A Measure of Quality for IDA* Heuristics

Robert K. P. Clausecker and Florian Schintke
Zuse Institute Berlin, Takustraße 7, 14195 Berlin, Germany
{clausecker,schintke}@zib.de

Abstract

We present a novel way to measure the performance of IDA* heuristics. With this measure of *heuristic quality* η, different heuristics for the same problem space can be compared objectively without regards to a particular problem instance. We show how η can be used to model the performance expectations of PDB heuristics. By drawing histograms of the contributions of different parts of the search space to η, we show what parts are most critical to the quality of a heuristic and contribute to the long-standing question on what h values are most critical to the performance of an IDA* heuristic.

1 Introduction

Heuristic search algorithms such as *A** (Hart, Nilsson, and Raphael 1968) and *iterative deepening A** (IDA*, Korf 1985) are driven by a heuristic function $h(v)$. Giving a lower bound to the number of steps needed to reach the goal vertex from the current vertex v, the heuristic function allows the search algorithm to disregard nodes in the search tree that cannot possibly lead to the goal within the remaining distance budget.

Of particular interest are *pattern database* or PDB heuristics (Culberson and Schaeffer 1996; Korf and Felner 2002). A PDB roughly simplifies a search problem, such that the solutions of the simpler problem can be tabulated and looked up at runtime. The distances of this simpler problem form an admissible heuristic, often with considerable pruning power.

Heuristic functions are the key factor to search performance. Designing heuristic functions is an active research area with many breakthroughs in the past years. However, not every improvement to heuristics gives equal results, and the impact of adjusting heuristics for different parts of the search space remains poorly understood.

In this paper, we introduce a new measure of *heuristic quality* allowing us to compare heuristic functions independently of a specific problem instance. In contrast to prior measures such as average h value, this heuristic quality η has a direct linear relation to the number of expanded nodes and admits further analysis on how the h values of different parts of the search space affect overall search performance.

2 Background

Given a directed or undirected graph $G = (V, E)$ and a pair of vertices v_0 and z, the *iterative-deepening A* algorithm* IDA* (Korf 1985) finds a shortest path from v_0 to z by a series of bounded depth-first searches with progressively larger depth d until a path is found. An admissible heuristic function $h(v)$ guides the search by providing a lower bound for the number of steps needed to reach z from the current candidate vertex $v \in V$, telling the search if z can be expected to be reached within the remaining depth budget.

In their foundational paper *Time Complexity of iterative-deepening A**, Korf, Reid, and Edelkamp (2001) estimated the number of expanded nodes in one round of such an IDA* search from a vertex v_0 to depth d by

$$E(v_0, d, P) = \sum_{i=0}^{d} N_i \, P(d - i) \tag{1}$$

where N_i is the number of vertices encountered at depth i of an uninformed depth-first search from v_0 and

$$P(i) = \mathrm{P}[\, h(u) \leq i \,] \tag{2}$$

is the probability that a vertex u chosen from the graph at random has an h value of no more than i.

For Eq. 1 to apply, $P(i)$ must be computed with respect to the *equilibrium distribution* of the search space, which is the distribution achieved by taking the limit of the distribution obtained after a random walk of k steps without returns as $k \to \infty$ (see Korf 2007).

3 The Heuristic Quality η

While the limitations of Eq. 1 are well known (see Zahavi et al. 2010) and more sophisticated analyses of the performance of IDA* exist (Lelis, Zilles, and Holte 2013), we would like to use $E(v_0, d, P)$ to measure the performance (or *quality*) of heuristics for a problem space in general as opposed to a specific instance within this space. This makes the limitations less important as we are not interested in an accurate prediction of expanded nodes for a specific search, but rather in understanding how the heuristic influences the search process.

We start by following an observation due to Korf (2007) that independently of v, the search front size N_i grows exponentially in i for many interesting search spaces. The base b

of this exponential growth is called the *branching factor* of the problem space. Setting

$$N_i = b^i \tag{3}$$

in Eq. 1, we obtain

$$E(b, d, P) = \sum_{i=0}^{d} b^i \, P(d-i) \tag{4a}$$

and then flip the order of summation by swapping i and $d-i$, allowing us to pull out b^d.

$$\begin{aligned} &= \sum_{i=0}^{d} b^{d-i} \, P(i) \\ &= b^d \sum_{i=0}^{d} \frac{P(i)}{b^i} \end{aligned} \tag{4b}$$

The sum is made independent of d by summing to $i = \infty$ and accounting for the difference with an error term.

$$= b^d \sum_{i=0}^{\infty} \frac{P(i)}{b^i} - b^d \sum_{i=d+1}^{\infty} \frac{P(i)}{b^i} \tag{4c}$$

Using $0 < P(i) \leq 1$, we establish bounds for the error term

$$0 < b^d \sum_{i=d+1}^{\infty} \frac{P(i)}{b^i} \leq b^d \sum_{i=d+1}^{\infty} \frac{1}{b^i} = \frac{1}{b} \sum_{i=0}^{\infty} \frac{1}{b^i} = \frac{1}{b-1} \tag{5}$$

and show how the number of expanded nodes predicted by Eq. 1 is proportional to b^d with the heuristic merely contributing a constant factor, regardless of what heuristic is used.

$$b^d \sum_{i=0}^{\infty} \frac{P(i)}{b^i} - \frac{1}{b-1} \leq E(b, d, P) < b^d \sum_{i=0}^{\infty} \frac{P(i)}{b^i} \tag{6}$$

We would like to express Eq. 6 in terms of $p(i)$, the probability that a vertex u chosen from the graph according to the equilibrium distribution has an h value equal to i.

$$p(i) = \mathrm{P}[\, h(u) = i \,] \tag{7}$$

It is

$$P(i) = \sum_{j=0}^{i} p(i), \quad p(i) = P(i) - P(i-1). \tag{8}$$

This is useful because as i surpasses the diameter of the graph, $P(i)$ assumes 1 but $p(i)$ drops to 0. While the sums in Eq. 6 have infinite terms, a reformulation in terms of $p(i)$ gives sums with only finitely many non-zero terms, making them easier to evaluate.

Rewriting Eq. 6, we find that $\sum_{i=0}^{\infty} P(i)/b^i$ is related to $\sum_{i=0}^{\infty} p(i)/b^i$ through the geometric sum $\frac{b}{b-1} = \sum_{i=1}^{\infty} b^{-i}$.

$$\begin{aligned} \frac{b-1}{b} \sum_{i=0}^{\infty} \frac{P(i)}{b^i} &= \sum_{i=0}^{\infty} \frac{P(i)}{b^i} - \sum_{i=0}^{\infty} \frac{P(i)}{b^{i+1}} \\ &= \sum_{i=0}^{\infty} \frac{P(i)}{b^i} - \sum_{i=1}^{\infty} \frac{P(i-1)}{b^i} \\ &= \sum_{i=0}^{\infty} \frac{P(i) - P(i-1)}{b^i} + \frac{P(-1)}{b^0} \\ &= \sum_{i=0}^{\infty} \frac{p(i)}{b^i}. \end{aligned} \tag{9}$$

From this, we define the *heuristic quality* η.

$$\begin{aligned} \eta &= \sum_{i=0}^{\infty} \frac{p(i)}{b^i} \\ &= \frac{b-1}{b} \sum_{i=0}^{\infty} \frac{P(i)}{b^i} \end{aligned} \tag{10}$$

This is the constant factor by which a given heuristic h reduces the number of expanded nodes compared to an uninformed iterative deepening search. We then restate Eq. 6 in terms of η:

$$\frac{b}{b-1} b^d \eta - \frac{1}{b-1} \leq E(b, d, P) < \frac{b}{b-1} b^d \eta. \tag{11}$$

With η, we have obtained a powerful tool to quantify how much a heuristic function $h(v)$ improves search performance. Eq. 11 shows that the factor by which a heuristic function improves the search is predicted to be independent of the depth or specifities of the problem instance we solve, making η a general, problem independent metric.

Intuitively, b^d is the number of nodes at depth d of uninformed depth-first search to depth d, $\frac{b}{b-1}$ accounts for the inner nodes in the search tree and η is the constant factor by which the heuristic reduces the number of expanded nodes. We decided to keep the factor $\frac{b}{b-1}$ separate from η to keep η normalised in the sense that $\eta = 1$ for a trivial heuristic h with $h(v) = 0$ everywhere.

Another way to see η is to view the heuristic $h(v)$ as reducing the depth of the search problem from d to an *effective depth*

$$d' = d - \log_b(1/\eta) \tag{12}$$

such that a depth-first search to depth d using $h(v)$ expands as many nodes as an uniformed depth-first search to depth d'.

4 The Quality of Arbitrary Heuristics

While the definition of η seems simple enough, finding the quality of arbitrary heuristics without knowledge about their construction takes more effort. The general idea is to treat η as an expected value

$$\eta = \mathrm{E}[b^{-h(v)}] \tag{13}$$

over the equilibrium-distributed search space, allowing an approximation of its value by the arithmetic mean of a random sample of vertices. However, this approach quickly

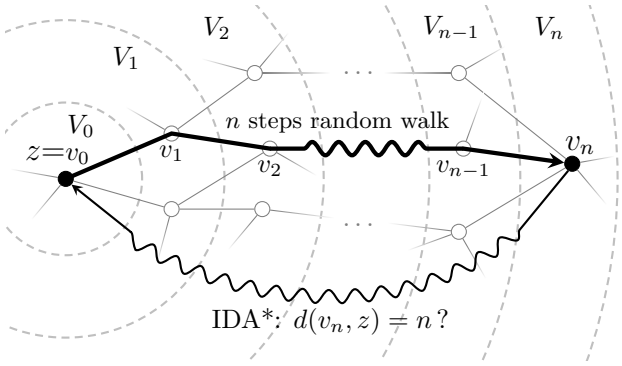


Figure 1: A vertex v_n drawn by random walk from $z = v_0$

fails for large search spaces as the vertices in the neighborhood of z dominate the value of η but are only a minuscule overall share of vertices.

For example, a PDB heuristic for the 24 puzzle using partitioning (a) from Fig. 3 has quality $\eta = 4.562 \times 10^{-22}$ with standard deviation $\sigma = 4.590 \times 10^{-13}$ (when interpreted as an expected value according to Eq. 13). To estimate this quality to just one significant digit (i. e. with an expected relative error of $10\,\%$ at $95\,\%$ confidence), about 4×10^{20} observations would be required, a clearly intractable number.

We get around this issue by sampling the search space stratified by distance to the goal vertex z. As distance to the goal vertex necessarily has a strong correlation to h value, this scheme, named *sphere stratified sampling*, captures the vertices with low h values in few small strata, allowing us to get a decent estimate of η with a feasible number of samples.

As an example, the quality given above was measured to an expected error of $0.73\,\%$ at $95\,\%$ confidence using only about 5×10^8 observations spread over 66 strata.

Sphere Stratified Sampling

With sphere stratified sampling, the search space is divided (stratified) into *spheres* V_0, V_1, …, V_{l-1} of vertices up to some limit l where each sphere V_n holds those vertices at distance n to the goal vertex z.

$$V_n = \{\, v \mid v \in V,\ d(v, z) = n \,\} \tag{14}$$

A final stratum $V_{\geq l}$ accounts for the rest of the graph.

$$V_{\geq l} = \{\, v \mid v \in V,\ d(v, z) \geq l \,\} \tag{15}$$

After having sampled the strata, the partial value of η for each stratum is estimated in $\eta_0, \eta_1, \ldots, \eta_{l-1}, \eta_{\geq l}$ and then combined into an estimate of η over the whole search space. The quality is then found as the sum of the partial qualities.

Sampling Spheres A sample from V_n is taken by performing a series of random walks of n steps from $v_0 = z$ (cf. Fig. 1). In case of a directed graph, this walk is performed with the direction of all edges flipped. At each step i, we uniformly draw a random vertex

$$v_i \in N'(v_0, v_1, \ldots, v_{i-1}) \tag{16}$$

where $N'(v_0, v_1, \ldots, v_{i-1})$ is the neighborhood of v_{i-1} pruned in some way such that each $v \in V_i$ is reachable from z by some random walk.[1] After n steps the walk concludes, and we check if $v_n \in V_n$ by means of some search algorithm. If it is, v_n is accepted and added to our collection of samples $\tilde{V}_n$. The observed probability of a sample being accepted is kept track of in the *sampling yield* y.

Unfortunately, the sample collected this way is not uniform. While it is difficult to make the sample uniform, we can compute the probability p_{v_n} of having accepted a vertex v_n into $\tilde{V}_n$ after an n step random walk and compensate for the bias later on. This can be done using information we have already collected: if the verification search from z to v_n is conducted to completion, we obtain the set $S(z, v_n)$ of shortest paths leading from z to v_n. The probability of reaching v_n by one path $(v_0, v_1, \ldots, v_n) \in S(z, v_n)$ is the product of the probabilities of having picked the right outgoing edge at each step of the random walk. The probability of having reached v_n after a random walk of n steps is the sum of the probability of each path by which it could have been reached. Finally, we divide by y to get the probability of v_n being drawn as a sample under the precondition $v_n \in V_n$. This gives us

$$p_{v_n} = \frac{1}{y} \sum_{\substack{(v_0,\ldots,v_n) \\ \in S(z,v_n)}} \prod_{i=0}^{n-1} \left| N'(v_0, \ldots, v_i) \right|^{-1}. \tag{17}$$

Using this probability, η_n can be obtained as

$$\eta_n = \frac{|V_n|}{|V|} \, |\tilde{V}_n|^{-1} \sum_{v_n \in \tilde{V}_n} \frac{w(v_n)}{|V_n| \, p_{v_n}} b^{-h(v_n)}, \tag{18}$$

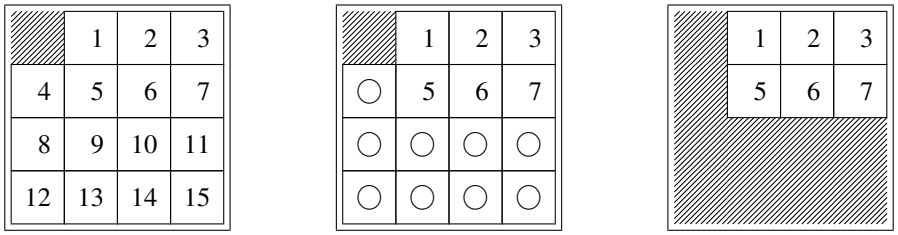
where $w(v_n)$ applies the *equilibrium weight* explained in detail in Sec. 5, $|V_n| \, p_{v_n}$ compensates for the sampling bias, and $|V_n|/|V|$ scales η_n to be the part V_n contributes to η.

As it usually encompasses most vertices in the graph, $V_{\geq l}$ is sampled by picking vertices uniformly from V and rejecting those that can be solved in less than l steps. This can be done using an IDA* search that aborts as soon as the budget f reaches l, indicating that the goal is at least l steps away. As this is a uniform sample, bias needs not be compensated and Eq. 19 can be used to find $\eta_{\geq l}$.

$$\eta_{\geq l} = \frac{|V_{\geq l}|}{|V|} \, |\tilde{V}_{\geq l}|^{-1} \sum_{v \in \tilde{V}_{\geq l}} \frac{w(v)}{b^{h(v)}} \tag{19}$$

Sphere Sizes To estimate η_n using Eq. 18, the size of the strata must be known. For small n, the size of V_n could be computed by means of an exhaustive breadth-first search to

[1] For this work, the authors used a variant of *finite state machine pruning* (Taylor and Korf 1993) extended to detect and prune *moribund states*. A state is 0-*moribund* if it is an accepting (i. e. pruned) state and k-*moribund* if each outgoing edge of the state is $k - 1$ or less moribund. If a k moribund state is reached with k or less steps left in the random walk, every possible continuation of the walk will lead to an accepting state before it could reach its goal. A move leading to such a situation is pruned immediately.



15 puzzle non-additive PDB additive PDB

Figure 2: The solved configuration of the 15 puzzle as seen by the non-additive and the additive $\{1, 2, 3, 5, 6, 7\}$ pattern database. While the non-additive PDB admits 2 moves out of this configuration, the additive PDB has 5 moves, reflecting its higher branching factor $B > b$.

distance n. While this approach becomes impracticable as n grows, we can luckily estimate $|V_n|$ to high precision using the probabilities p_{v_n} collected during the sampling process.

The reciprocal size $|V_n|^{-1}$ is the average probability of each vertex in V_n being drawn and thus the expected value of p_{v_n} assuming equidistributed vertices. We can model the sampling process as a *size-biased sample* drawing biased observations of p_{v_n} where the probability of having drawn each observation is proportional to its value as the value p_{v_n} is the probability itself. Applying a theorem due to Cox (2005), the expected value of reciprocals in a size-biased distribution is equal to the reciprocal of the expected value in a uniform distribution. That is, we have in the size-biased (i. e. sampled) distribution

$$\mathrm{E}[1/p_{v_n}] = \sum_{v_n \in V_n} p_{v_n}/p_{v_n} = \sum_{v_n \in V_n} 1 = |V_n| \qquad (20)$$

allowing us to estimate $|V_n|$ easily from the harmonic mean of probabilities p_{v_n}.

As for $V_{\geq l}$, its size is found by subtracting the size of all preceding spheres from the number of vertices in the graph.

$$|V_{\geq l}| = |V| - \sum_{i=0}^{l-1} |V_i| \qquad (21)$$

It is also possible to estimate $|V_{\geq l}|$ by dividing $|V|$ by the probability of a uniformly drawn vertex being in $V_{\geq l}$ (a side product of drawing samples from $V_{\geq l}$).

5 The Quality of Pattern Databases

The general idea behind PDBs is to abstract the problem into a simpler problem by ignoring some parts of its state. This yields a problem with a search space so small that the distance from each abstracted vertex v' to the abstracted solved configuration z' can be tabulated. If the abstraction is chosen such that each legal move in the original problem corresponds to a legal move in the abstracted problem, these distances form an admissible and consistent heuristic.

As an example, consider Fig. 2 for two ways to abstract the *15 puzzle*, a common benchmark problem for heuristic search. On the left, you can see the solved state of the 15 puzzle. Following the original Culberson and Schaeffer (1996) paper, the puzzle is abstracted by ignoring the identity of tiles 4 and 8–15, obtaining a $9! = 362\,880$ fold reduction of the search space size. This approach was improved by Korf and Felner (2002) into an *additive PDB* shown on the right, where we pretend the tiles 4 and 8–15 do not exist, allowing any moves into the hatched space. While this leads to a worse heuristic quality than the Culberson/Schaeffer approach, we gain *additivity*. This is the ability to add the h values of PDBs for disjoint tile sets, leading to much better h values with acceptable memory consumption through the combination of several small additive PDBs.

Computing the Quality

Glossing over the various kinds of pattern databases, we define a *table-based heuristic* of s entries over a search space $G = (V, E)$ as a pair (idx, tbl) of an *index function* $idx(v) : V \to \{0, 1, \ldots, s-1\}$ projecting the vertex set V to table entries and a *lookup table* $tbl[e] : \{0, 1 \ldots, s-1\} \to \mathbb{N}$ such that $h(v) = tbl[idx(v)]$ is a heuristic function. Additionally, the multi-valued *inverse index function*

$$idx^{-1}(e) = \{\, v \mid v \in V,\ idx(v) = e \,\} \qquad (22)$$

maps a table index back to the set of vertices it describes.

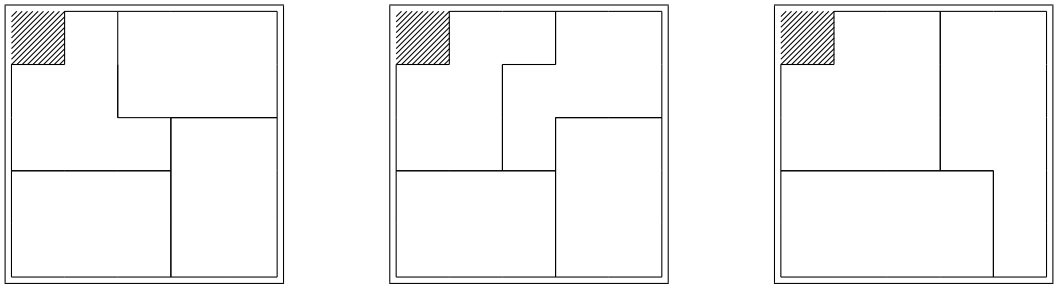
In the following paragraphs, we make use of the *equilibrium weight function* $w(v)$ giving weights to the search space's vertices proportional to the probability of drawing them in the equilibrium distribution. The proportionality constant is chosen such that $w(v) = 1$ for all v if the equilibrium distribution is an equidistribution. Letting u be a random vertex, it is thus

$$w(v) = |V|\,\mathrm{P}[\,u = v\,] \quad \text{with} \quad \sum_{v \in V} w(v) = |V|. \qquad (23)$$

This definition can then be used to express η as the sum of the contributions of each vertex to its value.

$$\eta = |V|^{-1} \sum_{v \in V} \frac{w(v)}{b^{h(v)}} \qquad (24)$$

Computing η is easy for table-based heuristics as $p(i)$ can be found by taking an appropriately weighted histogram over tbl. To do so, we first define the *index weight* $w_{idx}(e)$ as being the sum of $w(v)$ for all $v \in V$ with $idx(v) = e$.



(a) (b) (c)

Figure 3: Two partitionings of the 24 puzzle into 6-6-6-6 PDBs and one into 8-8-8 PDBs. (a) due to (Korf and Felner 2002), (b) from the author's previous work (Clausecker and Reinefeld 2019), and (c) due to (Döbbelin, Schütt, and Reinefeld 2013).

Similarly to Eq. 23, we have

$$w_{idx}(e) = \sum_{v \in idx^{-1}(e)} w(v) \quad \text{with} \quad \sum_{e=0}^{s-1} w_{idx}(e) = |V|. \tag{25}$$

With this in hand, we can compute the probability mass function

$$p(i) = |V|^{-1} \sum_{tbl[e]=i} w_{idx}(e) \tag{26}$$

and directly obtain η in analogy to Eq. 24.

$$\eta = |V|^{-1} \sum_{e=0}^{s-1} \frac{w_{idx}(e)}{b^{tbl[e]}} \tag{27}$$

Predicting the Quality

Non-additive PDBs In a performance analysis due to Korf (2007), *pattern database* (PDB) heuristics are modeled as subspaces of the problem space where each vertex in the subspace corresponds to an equal number of vertices in the problem space. The subspace described by the PDB is modeled as having the same branching factor b as the problem space such that the subspace is made up of $k+1$ classes of vertices with distances $i = 0, 1, \ldots, k$ and b^i vertices each.

This model accurately describes some kinds of *non-additive PDBs* like those for sliding tile puzzles described by Culberson and Schaeffer (1996) but due to the differing branching factors fails for *additive PDBs* such as those for sliding tile puzzles (Korf and Felner 2002) or gained by operator partitioning (Pommerening et al. 2015). An illustration of this effect can be seen in Fig. 2: because the additive PDB treats tiles not part of its pattern as nonexistent, it admits many more moves out of most positions than the unabstracted configuration.

In Korf's model, a pattern database with size

$$s = \sum_{i=0}^{k} b^i = \frac{b^{k+1} - 1}{b - 1}, \tag{28}$$

has probability mass function

$$p(i) = \begin{cases} b^i/s & \text{if } 0 \le i \le k \\ 0 & \text{otherwise.} \end{cases} \tag{29}$$

We can rewrite Eq. 28 to express k in terms of s and b

$$\begin{aligned} k &= \log_b\big(s\,(b-1)+1\big) - 1 \\ &= \log_b s + \log_b(b-1) - 1 + \mathcal{O}(s^{-1}) \end{aligned} \tag{30}$$

and then compute η by plugging s, $p(i)$, and k into Eq. 10.

$$\begin{aligned} \eta &= \sum_{i=0}^{k} \frac{b^i/s}{b^i} = \frac{k+1}{s} \\ &= \frac{\log_b\big(s\,(b-1)+1\big)}{s} \\ &= \frac{\log_b s + \log_b(b-1)}{s} + \mathcal{O}(s^{-2}) \end{aligned} \tag{31}$$

Hence, the pruning power of a PDB following this model is proportional to its size by the logarithm of its size. Korf (2007) gets the slightly different result

$$\eta = \frac{\log_b s + \log_b(b-1) + (b-1)^{-1}}{s} \approx \frac{\log_b s + 1}{s}, \tag{32}$$

where the extra $(b-1)^{-1}$ term is an artifact of some coarser approximations in his derivation.

Additive PDBs As an extension to Korf's model, it is useful to consider PDBs with a subspace branching factor B strictly larger[2] than the search space branching factor b, such as an additive PDB. This gives a heuristic with

$$s = \sum_{i=0}^{k} B^i = \frac{B^{k+1} - 1}{B - 1}, \tag{33}$$

$$\begin{aligned} k &= \log_B\big(s\,(B-1)+1\big) - 1 \\ &= \log_B s + \log_B(B-1) - 1 + \mathcal{O}(s^{-1}), \end{aligned} \tag{34}$$

$$\text{and} \quad p(i) = \begin{cases} B^i/s & \text{if } 0 \le i \le k \\ 0 & \text{otherwise.} \end{cases} \tag{35}$$

The quality is somewhat messy to derive. Proceeding as with Eq. 31, we start with

[2] $B < b$ cannot occur for admissible heuristics.

$$\eta = \sum_{i=0}^{k} \frac{B^i/s}{b^i} = \frac{1}{s}\sum_{i=0}^{k}\left(\frac{B}{b}\right)^i$$
$$= \frac{(B/b)^{k+1}-1}{s\,(B/b-1)} \tag{36a}$$
$$= \frac{b}{s\,(B-b)}\left(\left(\frac{B}{b}\right)^{k+1}-1\right)$$

and substitute k from Eq. 34.

$$= \frac{b}{s\,(B-b)}\left(\left(\frac{B}{b}\right)^{\log_B(s\,(B-1)+1)}-1\right) \tag{36b}$$

Then, the exponent is exchanged with the base using the identity $x^{\log y} = y^{\log x}$.

$$= \frac{b}{s\,(B-b)}\Big(\big(s\,(B-1)+1\big)^{\log_B(B/b)}-1\Big) \tag{36c}$$

We rewrite $\log_B(B/b)$ as $1 - 1/\log_b B$ and then simplify further. Assuming $b < B$, the exponent $\log_b B$ in the denominator is always greater than 1.

$$= \frac{b}{s\,(B-b)}\Big(\big(s\,(B-1)+1\big)^{1-1/\log_b B}-1\Big)$$
$$= \frac{b}{s\,(B-b)}\left(\frac{s\,(B-1)+1}{\sqrt[\log_b B]{s\,(B-1)+1}}-1\right)$$
$$= \frac{b}{B-b}\left(\frac{B-1+s^{-1}}{\sqrt[\log_b B]{s\,(B-1)+1}}-\frac{1}{s}\right) \tag{36d}$$
$$= \frac{B-1}{B/b-1}\,\frac{1}{\sqrt[\log_b B]{s(B-1)+1}}-\mathcal{O}(s^{-1})$$

As we have $\mathcal{O}\big(1/\sqrt[\log_b B]{s(B-1)+1}\big) > \mathcal{O}\big((\log_b s)/s\big)$ for $b < B$, the pruning power of an additive PDB given by Eq. 36 grows slower than that of a non-additive PDB, given by Eq. 31. The higher the difference between b and B is, the worse this effect. This predicts highly diminished returns when increasing PDB size for additive PDBs, as opposed to non-additive PDBs.

6 Results

To illustrate the use of η, we have computed the quality of some heuristics for the 24 puzzle using the method mentioned in Sec. 4. By plotting the contribution of each stratum to η, we then visualise what parts of the search space are most critical to the quality of a heuristic. This contributes to a long-standing question by Holte et al. (2004).

Examples

Tbl. 1 shows the values of η for the well-known *Manhattan heuristic* (Korf 1985), for three *additive PDB heuristics* (Clausecker and Reinefeld 2019) based on two 6-6-6-6 (Korf and Felner 2002; Clausecker and Reinefeld 2019) and one 8-8-8 partitioning (Döbbelin, Schütt, and Reinefeld 2013), as well as for two *PDB collections* (Clausecker and Reinefeld 2019). The effect of transposition search[3] on the heuristic's quality is shown as well, except for the Manhattan heuristic, as it is invariant under transposition.

heuristic	*quality*	*error*
Manhattan heuristic	$\eta = 9.926 \times 10^{-20}$	$(9.08\,\%)$
partitioning (a)	$\eta = 4.562 \times 10^{-22}$	$(0.73\,\%)$
with transp. search	$\eta = 1.359 \times 10^{-22}$	$(0.39\,\%)$
partitioning (b)	$\eta = 4.391 \times 10^{-22}$	$(2.13\,\%)$
with transp. search	$\eta = 1.611 \times 10^{-22}$	$(0.57\,\%)$
partitioning (c)	$\eta = 1.097 \times 10^{-22}$	$(0.49\,\%)$
with transp. search	$\eta = 6.780 \times 10^{-23}$	$(0.33\,\%)$
small collection	$\eta = 8.548 \times 10^{-23}$	$(0.50\,\%)$
with transp. search	$\eta = 3.787 \times 10^{-23}$	$(0.35\,\%)$
large collection	$\eta = 6.751 \times 10^{-23}$	$(0.71\,\%)$
with transp. search	$\eta = 3.208 \times 10^{-23}$	$(0.24\,\%)$

Table 1: The quality of various 24 puzzle heuristics without and with transposition search and expected relative error at $95\,\%$ confidence. Partitionings refer to Fig. 3, PDB collections to (Clausecker and Reinefeld 2019).

These measurements were taken using *sphere stratified sampling* with 66 strata (corresponding to the sets V_0 to V_{64} with a stratum $V_{\geq 65}$ accounting for the remaining vertices). Up to 10^7 observations per stratum were taken for V_0 to V_{64} with 10^8 observations being taken for $V_{\geq 65}$, giving a total of 5×10^8 observations.

These results give a number of interesting insights into the various heuristics. They show how the Manhattan heuristic's quality is 220 times worse than that of partitioning (a), the next best heuristic considered. Thus, we can expect partitioning (a) to expand on average 220 times less vertices than the Manhattan heuristic when solving difficult problem instances.

We also see how use of transposition search generally improves the quality by more than a factor of 2, offsetting the cost of the double lookup. An exception to this is partitioning (c) whose almost symmetrical structure likely causes the transposed puzzle's h-value to be very close to the original puzzle's h-value.

Lastly, we can see how the small collection of 7 partitionings and 14 PDBs outperforms the 8-8-8 partitioning (c) slightly without transposition and much more strongly with transposition. This is contrasted with the large collection of 14 partitionings made of 20 PDBs which does not manage to be all that better than the small collection, highlighting the diminished returns from adding more partitionings to a collection.

The Predictive Power

To show the effectiveness of η in predicting the pruning power of a heuristic, we have solved the 50 random instances of Korf and Felner (2002) with partitionings (a) and (b) as

[3] A method where the heuristic is queried both on the configuration v itself and on its transposition v^T across the main diagonal, with $\max\big(h(v), h(v^T)\big)$ being used as the h value for the search.

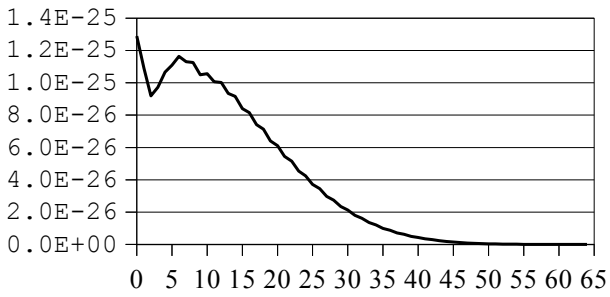


Figure 4: Baseline histogram: quality of the perfect heuristic $h(v) = d(v, z)$ by stratum

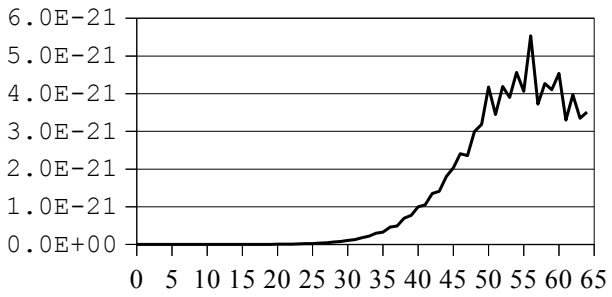


Figure 5: Manhattan heuristic quality histogram, $\eta_{\geq 65} = 2.364 \times 10^{-20}$

well as the small PDB collection; transposition search was used in all cases. Taking the geometric mean of the expanded vertices in these searches, we found that partitioning (b) expanded 1.022 times as many nodes as partitioning (a) while a ratio of 1.185 is predicted by the ratio of qualities. Likewise, we found that the small collection expanded 0.244 times as many nodes as partitioning (a) while η predicts a ratio of 0.279. While the predictions do not match exactly, they are close and may match even better given a larger body of instances.

Histograms

In their paper *Multiple Pattern Databases*, Holte et al. (2004) proposed that "eliminating low h-values is more important for improving search performance than retaining large h-values."

The quality η in conjunction with sphere stratified sampling provides us with the tools needed to test and expand this idea. By plotting the value of η_k for each sphere V_k, we obtain a *quality histogram* that tells us which parts of the search space we predict to contribute how much to the total number of expanded vertices. This partial η value η_k is defined similarly to Eq. 24 and obtains as a part of the sphere stratified sampling process.

$$\eta_k = |V|^{-1} \sum_{v \in V_k} \frac{w(v)}{b^{h(v)}} \tag{37}$$

Baseline The quality of the hypothetical perfect heuristic $h_{\min}(v) = d(v, z)$ given by

$$\eta_{\min} = \sum_{i=0}^{\infty} \frac{p(i)}{b^i} = \sum_{i=0}^{\infty} \frac{|V_i|/|V|}{b^i} \tag{38}$$

serves as a baseline for these histograms. No admissible heuristic can surpass $h_{\min}$ by definition and the quality histogram of no other heuristic can dip below $\eta_{\min}$. This tells us what the margin for possible improvement there is within the model of η.

For the 24 puzzle, we have $\eta_{\min} = 2.506 \times 10^{-24}$; Fig. 4 shows the corresponding quality histogram. The shape of the curve can be explained by $|V_i|$ growing slower than b^i as not all moves possible in a given configuration bring us further away from z. The initial spike in the graph is a consequence of the solved configuration being a single state with the blank tile in the corner that starts with a lower than average branching factor. As the distribution of states within a sphere gradually approaches the equilibrium distribution, the curve smoothes.

Example Histograms Figures 5, 6, and 7 show the quality histograms of some of the heuristics listed in Tbl. 1. For sufficiently good heuristics, it can be seen how a *critical region* between about V_{30} and V_{60} accounts for most of the value of η with contribution falling off to both sides.

We explain this intuitively through the interaction of two factors: (a) Close to z, the heuristic is fairly good at approximating $d(v, z)$, but as the spheres get more distant, the heuristic does a worse and worse job. The difference $d(v, z) - h(v)$ affects the contribution to η exponentially, so the contribution of the spheres to η starts out low and then rises sharply. (b) The farther we are from z, the more does the b^{-i} weighting pull down the curve. This is because as i grows, the growth rate of the sphere size $|V_i|$ keeps falling while b^i stays at the same exponential growth. This is exacerbated once the "belly" of the graph is reached and exponential growth in sphere size makes place for exponential decline in sphere size. Between these two processes, there is an interval where the contribution to η peaks, falling off to both sides.

In comparison with one another, it becomes evident that the peak moves slightly to the left as the heuristics get better. We interpret this as the heuristics reducing the h value contribution of more distant vertices more than that of vertices closer to z.

What h-values are critical? While the insight on the critical region does not directly tell us whether low or high h values are critical to search performance, we can use the strong correlation between distance to z and h value to observe what sphere likely corresponds to what h value. Working backwards from the definition of η_k and ignoring the equilibrium weighting, we can obtain $\tilde{h}_k$, the average h value for the vertices in V_k.

$$\tilde{h}_k = -\log_b \left(\frac{|V|}{|V_k|} \eta_k \right) \tag{39}$$

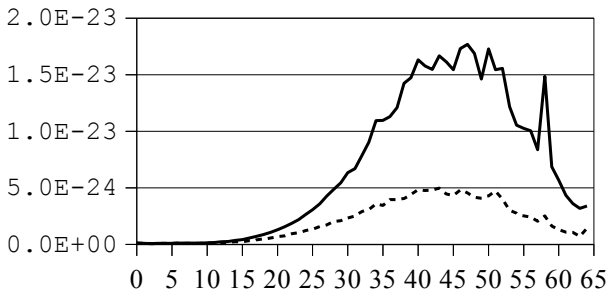


Figure 6: Partitioning (a) quality histogram without (solid line, $\eta_{\geq 65} = 1.098 \times 10^{-23}$) and with (dashed line, $\eta_{\geq 65} = 2.573 \times 10^{-23}$) transposition search

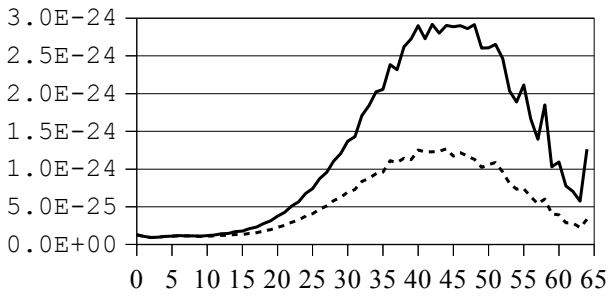


Figure 7: Small collection quality histogram without (solid line, $\eta_{\geq 65} = 2.008 \times 10^{-24}$) and with (dashed line, $\eta_{\geq 65} = 2.623 \times 10^{-25}$) transposition search

Under the precarious assumption that the h values within one stratum are very close, we can use $\tilde{h}_k$ on the boundaries of the critical region to obtain a critical region of h values to target when optimising heuristics.

For example, we find $\tilde{h}_{30} = 23.39$ and $\tilde{h}_{60} = 45.46$ for the histogram of partitioning (a) without transposition from Fig. 6, so the h values critical to search performance are expected to be about $h(v) \in \{23, 24, \dots 46\}$.

We can use this knowledge to guide our choices on how to spend the memory budget on heuristics. For example, it could be used to choose more effective ranges for *value compressed PDBs* (Sturtevant, Felner, and Helmert 2017) or to decide if adding another PDB to a PDB collection sufficiently increases h values in this region to be worth the extra memory and lookup cost.

7 Conclusions

With the *heuristic quality* η, we have introduced a novel measure for the quality of heuristic functions. It allows us to model the influence of the heuristic function on an IDA* search as a constant factor on the number of expanded nodes and is a convenient tool to compare different heuristics for pruning power.

Its simple definition enables the derivation of results about the pruning power of classes of heuristics, affirming a previous result (Korf 2007) for the pruning power of non-additive PDBs and extending it to the general case of table-based heuristics with possibly differing branching factors.

Using a sphere-stratified sample, we can effectively estimate η for arbitrary heuristics without having to know any details about the heuristic's construction. This gives us a tool to objectively compare the effectiveness of different heuristics in the general case and to improve heuristics by tuning their parameters to maximise their quality.

Plotting a *quality histogram* of η by distance from the solved configuration, we can see how the h values of a critical region in the search space contribute most to the number of expanded nodes. This gives us new insights into what parts of the search space are most critical to the performance of an IDA* heuristic and may guide decisions on what parts of the search space to spend a heuristic's memory budget on.

Acknowledgements

This work is based on the first author's master's thesis (Clausecker 2020). The first author would like to thank his advisor Alexander Reinefeld for his supervision and guidance. Both authors would also like to thank Benjamin Kaiser for his tireless work in reviewing the paper and for his many useful suggestions and improvements.

References

Clausecker, R. K. P. 2020. The Quality of Heuristic Functions for IDA*. ZIB Report 20-17, Zuse Institute Berlin.

Clausecker, R. K. P.; and Reinefeld, A. 2019. Zero-Aware Pattern Databases with 1-Bit Compression for Sliding Tile Puzzles. In *Proceedings of the Twelfth International Symposium on Combinatorial Search (SoCS 2019)*, 35–43.

Cox, D. R. 2005. Some sampling problems in technology. In *Selected Statistical Papers of Sir David Cox*, volume 1, chapter 1, 81–92. Cambridge University Press.

Culberson, J. C.; and Schaeffer, J. 1996. Searching with Pattern Databases. *Advances in Artifical Intelligence* 402–416.

Döbbelin, R.; Schütt, T.; and Reinefeld, A. 2013. Building Large Compressed PDBs for the Sliding Tile Puzzle. ZIB Report 13-21, Zuse Institute Berlin.

Hart, P. E.; Nilsson, N. J.; 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.; Newton, J.; Felner, A.; Meshulam, R.; and Furcy, D. 2004. Multiple Pattern Databases. In *ICAPS-04 Proceedings*, 122–131. AAAI Press / The MIT Press.

Korf, R. E. 1985. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence* 27(1): 97–109. ISSN 0004-3702.

Korf, R. E. 2007. Analyzing the Performance of Pattern Database Heuristics. In *Proceedings of the National Conference on Artificial Intelligence*, 1164–1170. AAAI Press / The MIT Press.

Korf, R. E.; and Felner, A. 2002. Disjoint pattern database heuristics. *Artificial Intelligence* 134(1-2): 9–22. ISSN 0004-3702.

Korf, R. E.; Reid, M.; and Edelkamp, S. 2001. Time complexity of iterative-deepening-A*. *Artificial Intelligence* 129(1–2): 199–218. ISSN 0004-3702.

Lelis, L. H. S.; Zilles, S.; and Holte, R. C. 2013. Predicting the size of IDA*'s search tree. *Artificial Intelligence* 196: 53–76. ISSN 0004-3702.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015. From Non-Negative to General Operator Cost Partitioning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 3335–3341. AAAI Press / The MIT Press.

Sturtevant, N. R.; Felner, A.; and Helmert, M. 2017. Value Compression of Pattern Databases. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 912–918. AAAI Press / The MIT Press.

Taylor, L. A.; and Korf, R. E. 1993. Pruning Duplicate Nodes in Depth-First Search. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 1202–1207. AAAI Press.

Zahavi, U.; Felner, A.; Burch, N.; and Holte, R. C. 2010. Predicting the Performance of IDA* using Conditional Distributions. *Journal of Artificial Intelligence Research* 37: 41–83.