# Investigating Exploration Techniques in Anytime Heuristic Search

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

**Abstract**

My Abstract

## Introduction

Evaluation Metrics:

1. Total Stored nodes (Hansen and Zhou 2007)
2. Total Expanded nodes (Hansen and Zhou 2007)
3. Solution quality at fixed CPU intervals (Thayer, Benton, and Helmert 2012)
4. Average time between solutions (Thayer, Benton, and Helmert 2012)
5. Average number of solutions found before optimal solution was found
6. Average time taken to find optimal solution
7. Lower bound on optimal solution at fixed CPU intervals

Parameters:

1. Weights: 1.3, 1.5, 2 (Hansen and Zhou 2007) (1.3 performed best, so maybe just do that?)
2. Epsilon: 0.1, 0.2, 0.3 (Valenzano et al. 2014)
3. Unit cost and inverse cost

## Background: AWA*

## 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 $\epsilon$ 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 $\epsilon$-AWA$^*$ and $\alpha\beta$-AWA$^*$.

### $\epsilon$-AWA$^*$

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



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 then**
10: $f(n_i) \leftarrow g(n_i) + c(n, n_i)$
11: $best \leftarrow n_i$
12: Continue
13: **end if**
14: $\vdots$
15: **end for**
16: **end if**
17: **end while**
18: **return** $best$

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 $\epsilon$-AWA$^*$ and is meant to address some of the shortcomings in $\epsilon$ random sampling. Strictly random explo-

ration (with probability $\epsilon$) runs into problems when large local minima or plateaus are encountered 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 precisely what exploration is trying (in part) 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 $\epsilon$ random sampling (Xie et al. 2014) and serves as an inspiration for the $\alpha\beta$ sampling devised for $\alpha\beta$-AWA$^*$.

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 $\alpha$ and $\beta$ (hence $\alpha\beta$-AWA$^*$). Sampling works by evenly spacing the row indices 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($OPEN$, $\gamma$)
    $start \leftarrow 2^{row} - 1$
    $end \leftarrow 2 \cdot start$
    **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.

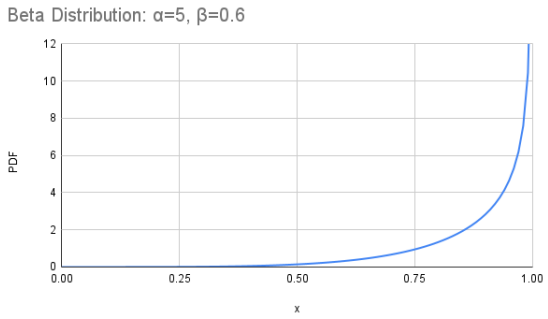


Figure 1: Beta Distribution used in $\alpha\beta$-AWA*

## Evaluation

In order to evaluate the usefulness of exploration in AWA$^*$ a number of experiments were conducted on 2 problem domains–the unit-cost and the inverse-cost sliding tile puzzles. In each domain, multiple weights and epsilon values will be used to parameterize the three algorithms. For the weights, 1.3, 2, and 5, will be used in order to see how weighing the heuristic more or less impacts the search. For $\epsilon-$AWA$^*$ and $\alpha\beta-$AWA$^*$, 0.1 and 0.3 will be used as the $\epsilon$ value in order to see how more or less random exploration impacts the search.

Degraded heuristic.

Summarize architecture.

### Unit-Cost Tile Puzzle

**Degraded Heuristic**

### Inverse-Cost Tile Puzzle

**Degraded Heuristic**

## Discussion and Conclusion

## 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.

Thayer, J.; Benton, J.; and Helmert, M. 2012. Better parameter-free anytime search by minimizing time between solutions. In *International Symposium on Combinatorial Search*, volume 3.

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.