Suboptimal Search with Dynamic Distribution of Suboptimality (Extended Abstract)

Mohammadreza Hami[1], **Nathan R. Sturtevant**[1, 2]

[1] University of Alberta
[2]Alberta Machine Intelligence Institute (Amii)
hami@ualberta.ca, nathanst@ualberta.ca

Abstract

In bounded-suboptimal heuristic search, the aim is to find a solution path within a given bound as quickly as possible, which is crucial when computational resources are limited. Recent research has demonstrated Weighted A* variants such as XDP that find bounded suboptimal solutions without needing to perform state re-expansions; they work by shifting where the suboptimality in the search is allowed. However, the suboptimality distribution is fixed before the search begins. This abstract describes Dynamic Suboptimality Weighted A* (DSWA*), an algorithm introduced at AAAI 2025 that allows suboptimality to be dynamically distributed at runtime based on the properties of the search.

Introduction

This abstract describes the ideas behind Dynamic Suboptimality Weighted A* (DSWA*) (Hami and Sturtevant 2025), a recent best-first search algorithm that can explicitly reasons about where suboptimality will be useful during a search, and re-distribute suboptimality to improve performance. The goal of this abstract is to describe the high-level ideas behind the algorithm. The technical details with additional experimental results can be found elsewhere (Hami and Sturtevant 2025; Hami 2025).

Background

We begin by covering a number of key ideas that DSWA* builds upon.

Problem Definition

DSWA* is designed for bounded suboptimal search. In particular, a bounded suboptimal search (BSS) problem P is characterized by a 5-tuple $P = \{G, start, goal, h, w\}$. The state space, G, is a finite graph that contains states and edges. Each edge $(u, v) \in G$ has a non-negative cost of $c(u, v)$. A path in G is a finite sequence $U = (u_0, ..., u_n)$ of states in G where (u_i, u_{i+1}) is an edge in G for $0 \leq i < n$. The cost of the least cost path from state u to state v in graph G is denoted as $d(u, v)$. The optimal solution cost denoted as C^*, is equal to $d(start, goal)$, the cost of the least cost path from $start$ state to the $goal$. The heuristic function, h, estimates the distance between any state and the $goal$. h is assumed to be strongly consistent, in that $|h(n) - h(m)| \leq d(n, m)$ holds for all m and n. The BSS problem is satisficing: return any path U such that $c(U) = \sum_i c(u_i, u_{i+1}) \leq w \cdot d(start, goal)$.

Priority Functions for Best-First Search

Recent work has studied priority functions for best-first search that do not perform node re-expansions (Chen and Sturtevant 2019, 2021). The work has developed a variety of new priority functions that can be used with best-first search guided by $f(n) = \Phi(h(n), g(n)$, where Φ must meet necessary and sufficient conditions to avoid re-expansions. Two example Φ functions are Φ_{XDP} (convex-downward parabola) and Φ_{PWXD} (piece-wise convex-downward).

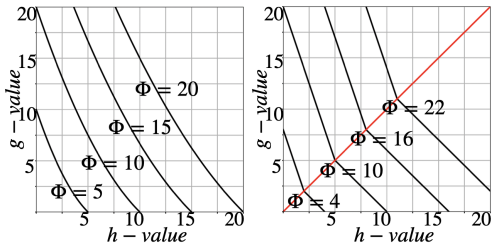
$$\Phi_{\text{XDP}}(h, g) = \frac{1}{2w}[g + (2w - 1)h + \sqrt{(g - h)^2 + 4whg}]$$

$$\Phi_{\text{PWXD}}(h, g) = \begin{cases} g + h & \frac{g}{h} < 1 \\ \frac{1}{w}(g + (2w - 1)h) & \frac{g}{h} \geq 1 \end{cases}$$

Φ describes a three dimensional surface. Every $h(n)$ and $g(n)$ pair have a corresponding priority. Because this surface is difficult to visualize, in Figure 1 we plot isolines for XDP and PWXD - all of the states which have the same priority. The key insight gained from these plots is that the slope of the isoline corresponds to the suboptimality in that part of the state space. Both XDP and PWXD have slope -1 near the h-axis, meaning that the search is optimal when the g-cost is low. They have slope $-(2w - 1)$ near the g-axis, meaning they are allowed to have suboptimality of $2w - 1$ as they approach the goal (with $h = 0$).

These priority functions allow a designer to choose *a priori* where a search should allow suboptimality. PWXD, for example, effectively runs A* to start (no suboptimality), so that it can effectively run Weighted A* with a weight of $2w-1$ for the remainder of the search. This works very well on very hard problems, but on easy problems this approach doesn't work well. For instance, on random maps with relatively few obstacles it is costly to go around obstacles early in the search, where using a weight of w makes it easy. Instead of fixing the regions of higher or lower suboptimality,

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.



(a) XDP (b) PWXD

Figure 1: Isolines for $w = 2$ for (a) XDP and (b) PWXD

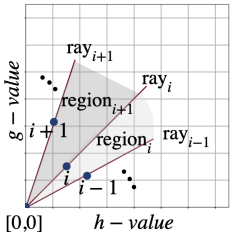


Figure 2: DSWA* regions

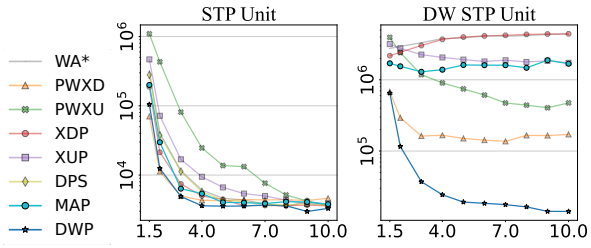
DSWA* allows an algorithm to determine *at runtime* where greater suboptimality is needed.

DSWA*

DSWA* was inspired by the observation that searches incrementally proceed from states with low g-cost and high h-cost (on the h axis) to states with low h-cost and high g-cost on the g-axis. Thus, the Φ-function can be incrementally defined at runtime, as the search proceeds.

In particular, the state space is broken up into regions defined by rays that proceed from the origin, as shown in Figure 2. Each region can use a different weight, w_i, with the general priority in the ith region being of the form $\kappa_i(g+w_ih)$, where κ_i is a normalizing constant that ensures the isolines are contiguous at region borders. The weight used in region i is defined when the first state in that region is generated. The weight is limited to be between $w_{min} = 1$ and $w_{max} = 2w-1$, with the exception of edge cases where a smaller range of values is permitted.

DSWA* can then use different policies to decide where the search should be more or less suboptimal. One simple policy, called the Dynamic Weighted Policy (DWP), measures the ratio $c(a,b)/h(a,b)$ for neighboring states in the search. If this ratio is 1, the heuristic exactly captures the distances in that part of the state space. In this case the DWP policy calculates the angle between the newest ray and h-axis, and returns $w_{min} + (w_{max} - w_{min}) \cdot (angle/90)^i$ for $i = 3$. If the ratio is greater than 1, then there is some cost that is not accounted for by the heuristic, and w_{max} is used. The Moving Average Policy (MAP) measures the distribution of recent expansions, and increases the weight according to this distribution.



(a) Unit weights (b) Dynamic Weights

Figure 3: Average number of node expansions (y-axis) to find solutions within a given suboptimality bound (x-axis)

Results

In Figure 3 we show a single result in the Sliding-Tile Puzzle. Here we compare fixed Φ functions in WA*, PWXD, PWXU, XDP, and XUP, along with Dynamic Potential Search (DPS) (Gilon, Felner, and Stern 2016), and DSWA* with a moving-average policy (MAP) and the dynamic weighted policy (DWP).

When we use a unit-cost version of the problem with the Manhattan distance heuristic, the DWP policy give a static curve similar to XDP. It performs well, but similar to other algorithms. When we weight some regions of the state space, the DWP policy significantly outperforms other approaches.

Conclusions

DSWA* is a new algorithm which can determine at runtime where to be more or less suboptimal. For DSWA* to work well, it needs a signal that tells it when/where it should be suboptimal. The open challenge is to understand where those signals are found in real-world, and to find policies that DSWA* can use to exploit them.

Acknowledgments

This work was supported by the National Science and Engineering Research Council of Canada Discovery Grant Program and the Canada CIFAR AI Chairs Program. This research was enabled in part by support provided by Prairies DRI and the Digital Research Alliance of Canada.

References

Chen, J.; and Sturtevant, N. R. 2019. Conditions for Avoiding Node Re-expansions in Bounded Suboptimal Search. *International Joint Conference on Artificial Intelligence (IJCAI)*.

Chen, J.; and Sturtevant, N. R. 2021. Necessary and sufficient conditions for avoiding reopenings in best first suboptimal search with general bounding functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 3688–3696.

Gilon, D.; Felner, A.; and Stern, R. 2016. Dynamic potential search—a new bounded suboptimal search. In *Proceedings of the International Symposium on Combinatorial Search*, volume 7, 36–44.

Hami, M. 2025. *Suboptimal Search with Dynamic Distribution of Suboptimality*. Master's thesis, University of Alberta.

Hami, M.; and Sturtevant, N. R. 2025. Suboptimal Search with Dynamic Distribution of Suboptimality. In *AAAI Conference on Artificial Intelligence*.