Requirements on Heuristic Functions when Using A* in Domains with Transpositions

Nir Pochter and Jeffrey S. Rosenschein
School of Engineering and Computer Science
The Hebrew University of Jerusalem, Israel
{nirp, jeff}@cs.huji.ac.il

Abstract

Heuristic functions play a crucial rule in optimal planning, and the theoretical limitations of algorithms using such functions are therefore of interest. Much work has focused on finding bounds on the behavior of heuristic search algorithms, using heuristics with specific attributes. Recently, it has been shown that in domains that contain transpositions, even heuristics that are intuitively considered to be very good, perform very badly. We show that a large family of non-perfect heuristic functions may perform well in such domains, and demonstrate some properties of a heuristic function that support success in those domains.

Introduction

Many state-of-the-art planners that optimally solve classical planning problems use heuristic functions to guide their search. While planning problems are computationally hard to solve, heuristics are intended to help the planner solve the problem quickly, in practice. The importance of heuristics has encouraged researchers to exert much effort on finding methods that automatically create heuristics for a domain, based on its description (Katz and Domshlak 2008; Helmert, Haslum, and Hoffmann 2008).

The extensive use of heuristic functions also gives rise to questions about the theoretical limits of their use. For example, given a domain, can we always find a heuristic function, which can be calculated efficiently, that will help solve problems quickly? Or given a heuristic function, how good a job will it do in the domain? These questions have more than theoretical importance—if one can prove that it is impossible to efficiently find a good heuristic for a family of domains, then research should focus on other directions.

To analyze the performance of heuristic functions, one usually defines a family of heuristics (generally those which are considered intuitively to be good heuristics), and analyzes their average and worst-case behavior. Two popular families of heuristics are additive heuristics (where the heuristic is accurate up to an additive constant), and relative heuristics (where the heuristic is accurate up to a multiplicative constant).

In previous work, heuristics were generally evaluated using the notion of "accuracy"; that is, given a state, what is the discrepancy between the heuristic's estimate for the state's distance from the goal, and the state's real distance from the goal? In this paper, we will use a different measurement: the chance that the heuristic function gives a precisely correct value for the distance to the goal from a random state. We call this measurement the *Probably Accurate* measurement, or *PA*(h).[1] We will show that in domains that have transpositions, *PA*(h) is highly correlated with performance, and that heuristic functions that are ϵ-$approximate$ with the same ϵ, but have different *PA*(h) values, can behave dramatically differently in such domains, when using the A* algorithm (Hart, Nilsson, and Raphael 1968) .

Related Work

Much research has studied the behavior and complexity of the A* algorithm, using various heuristic functions in domains with certain restrictions.

Pohl (1977) considered heuristic functions that have an error which is bounded by a constant, in which case the A* algorithm expands only a linear number of nodes; he also considered the case of constant relative error. However, his analysis required some conditions on the domain, which are often violated specifically in planning domains. First, his analysis requires that the branching factor be constant. Second, it requires that there is only a single goal state. Finally, it requires that there be no transpositions.

Gaschnig (1977) showed that with the same conditions, A* using a heuristic function with logarithmic absolute error will require expansion of a polynomial number of nodes. Pearl (1984) showed that these results also hold in the average case. Dinh et al. (2007) have extended these results for the case of additive error on the heuristic, with fewer restrictions on the domain (as multiple goals were allowed). However, transpositions were still not considered in their analysis.

Some work has also been done on trying to predict the number of nodes expanded by the IDA* algorithm (Korf 1985). Korf et al. (2001) showed that IDA* expands at most $\sum_{i=0}^{d} N(i)P(d-i)$ nodes when the search is bounded by depth d, where $N(i)$ is the number of nodes at depth i in the search tree, and $P(v)$ is the equilibrium distribution.

Zahavi et al. (2008) extended this work by replacing a static distribution with a conditional distribution (which took into account certain properties of the search space). They

[1] We are using *accurate* here as a synonym for perfect.

derived a formula based on the conditional distribution that also worked with inconsistent heuristics, and using a set of nonrandom start states. The Korf and Zahavi formulas, are limited to IDA*. although (Breyer and Korf 2008) showed empirically that these formulas with minor modifications can be used to predict the performance of A* as well,

Recently, Helmert and Röger (2008) investigated the behavior of A* on domains that contain transpositions, using an *almost perfect* heuristic. An almost perfect heuristic is a heuristic that for each state returns the real distance minus a constant c (i.e., the almost perfect heuristic is $h^* - c$, where h^* is the real distance). Their results showed that A* with an almost perfect heuristic will perform very badly in some domains that contain transpositions, no matter how small c is. We will show that while this is true, a large family of heuristic functions will perform well in such domains—even though those heuristics may not intuitively seem superior to an almost perfect heuristic function, nor surpass it by other measurements (like average error).

Evaluating Heuristics

While the almost perfect heuristic is significant for theoretical analysis, it is not common to find such a heuristic in practice. Heuristic functions generally have different errors in different states. How, then, should we analyze a given heuristic?

As previously mentioned, one way is to bound the error that the heuristic function gives for any state in the search space, either by additive or relative constants. Another way is to use the expected error of the heuristic on each node. In this paper, we focus on the *probably accurate* measurement. We say that a heuristic function h has $PA(h) = p$ if $Pr[h^*(v) - h(v) = 0] = p$ for v which is chosen randomly using the uniform distribution, and also that the probability of v getting a perfect heuristic value is independent of the probability of any other node getting a perfect heuristic value. Under this measurement, one heuristic will be considered better than another if it achieves a perfect estimate in more states. Specifically, the *almost perfect* heuristic has the lowest possible *probably accurate* measurement, since the former does not give a perfect estimate in *any* state. In fact, many heuristic functions that give an additive ϵ-approximate estimate would be dominated, using the *PA*(h) measurement, by heuristic functions that are not additive ϵ-approximate.

Obviously, *PA*(h) is not enough by itself to show that a heuristic is good; in some cases, average error may play a more important role in performance. However, in the next section we will show that in domains with many transpositions, this measurement can be crucial.

Transpositions on Shortest Paths

We now evaluate how many nodes will be expanded by a heuristic with *PA*$(h) = p$ in a search problem with many transpositions. We demonstrate that in domains that have a large number of transpositions, it is important for the heuristic to have a "large enough" p. To show this, we will prove the expected estimate of such a heuristic where the only available paths are transpositions, and compare that with the performance of the almost perfect heuristic.

Our domain will consist of two states, s and t, which are connected via m distinguished separate paths, each of length n. That is, for $i = 1, \ldots, m$:

1. s is connected to $v_{i,1}$
2. For $j = 1, \ldots, n-1$, $v_{i,j}$ is connected to $v_{i,j+1}$
3. $v_{i,n}$ is connected to t.

Our search task will be to find the shortest path from s to t, using the A* algorithm. First, we claim that using an almost perfect heuristic, the number of nodes that the A* algorithm will expand equals the number of states in the search space.

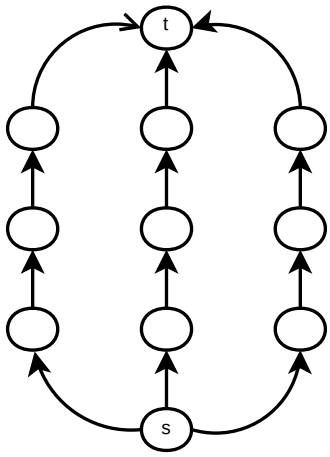

Figure 1: Example of a domain with m shortest paths of length n

Theorem 1. *The A* algorithm using an almost perfect, i.e., $h^* - c$, heuristic will expand $nm + 1$ nodes when searching from s to t on the domain defined above.*

Proof. Let us look at some path p_i. Since the start state is connected to all the paths that lead to the target state, when it is removed from the open list all its children will be added to the open list, which includes one node from p_i. From this stage on, if this node is expanded then its son is also on p_i; therefore, at each stage of the execution of A* we have at least one node that belongs to p_i on the open list, until the algorithm stops. Given that the almost perfect heuristic always estimates the heuristic with a $-c$ constant, the f-value of the node n_i that belongs to p_i and is in the open list is $g(n_i)+h(n_i)$=$g(n_i)+h^*(n_i)-c$=$h^*(s)-c$. Since the f-value of n_i is smaller than the real cost, it has to be expanded before A* terminates. This is true for all $n_i \in p_i$ and for all p_i, so all nodes from all paths have to be expanded, giving us nm nodes, plus the start node, i.e., $nm + 1$ nodes. □

We will now bound the number of nodes using an admissible heuristic function that has *PA*$(h) = p$, starting with the following theorem:

Theorem 2. *When using A* to search from s to t in the domain defined above, and given that two nodes n_i and n_j, from different optimal paths p_i and p_j, are in the open list, and $h^*(n_i) - h(n_i) > h^*(n_j) - h(n_j)$, then n_i will be expanded before n_j.*

Proof.

$$\begin{aligned} f(n_i) &= g(n_i) + h(n_i) \\ &= g(n_i) + h^*(n_i) - (h^*(n_i) - h(n_i)) \\ &= h^*(s) - (h^*(n_i) - h(n_i)) \\ &< h^*(s) - (h^*(n_j) - h(n_j)) \\ &= g(n_j) + h^*(n_j) - (h^*(n_j) - h(n_j)) \\ &= g(n_j) + h(n_j) = f(n_j) \end{aligned}$$

□

Next we will prove that if the open list contains a node each from two paths, where h gives an accurate estimate on both nodes, then no more nodes will be expanded from the path with the lower g-value.

Theorem 3. *Given that two paths p_1 and p_2 are both optimal, and that node n_1 from path p_1 and node n_2 from path p_2 were both added to the open list, and $g(n_2) > g(n_1)$, and $h(n_1) = h^*(n_1)$, then no more nodes will be expanded from path p_1 (when using an implementation of A* that breaks ties by choosing the node with the higher g-value).*

Proof. Since the heuristic is admissible, it cannot overestimate the distance to the goal; therefore, the f-value of n_2 has to be smaller than or equal to the real distance from the start to the target. If it is smaller, then it will be expanded before n_1, as n_1's f-value is the exact distance (since the heuristic there is accurate). If the heuristic is accurate on n_2, its f-value will be equal to n_1's f-value, but it will be expanded first because it has a higher g-value. This process continues on all nodes that lie on p_2 after n_2, until the goal is reached. When the goal is reached, the f-value is accurate, and n_1 will not be expanded because its f-value is not lower than the real distance. Therefore, no more nodes from p_1 will be expanded. □

Now, we will see what the expected number of node expansions is, until a node that has a perfect estimate is found:

Theorem 4. *When searching from s to t on the domain defined above, and given that PA$(h) = p$, the expected number of nodes that will be expanded on a path that has length L, before the expansion of the first node on that path where h provides a perfect heuristic or the end of the path was reached, is $\frac{L(1-p)L+1-(L+1)(1-p)^L+1}{p} + L(1-p)^L$.*

Proof. Let X be a random variable representing the expected number of nodes expanded on a path before either a node on that path that has a perfect heuristic is expanded, or the path reaches to an end. For $i = 0 \dots L-1$, X has the same probability function as a geometric variable with probability p: $Pr(X = i) = (1-p)^{i-1}p$. However, for L, we have to add all the values in which it would take more than L turns until success. We thus get that $E(x) = \sum_{i=0}^{l} p(1-p)^{i-1} * i + \sum_{i=l+1}^{\infty} Lp(1-p)^{i-1}$. We split the expression, and first get that:

$$\begin{aligned} &\sum_{i=0}^{L}(p(1-p)^{i-1})i \\ &= \frac{p}{1-p}\sum_{i=0}^{L}(1-p)^i i \\ &= \frac{p}{1-p}\frac{L(1-p)^{L+2} - (L+1)(1-p)^{L+1} + (1-p)}{(1-(1-p))^2} \\ &= \frac{L(1-p)L + 1 - (L+1)(1-p)^L + 1}{p} \end{aligned}$$

Now we look at the tail:

$$\begin{aligned} &\sum_{i=L+1}^{\infty}(p(1-p)^{i-1}) * L = Lp\sum_{i=L+1}^{\infty}((1-p)^{i-1}) \\ &= Lp\sum_{i=0}^{\infty}((1-p)^{i+L} = Lp(1-p)^L\sum_{i=0}^{\infty}((1-p)^i) \\ &= Lp(1-p)^L\frac{1}{1-(1-p)} = L(1-p)^L \end{aligned}$$

This means that all but one path will stop expanding nodes after an expected number of $e(n,p)$ nodes, and the one last path will expand n nodes. We can look at each path as a random variable with expected value of $e(n,p)$. Taking m such variables plus the start node would give $me(n,p)+1$. However, the leading path will continue until reaching the end, so all of its n nodes will be expanded. This gives $(m-1)e(n,p)+1+n$ nodes. This is, however, an upper bound, as the path that will continue until its end was not chosen randomly - it was the leading path, therefor the expected number of nodes that will be expanded from all other paths is a number that is smaller than $e(n,p)$. This gives us an upper bound of $(m-1)e(n,p)+n+1$.

Adding the two together we get:

$E(X) = \frac{L(1-p)L+1-(L+1)(1-p)^L+1}{p} + L(1-p)^L$ □

To more easily understand the meaning of this expression, one can clearly see that it is smaller than $\frac{1}{p}$. We will use this expression often in the rest of the paper; it is therefore convenient to frame it as a function of L and p:

$e(L,p) \equiv \frac{L(1-p)L+1-(L+1)(1-p)^L+1}{p} + L(1-p)^L$.

Theorem 5. *When searching from s to t on the domain defined above using A*, and given that there are m paths with length n that are optimal, and PA$(h) = p$, the expected number of nodes that will be expanded is smaller than $(m-1)e(n,p)+n+1$.*

Proof. After the start node is expanded from the open list, all of its sons enter the open list. All paths have a state connected to the start state, and therefore all paths will have a node in the open list until the target is reached through them. Let us look at path p_i. According to Theorem 4, we know that the expected number of node expansions before reaching a node $n_i \in p_i$ that has $h(n_i) = h^*(n_i)$ is $e(n,p)$. After

this happens, according to Theorem 2, no more nodes will be expanded from this path until all other paths have reached a node with a perfect estimate (this includes reaching the target). When this happens, there are two options:

1. There exists a path p_j, such that the node n_j that is on p_j and is on the open list, has a higher g-value. According to Theorem 3, no additional nodes will be expanded from p_i.
2. p_i is the leading path (i.e., has the highest g-value), and will continue until the end.[2]

This means that all but one path will stop expanding nodes after an expected number of $e(n,p)$ nodes, and the one last path will expand n nodes. We can look at each path as a random variable with expected value of $e(n,p)$. Taking m such variables plus the start node would give $me(n,p)+1$. However, the leading path will continue until reaching the end, so all of its n nodes will be expanded. This gives $(m-1)e(n,p)+1+n$ nodes. This is, however, an upper bound, as the path that will continue until its end was not chosen randomly—it was the leading path. Therefore, the expected number of nodes that will be expanded from all other paths is a number that is smaller than $e(n,p)$. This gives us an upper bound of $(m-1)e(n,p)+n+1$.

□

Transpositions on Non-Shortest Paths

So far, we have only considered the penalty taken on alternative shortest paths. Now, we will look at a broader picture, namely the penalty paid in transposition on off-track paths—paths that are not shortest paths to the target.

First, consider the following scenario. The state space is composed of a start state s and a goal state t, and there is one shortest path from s to t, with length n. There are also m disjoint paths from s to t with length $n+1$ (as in the previous domain, these lengths do not include s and t). With good tie-breaking and some luck, the almost perfect heuristic with $c=1$ will expand only $n+2$ nodes (this will happen if the search follows the shortest path until it reaches the target). In the case of $c=2$, even luck will not help: using a similar argument as before, one can see that at least n nodes from each non-shortest path will have to be expanded before A* terminates. Again, using a heuristic with *PA*$(h)=p$ will give better results on the expected number of nodes expanded from these paths, even if we do not know the degree of error it gives on states that do not get a perfect estimate. In fact, a weaker demand will suffice here: instead of $Pr[h^*(v)-h(v))=0]=p$, we will only need $Pr[h^*(s)-h(s)\leq 1]=p$ to get these results.

This is true for the transpositions of paths that are longer than the shortest path by any length. Consider a domain with two nodes, s and t, that are connected by distinct paths. There is one shortest path, with length n. Also, for $i=1,\ldots,L$ there are m_i distinct paths with length $n+i$. The task is to find the shortest path from s to t. We will compare a variant of the almost perfect heuristic to a heuristic function that satisfies $Pr[h^*(v)-h(v)<i]=p$ for each v that lies on a path which is longer by i than the shortest path. We call this the *Probably Accurate Enough* measurement with parameter p, or *PAE*$(h)=p$.

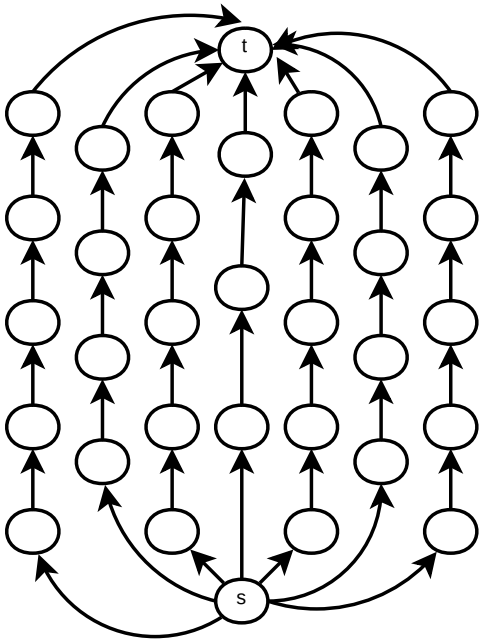

Figure 2: Example of a domain with one shortest path and many non-shortest paths

Theorem 6. *When searching from s to t in the domain defined above using A* and a heuristic function that for each state $state$ that lies on a path that is of length $n+i+1$ gives a value of $h^*(state)-(i+1)$, the number of nodes expanded will be at least $\sum_{i=1}^{L} nm_i+n+2$.*[3]

Proof. Since the start state is connected to all paths, at each stage at least one node from each path will be in the open list. Let p_i be a path with length $n+i$. Now let us look at a node n_i that is on path p_i, and is on the open list.

$$\begin{aligned} f(n_i) =& g(n_i)+h(n_i) \\ &= g(n_i)+h^*(n_i)-(h^*(n_i)-h(n_i)) \\ &= n+i-(i+1) \\ &< h^*(s). \end{aligned}$$

This is true for at least the first n nodes in p_i; therefore, at least the first n nodes from p_i will be expanded. This is true for all paths. In addition, s need to be expanded once, giving a total of at least: $\sum_{i=1}^{L} nm_i+n+1$. □

Theorem 7. *Let h be a heuristic function with PAE$(h)=p$. The expected number of nodes that will be expanded is smaller than $\sum_{i=1}^{L}(e(m_i,p))+n+1$.*

[2] We assume that there is arbitrary tie-breaking if there are several leading paths. After the first tie-breaking, there is only one leading path that will continue until it reaches the target.

[3] An almost perfect heuristic would appear to give negative h-values to states that are closer than c from the target state. Our statement of the theorem is adjusted to avoid this apparent problem.

Proof. s is connected to all paths; therefore, at all times there will be one node from each path on the open list. For each path that is not a shortest path to t, all nodes will be expanded until a node with f-value larger than $h^*(s)$ enters the open list. Let p_i be a path with length $n+i$. The f-value of a node n_i that belongs to p_i is:

$$\begin{aligned} g(n_i) + h(n_i) &= g(n_i) + h^*(n_i) - (h^*(n_i) - h(n_i)) \\ &= n + i - (h^*(n_i) - h(n_i)) \end{aligned}$$

for nodes with $g(n_i) \leq n$. n_i will be expanded if its f-value is smaller than $h^*(s)$, and may or may not be expanded if its f-value equals $h^*(s)$ (depending on tie-breaking). $p[f(n_i) > h^*(s)] = p[h^*(n_i) - h(n_i) < i]$; therefore, since $PAE(h) = p$, the probability that n_i will not be expanded is at least p, meaning that the expected number of nodes that will be expanded from path p_i is at most $e(m_i, p)$. From linearity of expectation we get that the total number of nodes that will be expanded from paths with length $n + i$ is therefore $e(m_i, p)$, giving a total of at most $\sum_{i=1}^{L}(e(m_i, p)) + n + 1$ nodes. □

Discussion and Future Work

We have shown that the A* algorithm can do well in domains that have transpositions, depending on the percentage of states for which its heuristic function gives accurate values, that is, $PA(h)$. Our theoretical result uses this measurement to bound the number of nodes that the A* algorithm will explore on average.

These results support the argument, that when developing heuristics for such domains, or creating heuristics that are domain independent, $PA(h)$ should be taken into account, and a heuristic with a favorable $PA(h)$ might be preferred, even if that heuristic provides values of inferior accuracy on average.

While it is possible to create a heuristic which is *PA(h)* (a deterministic or probabilistic one), this heuristic will be inconsistent, as the error of each state must be independent of the error of its neighboring states. Even though most heuristics used in planning today are consistent, we believe that a heuristic that is "close" to being probably accurate might still have an advantage; this work can lead to a better understanding of why certain heuristics do well in some domains (or families of states within some domains), by analyzing the number of states in which they give accurate results. In future work, we hope to examine this assumption empirically. Another interesting direction for future work is to drop the independence condition, and instead take the conditional expectation into account.

In our proofs we also assumed that the paths only meet in the last state and are distinct before that point. While this is not usually the case in planning, this scenario is probably a sub-problem in many cases, in which case the results apply for the sub-cases (for example, a search problem that connects a state s to state m via n different shortest paths, and then connects m to t via n distinct shortest paths). It is therefore our conjecture that probably accurate heuristics can do well in practice in planning domains with transpositions, even if those domains do not exhibit the exact structure posited in our theorems. In future work, we hope to experimentally support this conjecture.

References

Breyer, T. M., and Korf, R. E. 2008. Recent results from analyzing the performance of heuristic search. In *The First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR-08)*.

Dinh, H. T.; Russell, A.; and Su, Y. 2007. On the value of good advice: The complexity of A* search with accurate heuristics. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI 2007)*, 1140–1145.

Gaschnig, J. 1977. Exactly how good are heuristics?: Toward a realistic predictive theory of best-first search. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI 1977)*, 434–441.

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

Helmert, M., and Röger, G. 2008. How good is almost perfect? In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, 944–949.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2008. Explicit-state abstraction: A new method for generating heuristic functions. In Fox, D., and Gomes, C. P., eds., *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, 1547–1550. AAAI Press.

Katz, M., and Domshlak, C. 2008. Structural patterns heuristics via fork decomposition. In Rintanen, J.; Nebel, B.; Beck, J. C.; and Hansen, E. A., eds., *Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 2008)*, 182–189. AAAI.

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

Korf, R. E. 1985. Iterative-deepening-A*: An optimal admissible tree search. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI 1985)*, 1034–1036.

Pearl, J. 1984. *Heuristics: intelligent search strategies for computer problem solving*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Pohl, I. 1977. Practical and theoretical considerations in heuristic search algorithms. In Elcock, E. W., and Michie, D., eds., *Machine Intelligence 8*. Ellis Horwood Ltd. and John Wiley and Sons. 55–72.

Zahavi, U.; Felner, A.; Burch, N.; and Holte, R. C. 2008. Predicting the performance of IDA* with conditional distributions. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, 381–386.