Predicting the Performance of IDA* with Conditional Distributions

Uzi Zahavi
Computer Science
Bar-Ilan University
Ramat-Gan, Israel 92500
zahaviu@cs.biu.ac.il

Ariel Felner
Information Systems Engineering
Deutsche Telekom Labs
Ben-Gurion University
Be'er-Sheva, Israel 85104
felner@bgu.ac.il

Neil Burch **Robert C. Holte**
Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E8
{burch, holte}@cs.ualberta.ca

Abstract

(Korf, Reid, and Edelkamp 2001) introduced a formula to predict the number of nodes IDA* will expand given the *static distribution* of heuristic values. Their formula proved to be very accurate but it is only accurate under the following limitations: (1) the heuristic must be consistent; (2) the prediction is for a large random sample of start states. In this paper we generalize the *static distribution* to a *conditional distribution* of heuristic values. We then propose a new formula for predicting the performance of IDA* that works well for inconsistent heuristics (Zahavi et al. 2007) and for any set of start states, not just a random sample. We also show how the formula can be enhanced to work well for single start states. Experimental results demonstrate the accuracy of our method in all these situations.

Introduction

The goal of this research is to accurately predict the number of nodes IDA* (Korf 1985) will expand for a specific start state (or set of start states) and a specific depth bound and heuristic function. Our starting point is the formula developed in (Korf and Reid 1998; Korf, Reid, and Edelkamp 2001) to predict the number of nodes expanded by IDA*. These papers, the formula they present, and the predictions it makes, will all be referred to as KRE in this paper.

Traditionally, the standard method for comparing two heuristic functions was to compare their average values (Korf 1997; Korf and Felner 2002; Felner et al. 2007). A heuristic with a higher average value was considered preferable. KRE made a significant improvement on this by characterizing the quality of a heuristic function by its static distribution of values. Using the static distribution they developed the KRE formula to predict the number of nodes expanded by IDA* searching with a specific heuristic and depth bound. Finally, they compared KRE to the actual number of nodes expanded by IDA* for all depths on several benchmark search spaces and showed that it gave virtually perfect predictions. This was a major advance in the analysis of search algorithms and heuristics.

Despite its impressive results, the KRE formula has two main shortcomings. The first is that it assumes that the given heuristic is consistent in addition to being admissible.[1] The term *inconsistency* sounds negative but recent studies have shown that inconsistent heuristics are easy to define in many search applications and can produce impressive substantial performance improvements (Felner et al. 2005; Zahavi et al. 2007). Therefore, it is important to extend KRE to be able to predict IDA*'s performance on inconsistent heuristics because they are likely to become very important for future applications.

The second shortcoming of the KRE formula is that its predictions are accurate only when aggregated over a large random sample of start states; as will be shown below, it can be very inaccurate on single start states and even on a set of start states that were not chosen at random.

The contribution of this paper is twofold. First we extend KRE's idea of a *static distribution* of heuristic values to a *conditional distribution*, in which the probability of a specific heuristic value is not constant, as in KRE, but is conditioned on certain properties of the search space. Such properties can be the heuristic of the neighbors, the operators that are being applied, properties of the ancestors etc. Second, we derive a formula, G_KRE, based on the conditional distribution, that predicts IDA*'s performance on any set of start states (not necessarily random) whether the heuristic is consistent or not. We also show an extension of G_KRE which works well for a specific single start state. Experimental results confirm that G_KRE works well in all these scenarios.

The KRE formula

This section sketches the derivation of the KRE formula; for full details see the KRE papers. We assume that all state transitions cost 1, but this can easily be relaxed.

Let d be the given *depth bound* and BFS_s^d be the brute-force search tree of depth d rooted at start state s. Define a node n at level i of BFS_s^d to be *fertile* if $h(n) \leq d - i$. Node n being at level i with a heuristic value *less than or equal to* $d - i$ ensures that $f(n) = g(n) + h(n) \leq d$ which is the condition for expansion with an IDA* depth bound of d. Thus a fertile node will be expanded if it is generated. The key insight in KRE is that if the given heuristic is *consistent*

[1] A heuristic h is "admissible" if $h(x) \leq dist(x, goal)$ for all states x, where $dist(x, y)$ is the cost of the least-cost path from x to y. Heuristic h is "consistent" if $|h(x) - h(y)| \leq dist(x, y)$ for all states x and y.

the number of nodes expanded by IDA* at level i of BFS_s^d is exactly the number of fertile nodes at level i, which can be calculated using the following equation:

$$Fertile(s, d, i) = N(s, i) \cdot P(d - i)$$

where $N(s, i)$ is the number of nodes in level i of BFS_s^d and $P(v)$ is the percentage of states in the state space (not nodes in the brute-force tree) that have a heuristic value *less than or equal to* v. Typically this distribution is computed over the space of all possible states or, if that is impractical, over a large random sample of states.

In KRE, the $N(s, i)$ term is written as $N(i)$, i.e. without the dependence on the start state s. This is perfectly correct for state spaces, such as Rubik's Cube, with a uniform branching factor b, because $N(s, i)$ in such cases is simply b^i. For state spaces with a non-uniform but "regular" branching structure, $N(i)$ was defined recursively in KRE in terms of $N(i - 1)$ in a way that is independent of s. However, the base case of the recursion, $N(0)$, does depend on s so their notation $N(i)$ instead of $N(s, i)$ is reasonable but not strictly correct. Thus, the total number of nodes expanded by IDA* in searching BFS_s^d according to KRE is:

$$\texttt{KRE} = \sum_{i=0}^{d} N(i) \cdot P(d - i) \quad (1)$$

Limitations of the KRE formula

The KRE formula has two main shortcomings and we now examine each of them.

KRE fails for inconsistent heuristics

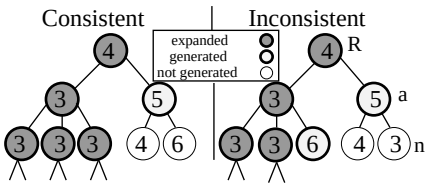


Figure 1: Consistent versus inconsistent heuristics

The KRE formula counts the number of *fertile* nodes at each level in BFS_s^d. These nodes have the *potential* to be expanded in the sense that IDA* will expand them if it generates them. With consistent heuristics, the heuristic value of neighboring states never changes by more than the change in the g-value (1 in this paper) as shown in the left side of Figure 1, where the number inside a node is its heuristic value. Thus, the f-value is *monotonically increasing* along any given branch. Therefore, it is easy to prove that with consistent heuristics all the ancestors of a fertile node are also fertile (cf. the KRE paper) and consequently IDA* will expand all and only the fertile nodes in BFS_s^d. Hence, a formula such as KRE that counts the number of fertile nodes in BFS_s^d can be used to predict the number of nodes IDA* will expand when given a consistent heuristic.

For inconsistent heuristics this reasoning does not apply. The heuristic values of neighboring states can change by much more than the change in g and the f-value can dramatically increase or decrease along a path. Therefore, the ancestors of a fertile node are not guaranteed to be fertile themselves, with the consequence that a fertile node might never be generated. For example, consider the search tree in the right side of Figure 1. Assume that the start node is R and that the depth bound is 5. There are 3 fertile nodes at depth 2 (all with heuristic value 3). Consider the fertile node n. The path to it is through node a but node a is infertile and will be generated but not expanded. Therefore, node n will never be generated, preventing IDA* from expanding it. Since the KRE formula counts the number of fertile nodes, it will count node n and thus overestimate the number of expanded nodes when an inconsistent heuristic is used.

The amount by which KRE overestimates the number of nodes expanded by IDA* with inconsistent heuristic can be very large. To illustrate this, consider the state space for Rubik's Cube and a pattern database (PDB) heuristic (Culberson and Schaeffer 1998) defined by the locations of 6 of the edge cubies. The regular method for looking up a heuristic value in a PDB produces a consistent heuristic. (Zahavi et al. 2007) discussed two alternative PDB lookups that produce inconsistent heuristics. The first is called the *dual* lookup. In permutation spaces, for each state s there exists a dual state s^d which has the same distance to the goal as s (Felner et al. 2005; Zahavi et al. 2006). When using PDBs, a dual lookup is to look up s^d in the PDB. This produces admissible but inconsistent heuristic values. The second method, *random lookup*, is to *randomly* select a heuristic from a number of available heuristics. This will produce inconsistent heuristics. Multiple PDB heuristic lookups arise in Rubik's Cube because it has 24 symmetries and each can be applied to any state to create a new way to perform a PDB lookup for it.

Because all three lookups (regular, dual and random) consult the same PDB they have the same static distribution of heuristic values, $P(v)$, and therefore KRE will predict that IDA* will expand the same number of nodes regardless of whether a regular, dual or random lookup is done.

The "IDA*" columns of Table 1 show the number of nodes IDA* expands when it performs either a regular, dual or random lookup in the same 6-edge PDB for Rubik's Cube. The "KRE" column, based on the static distribution, shows the KRE prediction. Each row represents a specific depth bound (d), and the numbers shown are averages over 1000 random initial states. The KRE prediction is within 8% of the actual number of nodes expanded when IDA* uses the regular (consistent) PDB lookup (second column) but it significantly overestimates the number of nodes expanded when IDA* does dual or random (inconsistent) lookups in the same PDB (sixth and eighth columns). The other columns will be discussed below.

KRE fails for non-random set of start states

For a consistent heuristic, and for most inconsistent heuristics too, the heuristic value of a state is highly correlated with the heuristic value of neighboring states. Consequently, the distribution of heuristic values in the search tree near the start state will be highly correlated with the heuristic value of the start state, and therefore will not be the same in search trees with start states having different heuristic values.

	Regular Lookup				Dual Lookup		Random Lookup		Static
d	**IDA***	KRE	G_KRE(1)	G_KRE(2)	**IDA***	G_KRE(2)	**IDA***	G_KRE(2)	MaxInc
9	3,624	3,431	3,151	3,446	518	508	346	346	210
10	47,546	45,801	41,599	45,985	6,809	6,792	4,608	4,601	2,813
11	626,792	611,385	546,808	613,332	92,094	90,664	61,617	61,174	37,553
12	8,298,262	**8,161,064**	7,188,863	8,180,676	1,225,538	1,210,225	823,003	815,444	501,293
13	110,087,215	108,937,712	94,711,234	109,133,021	16,333,931	16,154,640	10,907,276	10,878,227	6,691,518

Table 1: Rubik's Cube - regular, dual and random PDB lookups and their predictions

The reason that KRE is able to produce extremely accurate predictions in its experiments using just one distribution of heuristic values, $P(v)$, for all levels and all start states is that its experiments report average predictions and performance over a large number of randomly drawn start states. The heuristic values of a large random sample of states will be distributed according to $P(v)$, and, because the heuristic values at successive levels are strongly correlated with the values at previous levels, they too will be distributed according to $P(v)$. However, if the set of start states does not have its heuristic values distributed according to $P(v)$, KRE should not be expected to make good predictions. For example, a great deal of pruning is likely to occur near the top of the search tree for a start state with a large heuristic value, resulting in many fewer nodes expanded than for a start state with a small heuristic value. Yet KRE will make the same prediction for both start states because it uses $P(v)$ as the distribution in both cases.

h	IDA*	KRE	G_KRE(1)	G_KRE(2)
5	30,363,829	8,161,064	48,972,619	20,771,895
6	18,533,503	8,161,064	17,300,476	13,525,425
7	10,065,838	8,161,064	7,918,821	9,131,303
8	6,002,025	8,161,064	5,094,018	6,743,686

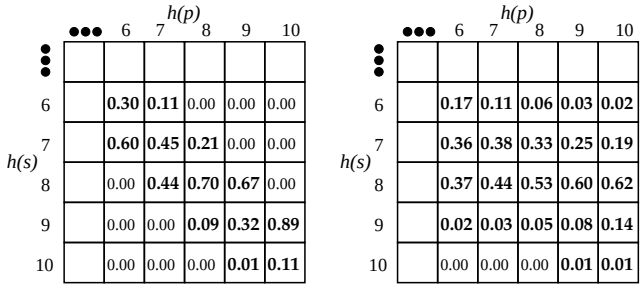
Table 2: Results with different start state heuristic.

Table 2 demonstrates this phenomenon on Rubik's Cube with regular PDB lookups for one depth bound ($d = 12$). The "IDA*" column shows the average number of nodes expanded for 1000 start states with the same heuristic value h. KRE ignores h and predicts that 8,161,064 nodes will be expanded by IDA* for each start state. The row for $d = 12$ in Table 1 shows that this is an accurate prediction when performance is averaged over a large random sample of start states, but in Table 2 we see that it is too low for start states with small heuristic values and too high for ones with large heuristic values. The last two columns of Table 2 show that the prediction of two variations of our new method (defined below), which takes the heuristic value of the start state into account, is much more accurate than KRE.

Conditional static distribution

These two shortcomings of KRE can both be overcome by extending the static distribution of heuristic values it uses, $P(v)$, to be a conditional distribution, $P(v|context)$, where $context$ represents properties of the search space that influence the distribution of heuristic values in a local neighborhood. We use $p(v|context)$ (lower case p) to denote the probability that a state with heuristic value *equal* to v will be produced when a state satisfying the conditions specified by $context$ is expanded. We use $P(v|context)$ (upper case P) to denote the probability that a state with heuristic value *less than or equal to* v will be produced when a state satisfying the conditions specified by $context$ is expanded. Obviously, $P(v|context) = \sum_{i=0}^{v} p(i|context)$.

The simplest conditional distribution is $p(v|v_p)$, the probability of a state with a heuristic value *equal to* v being produced when a state with value v_p is expanded. In special circumstances, $p(v|v_p)$ can be determined exactly by analysis of the state space and the heuristic, but in general it must be approximated empirically by sampling the state space. In our sampling method distribution $p(v|v_p)$ is represented by the entry $p_2[v][v_p]$ in a two-dimensional matrix $p_2[0..h_{max}][0..h_{max}]$, where h_{max} is the maximum possible heuristic value. To build the matrix we first set all values in the matrix to 0. We randomly generate a state and calculate its heuristic value v_p. We then generate each child of this state one at a time, calculate the child's heuristic value, v, and increment $p_2[v][v_p]$. We repeat this process a large number of times. Finally, we divide each entry in column v_p by the sum of that column, so that entry $p_2[v][v_p]$ represents the percentage of children generated with value v when a state with value $v_p$is expanded.



(a) Consistent heuristic (b) Inconsistent heuristic

Figure 2: A portion of the p_2 matrix for Rubik's Cube

Figure 2 shows the bottom right corner of two such matrices for the 6-edges PDB of Rubik's Cube. The left matrix (a) shows $p(v|v_p)$ for the regular (consistent) lookup in this PDB and the right matrix (b) shows $p(v|v_p)$ for the inconsistent heuristic created by the dual lookup in this PDB. The matrix in (a) is tridiagonal because neighboring values cannot differ by more than 1. For example, states with a heuristic value of 8 can only have children with heuristics of 7, 8 and 9; these occur with probabilities of 0.21, 0.70 and 0.09 respectively (see column 8). By contrast, the matrix in (b) is not tridiagonal. In column 8, for example, we see that 6% of

the time states with heuristic value of 8 have children with heuristic values of 6.

When IDA* expands a node, it eliminates some nodes because of operator pruning. Distribution $p(v|v_p)$ does not take this into account. In order to do so it is necessary to extend the context of the conditional probability to include the heuristic value of the grandparent. We denote this by $p(v|v_p,v_{gp})$ and call this a "2-step" model because it conditions on information from two ancestors in contrast the $p(v|v_p)$, which is a "1-step" model. $p(v|v_p,v_{gp})$ gives the probability of a state with a heuristic value equal to v is being produced when expanding a state with heuristic value v_p and whose parent's heuristic value is v_{gp}. It is estimated by sampling in the same way as was done to estimate $p(v|v_p)$, except that each sample generates a random state, *gp*, then all its neighbors, and then all of their neighbors except those eliminated by operator pruning.

The context of the conditional distribution can be extended in other ways as well. For the sliding tile puzzles, KRE conditions the static distribution on the "type" of the state being expanded, where the type indicates if the blank is in a corner, edge, or interior location. In our experiments with the sliding tile puzzle below, we extend $p(v|v_p,v_{gp})$ with this type information: $p(v,t|v_p,t_p,v_{gp},t_{gp})$ gives the probability of a state with type t and heuristic value equal to v being produced when a state is expanded whose heuristic value and type are v_p and t_p, respectively, and whose parent's heuristic value and type are v_{gp} and t_{gp}, respectively.

A new prediction formula, G_KRE

In this section we use the conditional distributions just described to develop G_KRE, an alternative to the KRE formula for predicting the number of nodes IDA* will expand for a given heuristic, depth bound, and set of start states. For simplicity, we assume there is no type system imposed on the states; the development of the formula when there are types is exactly analogous.

Define $N_i(s,v)$ to be the number of nodes that IDA* will *generate* at level i with a heuristic value *exactly equal* to v when s is the start state. For depth bound d, $\sum_{v=0}^{d-i} N_i(s,v)$ is the number of nodes IDA* will expand at level i (whether the given heuristic is consistent or not), precisely the quantity we wish to calculate. If the 1-step conditional distribution $p(v|v_p)$ is being used, $N_i(s,v)$ can be estimated recursively as follows:

$$N_i(s,v) = \sum_{v_p=0}^{d-(i-1)} N_{i-1}(s,v_p) \cdot b \cdot p(v|v_p) \qquad (2)$$

where b is the brute-force branching factor.[2] The reasoning behind this equation is that $N_{i-1}(s,v_p)\cdot b$ is the total number of children IDA* generates via the nodes it expands at level $i-1$ with heuristic value equal to v_p. This is multiplied by $p(v|v_p)$ to get the expected number of children of these nodes that have heuristic value v. Nodes at level $i-1$ are expanded if and only if their heuristic value is *less than or equal to* $d-(i-1)$, hence the summation only includes v_p values in this range. By restricting v_p to be *less than or equal to* $d-(i-1)$ in every recursive application of this formula, we ensure (even for inconsistent heuristics) that a node is only counted at level i if all its ancestors are expanded by IDA*. The base case of this recursion, $N_0(s,v)$, is 1 for $v = h(s)$ and 0 for all other values of v.[3] The number of nodes expanded by IDA* given start state s, depth bound d, and a particular heuristic can be predicted as follows:

$$\texttt{G\_KRE} = \sum_{i=0}^{d} \sum_{v=0}^{d-i} N_i(s,v) \qquad (3)$$

If a set of start states is given instead of just one, the calculation is identical except that the base case of the recursion ($N(0)$) is seeded using all the start states. That is, we increment $N_0(s_j,v)$ for each start state s_j with a heuristic of v. This cumulative N_0 is used as the base case and the rest is just the same.

Between "informedness" and inconsistency

Traditionally, more informed heuristics (i.e. with high heuristic values) were considered preferable. (Zahavi et al. 2007) showed that this is not enough for inconsistent heuristics. They introduced the *degree of inconsistency* which intuitively, is the amount of correlation between the heuristics of neighboring nodes. They showed that heuristics with the same "informedness" (i.e., static distribution) but with higher degree of inconsistency (less correlation) will perform better as more pruning will occur. The maximal degree of inconsistency is achieved when there is no correlation between the heuristic values of any two states in the search space and they all independently obey the static distribution P. In such case we can set $p(v|v_p) = p(v)$ in equation 2.

An outcome of our new formula is that we are now able to predict the performance of IDA* while assuming maximal degree of inconsistency. The "MaxInc" column in Table 1 presents the maximal possible improvement by inconsistent heuristics when the maximal degree of inconsistency is assumed. This can be seen as a lower bound on the number of expanded nodes for all possible heuristics with a given static distribution.

Experimental results – Rubik's Cube

We compared the performance of G_KRE, to KRE,[4] on Rubik's Cube with our 6-edges PDB on the regular, dual and random heuristics on 1000 random start states.

[2] In general, b can depend on the context used to define the conditional distribution.

[3] With a 2-step model the base case would be $N_1(s,v)$, the number of children of the start state that have heuristic value v.

[4] Our experiments on Rubik's Cube and the sliding tile puzzle differ from the experiments reported in KRE in the way we chose the start states. In KRE, the same large randomly chosen set of start states is used for every depth bound d and the search continues to depth d for every start state, even if the goal is encountered for a start state at a depth less than d. But, IDA* would not do this, it would stop as soon as the goal was encountered. To mimic this, for each value of d we used state s as a start state only if IDA* actually performs an iteration with a depth bound of d when s is the start state.

The results are presented in Table 1. G_KRE(1) and G_KRE(2) denote the 1-step and 2-step models, respectively. For the regular (consistent) lookup (2nd column) the predictions of KRE, G_KRE(1) and G_KRE(2) are all very accurate. Here we see that, overall, G_KRE(2) is more accurate than both G_KRE(1) and KRE. As discussed above, KRE produces the same prediction for all these heuristics and is overestimating for the inconsistent heuristics. By contrast G_KRE(2) predicts IDA*'s performance extremely accurately for the inconsistent heuristics. The predictions of G_KRE(1) for the inconsistent heuristics are not shown but are a little inferior to G_KRE(2).

Table 2, presented above, and the related discussion, shed light on the fact that KRE cannot make accurate predictions when start states are not selected uniformly at random. KRE will always predict a value of 8,161,064 even though the value depends on the exact set of start states used. In that table we can see that both versions of G_KRE significantly outperform KRE on a particular given set of start states.

Experimental results - Sliding Tile puzzle

As in the KRE experiments, three state types are used, based on whether the blank is in a corner, edge, or interior location, exact recurrence equations are used for $N(i,t)$ in the type-dependent version of the KRE formula, and the heuristic used was Manhattan Distance (MD). We used the 2-step type-dependent version of G_KRE.

h	#States	IDA*	KRE	G_KRE(2)
8-puzzle depth 22				
12	11454	1499	1391	1809
14	19426	1042	1404	1051
16	18528	660	1419	544
18	10099	377	1447	246
15-puzzle depth 52				
38	2999	16,226,330	428,883,700	21,505,426
40	3028	6,310,724	433,096,514	6,477,903
42	2454	2,137,488	438,475,079	1,749,231
44	1507	620,322	444,543,678	409,341

Table 3: Tile puzzles with a consistent heuristic (MD).

Prediction results of KRE and G_KRE for the 8- and 15-puzzles are shown in Table 3. For the 8-puzzle the predictions were made for depth 22 and each row corresponds to the group of **all** 8-puzzle states with the same heuristic value h (shown in the first column). The second column gives the number of states in each group. Clearly, as shown in the "IDA*" column, states with higher initial heuristics expanded a smaller number of nodes. This trend is not reflected in the KRE predictions since KRE does not take h into account. For KRE the only difference between the attributes of different rows is the different type distribution (of the blank's location) among the given group. Thus, the predicted number of expanded nodes of KRE is very similar for all rows (around 1400). The G_KRE formula takes the heuristic of the start state into account and was able to predict the number of expanded nodes much better than KRE. The bottom part of Table 3 show results for the 15 puzzle. Similar tendencies are observed.

d	IDA*	KRE	G_KRE
26	328	6,570	251
27	562	12,475	432
28	818	19,516	619
29	1,432	37,425	1,075

Table 4: Inconsistent heuristic for the 8-puzzle.

Inconsistent heuristics for the tile puzzle Our next experiment is for an inconsistent heuristic on the 8-puzzle. We defined two PDBs, one based on the location of the blank and tiles 1–4, the other based on the location of the blank and tiles 5–8. To create an inconsistent heuristic, only one of the PDBs was consulted by a regular lookup. The choice of PDB was made systematically, not randomly, based on the position of the blank. Thus, neighboring states were guaranteed to consult different PDBs. The results, presented in Table 4, show that G_KRE's predictions are reasonably accurate, and very much more accurate than KRE's.

Single start states

We have seen that G_KRE works well when the base case for the recursion ($N(0)$) is seeded by a set of start states, even if the set is not chosen at random. However, the actual number of expanded nodes for a specific **single** start state can deviate from the number predicted by G_KRE for its context. This is because there is no way to predict the heuristic of a neighbor of a **single** state as it does not necessarily obey the observed distribution. Consider a Rubik's Cube state with a heuristic value of 8. G_KRE predicts that IDA* will expand 6,743,686 for such a state (with depth bound 12). Indeed, Table 2 shows that on the average (over 1000 such start states) 6,002,025 states are expanded. Examining all these 1000 start states with a heuristic of 8 showed that the actual number of expanded nodes ranged between 2,398,072 to 15,290,697 nodes.

In order to predict the number of expanded nodes for a single start state we suggest the following enhancement to G_KRE. Assume that we want to predict the number of expanded nodes with depth bound d for start state s. First, we perform a small initial IDA* search from s up to radius r. We then seed all the states at radius r in the base case $N(0)$ of the G_KRE formula and compute the formula with depth $d-r$. This will cause a larger set of nodes to be seeded at level 0 of the G_KRE formula and more accurate results will be achieved.

h	IDA*	G_KRE(r=2)	G_KRE(r=5)	G_KRE(r=6)
8	2,398,072	4,854,485	3,047,836	2,696,532
8	4,826,154	7,072,952	5,495,475	5,184,453
8	15,290,697	9,432,008	13,384,290	14,482,001

Table 5: Single state ($d = 12$).

Table 5 shows results for three specific Rubik's Cube states with a heuristic of 8 (of the regular 6-edges PDB lookup) when the depth bound was set to 12. We chose the states with the least, median and greatest number of expanded nodes. The first column shows the actual number of nodes IDA* expands for each state. The next columns show

the number of expanded nodes predicted by our enhanced G_KRE formula where the initial search was performed to radiuses of 2, 5 and 6. Clearly, these initial searches give much better predictions than the simple G_KRE which predicts 6,743,686 for all these states. With an initial search to radius 6, the predictions were very accurate.

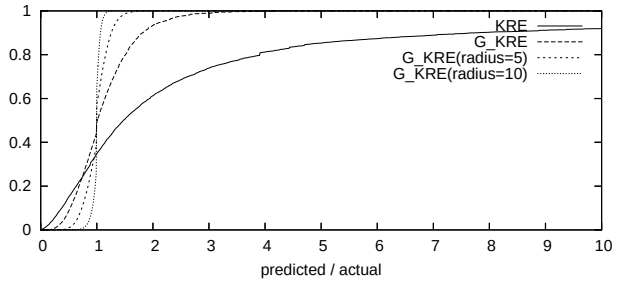


Figure 3: Relative error for the 8-puzzle

The 8-puzzle We performed experiments with the enhanced G_KRE formula on all the states of the 8-puzzle with the (consistent) MD heuristic. We use the term "trial" to refer to each pair of a single start state and a given depth bound d. The trials included all possible values of d and for each d all start states for which IDA* would actually perform a search with depth bound d. Predictions were made for each trial separately, and the relative error, $predicted/actual$, for the trial was calculated. The results are shown in Figure 3. There are four curves in the figure, for KRE, for G_KRE, and for the enhanced G_KRE with radiuses of 5 and 10. The x-axis is relative error. The y-axis is the percentage of trials that the prediction had a relative error of x or less. For example, the y-value of 20% for the KRE curve at $x = 0.5$ means that KRE underestimated by a factor of 2 or more on 20% of the trials. The rightmost point of the KRE plot ($x = 10$, $y = 94\%$) indicates that on 6% of the trials KRE's prediction was more than 10 times the actual number of nodes expanded. By contrast G_KRE has a much larger percentage of highly accurate predictions, with over 99% of its predictions within a factor of two of the actual number of nodes expanded. The figure clearly shows the advantage of using the enhanced G_KRE with an initial search to a radius 10, 90% of the cases were within 10% of the correct number.

IDA* with BPMX

With an inconsistent heuristic, the heuristic value of a child can be much larger than that of the parent. When generating such a child this large heuristic can be propagated to the parent. For example a child heuristic value of 7 can be propagated to the parent and the parent can use the value of 6 even if its own value was much smaller. This propagation technique is called *bidirectional pathmax* (BPMX) (Felner et al. 2005; Zahavi et al. 2007). It was shown to be very effective in pruning subtrees that would otherwise be explored and the number of expanded nodes can be reduced by a factor of up to 18 in some cases.

Our prediction formula G_KRE can be generalized to handle the BPMX method as well. Here we give a sketch of the general idea. (Zahavi et al. 2007) showed that when applying BPMX to inconsistent heuristics the *dynamic branching factor* (the average number of nodes that are actually generated) might be smaller than the brute-force branching factor. Assume that p is a fertile node that was just generated. When BPMX is used, a child c (of p) will be generated only if its preceding siblings and their subtrees will not prune p. We define $p_{bx}(c|context)$ to be the probability that a state p will not be pruned by its first c children or by the subtrees below them. With this new definition we can extend equation 2 to:

$$N_i(s,v) = \sum_{v_p=0}^{d-(i-1)} \sum_{c=1}^{b} N_{i-1}(s,v_p) \cdot p(v|v_p) \cdot p_{bx}(c-1|d, i-1, v_p)$$

The exact development of $p_{bx}(c|context)$ is complicated and we leave the detailed discussion for another paper.

Conclusions and future work

Historically, heuristics were characterized by their average (a single value). KRE introduced the static distribution (a vector of values) and presented their prediction formula. We took this another step to the conditional distribution (at least a 2-dimensional matrix of values) and presented the G_KRE prediction formula. It advances KRE in that it works for any set of start states as well as for inconsistent heuristics. We have also shown how to use it to make a prediction for a single start state.

Future work will address a number of issues. First, a full treatment of BPMX should be given. The conditional probability widens our abilities to characterize heuristics. A deeper study of the attributes of a heuristic given its conditional probability distribution is likely to advance our understanding of heuristics.

Acknowledgments

This research was supported by the Israel Science Foundation (ISF), grant #728/06 to Ariel Felner and by iCORE.

References

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

Felner, A.; Zahavi, U.; Holte, R.; and Schaeffer, J. 2005. Dual lookups in pattern databases. In *IJCAI*, 103–108.

Felner, A.; Korf, R. E.; Meshulam, R.; and Holte, R. C. 2007. Compressed pattern databases. *JAIR* 30:213–247.

Korf, R. E., and Felner, A. 2002. Disjoint pattern database heuristics. *Artificial Intelligence* 134:9–22.

Korf, R. E., and Reid, M. 1998. Complexity analysis of admissible heuristic search. *AAAI* 305–310.

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

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

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

Zahavi, U.; Felner, A.; Holte, R.; and Schaeffer, J. 2006. Dual search in permutation state spaces. In *AAAI*, 1076–1081.

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