Necessary and Sufficient Conditions for Avoiding Reopenings in Best First Suboptimal Search with General Bounding Functions

Jingwei Chen,[1] **Nathan R. Sturtevant,**[1]
[1] Department of Computing Science, Alberta Machine Intelligence Institute (Amii), University of Alberta, Canada
jingwei5@ualberta.ca, nathanst@ualberta.ca

Abstract

Recent work introduced XDP and XUP priority functions for best-first bounded-suboptimal search that do not need to perform state re-expansions as long as the search heuristic is consistent. However, that work had several limitations that are rectified here. This paper analyzes the sufficiency and necessity of the conditions used to formulate XDP and XUP. The analysis presents a simpler proof and generalizes the result in three aspects: (1) the priority function no longer has to be differentiable everywhere, (2) the quality of the solution does not have to be bounded by a constant factor, and (3) directed graphs are handled correctly. These results allow the introduction of more priority functions, such as piecewise linear functions, and more variants of bounded-suboptimal search, such as constant suboptimality. Several new priority functions are presented in this paper that, according to empirical results, can significantly outperform existing approaches including XDP.

1 Introduction

Heuristic search algorithms such as A* are able to find optimal paths between a given start and goal state. For many time-sensitive applications, such as embedded systems (Benton, Do, and Ruml 2007) or video games (Bulitko et al. 2011), a quick but suboptimal solution is more favorable than a slow but optimal solution. Algorithms aimed at finding suboptimal solutions are called suboptimal search algorithms. Research has focused on bounded suboptimal search (BSS), which guarantees that the solution found has cost no greater than $B(C^*)$, where $B : \mathbb{R} \to \mathbb{R}$ is a given bounding function and C^* is the optimal solution cost. The most popular bounding function is $B_w(x) = wx$, which means when the optimal solution is C^*, the returned solution is guaranteed to fall in the range of $[C^*, wC^*]$. Algorithms designed for solving this problem include weighted A* (WA*) (Pohl 1970), A*$_\epsilon$ (Pearl and Kim 1982), Optimistic Search (Thayer and Ruml 2008), EES (Thayer and Ruml 2011), and DPS (Gilon, Felner, and Stern 2016).

There are other types of bounding functions, such as additive bounding function $B_\gamma(x) = x + \gamma$, where γ is a given constant (Valenzano et al. 2013). Unfortunately, the only algorithm available for additive bounding requires performing node re-openings, which are expensive in some domains.

Our recent research studied the sufficient conditions for a Best-First Search (BFS) using a priority function Φ to find bounded suboptimal solutions without having to reopen nodes (Chen and Sturtevant 2019). Two example priority functions (or Φ functions) were introduced, *XDP* and *XUP*, which improve the performance of WA*.

However, there are a few limitations to this work. First of all, Φ was implicitly required to be differentiable. It was unclear whether non-differentiable functions, such as piecewise functions could be used. Secondly, the work only provided sufficient conditions for avoiding re-expansions. The conditions can be used to determine whether Φ will find bounded suboptimal solutions without reexpansions; however, without necessary conditions, we can not decide whether a Φ will cause the algorithm to find unbounded solutions. Thirdly, the conditions relied on consistent heuristics and undirected graphs. Thus, it was an open question whether some Φ functions will work with inconsistent heuristics. Fourthly, those results only applied to $B = B_w$.

In this paper, we resolve those issues in the following ways: (1) we show that, given a few basic assumptions on Φ, a consistent heuristic is a necessary condition for avoiding re-openings; (2) we present necessary conditions and the cases where they apply; (3) we relax previous assumptions by allowing continuous but not differentiable Φ-functions; (4) we extend the results to other bounding functions. Our new priority functions result in significant improvement over *XDP* and *XUP*.

2 Background

A bounded suboptimal search (BSS) 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 $c(u, v)$. We assume there is at most one edge between each pair of states, and the edge costs are non-negative. A path in G is a finite sequence $U = (u_0, \dots, u_n)$ of states in G where (u_i, u_{i+1}) is an edge in G for $0 \le i < n$.

Let $d(u, v)$ be the cost of the cheapest path from state u to state v in G. If there is no path from u to v then $d(u, v) = \infty$. We let $g^*(u) = d(start, u)$. The shortest path from start to goal is the optimal solution cost $C^* = d(start, goal)$. In

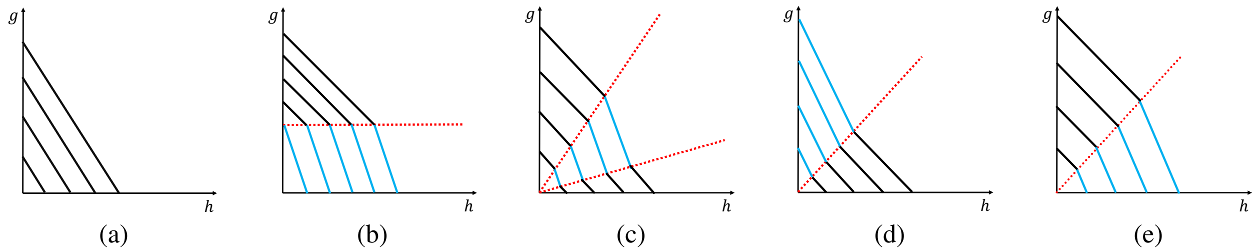

Figure 1: The contour plot of (a) Φ_{WA^*} (b) Φ_{AB} (c) Φ_z (d) Φ_{pwXD} (e) Φ_{pwXU}. Red lines indicate free parameters.

suboptimal search, the currently explored path to a state is not necessarily the shortest path. Let the g-cost of a state be the cost of the current best path from start to that state. Thus, $g(n) \geq d(start, n)$

The heuristic function, h, is an input parameter which maps a state to a real number. A heuristic is admissible if $\forall n, h(n) \leq d(n, goal)$. A heuristic is consistent on an undirected graph if $\forall m, n, |h(n) - h(m)| \leq d(n, m)$. If this property holds on a directed graph, we say the heuristic is *strongly consistent*. A heuristic is then *weakly consistent* if $\forall m, n, h(n) \leq d(n, m) + h(m)$. A weakly consistent heuristic only requires that the h-cost does not decrease more than the edge cost, but can go up along an edge as much as it wants. A strongly consistent heuristic limits the heuristic change in both directions, whether increasing or decreasing, to not exceed the edge cost. This paper assumes that the heuristic is admissible, and will discuss the conditions of Φ under each consistency case (inconsistent, weakly consistent, and strongly consistent).

The BSS problem is to find a solution path with cost $\leq B(C^*)$, where $B : \mathbb{R} \to \mathbb{R}$ is a given bounding function that satisfies $\forall x \geq 0, B(x) \geq x$.

We assume that search algorithms only have black box access to the state space. That is, they are only allowed to explore the state space by node expansions, and they can only prioritize states based the g-cost and h-cost of a state.

2.1 BFS-NR Guided by Φ

Best-first search is a generic algorithm whose pseudocode is Algorithm 1. The algorithm keeps expanding the state with minimum f-cost according to the provided Φ function until goal state is expanded.

In the following text, we assume $\Phi(x, y)$ is a function $\mathbb{R}^2 \to \mathbb{R}$. We assume $\Phi(x, y)$ is continuous, but not necessarily differentiable. Given a state u in OPEN reached with cost $g(u)$, the priority of u is $f(u) = \Phi(h(u), g(u))$.

Once a state is expanded, it is removed from OPEN list and put onto the *Closed* list, while its successors are put on OPEN or get updates to lower cost. If a shorter path is found to a state on *Closed*, it may optionally be reopened.

In heuristic search, reopening, or re-expanding, means taking a state from the *Closed* list and placing it back onto the OPEN list. Reopening is not mandatory for suboptimal search: suboptimal search algorithms can be complete even if they do not reopen states (Chen and Sturtevant 2019). Therefore, one could try different reopening policies, such as always re-expand (AR) and never re-expand (NR) (Sepetnitsky, Felner, and Stern 2016). When we apply the NR policy to best-first search, we get Best-First Search with the NR policy (BFS-NR) (Valenzano, Sturtevant, and Schaeffer 2014). In Algorithm 1, by simply setting the parameter RP to be $False$, we can get BFS-NR.

However, for bounded suboptiaml search, we not only want a solution but also care about the solution quality; we want the algorithm to be bounded. The definition of bounded algorithm is as following:

Definition 1. *If algorithm A is guaranteed to return a solution bounded by B for a set of instances I, then A is a bounded algorithm on I.*

BSS algorithms such as A$^*_\epsilon$ (Pearl and Kim 1982) and EES (Thayer and Ruml 2011) are forced to reopen states in order to be bounded.

In many cases, however, reopening can hurt performance (Chen et al. 2019). Thus, in the following context, we will discuss how to find bounded suboptimal solutions using the NR policy.

This paper defines the necessary and sufficient conditions on Φ to make BFS-NR a bounded algorithm.

3 Generalized Conditions

In our previous work (Chen and Sturtevant 2019), we presented 4 properties which are sufficient for BFS-NR to be bounded when using Φ :

$$\begin{cases} (a)\ \frac{\partial \Phi}{\partial x} > 0, \frac{\partial \Phi}{\partial y} > 0 \\ (b)\ \frac{\partial \Phi}{\partial y} \leq \frac{\partial \Phi}{\partial x} \\ (c)\ \Phi(0, w \cdot t) = \Phi(t, 0) = t \\ (d)\ \frac{\partial \Phi}{\partial x} + \frac{\partial \Phi}{\partial y} \leq 2 \end{cases}$$

These properties are limited for these reasons: (1) They implicitly assume that Φ is differentiable. (2) The bounding function $B(x)$ is fixed to be $B_w(x) = wx$. (3) The numeric value 2 in equation (d) is not justified. It is unclear whether the bound is tight or necessary.

This paper starts from the beginning, examining the properties one by one, and then describes precisely the set of Φ functions under which BFS-NR is a bounded algorithm. We start with a few intuitive assumption that Φ will satisfy.

Property 1. *For any given $\delta > 0$, $\Phi(x + \delta, y) > \Phi(x, y)$, $\Phi(x, y + \delta) > \Phi(x, y)$.*

Algorithm 1 Best-First Search Guided by Φ

```
Input: $start$, $goal$,$G$,$h$,$\Phi$,$RP$
1: Push($start$, $Open$)
2: while $Open$ not $empty$ do
3:     Remove state $s$ with minimum $\Phi(h(s), g(s))$ from $Open$
4:     if $s == goal$ then
5:         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 $g(s_i)$ of $s_i$ on $Open$ if shorter
11:        else if $s_i$ not on $Closed$ then
12:            Add $s_i$ to $Open$
13:        else if $RP$ == True then
14:            reopen $s_i$ if shorter
15:        end if
16:    end for
17: end while
18: return $failure$
```

Property 2. *For any given $\delta > 0$, $\Phi(x, y+\delta) \leq \Phi(x+\delta, y)$*

Property 1 means that the Φ-value grows monotonically along each axis. In terms of the behavior of the search algorithm, it is equivalent to say that for two states with same h-cost, the one with lower g-cost should be preferred; similarly, for two states with same g-cost, the one with lower h-cost should be preferred.

Property 2 means when there are two states with the same sum of h-cost and g-cost, the one with lower g-cost should not be preferred over the one with higher g-cost, although they could be equally preferred.

Property 3. $\Phi(x, 0) = x$.

The return value of the Φ function provides a partial ordering over states in OPEN. In this sense, there are many possible partial orderings that Φ could use. We assume $\Phi(x, 0) = x$ because (1) it gives Φ a semantic meaning: Φ is a lower bound on the optimal solution cost, not only for the start state, but also for every state expanded; and (2) it will simplify our remaining derivations.

With the first 3 assumptions made, now we can say something interesting about Φ.

Definition 2. *Let I_{AD} be all problem instances with admissible heuristics and I_{CON} be all instances with consistent heuristics. Furthermore, let I_{CON_S} stand for instances with strongly consistent heuristics, while I_{CON_W} stands for instances with weakly consistent heuristics. The instances with inconsistent but admissible heuristics are denoted by I_{INC}.*

Theorem 1. *If, for a given Φ, there exists an $x \geq 0$ such that $\Phi(x, 0) > \Phi(0, B(x))$, where B is the given bounding function, then BFS-NR using Φ is not guaranteed to be a bounded algorithm on I_{AD}.*

Proof. Suppose $\Phi(x, 0) > \Phi(0, B(x))$. Then, according to Property 1, there exists an $\epsilon > 0$ such that $\Phi(x, 0) = \Phi(0, B(x) + \epsilon)$. Then, we can create a problem instance that belongs to I_{AD} with 3 states, $start$, n, $goal$, as illustrated in Figure 2. In this example $h(start) = h(n) = x$, $c(start, n) = 0$, $c(start, goal) = B(x) + \frac{\epsilon}{2}$, $c(n, goal) = x$. On this problem BFS-NR will expand the goal and find a solution of cost $B(x) + \frac{\epsilon}{2}$, meaning it is not bounded. □

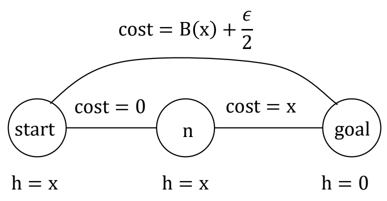

Figure 2: Illustration for Theorem 1

We now introduce two properties, the first of which, Property 4, is necessary according to Theorem 1.

Property 4. $\Phi(x, 0) \leq \Phi(0, B(x))$*, where B is the given bounding function.*

Property 5. $\Phi(x, 0) = \Phi(0, B(x))$*, where B is the given bounding function.*

Note that Property 5 is more strict than Property 4. Any Φ that meets Property 5 will also meet Property 4.

Weighted A*, *XDP* and *XUP* meet the stronger Property 5. This seems to be a natural choice, since it makes full use of the allowable suboptimality. But, some algorithms, such as Dynamic WA* (Pohl 1973), have $\Phi(x, 0) < \Phi(0, B(x))$. Also, in the next section, the Φ_{AB} we introduce, only meets Property 4 but not Property 5.

Given properties 1–4, we present an important result, the Φ-inequality. This inequality is used by Chen and Sturtevant (2019); we describe it here in a way that is consistent with our other derivations.

Definition 3. *Suppose BFS-NR expands a state n with cost $g(n)$. The Φ-inequality is $\Phi(h(n), g(n)) \leq g^*(n) + h(n)$*

$g^*(n) + h(n)$ is a lower bound on the optimal cost from $start$ to $goal$ through n. Thus, Φ can be interpreted as an estimate of the optimal path cost through the each state. (Note that this contrasts with how f is used in algorithms like WA* – as an estimate of the solution cost that will be found.)

In the following, we will show that BFS-NR is bounded if and only if the Φ-inequality holds. The Φ function distributes the potential suboptimality across a path, potentially allowing more suboptimality at the beginning or end of the search. Intuitively, the Φ-inequality guarantees that the local suboptimality is bounded such that a bounded suboptimal path to the goal always exists.

Theorem 2. *Suppose that Φ satisfies Property 1 - 3 and Property 4. Then, if the Φ-inequality holds for every state expanded, BFS-NR is a bounded algorithm on I_{AD}.*

Proof. Suppose the Φ-inequality holds for every state expanded, then it also holds for $goal$. Therefore, $\Phi(0, g(goal)) \leq g^*(goal) + h(goal) = C^*$. According to Property 4, $C^* \leq \Phi(0, B(C^*))$. According to Property 1, $g(goal) \leq B(C^*)$, therefore the solution is bounded. □

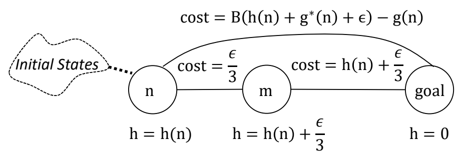

Figure 3: Illustration for Theorem 4

Since Property 5 is stronger than Property 4, then Theorem 2 also holds for Property 5, which we state below, as Corollary 3.

Corollary 3. *Suppose that Φ satisfies Property 1 - 3 and Property 5. Then, if the Φ-inequality holds for every state expanded, BFS-NR is a bounded algorithm on I_{AD}.*

Theorem 2 suggests that the Φ-inequality is a sufficient condition for BFS-NR to be bounded. But, is it a necessary condition? The short answer is, yes. The next theorem, Theorem 4, tells us that this condition must hold for every single expansion. Otherwise, there could be some instance $I \in I_{AD}$ on which BFS-NR will not return a bounded solution. This theorem requires Property 5; there is no equivalent theorem when only Property 4 holds.

Theorem 4. *Suppose Φ satisfies Property 1 - 3 and Property 5. Then, only if the Φ-inequality holds for every state expanded will BFS-NR be bounded algorithm on I_{AD}.*

Proof. We prove the contrapositive.

Suppose there exists one state n expanded by BFS-NR on a problem $I \in I_{AD}$ that violates the Φ-inequality. That is, $\Phi(h(n), g(n)) > h(n) + g^*(n)$. Then, we can show that BFS-NR is not bounded on I. Suppose BFS-NR expands a few initial states and comes to the situation illustrated in Figure 3, where BFS-NR decides to expand the state n whose $\Phi(h(n), g(n)) > h(n) + g^*(n)$ without knowing which states are after n. Since Φ is continuous, there must exist $\epsilon > 0$ such that $\Phi(h(n), g(n)) = h(n) + g^*(n) + \epsilon$. Now, we can alter the remaining graph such that there are only 2 successors of n: m and $goal$. $c(n, goal)$ satisfies that $\Phi(0, g(n) + c(n, goal)) = \Phi(h(n), g(n))$. $h(m) = h(n) + \frac{\epsilon}{3}$, $c(n, m) = \frac{\epsilon}{3}$, $c(m, goal) = h(n) + \frac{\epsilon}{3}$.

Since the algorithm just expanded n, we know n was the state with minimum priority on OPEN. Thus, once we expand n and put m and $goal$ on OPEN, the $goal$ will be chosen for expansion immediately, since it will be the state with minimum Φ value. In this case, $\Phi(0, g(goal)) = h(n) + g^*(n) + \epsilon = \Phi(0, B(h(n) + g^*(n) + \epsilon))$. However, there exists a path from $start$-n-m-$goal$, whose cost is $h(n)+g^*(n)+\frac{2\epsilon}{3}$. In this problem, $C^* \leq h(n)+g^*(n)+\frac{2\epsilon}{3}$, while the solution we returned has cost $B(h(n)+g^*(n)+\epsilon)$, which is strictly greater than $B(C^*)$.

Hence, the BFS-NR is not bounded on I. □

In short, if BFS-NR expands a state that violates the Φ-inequality, we can construct an new problem instance where BFS-NR will not be bounded.

Thus, to understand when BFS-NR is guaranteed to find bounded-suboptimal solutions, we must study the properties that will cause the Φ-inequality to hold. As we will now show, the Φ-inequality holds for each expansion if and only if we have Properties 1–3, 5–6, and a consistent heuristic.

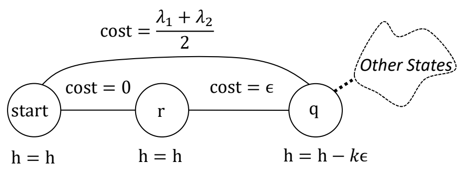

Figure 4: Illustration for Theorem 5

Theorem 5. *Suppose that Φ satisfies Property 1 - 3. Then there does not exist a Φ that can guarantee that the Φ-inequality holds for all states expanded on problem instances $I \in I_{INC}$.*

Proof. [By construction.]

We create a parametric example where using any Φ function that meets Property 1 - 3 with an inconsistent heuristic will cause BFS-NR to not be bounded. As is illustrated in Figure 4, create a $I \in I_{INC}$ where $start$ has 2 successors, q and r. $h(start) = h$, $h(q) = h - k\epsilon$, $h(r) = h$, $c(start, r) = 0$, $c(r, q) = \epsilon$. where $k > 1$. Then we can see that the heuristic is inconsistent between q and r.

Since Φ is continuous, we can let $\Phi(h - k\epsilon, \lambda_1) = h - k\epsilon + \epsilon$ and $\Phi(h - k\epsilon, \lambda_2) = h$. According to Property 1, $\lambda_1 < \lambda_2$. Then set $c(start, q) = \frac{\lambda_1+\lambda_2}{2}$. We can compute

$$f(q) = \Phi(h(q), g(q)) = \Phi(h - k\epsilon, \frac{\lambda_1 + \lambda_2}{2})$$

$$f(r) = \Phi(h(r), g(r)) = \Phi(h, 0) = h$$

Since $\lambda_1 < \lambda_2$, according to Property 1,

$$\Phi(h - k\epsilon, \frac{\lambda_1 + \lambda_2}{2}) < \Phi(h - k\epsilon, \lambda_2) = h$$

, therefore $f(q) < f(r)$. Thus we will expand q instead of r.

On the other hand,

$$h(q) + g^*(q) = h(q) + c(p, r) + c(r, q) = h - k\epsilon + \epsilon$$

$$\Phi(h - k\epsilon, \frac{\lambda_1 + \lambda_2}{2}) > \Phi(h - k\epsilon, \lambda_1) = h - k\epsilon + \epsilon$$

which breaks the Φ-inequality on state q. □

Theorem 6. *Suppose that Φ satisfies Property 1 - 3. Then, there does not exist a Φ that can guarantee that BFS-NR is a bounded algorithm for instances $I \in I_{INC}$.*

Proof. Theorem 4 tells us the Φ-inequality must hold for every single expansion. Theorem 5 shows that it will not on problems in I_{INC}. □

If the heuristic is not consistent, then without further assumptions, we cannot guaranteed that the algorithm returns bounded solution on every instance. It may return bounded solutions for some instances, but not for all. Thus, Theorem 6 indicates that consistent heuristics are a necessary condition to guarantee that BFS-NR is a bounded algorithm. This is not counter-intuitive; A^* needs to perform re-expansions to guarantee optimal solutions when the heuristic is inconsistent (Martelli 1977; Felner et al. 2011).

Now we will study Φ functions given problems in I_{CON}.

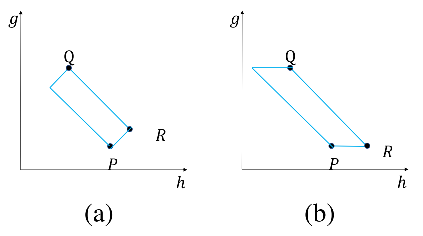

Figure 5: The successor parallelogram when heuristic is (a) strongly consistent (b) weakly consistent

Definition 4. *Successor Parallelogram.*

Figure 5 (a) and (b) illustrate the successor parallelogram for strongly consistent heuristics and weakly consistent heuristics, respectively. Suppose that state q is a descendant of state p. p is an expanded state. The h-cost and g-cost of p are $h(p)$, $g(p)$, respectively. We put a point P on h-g plane, whose coordinates are $(h(p), g(p))$. We put another point Q, with coordinates $(h(q), g(p) + d(p, q))$. This point represents the cost of q if we reach q optimally from p.

After that, we draw out two 45° straight line from P and Q (45° straight lines correspond to consistency). For a directed graph, we draw out two -45° straight line from P and Q; for undirected graph, we draw out two horizontal lines from P and Q.

These four lines will give us a parallelogram in every possible case. Such a parallelogram is called a successor parallelogram.

As the name indicates, we can prove that for any state p_1 on the optimal path from p to q, p_1 must be in the successor parallelogram, given that the heuristic is consistent.

Definition 5. *We define R as the right most corner of successor parallelogram.*

Point R is interesting because that is the point with the maximum Φ value in the parallelogram.

Lemma 7. *Point R is the point with maximum Φ value.*

Proof. According to Property 1, the maximum Φ value is on the line segment QR.

For any given point S in line segment QR, Let $x = x_S$, $y = y_R$, $\delta = x_R - x_S$. Finally, According to Property 2, we know $\Phi(x_R, y_R) \geq \Phi(x_S, y_S)$. □

The next property is the key Property we need for avoiding reexpansions. It is a sufficient condition to guarantee the Φ-inequality under Property 4. It is necessary when Φ meets Property 5. This condition is different on directed/undirected graphs.

Property 6. *For a strongly consistent heuristic: for any given $\delta > 0$, $\Phi(x + \delta, y + \delta) \leq \Phi(x, y) + 2\delta$*

For a weakly consistent heuristic: for any given $\delta > 0$, $\Phi(x + \delta, y) \leq \Phi(x, y) + \delta$

Theorem 8. *Assume BFS-NR is using a priority function Φ with a bounding function $B(x)$ on a problem instance $I \in I_{CON}$. If there exists h_0, g_0, δ_0 such that (1) for a strongly consistent heuristic $\Phi(h_0 + \delta_0, g_0 + \delta_0) > \Phi(h_0, g_0) + 2\delta_0$, or (2) for a weakly consistent heuristic $\Phi(h_0 + \delta_0, g_0) > \Phi(h_0, g_0) + \delta_0$, then Φ-inequality is not always guaranteed to hold.*

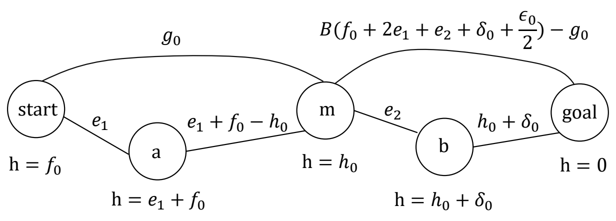

Figure 6: The examples for Theorem 8

Proof. [By construction.]

We will construct an example where BFS-NR fails. In the following proof, $e_2 = \delta_0$ if the heuristic is strongly consistent, $e_2 = 0$ if the heuristic is weakly consistent.

Let $\Phi(h_0, g_0) = f_0$, $\Phi(h_0 + \delta_0, g_0 + e_2) = \Phi(h_0, g_0) + \delta_0 + e_2 + \epsilon_0$. Figure 6 shows the instance where this holds where the h-costs are $h(m) = h_0$, $h(a) = f_0 + e_1$, $h(b) = h_0 + \delta_0$, and the edge costs are $c(start, m) = g_0$, $c(start, a) = e_1$, $c(a, m) = e_1 + f_0 - h_0$, $c(m, goal) = B(f_0 + 2e_1 + e_2 + \delta_0 + \frac{\epsilon_0}{2}) - g_0$, $c(m, b) = e_2$, $c(b, goal) = \delta + h_0$.

Where

$$e_1 \leq \frac{h_0 + g_0 - f_0}{2} \tag{1}$$

and

$$e_1 < \frac{\epsilon_0}{4} \tag{2}$$

.

(Note that $\Phi(h_0, g_0) = f_0$ and $h_0 + g_0 = \Phi(h_0 + g_0, 0)$. According to Property 2, $\Phi(h_0, g_0) \leq \Phi(h_0 + g_0, 0)$, which guarantees $h_0 + g_0 - f_0 \geq 0$. Therefore, there must exist a non-negative e_1.)

BFS-NR will expand $start$ first and put m and a on OPEN. At that point, $f(m) = \Phi(h_0, g_0) = f_0$, $f(a) = \Phi(h(a), g(a)) = \Phi(f_0 + e_1, e_1)$, which means we will expand m before a.

According to Equation 1, the cost of $start$-a-m is $2e_1 + f_0 - h_0 \leq (h_0 + g_0 - f_0) + f_0 - h_0 = g_0$, which means $start$-a-m is shorter than $start$-m.

Since we do not re-open nodes, when we reach b, $h(b) = h_0 + \delta_0, g(b) = g(m) + c(m, b) = g_0 + e_2$.

Since $f(b) = \Phi(h(b), g(b))$, we can compute that $f(b) = f_0 + \delta_0 + e_2 + \epsilon_0$.

At the same time,

$f(goal) = \Phi(0, g(goal))$
$= \Phi(0, B(f_0 + 2e_1 + \delta_0 + e_2 + \frac{\epsilon_0}{2}))$
$= f_0 + 2e_1 + \delta_0 + e_2 + \frac{\epsilon_0}{2}$

According to Equation 2, $f_0 + 2e_1 + \delta_0 + e_2 + \frac{\epsilon_0}{2}$
$< f_0 + 2\frac{\epsilon_0}{4} + \delta_0 + e_2 + \frac{\epsilon_0}{2}$
$= f_0 + \delta_0 + e_2 + \epsilon_0$

Therefore, $f(goal) < f(b)$ and we will expand $goal$. In such a case, the total cost of the path will be $B(f_0 + 2e_1 + e_2 + \delta_0 + \frac{\epsilon_0}{2})$.

However, the shortest path from $start$ to $goal$ should be $start$-a-m-b-$goal$, whose cost is $f_0 + 2e_1 + e_2 + \delta_0$. We can see that

$B(f_0 + 2e_1 + e_2 + \delta_0 + \frac{\epsilon_0}{2}) > B(f_0 + 2e_1 + e_2 + \delta_0)$, which means the expansion of the *goal* breaks the Φ-inequality. □

Theorem 9. *Assume BFS-NR is using a priority function* Φ *which meets Properties 1 to 4 and 6 on an problem instance* $I \in I_{CON}$*. Then for all expansions the* Φ*-inequality holds.*

The proof is omitted due to space, but follows from the same argument as Chen and Sturtevant (2019).

Theorem 10. *For BFS-NR, assume that* Φ *satisfies Property 1 to 4. Then if Property 6 holds, the algorithm will be bounded on* I_{CON} *.*

Proof. Theorem 9 tells us that for all expansions the Φ-inequality will hold. According to Theorem 2, we can draw this conclusion. □

To summarize the theoretical results: (1) on instances in I_{INC}, BFS-NR is not bounded. (2) On instances in I_{CON}, to make BFS-NR bounded, Φ need to meet a few properties. We always assume Properties 1 to 3. Then, Property 6 is sufficient and necessary condition for BFS-NR to be a bounded algorithm on I_{CON} when Φ meets Property 5.

If Φ only meets Property 4, then Property 6 is sufficient but not necessary for BFS-NR to be bounded algorithm on I_{CON}.

4 New Φ Functions

In this section, we introduce methods for constructing Φ functions that have parameter that can be tuned.

4.1 Additive Bounds

First we look at additive suboptimality bounds. For reference, the Φ function used by WA* with weight w corresponds to straight contour plots with a slope of $-w$, as shown in Figure 1(a). We can achieve constant suboptimality with Φ_{AB} and a parameter K, $K \geq \gamma$ that can be tuned.

$$\Phi_{AB}(x, y) = \begin{cases} x + \frac{K-\gamma}{K} y & y < K \\ x + y - \gamma & y \geq K \end{cases} \tag{3}$$

Figure 1(b) illustrates Φ_{AB}, a priority function that allows a BFS-NR to find a solution with cost $C^* + \gamma$. The behavior of this algorithm is easy to understand: Φ_{AB} searches with WA* and then switches to A*. There is one parameter K, which defines the size of the region where WA* is performed. A larger K performs WA* for longer distances; as a trade off, the weight of WA*, $\frac{K}{K-\gamma}$, must be smaller. We can verify that all the required properties are satisfied. Note that when $x < K - \gamma$, $\Phi_{AB}(x, 0) < \Phi_{AB}(0, B(x))$; when $x \geq K - \gamma$, $\Phi_{AB}(x, 0) = \Phi_{AB}(0, B(x))$.

An interesting fact is that reversing the approach by doing A* and then switching to WA* won't work. The following Φ is not guaranteed to find bounded suboptimal solutions because when $x \geq b$, $\Phi(x, 0) = \Phi(0, B(x))$, but $\Phi(x + \delta, y + \delta) > 2\delta$.

$$\Phi(x, y) = \begin{cases} x + y & x < b \\ \frac{\gamma+b}{b} x + y - \gamma & x \geq b \end{cases}$$

Domain	Φ	Suboptimality Bound/ (w)			
		1.5	2.0	3.0	10.0
15-Puzzle	Φ_{WA^*}	273,101	40,544	11,600	3,758
	Φ_{XDP}	166,447	21,338	7,550	3,586
	Φ_{XUP}	373,023	71,014	16,934	3,859
	Φ_{pwXD}	70,799	**11,230**	**4,978**	4,621
	Φ_{pwXU}	-	2,792,255	823,029	67,065
	Φ_{z1}	**43,009**	12,323	7,397	**3,321**
Heavy 15-Puzzle	Φ_{WA^*}	333,320	114,848	57,778	44,207
	Φ_{XDP}	200,318	82,295	**48,203**	43,141
	Φ_{XUP}	702,468	161,126	82,916	34,065
	Φ_{pwXD}	101,498	**52,386**	50,664	**30,438**
	Φ_{pwXU}	-	4,335,932	1,970,396	125,473
	Φ_{z1}	**95,674**	53,274	54,537	34,315
Heavy Pancake Puzzle	Φ_{WA^*}	8,498,635	973,556	22,732	33
	Φ_{XDP}	2,460,235	93,355	1,123	**20**
	Φ_{XUP}	-	7,072,634	404,622	318
	Φ_{pwXD}	**988,899**	**23,718**	**367**	29
	Φ_{pwXU}	51,245,052	-	11,244,399	24,377
	Φ_{z1}	3,323,705	535,430	98,353	1,087

Table 1: Average state expansions for each priority functions on different domains

Domain	Φ	Suboptimality Bound/ (w)			
		1.5	2.0	3.0	10.0
15-Puzzle	Φ_{WA^*}	56.37	63.71	77.99	118.87
	Φ_{XDP}	56.79	64.35	78.65	121.07
	Φ_{XUP}	55.87	63.73	77.69	121.99
	Φ_{pwXD}	66.83	75.23	89.33	119.57
Heavy 15-Puzzle	Φ_{WA^*}	442.34	509.10	619.62	930.06
	Φ_{XDP}	454.28	516.78	620.98	933.08
	Φ_{XUP}	438.34	502.56	619.58	988.26
	Φ_{pwXD}	519.14	586.26	685.66	1038.12
Heavy Pancake	Φ_{WA^*}	81.18	82.78	88.94	105.16
	Φ_{XDP}	81.56	84.34	92.78	106.26
	Φ_{XUP}	80.21	82.08	86.70	102.30
	Φ_{pwXD}	82.22	86.36	95.74	104.90

Table 2: Average path costs for each priority functions on different domains

4.2 Piecewise Linear Bounds

A general piecewise function for $B_w(x)$ is:

$$\Phi_{B_w}(x, y) = \begin{cases} x + y & y < K_1 x \\ A(y + Bx) & K_1 x \leq y < K_2 x \\ \frac{1}{w}(x + y) & y \geq K_2 x \end{cases} \tag{4}$$

Figures 1(c)-(d) illustrate several possible contour plots for this priority function, where (c) is the most general form. The behavior of the most general Φ function will perform optimally near *start* and *goal* with suboptimal search in the middle region (where $K_1 x \leq y < K_2 x$). Note that equation set has 4 parameters, K_1,K_2, A and B. However, there are only 2 free variables. Once we choose K_1, A is determined. As for the remaining variables, if we fix one, the other is determined. We manually choose K_1 and $B = (2w - 1)$, and then compute K_2 and A.

Intuitively, there are a few advantages of this approach over previous *XUP* and *XDP* functions. Firstly, *XDP* and *XUP* are quadratic functions and are more complicated to

		Suboptimality Bound/ (w)		
Domain	Φ	2.0	3.0	10.0
Data-network	Φ_{WA^*}	15	14	16
	Φ_{pwXD}	16	16	16
Spider	Φ_{WA^*}	13	17	20
	Φ_{pwXD}	18	18	18
Termes	Φ_{WA^*}	11	12	16
	Φ_{pwXD}	12	13	15
Total of all 8 domians	Φ_{WA^*}	69	72	77
	Φ_{pwXD}	74	79	74

Table 3: IPC Problems solved (each domain contains 20 problems)

		Suboptimality Bound/ (w)		
Domain	Φ	2.0	3.0	10.0
Data-network	Φ_{WA^*}	302,558	996,910	827,652
	Φ_{pwXD}	328,049	668,416	524,842
Spider	Φ_{WA^*}	44,904	30,017	48,977
	Φ_{pwXD}	12,650	37,827	112,482
Termes	Φ_{WA^*}	21,742,548	18,605,174	12,429,415
	Φ_{pwXD}	13,941,669	6,327,308	7,023,560

Table 4: Average expansions on commonly solved problems

implement and analyze, involving a square root operation. The piecewise curves are slightly easier to implement in A*, although both approaches only require small changes to A* compared to alternate algorithms. But, these new priority functions also give us more degrees of freedom to distribute the suboptimality. When we are near $start$, we usually need to expand a few states optimally to validate the final solution; when we are close to $goal$, the heuristics are usually perfect, which means the path we find is typically optimal in those portions. It is in the middle where we actually have the freedom to find suboptimal paths - and we can use a higher degree of suboptimality there, instead of evenly distributing it across the solution path, as WA* does. Φ_{z1} implements this approach, as shown in Figure 1(c).

$$\Phi_{z1} = \begin{cases} y + x & y < \frac{1}{w}x \\ \frac{w+1}{2w^2-w+1}(y + (2w-1)x) & \frac{1}{w}x \le y < \frac{2w^2+w+1}{w-1}x \\ \frac{1}{w}(y+x) & \frac{2w^2+w+1}{w-1}x \le y \end{cases} \tag{5}$$

We can get simpler forms by setting $K_1 = 0$ or $K_2 = \infty$ resulting in a 2-case piecewise function. We call these the piecewise Convex Downward (pwXD) function Figure 1(d) and piecewise Convex Upward (pwXU) Figure 1(e). Each function has one free parameter, K. In our experiments, we always use $K = 2w - 1$ giving:

$$\Phi_{pwXD} = \begin{cases} y + x & y < \frac{K-w}{w-1}x \\ \frac{1}{w}(y + Kx) & \frac{K-w}{w-1}x \le y \end{cases} \tag{6}$$

$$\Phi_{pwXU} = \begin{cases} \frac{1}{K}y + x & y < \frac{K(w-1)}{K-w}x \\ \frac{1}{w}(y+x) & \frac{K(w-1)}{K-w}x \le y \end{cases} \tag{7}$$

		Additive Suboptimality Bound/ (γ)				
Algorithm	Parameter	0	4	16	64	256
BFS^{F_γ}		1,240	1,182	1,219	986	1,078
Φ_{AB}	$K = \gamma + 1$	1,240	1,237	1,169	711	**368**
Φ_{AB}	$K = 2\gamma$	1,240	1,233	1,107	**485**	481
Φ_{AB}	$K = h_0$	1,240	**1,137**	**929**	518	**368**

Table 5: Average state expansions for algorithm on DAO for additive bound ($h_0 = max\{h(start), \gamma + 1\}$)

		Additive Suboptimality Bound/ (γ)				
Algorithm	Parameter	0	4	16	64	256
BFS^{F_γ}		12,325	7,494	**878**	**356**	235
Φ_{AB}	$K = \gamma + 1$	12,325	12,265	7,186	446	**204**
Φ_{AB}	$K = 2\gamma$	12,325	12,010	1,972	399	399
Φ_{AB}	$K = h_0$	12,325	**6,273**	1,160	446	**204**

Table 6: Average state expansions for algorithm on STP for additive bound ($h_0 = max\{h(start), \gamma + 1\}$)

Not that for Equation (6), if $K > 2w - 1$, then property 6 is violated.

5 Experimental Results

In the following, we run experiments on a server with 16 GB RAM and 24 processors 6-core Intel Xeon CPU E5-2630 (2.30GHz). We tested on the 15-puzzle with unit edge costs and the heavy tile setting, where the cost of moving tile X is X. Manhattan Distance (MD) and PDB heuristics are 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). We also tested on 1098 problems with solution length $[128 - 132)$ from the Dragon Age: Origins benchmark set (Sturtevant 2012). Further experiments evaluated performance on a heavy variant of pancake puzzle (Gilon, Felner, and Stern 2016), where the cost of flipping a prefix $(V[1] \cdots V[i+1])$ is the $max(V[1]; V[i+1])$. We also use their *HGAP* heuristic. The problem set was provided by (Chen and Sturtevant 2019), which consists of 50 randomly generated 12-pancake instances. “-” means the algorithm is not able to solve all instances due to running out of memory.

Results on linear bounds in Table 1 show that Φ_{pwXD} and Φ_{z1} typically perform well. For instance, on the heavy pancake puzzle with $w = 1.5$, Φ_{pwXD} is 8x better than WA* and 2.48x better than Φ_{XDP}. The cost for this improved performance is slightly worse solution quality. However, under the problem definition the task is to find a solution within the bound, not to optimize solution quality.

Results on additive suboptimality bounds are presented in Tables 6 (for sliding tile puzzle) and Table 5 (for grid maps). In our experiments, we tried 3 different K: $\gamma + 1$, 2γ and $h_0 = max\{h(start), \gamma + 1\}$, and compared against BFS^{F_γ} (Valenzano et al. 2013). In most cases h_0 gives us the best performance. On grid maps, the poor performance of BFS^{F_γ} is fully due to reopening states.

We also run experiments in FastDownward with the CE-

GAR heuristic (Seipp and Helmert 2014) on planning domains from IPC 2018 problem sets. Table 3 shows the number of problems WA* and piecewise XD can solve; while Table 4 demonstrate the average number of node expanded for solving the common set. These problems were run with a 15 minute timeout on a cluster with 2.1 GhZ Intel Xeon E5-2683 CPUs where each job was given 4 GB RAM. Finally, Table 4 shows that the algorithms with fewer node expansions usually find a higher cost solution.

6 Conclusions and Future Work

This work studies the general properties of Φ which guarantee BFS-NR will find bounded suboptimal solutions, providing necessary and sufficient conditions on Φ. Experimental results show additional improvements over existing methods for both linear and constant suboptimality bounds.

Acknowledgments

This work was funded by the Canada CIFAR AI Chairs Program. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Benton, J.; Do, M.; and Ruml, W. 2007. A simple testbed for on-line planning. In *ICAPS Workshop on Moving Planning and Scheduling Systems into the Real World*.

Bulitko, V.; Björnsson, Y.; Sturtevant, N. R.; and Lawrence, R. 2011. Real-time heuristic search for pathfinding in video games. In *Artificial Intelligence for Computer Games*, 1–30. Springer.

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.; Sturtevant, N. R.; Doyle, W.; and Ruml, W. 2019. Revisiting Suboptimal Search. *Symposium on Combinatorial Search (SoCS)* 18–25.

Felner, A.; Zahavi, U.; Holte, R.; Schaeffer, J.; Sturtevant, N.; and Zhang, Z. 2011. Inconsistent heuristics in theory and practice. *Artificial Intelligence (AIJ)* 175(9-10): 1570–1603.

Gilon, D.; Felner, A.; and Stern, R. 2016. Dynamic Potential Search—A New Bounded Suboptimal Search. In *Symposium on Combinatorial Search (SoCS)*, 36–44.

Korf, R. E. 1985. Depth-first Iterative-Deepening: An Optimal Admissible Tree Search. *Artificial Intelligence* 27: 97–109.

Martelli, A. 1977. On the complexity of admissible search algorithms. *Artificial Intelligence* 8(1): 1–13.

Pearl, J.; and Kim, J. H. 1982. Studies in Semi-Admissible Heuristics. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-4(4): 392–399. ISSN 0162-8828. doi:10.1109/TPAMI.1982.4767270.

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

Pohl, I. 1973. 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*, 12–17. Morgan Kaufmann Publishers Inc.

Seipp, J.; and Helmert, M. 2014. Diverse and additive Cartesian abstraction heuristics. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 289–297.

Sepetnitsky, V.; Felner, A.; and Stern, R. 2016. Repair policies for not reopening nodes in different search settings. In *Symposium on Combinatorial Search (SoCS)*, 81–88.

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

Thayer, J. T.; and Ruml, W. 2008. Faster than Weighted A*: An Optimistic Approach to Bounded Suboptimal Search. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 355–362.

Thayer, J. T.; and Ruml, W. 2011. Bounded suboptimal search: A direct approach using inadmissible estimates. In *International Joint Conference on Artifical Intelligence (IJCAI)*, 674–679.

Valenzano, R.; Arfaee, S. J.; Stern, R.; Thayer, J.; and Sturtevant, N. 2013. Using Alternative Suboptimality Bounds in Heuristic Search. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 233–241.

Valenzano, R.; Sturtevant, N.; and Schaeffer, J. 2014. Worst-Case Solution Quality Analysis When Not Re-Expanding Nodes in Best-First Search. In *AAAI Conference on Artificial Intelligence*, 885–892.