. The sum-of-pairs cost of the already aligned sequence prefixes is used as the g-cost, and the pairwise heuristic of the unaligned postfixes is the h-cost.

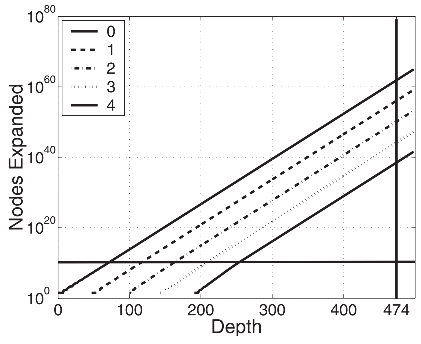

Figure 1: Actual and Predicted Number of Nodes Expanded Aligning 5 Random Sequences using IDA* with 0 to 4 Subalignments of Two Sequences as Heuristics

Sequence alignment can also be formulated as the task of finding a lowest-cost path through a directed hypercube. We refer the reader to Needleman and Wunsch's (1970) work. Bounded Diagonal Search (BDS) (Thayer 2003) and Sweep-A* (Zhou and Hansen 2003), the algorithms of choice for this problem, search this graph and detect all duplicate nodes, while IDA* searches the tree expansion of this graph and generates a large number of duplicate nodes.

Experimental Results

We used 5 random sequences of length 50 over an alphabet of 20 characters and the simple cost function described above for our experiments. We ran IDA* even though it is not the algorithm of choice for this problem, but our theorem was derived for IDA* only. Figure 1 has a combination of experimental results and predicted numbers of nodes expanded. The search depth is plotted on the x-axis, and the number of nodes expanded in a search to that threshold on the y-axis on a log scale. From top to bottom the graphs represent an iterative-deepening depth-first search, IDA* using one pairwise subalignment cost, the sum of two pairwise subalignment costs, up to the sum of four pairwise subalignment costs on the bottom. The lines only reflect actual experimental data until they cross approximately 10^{10} on the y-axis. Then we simply extended all lines to predict the number of nodes expanded for deeper depths. We drew a vertical line at the solution depth 474, which we determined using BDS. The intersections of all graphs with this line gives the predicted number of nodes expanded using the corresponding heuristic. They are all separated by the same vertical distances on this log scale, which means a constant multiplicative reduction in nodes expanded. Again, we were able to show the multiplicative effect of independent additive PDBs, this time on a non-unit edge-cost directed graph.

IDA* with Pattern Databases

Here we analyze the performance of IDA* with PDBs using only the brute-force branching factor, the size of the PDBs, and the solution depth. First we review an existing model using a single PDB, then we extend it to independent PDBs.

IDA* with a Single Pattern Database

Korf (2007) introduced a model for analyzing the performance of IDA* with a single PDB that builds on (1). N_i, the number of nodes at depth i in the brute-force search tree, is approximated by b^i, where b is the brute-force branching factor. The equilibrium distribution P is approximated using b and the size s of the PDB. 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, since PDBs are constructed through a backward breadth-first search from the goal state, Korf assumes that there is one node with h-cost 0, b nodes with h-cost 1, b^2 nodes with h-cost 2, etc., up to b^m nodes with maximum h-cost m, such that $\sum_{i=0}^{m} b^i \geq s$. In other words this model only depends on the branching factor b, the size of the PDB s, and the search depth d. The number of nodes expanded by IDA* for a search to depth d is approximately

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

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

IDA* with Independent Disjoint Additive Pattern Databases on Unit Edge-Cost Undirected Graphs

Here we extend Korf's (2007) theoretical analysis for IDA* to the sum of two independent disjoint additive PDBs of size s_1 and s_2, respectively. We assume without loss of generality that $s_1 \leq s_2$. The forward and backward branching factors of the problem space are both assumed to be b. $Pr[X_1 = x]$ equals $\frac{b^x}{s_1}$ for $x \leq m_1$, where m_1 is the maximum h-cost in PDB 1. The equivalent definitions hold for PDB 2.

Cumulative Equilibrium Heuristic Distribution

We assume independence of the individual PDBs. Thus, the equilibrium distribution of the sum of the two disjoint additive PDBs is the convolution of the equilibrium distributions of the individual PDBs. A heuristic estimate x is the sum of two terms, 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_1 always has to be less than or equal to m_1, and x_2 less than or equal to m_2. Thus, we have to look at different ranges for x. As an example we look at $P(x)$ for $x \leq m_1$:

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

We substitute $Pr[X_1 = i] = \frac{b^i}{s_1}$ and $Pr[X_2 = i] = \frac{b^i}{s_2}$ and derive the cumulative heuristic distribution function using simple algebraic transformations. For $m_1 < x \leq m_2$ and $m_1 \leq m_2 < x \leq m_1 + m_2$, $P(x)$ can be derived similarly.

Number of Nodes Expanded

The number of nodes expanded by IDA* for a search to depth d using the sum of two independent additive PDBs of

Depth	Experimental	Sampling	Error	Convolution	Error
45	20,397	20,641	1.19%	4,802	76.46%
46	42,659	43,974	3.08%	10,231	76.02%
47	92,579	93,684	1.19%	21,797	76.48%
48	194,331	199,585	2.70%	46,438	76.10%
49	419,795	425,198	1.29%	98,932	76.43%
50	883,854	905,843	2.49%	210,767	76.15%

Table 4: Actual and Predicted Number of Nodes Expanded by IDA* on 15 Puzzle with 7-8 Tile Disjoint Additive PDBs

size s_1 and s_2 can be derived from (1). We plug in the distributions, which we calculate as mentioned above, for $P(x)$ as well as b^i for N_i. Pascal's second identity and simple algebraic transformations yield for large values of b:

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

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

IDA* with Dependent Additive Heuristics

Here we investigate whether our result holds for correlated heuristics as well. The heuristic distribution of the sum of several additive heuristics can be computed using random sampling or, in case of independence, by taking the convolution of the individual distributions. We already mentioned the 7-tile and 8-tile disjoint additive PDBs for the 15 puzzles as an example of dependent additive heuristics.

Experimental results are shown in Table 4. The first column gives the search depth. The second column gives the average number of nodes expanded by IDA* over $100,000$ random initial states. The third column gives the number of nodes expanded predicted by (1). For the equilibrium heuristic distribution we randomly sample 10 billion states and differentiate by the position of the blank. The fourth column gives the relative error of this analysis compared to experimental results. The fifth column gives the predicted number of nodes expanded when computing the equilibrium distribution using the convolution of the overall distributions. Again, we differentiate by the position of the blank. The last column gives the relative error of this analysis compared to experimental results. Table 4 shows that the convolution underestimates the number of nodes expanded by more than 75% because the convolution underestimates the probabilities of small heuristic values. A very small error in the probabilities of small heuristic values blows up into a big error in the predicted number of nodes expanded because in (1) these probabilities are multiplied by the largest N_i. The probabilities of very large heuristic values are underestimated as well, but they only matter at low search depths. Thus, they do not introduce a big error. Summarizing, we cannot assume independence to perform analysis on dependent additive heuristics, and we showed an example where assuming independence introduces a significant error.

Conclusion

We presented the first results analyzing the performance of independent additive heuristics. First, we proved a multiplicative effect in the reduction of nodes expanded when using independent additive heuristics on unit edge-cost undirected graphs with a uniform branching factor. We experimentally verified this result using a model of the 4-peg Towers of Hanoi problem. Then, we empirically showed more general applicability to non-unit edge-cost directed graphs using the multiple sequence alignment problem. Finally, we extended Korf's (2007) model to predict the performance of IDA* with a single PDB to independent additive PDBs.

Acknowledgment

This research was supported by NSF grant No. IIS-0713178.

References

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

Frame, J. S. 1941. Solution to advanced problem 3918. *American Mathematical Monthly* 48:216–217.

Hohwald, H.; Thayer, I.; and Korf, R. E. 2003. Comparing best-first search and dynamic programming for optimal multiple sequence alignment. In *IJCAI*, 1239–1245.

Holte, R. C., and Hernádvölgyi, I. T. 2004. Steps towards the automatic creation of search heuristics. Technical Report TR04-02, 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 Zhang, W. 2000. Divide-and-conquer frontier search applied to optimal sequence alignment. In *AAAI-00*, 910–916.

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.

Needleman, S. B., and Wunsch, C. D. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Mol. Biol.* 48(3):443–453.

Pearl, J. 1984. *Heuristics*. Reading, MA: Addison-Wesley.

Stewart, B. 1941. Solution to advanced problem 3918. *American Mathematical Monthly* 48:217–219.

Thayer, I. 2003. Methods for optimal multiple sequence alignment. Master's thesis, UCLA, Los Angeles, CA.

Zhou, R., and Hansen, E. A. 2003. Sweep A*: Space-efficient heuristic search in partially ordered graphs. In *ICTAI-03*, 427.