Investigating Exploration Techniques in Anytime Heuristic Search

Dawson Brown (500780579)
dawson.brown@ryerson.ca

Abstract

Knowledge free, randomized exploration techniques in heuristic search algorithms have proved very useful. Randomized exploration seeks to avoid large plateaus and even improve the complexity and runtime of a heuristic search algorithm by devoting some time to exploration. The hope is that by improving the coverage of the state space during search, a solution might be yielded faster. Anytime Heuristic Search employs both an admissible and inadmissible heuristic. It uses the inadmissible heuristic to arrive at solutions quickly, while the admissible heuristic is used to prune the open list and slowly converge on the optimal solution; in this way it gets the best of both worlds–the optimal solution in the end, but fast suboptimal solutions in the mean time. In this study, I seek to examine the benefits of employing randomized exploration in an Anytime context. For this, I will use Anytime Weight A* as the baseline and create two exploration techniques giving two Exploration Anytime Algorithms: $\epsilon-$AWA* and $\alpha\beta-$AWA*. Several experiments using the sliding tile puzzle are used to compare the three algorithms. These experiments convincly demonstrate that using knowledge-free randomized exploration can be very effective in anytime search algorithms and more investigation into the combination of randomized exploration and anytime search is needed.

Introduction

Background

For this study, I drew on Anytime Weighted A* (AWA*) as well as concepts in knowledge free exploration techniques.

AWA*

Hansen and Zhou introduced a method for converting Weighted A* into an Anytime algorithm giving Anytime Weighted A* (Hansen and Zhou 2007). To do so, three things are required:

1. An inadmissible evaluation function for selecting nodes for expansion. In the case of AWA* that's a weighted heuristic (Hansen and Zhou 2007).
2. The search must continue after a (likely suboptimal) solution is so improved solutions can be found (Hansen and Zhou 2007).
3. An admissible evaluation function which serves as a lower bound, along with the upper bound being the cost of the best solution found so for, both of which can be used to prune the open list until it's empty at which point the last found solution must be optimal (Hansen and Zhou 2007).

Algorithm 1 gives the details of the 'anytime' part of AWA*–the steps omitted at line 14 are those that are part of normal WA*.

Algorithm 1: AWA*

Require: $f'(n) = g(n) + wh(n)$, $f(n) = g(n) + h(n)$
1: $g(init) = 0$
2: $OPEN \leftarrow \{init\}, CLOSED \leftarrow \emptyset$
3: $best \leftarrow$ None, $f(best) \leftarrow \infty$
4: **while** $OPEN \neq \emptyset$ **do**
5: $n \leftarrow \arg\min_{n' \in OPEN} f'(n')$
6: **if** $f(n) < f(best)$ **then**
7: $CLOSED \leftarrow CLOSED \cup \{n\}$
8: **for** $n_i \in Succ(n)$ **if** $f(n) < f(best)$ **do**
9: **if** n_i is goal **then**
10: $f(n_i) \leftarrow g(n_i) + c(n, n_i)$
11: $best \leftarrow n_i$
12: Continue
13: **end if**
14: ⋮
15: **end for**
16: **end if**
17: **end while**
18: **return** $best$

This approach is largely agnostic to the internals of the search being done so long as nodes are explored in some best first ordering that allows for open list pruning (Hansen and Zhou 2007). This makes it possible to include randomized exploration techniques in AWA*.

Exploration

Knowledge-free random exploration has shown promise in satisficing search, in particular GBFS (Valenzano et al. 2014). Here, Valenzano *et al.* describe ϵ-GBFS in which with probability $(1 - \epsilon)$ the node with the smallest heuris-

tic value is chosen for expansion, otherwise, a node is chosen uniformly at random from the open list (Valenzano et al. 2014). In their experimentation, adding ϵ-greedy node selection added a lot of value in several domains (Valenzano et al. 2014).

Despite showing improvement in some domains, and its relative simplicity, strictly random exploration (with probability ϵ) runs into problems when large local minima or plateaus are encountered. This is because the open list will come to be dominated by that plateau and so sampling randomly will still expand a node on the plateau which is in part what exploration is trying to avoid (Xie et al. 2014)(Cohen, Valenzano, and McIlraith 2021). To address this, Xie *et al.* proposed type based exploration in which nodes are bucketed by f-cost, and then a bucket is sampled uniformly at random and then a node within that bucket is sampled (Xie et al. 2014). This approach achieves a much better spread over the state space than simple ϵ random sampling (Xie et al. 2014).

Cohen *et al.* show that WA* can also fall prey to these large plateaus (termed uninformed heuristic regions) and Type-based exploration can alleviate this (Cohen, Valenzano, and McIlraith 2021). To this end, they merge Type-Based exploration with WA* to give Type-WA* (Cohen, Valenzano, and McIlraith 2021).

Exploration in AWA*

I implemented two exploration techniques within the framework of AWA*. The general idea is that for each node expansion, there is some probability ϵ that instead of expanding the best node according to the weighted evaluation function $h(n)$, some other node is 'randomly' chosen from the open list for expansion. The two proposed techniques are ϵ-AWA* and $\alpha\beta$-AWA*.

ϵ-AWA*

ϵ-AWA* employs the simplest technique exploration taken from ϵ-GBFS (Valenzano and Xie 2016)(Valenzano et al. 2014). With this technique, with probability ϵ, a node is chosen uniformly at random from the open list.

In an anytime context, this simply means that the node selection on Line 5 in Algorithm 1 would be replaced with the procedure outlined in Algorithm 2, where `randomSample` samples uniformly at random from the open list, and `randrange` samples uniformly at random from the given range.

Algorithm 2: $\epsilon-$AWA* node selection

$y \leftarrow$ randrange(0,1)
if $y \leq \epsilon$ **then**
  return randomSample($OPEN$)
else
  return $\arg\min_{x \in OPEN} f'(x)$
end if

$\alpha\beta$-AWA*

The sampling technique in $\alpha\beta$-AWA* is a bit more involved than the on in ϵ-AWA* and is meant to address some of the shortcomings in ϵ random sampling. Xie's Type-Based sampling, along with its use in WA* by Cohen *et al.*, serves as an inspiration for the $\alpha\beta$ sampling devised for $\alpha\beta$-AWA*– where the aim is to achieve a good spread over the state space during sampling.

Simply put, $\alpha\beta$-AWA* samples from the open list non-uniformly. It is assumed that the open list is stored as a minimum heap; then the sampling is done in two steps: first sample a row from the heap non-uniformly, second sample a node uniformly from that row. The non-uniform sampling of a row is done according to a beta distribution with parameters α and β (hence $\alpha\beta$-AWA*). Sampling a row can be thought of as evenly spacing the rows between 0 and 1 along the x-axis and then sampling one according to the probability density above it. The intuition behind this is that, while heaps don't make strong guarantees about the order of nodes, they do guarantee that a parent is larger than both its children. This means that, in general, sampling from higher rows will yield smaller f-cost nodes than sampling from lower rows. And so, if you bias your row sampling to middling or lower rows you should expect to see a better spread across the state space. Furthermore, I think it's a fair assumption that those nodes with smaller f-costs are likely to be expanded sooner in the usual way (ie by the selection done on Line 5 in Algorithm 1) regardless of plateaus, and so exploration should focus on other nodes–nodes that aren't close to expansion. This sampling technique is also quite simple to implement, unlike Type-Based exploration which is quite complex.

Algorithm 3: $\alpha\beta-$AWA* node selection

Require: $\gamma \leftarrow beta(\alpha, \beta)$
$y \leftarrow$ randrange(0,1)
if $y \leq epsilon$ **then**
  $row \leftarrow$ sampleRow($OPEN$, γ)
  $start \leftarrow 2^{row} - 1$
  $end \leftarrow 2 \cdot start$
  return randomSample($OPEN[start : end]$)
else
  return $\arg\min_{x \in OPEN} f'(x)$
end if

In an anytime context, the procedure outlined in Algorithm 3 can be invoked instead of Line 5 in Algorithm 1. For the rest of this paper the beta distribution in $\alpha\beta-$AWA* will be such that $\alpha = 5$ and $\beta = 0.6$ as seen in Figure 1 which very heavily favours lower rows in the heap.

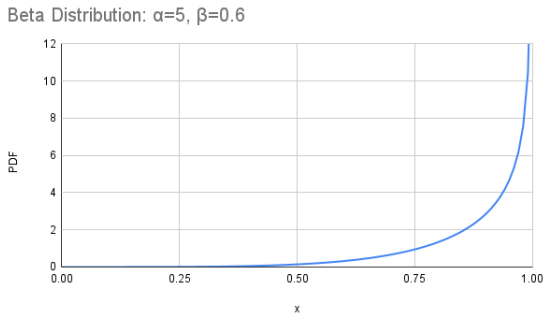

Figure 1: Beta Distribution used in $\alpha\beta$-AWA*

Evaluation

In order to evaluate the usefulness of exploration in AWA* a number of experiments were conducted on the sliding tile puzzle. All experiments are run on an 8-core 1.70GHz machine with 16 gigabytes of memory. For all experiments, the two exploration techniques will be compared to a standard AWA* implementation which will serve as the benchmark. One hundred instances of the unit cost 12-puzzle were used to compare the three algorithms.

The primary focus was on the Unit Cost Sliding Tile puzzle using the Manhattan Distance Heuristic. Various weights are used across the three algorithms and two different values of ϵ, 0.1 and 0.3 (Valenzano et al. 2014), are used for $\epsilon-$AWA* and $\alpha\beta-$AWA*. Depending on the value of ϵ, $\epsilon-$AWA* and $\alpha\beta-$AWA* will be abbreviated to 0.1-ϵ, 0.3-ϵ, 0.1-$\alpha\beta$, and 0.3-$\alpha\beta$ respectively. The beta distribution in all $\alpha\beta-$AWA* instances is fixed at $\alpha = 5$ and $\beta = 0.6$.

There are also two small extensions. In the first, the Inverse Sliding Tile Puzzle is briefly investigated using the Manhattan Distance Heuristic. In these experiments, the value of ϵ is fixed at 0.3, and only 3 weights are considered: 1.3, 5, and 10. The Inverse Sliding Tile Puzzle was chosen because it tends to be harder than the Unit Cost variant. The second extension involves degrading the heuristic and using the Correct Tile Placement Heuristic. This was done to investigate how exploration may help or hinder with a highly uninformed heuristic.

Unit Cost Tile Puzzle

In general, it was found that adding exploration to AWA* for the sliding tile puzzle proved to be very effective. Both $\epsilon-$AWA* and $\alpha\beta-$AWA* required fewer incumbent solutions to zero in on the optimal solution, as summarized in Table 1.

In all cases, except for weight=1.3, 0.1-$\epsilon-$AWA* required the fewest incumbent solutions before finding the optimal. However, both exploration were consistently better than standard AWA* by this metric.

To gauge the complexity of each algorithm the number expanded nodes and the total number of generated nodes are considered. By this metric exploration shows a huge improvement over standard AWA*, as shown in Figure 2.

Average Number of Incumbent Solutions					
Weight	AWA*	0.1-ϵ	0.3-ϵ	0.1-$\alpha\beta$	0.3-$\alpha\beta$
1.3	**2.24**	2.25	2.28	2.26	2.28
2	3.91	4.09	**3.75**	3.94	3.86
3	7.07	6.69	**6.07**	6.54	6.19
5	11.15	10.37	**9.28**	10.05	9.65
10	15.49	15.03	**13.27**	14.56	13.74

Table 1: Incumbent Solutions

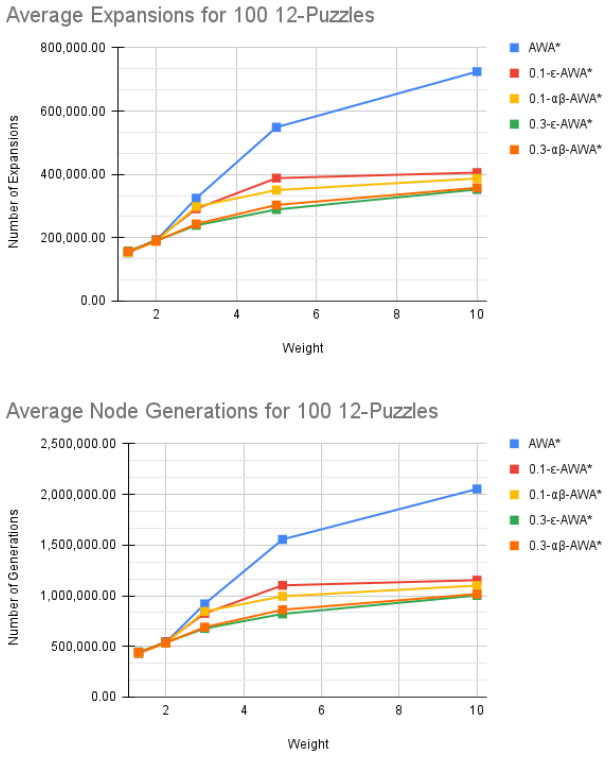

Figure 2: Complexity with Manhattan Distance Heuristic.

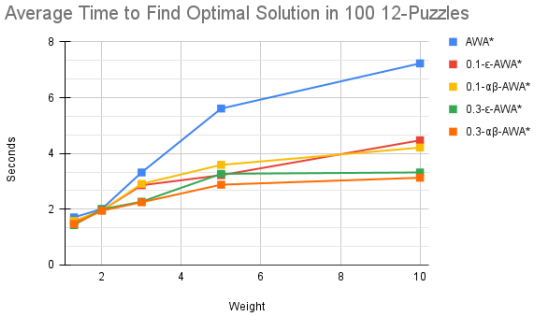

Figure 3: Average Time to Find Optimal Solution

Standard AWA* requires close to 800,000 expansions in the case of a weight of 10, while all four exploration algorithms require in the neighbourhood of 400,000. AWA* seems to be far more impacted by increasing the weight, and thereby degrading the accuracy of the evaluation function, than the ϵ and $\alpha\beta$ variants–which both require a similar number of expansions for a weight of 5 and a weight of 10. Perhaps most surprising is that setting ϵ to 0.3 gets the best results.

Adding to this, exploration in anytime sees an analogous improvement in its runtime when compared to standard AWA*, as seen in Figure 3. This is not guaranteed as it could be the case that the improved node selection procedure takes enough time so as to offset the time savings of expanding fewer nodes. In the cases of $\epsilon-$AWA* and $\alpha\beta-$AWA*, the node selection procedure is sufficiently simple that the expansion savings result in (proportionally) equal time savings.

To summarize the above results, it's useful to look at how quickly each algorithm converges on the optimal solution over time and over number of expansions. Given both the ϵ and $\alpha\beta$ variants required fewer incumbent solutions, fewer expansions to get those solutions, and took less time, it should be the case that they approach the optimal solution much faster than AWA*, which is precisely what we see in Figure 4.

Here, solution quality is $\frac{incumbent-cost}{optimal-cost}$ for a given problem and the solution qualities are averaged across all problem instances. For expansion convergence, the incumbent cost was polled every 50,000 expansions; for the time convergence the incumbent cost was polled every one second (notice the log x-axis in both charts). In this, only a weight of 10 and an epsilon value of 0.3 were considered.

Inverse Cost Tile Puzzle

Degraded Heuristic in Unit Cost Sliding Tiles

Given the promising results above, it would be expected that when the heuristic gets degraded, exploration should help even more as degraded heuristics result in more plateaus and less confidence about the 'best' node in the open list. However, in looking at the summary evaluation of solution quality over time and expansions using the Correct Tile Place-

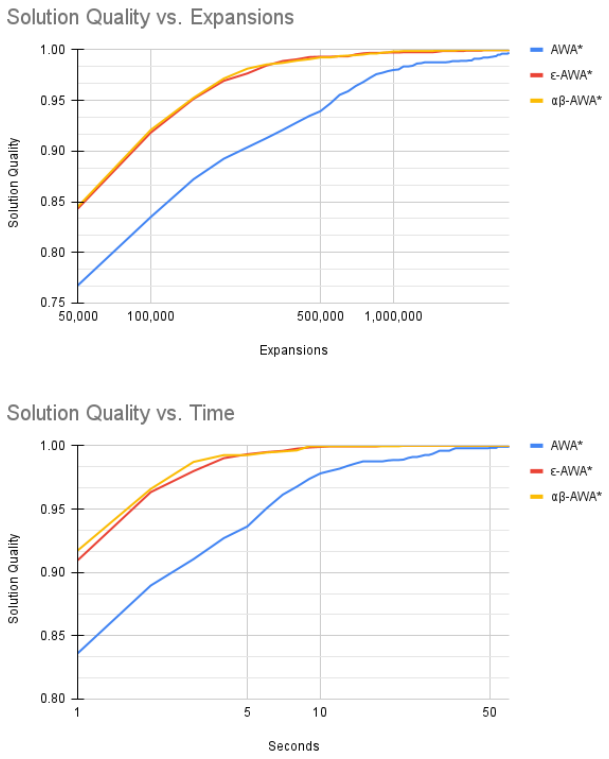

Figure 4: Quality Convergence with Manhattan Distance Heuristic.

Average Number of Incumbent Solutions			
Weight	AWA*	0.3-ϵ-AWA*	0.3-$\alpha\beta$-AWA*
1.3	3.14	3.13	**2.97**
5	21.91	15.66	**15.41**
10	30.8	**20.4**	21.26

Table 2: Incumbent Solutions

Average Number of Seconds to Find Optimal			
Weight	AWA*	0.3-ϵ-AWA*	0.3-$\alpha\beta$-AWA*
1.3	2.25	**2.15**	**2.15**
5	16.38	6.65	**6.34**
10	21.57	**7.3**	8.09

Table 3: Runtime

Average Number of Expansions			
Weight	AWA*	0.3-ϵ-AWA*	0.3-$\alpha\beta$-AWA*
1.3	**186,038.15**	211,198.92	206,175.07
5	1,408,646.74	622,837.79	**587,692.64**
10	1,901,236.58	**709,049.28**	769,546.95

Table 4: Expansions

Average Number of Node Generations			
Weight	AWA*	0.3-ϵ-AWA*	0.3-$\alpha\beta$-AWA*
1.3	**529,095.35**	603,706.60	589,437.60
5	4,003,219.50	1,780,788.61	**1,680,521.61**
10	5,401,559.63	**2,027,138.81**	2,200,575.94

Table 5: Generations

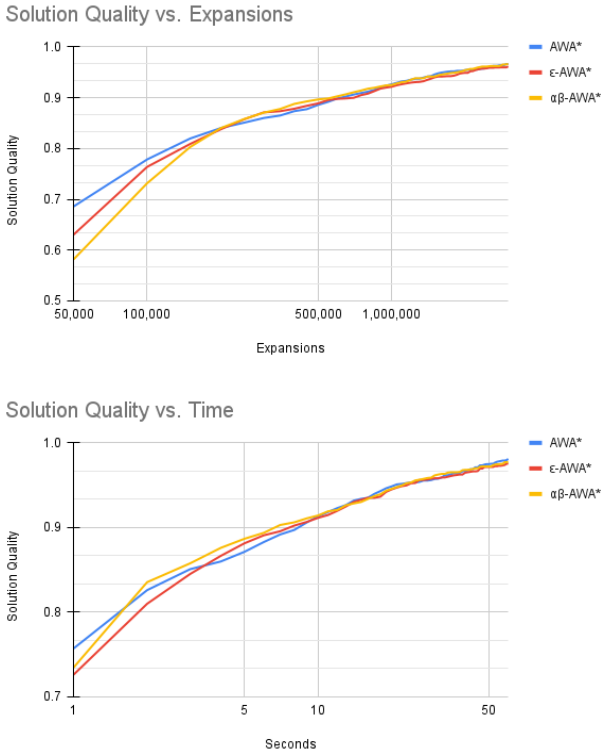

Figure 5: Quality Convergence with Correct Tile Placement Heuristic.

ment heuristic, this did not turn out to be case, as seen in Figure 5.

Here, again, only a weight of 10 and an epsilon value of 0.3 were considered. For expansions, each algorithm was limited to 3,000,000 expansions, and for time each algorithm was limited to one minute.

While in the beginning AWA* actually has the best incumbent solutions, they all more or less approach the optimal solution at the same rate. This seems to simply demonstrate just how bad the Correct Placement heuristic is. I think what is going on here is that the Correct Placement heuristic is so bad it affectively adds and pulls nodes from the open list at random and so adding an exploration technique in no way helps since choosing the 'best' node is also, basically, exploratory.

Discussion and Conclusion

While the results above are promising, the experimentation done was very limited in its scope and left a lot to be desired.

For starters, very few domains were investigated. Just investigating the tile puzzle makes it difficult to make any general statements about these results.

The investigation into $\alpha\beta$-AWA* was incomplete because the beta distribution was fixed for all experiments. Discovering any merits or shortcomings of $\alpha\beta$-AWA* would require a whole set of experiments examining different distributions across various domains. Based on these early results I'd say $\alpha\beta$-AWA* looks interesting, but the fact it was not convincingly better than random sampling means more experimentation needs to be done.

There are many other exploration techniques that could be evaluated in an anytime context. Type-Based exploration, which appears to perform very well in many domains would be an obvious starting point.

Ultimately, I think this was a useful pilot study into using randomized exploration in an anytime search algorithm, but still more investigation is required before anything definitive is said.

References

Cohen, E.; Valenzano, R.; and McIlraith, S. A. 2021. Type-WA*: Using Exploration in Bounded Suboptimal Planning. In *IJCAI*.

Hansen, E. A.; and Zhou, R. 2007. Anytime heuristic search. *Journal of Artificial Intelligence Research*, 28: 267–297.

Valenzano, R.; and Xie, F. 2016. On the completeness of best-first search variants that use random exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Valenzano, R. A.; Sturtevant, N. R.; Schaeffer, J.; and Xie, F. 2014. A comparison of knowledge-based GBFS enhancements and knowledge-free exploration. In *Twenty-Fourth International Conference on Automated Planning and Scheduling*.

Xie, F.; Müller, M.; Holte, R.; and Imai, T. 2014. Type-based exploration with multiple search queues for satisficing planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.