Conditions for Avoiding Node Re-expansions in Bounded Suboptimal Search

Jingwei Chen and **Nathan R. Sturtevant**
Department of Computing Science, University of Alberta, Edmonton, AB Canada
{jingwei5,nathanst}@ualberta.ca

Abstract

Many practical problems are too difficult to solve optimally, motivating the need to found suboptimal solutions, particularly those with bounds on the final solution quality. Algorithms like Weighted A*, A*-epsilon, Optimistic Search, EES, and DPS have been developed to find suboptimal solutions with solution quality that is within a constant bound of the optimal solution. However, with the exception of weighted A*, all of these algorithms require performing node re-expansions during search. This paper explores the properties of priority functions that can find bounded suboptimal solution without requiring node re-expansions. After general bounds are developed, two new convex priority functions are developed that can outperform weighted A*.

Algorithm 1 Generic Best-First Search

```
1:  procedure BEST-FIRST SEARCH($start$, $goal$)
2:      Push($start$, $Open$)
3:      while $Open$ not empty do
4:          Remove best state $s$ from $Open$
5:          if $s$ == $goal$ then return $success$
6:          end if
7:          Move $s$ to Closed
8:          for each successor $s_i$ of $s$ do
9:              if $s_i$ on $Open$ then
10:                 Update cost of $s_i$ on $Open$ if shorter
11:             else if $s_i$ not on Closed then
12:                 Add $s_i$ to $Open$
13:             end if
14:         end for
15:     end while
16:     return $failure$
17: end procedure
```

1 Introduction

If the cost of finding the optimal solution is too expensive, an alternate is to relax the solution quality in order to reduce the time required to find a solution. Such techniques are widely used in applications that are sensitive to response time such as embedded systems [Benton *et al.*, 2007] or video games [Bulitko *et al.*, 2011]. Research has focused on bounded suboptimal search (BSS), which guarantees that the solution found has cost no greater than $w \times C^*$, where w is the bound on the solution quality and C^* is the optimal solution cost.

BSS algorithms have been relatively well-studied in the literature. A classic example is weighted A* (WA*) [Pohl, 1970], which weights the heuristic in order to focus on states that have lower heuristic values. WA* uses a single priority function $f(n) = g(n) + w \cdot h(n)$, where w is the suboptimality bound. Later versions of the algorithm [Pohl, 1973] used a dynamic weight to try to improve performance.

In BSS there are two tasks that any algorithm must balance, (1) finding a solution and (2) proving that the solution is within the suboptimality bound. A* does both of these with a simple priority function, but more advanced algorithms split these two tasks and work on them somewhat independently.

Algorithms in this category include A*$_\epsilon$ [Pearl and Kim, 1982], Optimistic Search [Thayer and Ruml, 2008], EES [Thayer and Ruml, 2011], and DPS [Gilon *et al.*, 2016]. These algorithms can outperform WA* in some domains, however they have several shortcomings compared to WA*: (1) they sometimes need to re-sort the states on their priority queues, which can be an expensive operation; (2) they cannot avoid reopening states, even if the heuristic is consistent; and (3) they are far more difficult to implement than WA*, as they use more complicated data structures and may require other estimates or heuristics to be used in practice.

Looking at these drawbacks, this paper describes conditions for the priority function for A* (or more generally best-first search) that allow the algorithm to (1) maintain a single priority queue, avoiding the need for re-sorting during search and (2) avoid necessarily re-opening states when a shorter path is found to a state on the closed list. From these conditions new priority functions are derived which have these properties that can also outperform WA*.

2 Background

Suboptimal search algorithms fall in two broad categories, generic best-first search, whose pseudo code is Algorithm 1, and focal search. Best-first-search algorithms maintain a single priority queue and select the best state from that queue to expand at each time step. Focal list algorithm run an A* search, which is used to verify the optimality bound of the solution found, in parallel with a greedy search, which attempts to find the solution quickly. One exception to this is DPS

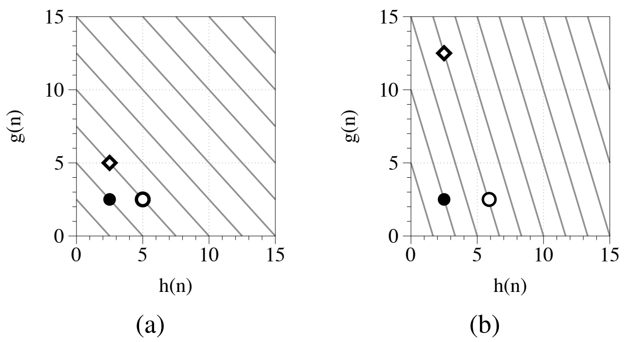

Figure 1: Isolines for states with the same priority in (a) A* and (b) Weighted A* with $w = 3$.

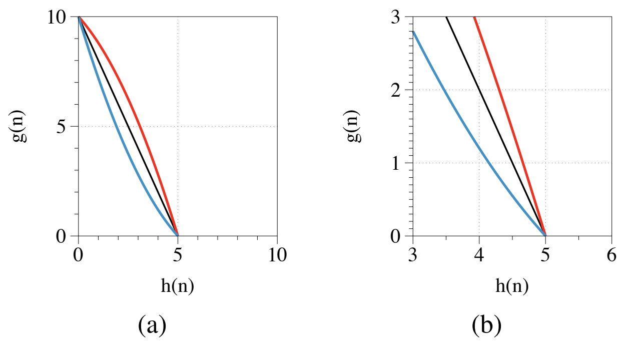

Figure 2: Different functions produce isolines with different properties.

[Gilon *et al.*, 2016], which maintains a single priority queue, but must frequently re-sort the queue.

In generic best-first search, what makes an algorithm unique is the definition of *best*. A priority function is typically used to define the best state to expand over all states. A* uses a priority function of $f(n) = g(n) + h(n)$ while WA* uses $f(n) = g(n) + w \cdot h(n)$. This paper analyzes the class of priority functions in the form of $f(n) = \Phi(h(n), g(n))$ that can avoid re-expansions during search.

2.1 Assumptions and Notation

A heuristic search problem is defined by a n-tuple $\{G, start, goal, h, B\}$. The state space, G, is a finite directed graph whose vertices are states and whose edges are pairs of states. Each edge $(u, v) \in G$ has a cost $c(u, v) \geq 0$ and we assume there is at most one edge between each pair of states. A path in G is a finite sequence $U = (U_0, \ldots, U_n)$ of states in G where (U_i, U_{i+1}) is an edge in G for $0 \leq i < n$. A path U contains edge (u, v) iff for some i $U_i = u$ and $U_{i+1} = v$.

In BSS the currently explored path to a state does not necessarily correspond to the shortest path. Thus, the g-cost of a state is the cost of the current path from the start to that state.

Let $d(u, v)$ be the shortest distance from state u to state v in G, i.e., the cost of the cheapest path from u to v. If there is no path from u to v then $d(u, v) = \infty$. We let $g^*(u) = d(start, u)$. We use $end(U)$ to refer to the state at the end of a path. Given two states $start$ and $goal$ in G, a solution path is a path U with $u_0 = start$ and $end(U) = goal$. The shortest path has solution cost $C^* = d(start, goal)$.

Provided with a problem is the heuristic function $h(n)$, which estimates the distance between n and the goal. This paper assumes that the heuristic is both admissible ($h(n) \leq d(n, goal)$) and consistent ($h(n) \leq c(n, m) + h(m)$). A problem also includes a bounding function $B(x)$ [Valenzano *et al.*, 2013] which provides the bound on the solution quality for a given problem. For a problem instance with solution cost C^*, the algorithm must return a solution $C \leq B(C^*)$. For WA*, $B(n) = w \cdot n$. This paper only studies w-suboptimal bounding functions, as other bounding functions introduce complications that are beyond the scope of this paper.

A node *re-opening* occurs when a shorter path is found to a state on closed and the state is removed from closed and placed back on open. A *re-expansion* occurs when the state is later removed and expanded from open a second time. WA* does not need to re-open or re-expand states to maintain its bound [Likhachev *et al.*, 2004].

Domain	no reopen	reopen
DAO grid maps	**7,550**	47,305
15-puzzle	40,544	**39,753**
Heavy 15-puzzle	**118,848**	173,729

Table 1: Average state expansions for WA*, with w=2 on different domains

3 Motivation

Reopenings have a significant impact in BSS [Sepetnitsky *et al.*, 2016]. This is illustrated in Table 1 which shows the cost of re-opening nodes with WA* in several domains. Recent work has shown that WA* with re-expansions may perform $O(N^2)$ expansions in a graph with N states [Chen *et al.*, 2019]. Performing or not performing re-openings also has impact on solution quality [Valenzano *et al.*, 2014].

The main contribution of this paper is to develop a set of conditions on the priority function used by a best-first search that ensures that the best-first search will find a bounded suboptimal solution without re-expanding states. This paper will fully develop two specific priority functions from a larger class that have this property. But, first, we will motivate why alternate priority functions are potentially interesting.

First, consider the priority function used in A* or WA*. This is a function $\mathbb{R}^2 \to \mathbb{R}$, so a priority function is described by a two-dimensional surface over the x-y plane, where the height of the surface is the priority of a state. Sets of points along this surface which have the same priority fall on isolines of the surface. Consider plotting every state on the open list as a point according to its g-cost and h-cost. States on the same isoline will have the same priority, and a best-first search will expand the state with lowest priority first. This is illustrated in Figure 1, where part (a) represents A* and part (b) represents WA* with a weight of 3. A* expands states according to $g(n) + h(n)$, so in part (a) all states with the same $g(n) + h(n)$ are on the same line. Because it is on a lower isoline, the state represented by the solid circle, with priority

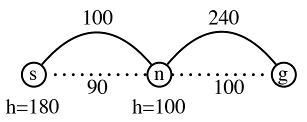

Figure 3: An example problem with a bounded-suboptimal solution that WA* cannot find.

5, will be expanded before the hollow states, which have priority 7.5. With WA* the steeper slope of the isolines allows suboptimal solutions.

Because the isolines are straight lines, WA* allows the same suboptimality along each portion of the path being explored. But, there are other possibilities. In Figure 2(a), we show three different isolines for different priority functions. Each of these priority functions will return w-suboptimal solutions, but they place different requirements on the path that is found. Figure 2(b) shows part of the curve in more detail. The straight black line in the middle is WA* with $w = 2$. Because the WA* isoline is straight, at any point in the search WA* can expand a state that increases the g-cost by 2 and decreases the h-cost by 1 while maintaining the same priority.

The blue line isoline on the bottom, however, is curved. The slope when g is close to 0 is almost -1. Thus, if the isolines of the priority function follow this shape, then best-first search with this priority function will allow very little suboptimality at the beginning of any path that is explored. As the g-cost increases, however, larger amounts of suboptimality will be allowed. The red line on the top has the opposite impact. Early in the search a larger suboptimality is allowed, but the closer the search gets to the goal the less suboptimality is allowed.

The practical use of such priority functions is illustrated in Figure 3. Suppose that WA* is searching from s to g with $w = 2$. We can reason from the h-cost of the start that the top path from s to n to g is within the 2x path cost bound. But, WA* requires that the portion of the path from n to g be at most 2x suboptimal, so all nodes on the bottom path from n to g will always have lower priority than the goal will along the top path. However, the priority function with the blue isoline in Figure 2 allows less suboptimality on the first half of the path and more suboptimality on the second half, so it would be able to find the solution length 340 in this example. Whether this is useful in practice will depend on the properties of a domain and a heuristic.

4 Conditions for Avoiding Re-expansions

This section develops the conditions on any function $\Phi(x, y)$ that can be used as a priority function for best-first search and does not require node re-expansions to find a bounded cost solution. First, let $\Phi(x, y)$ be a function $\mathbb{R}^2 \rightarrow \mathbb{R}$. Given a state u in open reached by a path U, $f(u) = \Phi(h(u), g(u))$. Φ describes a surface, but we are interested in the sets of states on that surface which have the same priority. Such states fall on isolines of the surface. For WA* we use $\Phi(x, y) = x + \frac{1}{w} \cdot y$.

It is worth noting that in a finite state space, a best-first search with any function Φ is complete even if it does not re-open states.

Lemma 1. *In a finite state space, a best-first search with any priority function Φ is complete (finds a solution if one exists) even if it does not re-open states.*

Proof. Assume that the optimal path from $start$ to $goal$ is $p_0, p_1, p_2, ...p_n$, where $p_0 = start$, $p_n = goal$. If $goal$ is not expanded, then prior to each expansion there always exist at least one p_i such that $0 \leq i \leq n$ and p_i is on $Open$ while none of $p_{i+1}, ...p_n$ is closed. This implies that the search always makes progress on exploring the optimal path (even if the costs used to explore the path are not optimal), and thus will eventually expand the goal and complete. We prove this by induction.

Base case: At the very beginning, p_0, which is $start$ is on open, meeting the requirement for this lemma.

Inductive step: Assume at some point there exists some p_i which meets this condition. If the next state that is chosen for expansion is not from $p_i, p_{i+1}, ...p_n$, then p_i is still valid after the next expansion. Otherwise, suppose we choose to expand p_k which is one of $p_i, p_{i+1}, ...p_n$. Then p_{k+1} will be placed on open and meet the condition. □

4.1 Properties of Φ that Avoid Re-expansions

The following properties are the sufficient conditions for Φ to avoid re-expansions. We provide a brief intuition of how these properties related to search, and we will provide the proofs for sufficiency in next subsection.

Property 1. $\frac{\partial \Phi}{\partial x} > 0$, $\frac{\partial \Phi}{\partial y} > 0$.

This property means we assume that for two states with same h-cost, we will alway prefer the one with lower g-cost; similarly, for two states with same g-cost, we will alway prefer the one with lower h-cost.

Property 2. $\frac{\partial \Phi}{\partial y} \leq \frac{\partial \Phi}{\partial x}$

This property requires that Φ grows faster when the h-cost is increased than when the g-cost is increased by the same amount. Namely, there is more weight on the h-cost term than the g-cost term. In terms of isolines, this property is equivalent to saying that the slope of any iosoline at any point is no greater than -1. A slope greater than -1 corresponds to WA* with a weight less than one, which is never beneficial with a consistent heuristic. Practically speaking it means that the hollow diamond state in Figure 1(a) can be expanded no later than the hollow circle, because the hollow diamond state has lower h-cost.

Property 3. $\Phi(0, w \cdot t) = \Phi(t, 0) = t$

A state with h-cost of v and g-cost of 0 should have equivalent priority to a state with g-cost of $w \cdot v$, because both of these can lead to a w-suboptimal path. The actual priority (value of Φ) at these points can be scaled by a constant, but for simplicity we assume the priority (value of Φ of these states is also t.

Property 4. $\frac{\partial \Phi}{\partial x} + \frac{\partial \Phi}{\partial y} \leq 2$

This last property is related to the rate of change of the isolines; this needs to be bounded (the value of 2 arises from the constant t used in Property 3) in our construction.

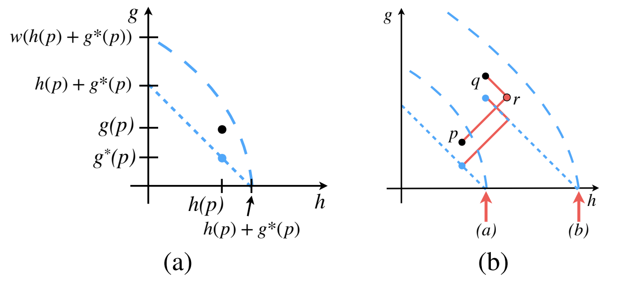

Figure 4: (a) An illustration of the Φ-inequality (b) Proof outline.

4.2 Proofs that Φ Avoid Re-expansions

The proof is done in a top-down manner. We begin by showing a condition that is sufficient for a best-first search algorithm to avoid re-expansions, and then work out the practical properties required to meet this condition. We call the primary condition the Φ-*inequality*; it relies on property 3.

Definition 1. *We define the* Φ*-inequality as* $\Phi(h(p), g(p)) \leq \Phi(0, w(h(p) + g^*(p))) = \Phi(h(p) + g^*(p), 0)$

The Φ-inequality is illustrated in Figure 4(a). The black dot is the point $(h(p), g(p))$ and the blue dot is the point $(h(p), g^*(p))$. The blue dot represents where p would be plotted if we found the optimal path to p instead of the current path with cost $g(p)$. The minimum path cost through p to the goal is $h(p) + g^*(p)$, and if the solution is w-suboptimal, the longest w-optimal solution will have cost $w(h(p)+g^*(p))$. The Φ-inequality says that $(h(p), g(p))$ will not be above the isoline that passes through $(h(p)+g^*(p), 0)$ and $(0, w(h(p) + g^*(p)))$.

For WA*, the Φ-inequality says that when p is expanded the portion of the path found so far is at most w-suboptimal. More generally, it is saying that the path so far is within the local bound allowed by the priority function.

Theorem 2. *Assume a best-first search is using a priority function* Φ *with a suboptimality bound of* w *that meets property 1. If, for every state* p *that is expanded the first time, the* Φ*-inequality holds, then it is not necessary to re-open (and thus re-expand) states during search to return a* w*-suboptimal solution.*

Proof. Lemma 1 shows that any best-first search algorithm that does not perform re-openings is complete. Thus, we know we will expand the goal state at some cost $g(\text{goal})$. If for any p expanded it holds that $\Phi(h(p), g(p)) \leq \Phi(0, w(h(p) + g^*(p)))$, then it also holds for the path discovered to the goal. In this case $\Phi(0, g(\text{goal})) \leq \Phi(0, wC^*)$. Because Φ is monotonically increasing with y due to Property 1, this means that $g(\text{goal}) \leq wC^*$ and the solution is w-suboptimal. □

Thus, it is sufficient to show that the Φ-inequality holds for every state expanded with a given priority function. The high-level approach to proving this is illustrated in Figure 4(b). Suppose we plan to expand a state q where p is the last state expanded on the optimal path to q. As before, the point in blue below p represents the optimal path to p and the curved line at (a) is the isoline for that point. We know by the Φ-inequality that p is under the isoline that begins at point (a). Thus, if the search moves optimally from p to q it should still hold that q (and all states on the optimal path from p to q) is under the $h(q) + g^*(q)$ isoline at point (b) on the h-axis because the optimal path does not accumulate any additional suboptimality. If this holds, then there will always be at least one path (the remaining portion of the optimal path) where it is possible to reach the goal under the optimality bound without re-expanding a previously expanded state. While the search may expand a less optimal path in practice, the explored path will still be guaranteed to be within the bound allowed by Φ.

To show this holds, we can compare the change in Φ from p to q to the change in Φ between (a) and (b). As long as the change in Φ between p and q is less than the change in Φ between (a) and (b) then the Φ inequality will hold when q is expanded. To do this, we want to measure the rate of change of Φ in a direction (a, b).

Lemma 3. *Suppose that state* q *is a descendant of state* p*. The coordinates of* p *on* h*-*g *plane are* $(h(p), g(p))$ *and the coordinates of* q *are* $(h(q), g(p) + d(p, q))$*. For any state* p_1 *on the optimal path from* p *to* q*,* p_1 *must be in the rectangle formed with opposite corners on* p *and* q*. (Two lower edges of this rectangle are shown in red in Figure 4(b).)*

Proof. This is a direct result of a consistent heuristic. □

Given that the path from p to q stays inside a rectangle around p and q, we need to know which point on the rectangle has maximum priority.

Definition 2. *We define* r *as the right most corner of the rectangle described in Lemma 3. It can be computed that* $x_r = h(p) + \frac{d(p,q)+h(q)-h(p)}{2}, y_r = g(p) + \frac{d(p,q)+h(q)-h(p)}{2}$.

We now present a general condition that will allow us to show that no state on the path between p and q has higher priority than r.

Lemma 4. $\phi(x+at, y+bt) \leq \phi(x, y)+ct$ *for any fixed point* (x, y) *if* ϕ *is a function* $\mathbb{R}^2 \to \mathbb{R}$ *satisfying* $a \cdot \frac{\partial \phi}{\partial x} + b \cdot \frac{\partial \phi}{\partial y} \leq c$.

Although we omit the proof of this property, note that this is a general condition related to whether ϕ is Lipschitz.

Corollary 5. $\Phi(x - t, y + t) \leq \Phi(x, y)$.

Proof. $-\frac{\partial \Phi}{\partial x} + \frac{\partial \Phi}{\partial y} \leq 0$ by property 2. Thus, Φ satisfies the requirement of Lemma 4 for $a = -1, b = 1, c = 0$ and so $\Phi(x - t, y + t) \leq \Phi(x, y)$. □

This tells us that in Figure 4(b), $\Phi(x_r, y_r) \geq \Phi(h(q), g(q))$.

Lemma 6. *For all states* s *on all optimal paths between* p *and* q*,* $\Phi(h(s), g(p) + d(p, s)) \leq \Phi(x_r, y_r)$ *where* r *is defined in Definition 2.*

Proof. First of all, we know s must be in the rectangle described in Lemma 3. Then, by Property 1, the maximum value of Φ on this rectangle must be a point on the segment qr. By Corollary 5, as we move from q to r, the Φ-value is monotonically non-decreasing. Therefore, $\Phi(x_r, y_r)$ has maximum value in the rectangle defined by p and q. □

Corollary 7. $\Phi(x + t, y + t) \leq \Phi(x, y) + 2t$.

Proof. $\frac{\partial \Phi}{\partial x} + \frac{\partial \Phi}{\partial y} \leq 2$ by property 4. Thus, Φ satisfies the requirement of Lemma 4 for $a = 1, b = 1, c = 2$ and so $\Phi(x+t, y+t) \leq \Phi(x, y) + 2t$. □

This bounds the change in Φ between p and r.

Given these results, we can now show that the Φ-inequality holds as long as Φ meets properties 1 – 4.

Theorem 8. *Assume a best-first search is using a priority function* Φ *which meets properties 1 – 4. Then, the* Φ*-inequality holds for all expansions:* $\Phi(h(n), g(n)) \leq \Phi(h(n) + g^*(n), 0)$

Proof. Proof by induction.

Base case: Initially *start* is chosen for expansion, so $\Phi(h(start), g(start)) = \Phi(h(start), 0) = \Phi(h(start) + g^*(start), 0)$, and the claim holds.

Inductive step: Assume the inequality holds for all expanded paths. And a state q is the next state from *Open* is chosen for expansion. We need to prove that $\Phi(h(q), g(q)) \leq \Phi(h(q) + g^*(q), 0))$.

Let the last node that is closed on optimal path from $start$ to q be p. The hypothesis guarantees that:

$$\Phi(h(p), g(p)) \leq \Phi(h(p) + g^*(p), 0) = h(p) + g^*(p) \quad (1)$$

Since p is closed, there must exist a direct successor of p, p_x on open, which is on the optimal path from p to q. Examining the rectangle formed around p and q, the absolute distance along each axis from p to the corner state r with maximum priority is:

$$t = \frac{d(p, q) + h(q) - h(p)}{2} \quad (2)$$

So, according to Lemma 6, we get:

$$\begin{aligned} \Phi(h(p_x), g(p_x)) &= \Phi(h(p_x), g(p) + d(p, p_x)) \\ &\leq \Phi(h(p) + t, g(p) + t) \quad (3) \end{aligned}$$

According to Corollary 7,

$$\begin{aligned} &\Phi(h(p) + t, g(p) + t) \\ &\leq \Phi(h(p), g(p)) + 2t \quad (4) \\ &= \Phi(h(p), g(p)) + d(p, q) + h(q) - h(p) \end{aligned}$$

By combining inequalities (1), (3) and (4) we can get

$$\begin{aligned} &\Phi(h(p_x), g(p_x)) \\ &\leq h(p) + g^*(p) + d(p, q) + h(q) - h(p) \\ &= h(q) + g^*(p) + d(p, q) \quad (5) \\ &= h(q) + g^*(q) \\ &= \Phi(h(q) + g^*(q), 0) \end{aligned}$$

In practice, we chose to expand q instead of p_x, so the priority of q will be no more than that of p_x. Thus:

$$\begin{aligned} \Phi(h(q), g(q)) &\leq \Phi(h(p_x), g(p_x)) \\ &\leq \Phi(h(q) + g^*(q), 0) \end{aligned}$$

This proves that the Φ-inequality holds for each expansion. □

Theorem 9. *If there is a path from start to goal, then a BFS using any* Φ *that meets properties 1 – 4, is guaranteed to find a solution within the given bound without re-opening states.*

Proof. This is a direct result of Theorem 2 and 8. □

Bound	Φ_{XDP}	WA*	Φ_{XUP}
1.11	**11,577**	11,825	12,152
1.25	**10,414**	10,666	11,041
1.50	**9,185**	9,254	9,418
1.60	8,885	**8,772**	8,811
1.75	8,279	**8,210**	8,372
1.90	7,787	**7,779**	8,036
2.00	7,843	7,550	**7,499**
3.00	6,675	6,225	**5,862**

Table 2: Average state expansions for WA*, XDP and XUP on DAO

	SC1			Random		
Bound	Total Expansions			Total Expansions		
	Φ_{XDP}	WA*	Φ_{XUP}	Φ_{XDP}	WA*	Φ_{XUP}
1.25	**30,159**	31,369	32,658	**28,004**	30,659	33,509
1.5	**23,767**	24,060	24,715	**22,863**	26,015	30,694
2.0	18,611	17,940	**17,492**	**18,002**	19,758	23,813
3.0	15,013	13,796	**12,989**	**14,458**	14,469	15,184
10.0	11,781	10,998	**10,663**	11,356	10,895	**10,600**

Table 3: Average state expansions for WA*, XDP and XUP on Grid Maps

5 Function Selection

Given now that we understand the properties required for Φ we can develop several different functions that meet these properties.

5.1 Linear Function

The first function we develop is a linear function, which is equivalent to WA*. In this case $\Phi(x, y) = \frac{1}{w} \cdot y + x$. While WA* typically puts a weight of w on the heuristic, it is equivalent to put a weight of $1/w$ on the g-cost.

We can, for instance, verify that property 4 holds for $w \geq 1$:

$$\frac{\partial \Phi}{\partial x} + \frac{\partial \Phi}{\partial y} = \frac{1}{w} + 1 \leq 2$$

In a similar manner, it can be shown that properties 1 – 3 also hold. Thus, as was known, WA* maintains a w-optimal solution without re-expanding states.

5.2 Non-linear Priority Functions

We can now derive new priority functions that meet properties Properties 1 – 4.

Convex Downward Parabola

Our first priority function that has isolines similar to the bottom parabola shown in blue in Figure 2. A parabola is a function is of the form $y = ax^2 + bx + c$, where a, b, c are to be determined. Assume that parabola goes through the points $(0, wU)$ and $(U, 0)$ and has slope -1 at $(U, 0)$. The slope of -1 at $(U, 0)$ means that paths with low g-cost must be near-optimal. This results in the following equation set:

$$\begin{cases} c = wU \\ a \cdot U^2 + b \cdot U + c = 0 \\ 2a \cdot U + b = -1 \end{cases} \quad (6)$$

Bound	15 Puzzle						Heavy 15-Puzzle						Heavy Pancake					
	Total Expansions			Time			Total Expansions			Time			Total Expansions			Time		
	XDP	WA*	XUP	XDP	WA*	XUP	XDP	WA*	XUP	XDP	WA*	XUP	XDP	WA*	XUP	XDP	WA*	XUP
1.50	**185,783**	318,953	496,412	**1.36**	1.91	3.94	**200,318**	333,320	702,468	**1.49**	2.00	5.95	**2,460,235**	8,498,635	19,995,382	**42.44**	140.69	672.95
2.00	**21,338**	40,544	71,014	**0.12**	0.20	0.44	**82,295**	114,848	161,126	**0.60**	0.69	1.31	**93,355**	973,556	7,072,634	**1.49**	15.49	181.63
3.00	**7,550**	11,600	16,934	**0.09**	0.11	0.21	**48,203**	57,778	82,916	**0.56**	0.57	1.12	**1,123**	22,732	404,622	**0.03**	0.49	10.66
10.00	**3,586**	3,758	3,859	0.03	0.03	0.03	43,141	44,207	**34,066**	0.56	0.49	**0.45**	**20**	33	318	0.00	0.00	0.00

Table 4: Average performance for WA*, Φ_{XDP} (XDP) and Φ_{XUP} (XUP) on Exponential Domains

Solving for a, b and c, the equation of the parabola is

$$y = \frac{w-1}{U} \cdot x^2 + (1-2w) \cdot x + wU \tag{7}$$

We can rewrite equation (7) as an equation of U:

$$wU^2 + (x - 2wx - y)U + wx^2 - x^2 = 0 \tag{8}$$

The Φ function we are looking for, is proportion to the larger root of equation (8):

$$\Phi_{XDP}(x, y) = \frac{1}{2w}[y + (2w-1)x + \sqrt{(y-x)^2 + 4wyx}] \tag{9}$$

A check of properties 1 – 4 shows that they all hold for this priority function. We call this priority function the convex downward parabola (XDP) priority function .

Convex Upward Parabola

Our second function corresponds to the top parabola shown in red in Figure 2. This parabola goes through points $(0, wU)$ and $(U, 0)$, has slope -1 at point $(0, wU)$. Because the slope is -1 near $(0, wU)$ it means that near the goal the path found must be near-optimal.

$$\begin{cases} c = wU \\ a \cdot U^2 + b \cdot U + c = 0 \\ b = -1 \end{cases} \tag{10}$$

Following the same steps as before we get the priority function:

$$\Phi_{XUP}(x, y) = \frac{1}{2w}(y + x + \sqrt{(y+x)^2 + 4w(w-1)x^2}) \tag{11}$$

This function is a convex upward parabola (XUP) and also meets properties 1 – 4.

6 Experimental Results

This paper characterizes priority functions that return w-suboptimal solutions without requiring node re-openings during search. The experimental results compare priority functions within this class to see if the new priority functions outperform WA*. Several standard domains are used for testing.

6.1 Experimental Settings

On grid maps we tested on 15,928 problem instances from the Dragon Age: Origins (DAO) map set, 6,444 problem instances from the StarCraft 1 (SC1) map set, and on 35,360 problem instances on maps with 40% random obstacles [Sturtevant, 2012]. The instances for these maps contain 10 problems for each length; we randomly select 10% of each of these problems for testing.

We also tested on the 15-puzzle with the unit edge cost setting and the heavy tile setting, where the cost of moving tile X is X. Manhattan Distance (MD) is used as heuristic for regular tiles, and the modified MD [Thayer and Ruml, 2011] is used for heavy tiles. The instance are the standard 100 Korf instance [Korf, 1985].

Finally, we followed the heavy variant of pancake puzzle created by Gilon *et al.* [2016], where the cost of flipping a prefix $(V[1] \cdots V[i+1])$ is the $max(V[1]; V[i+1])$. The heuristic we used is their *HGAP* heuristic. The problem set is 50 randomly generated 12-pancake instances.

We experiment with the various weights for WA*, and best-first search with the convex downward parabola (Φ_{XDP}) and the convex upward parabola (Φ_{XUP}). As a general trend, the difference in performance on grid maps was noticeable but not notable. The most interesting trend is the phase transition between Φ_{XUP}, WA*, and Φ_{XDP} in Table 2 as the weights increase. This same trend is seen in all experimental results. There is less than 5% difference in path quality between the different priority functions, with fewer node expansions leading to lower solution quality.

On the other exponential domains in Table 4 we can see a significant improvement with the right priority function. With low weights on the 15-puzzle, Heavy 15-puzzle and Heavy Pancake Puzzle, Φ_{XDP} significantly outperforms WA* both in node expansions and time with the same weight. With a weight of 10, Φ_{XUP} has better performance in the Heavy 15-puzzle. See Chen *et al.* [2019] for further experiments.

If we examine focal list algorithms, they run an A* search near the start to prove that the solution is within the bound, while greedily searching for the goal. Φ_{XDP} able to accomplish both of these in a single search, balancing a tight bound around the start with a looser bound around the goal.

7 Conclusions and Future Work

This work generalized WA* to a family of algorithms that can find w-suboptimal solutions while avoiding node re-expansions. We also provide two new priority functions, Φ_{XDP} and Φ_{XUP}. Φ_{XDP} only requires a minor change to a best-first search implementation, but offers a significant improvement over WA* in the exponential domains we tested.

This work just handles curves for w-suboptimal paths, however there are other classes of suboptimality which are not addressed, such as additive bounds [Valenzano *et al.*, 2013]. We have also developed w-suboptimal piece-wise continuous curves which are not presented here. Future research will continue develop this broader classes of bounds.

References

[Benton *et al.*, 2007] J Benton, Minh Do, and Wheeler Ruml. A simple testbed for on-line planning. In *ICAPS Workshop on Moving Planning and Scheduling Systems into the Real World*, 2007.

[Bulitko *et al.*, 2011] Vadim Bulitko, Yngvi Björnsson, Nathan R Sturtevant, and Ramon Lawrence. Real-time heuristic search for pathfinding in video games. In *Artificial Intelligence for Computer Games*, pages 1–30. Springer, 2011.

[Chen *et al.*, 2019] Jingwei Chen, Nathan R. Sturtevant, William Doyle, and Wheeler Ruml. Revisiting suboptimal search. In *Symposium on Combinatorial Search (SoCS)*, 2019.

[Gilon *et al.*, 2016] Daniel Gilon, Ariel Felner, and Roni Stern. Dynamic potential search—a new bounded suboptimal search. In *Symposium on Combinatorial Search (SoCS)*, 2016.

[Korf, 1985] Richard E Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial intelligence*, 27(1):97–109, 1985.

[Likhachev *et al.*, 2004] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in neural information processing systems*, pages 767–774, 2004.

[Pearl and Kim, 1982] Judea Pearl and Jin H. Kim. Studies in semi-admissible heuristics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(4):392–399, July 1982.

[Pohl, 1970] Ira Pohl. Heuristic search viewed as path finding in a graph. *Artificial intelligence*, 1(3-4):193–204, 1970.

[Pohl, 1973] Ira Pohl. The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving. In *Proceedings of the 3rd international joint conference on Artificial intelligence*, pages 12–17. Morgan Kaufmann Publishers Inc., 1973.

[Sepetnitsky *et al.*, 2016] Vitaly Sepetnitsky, Ariel Felner, and Roni Stern. Repair policies for not reopening nodes in different search settings. In *Ninth Annual Symposium on Combinatorial Search*, 2016.

[Sturtevant, 2012] Nathan R. Sturtevant. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games*, 4(2):144–148, 2012.

[Thayer and Ruml, 2008] Jordan T. Thayer and Wheeler Ruml. Faster than weighted A*: An optimistic approach to bounded suboptimal search. In *International Conference on Automated Planning and Scheduling*, pages 355–362, 2008.

[Thayer and Ruml, 2011] Jordan T. Thayer and Wheeler Ruml. Bounded suboptimal search: A direct approach using inadmissible estimates. In *International Joint Conference on Artifical Intelligence (IJCAI)*, pages 674–679, 2011.

[Valenzano *et al.*, 2013] Richard Valenzano, Shahab Jabbari Arfaee, Roni Stern, Jordan Thayer, and Nathan Sturtevant. Using alternative suboptimality bounds in heuristic search. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 233–241, 2013.

[Valenzano *et al.*, 2014] Richard Valenzano, Nathan Sturtevant, and Jonathan Schaeffer. Worst-case solution quality analysis when not re-expanding nodes in best-first search. In *AAAI Conference on Artificial Intelligence*, pages 885–892, 2014.