Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

Worst-Case Solution Quality Analysis When Not Re-Expanding Nodes in Best-First Search

Richard Valenzano
University of Alberta
valenzan@ualberta.ca

Nathan R. Sturtevant
University of Denver
sturtevant@cs.du.edu

Jonathan Schaeffer
University of Alberta
jonathan@ualberta.ca

Abstract

The use of inconsistent heuristics with A* can result in increased runtime due to the need to re-expand nodes. Poor performance can also be seen with Weighted A* if nodes are re-expanded. While the negative impact of re-expansions can often be minimized by setting these algorithms to never expand nodes more than once, the result can be a lower solution quality. In this paper, we formally show that the loss in solution quality can be bounded based on the amount of inconsistency along optimal solution paths. This bound holds regardless of whether the heuristic is admissible or inadmissible, though if the heuristic is admissible the bound can be used to show that not re-expanding nodes can have at most a quadratic impact on the quality of solutions found when using A*. We then show that the bound is tight by describing a process for the construction of graphs for which a best-first search that does not re-expand nodes will find solutions whose quality is arbitrarily close to that given by the bound. Finally, we will use the bound to extend a known result regarding the solution quality of WA* when weighting a consistent heuristic, so that it also applies to other types of heuristic weighting.

1 Introduction

When an A* search is using an *admissible* heuristic, any solutions it finds are guaranteed to be optimal (Hart, Nilsson, and Raphael 1968). If the heuristic is also *consistent*, then a node will only be expanded when the lowest cost path to it has been found. This cannot be guaranteed when the heuristic is *inconsistent*, in which case A* may expand the same node multiple times. In some domains, these node *re-expansions* can dominate runtime and thereby greatly decrease algorithm performance (Martelli 1977).

Re-expansions can also occur when using Weighted A* (WA*) (Pohl 1970). This algorithm performs an A* search using the *inadmissible* and inconsistent heuristic of $H = w \cdot h$, where h is an admissible heuristic and $w \geq 1$ is a constant. To avoid the negative impact of re-expansions, it is common to modify WA* so that it never expands any node more than once. For example, this technique has been applied in domains such as robot path planning (Likhachev, Gordon, and Thrun 2003) and binary decision diagram minimization (Ebendt and Drechsler 2009).

Yet there is still much that is not fully understood about the impact that not re-expanding nodes has on a search. For example, when this technique has been tested empirically it has been shown to improve algorithm runtime in some problems while harming it in others, and it also typically decreases the quality of the solutions found (Thayer and Ruml 2008; Ebendt and Drechsler 2009). However, there are still no theoretical results that identify the properties of a state-space that determine whether or not re-expanding nodes is beneficial, and the only known guaranteed bounds on the quality of solutions found when using this technique are when weighting a consistent heuristic in WA* and A$^*_\epsilon$.

The goal of this paper is to begin to address this gap in our understanding of this commonly used technique. Our specific focus will be on formally analyzing how not re-expanding nodes can impact the solution quality. In particular, we will show that the loss in quality that can result from not re-expanding nodes can be bound based on the amount of inconsistency along optimal solution paths. This will be proven for a large class of best-first search algorithms and will apply regardless of whether the heuristic is admissible or inadmissible. The bound will then be used to show that for admissible heuristics, the worst-case when using an A* which does not re-expand nodes is to find solutions that are quadratic in the optimal solution cost. We will then identify a family of worst-case graphs and corresponding heuristics for which the given bound is exact. Finally, we will consider the known bound on solution quality that is specific to the use of a consistent heuristic with WA*, and extend this bound so that it applies to other types of heuristic weighting.

2 Not Re-Expanding Nodes Can Be Helpful

To motivate the need to study the relationship between not re-expanding nodes and solution quality, we consider two examples in this section in which algorithm runtime suffers greatly when this technique is not used. The first is a well-known family of graphs $G_k(V, E)$ that were identified by Martelli (1977), where G_k has k vertices for any $k \geq 4$, each with a corresponding admissible but inconsistent heuristic. We omit a full description of G_k due to space constraints. Martelli showed that if an A* search which re-expands nodes is used on a problem that has G_k as a subgraph, then the search will require $O(2^{|V|})$ expansions to explore that region of the search space. Martelli introduced a modification to A*

Weight	Ratio of Total Nodes	Re-Expansion Percentage	NR Ratio of Total Nodes
1.0	1.0	0%	1.0
1.5	0.86	16%	0.74
2	1.52	65%	0.56
5	3.17	90%	0.33
10	3.28	91%	0.30

Table 1: The impact of re-expanding nodes on WA*.

which only requires $O(|V|^2)$ expansions to traverse through such subgraphs while still ensuring that all solutions found will still be optimal. However, by simply not re-expanding nodes we guarantee that at most $|V|$ expansions are needed, though with a loss of guaranteed optimality.

As an example in which it is essential to not perform re-expansions when using an inadmissible heuristic, consider the use of WA* on the $35,360$ pathfinding problems from the 10 8-connected, 512×512 maps with 40% random obstacles given by Sturtevant (2012). The average optimal solution cost of these problems is 733.87, and A* performs an average of $36,003$ expansions per problem when using the octile heuristic. When it is not weighted, this heuristic is consistent. Table 1 shows the average performance of WA* on these problems for a variety of weights. The first column shows the weight, the second column shows the total number of expansions relative to A*, and the third column shows what percentage of the total expansions were re-expansions. The table shows that the higher weights actually expand more nodes than A*, largely because of re-expansions. For example, 91% of the expansions made by the weight 10 search are re-expansions, which is why it is much slower than A* despite expanding only 30% as many unique nodes. The final column of the table shows the total number of node expansions relative to A* when WA* does not re-expand nodes. As shown, all weights greater than one now result in a substantially faster search than A*. Clearly, not performing re-expansions is necessary in this domain.

Since the above examples suggest that not re-expanding nodes can lead to significant runtime improvements, we now turn to formally analyzing the impact of this technique on solution quality. We will later compare the actual impact on solution quality of not re-expanding nodes in these problems to what the formal analysis predicts in Section 4.4.

3 Formal Notation

In this section, we will define the notation that will be used throughout the paper.

Problem definition. The tasks we consider consist of finding a *path* (defined below) in a graph $G = (V, E)$ from a given *start node* or *vertex* $n_{start} \in V$ to one of a set of goal nodes $V_{goals} \subseteq V$. If $(n, m) \in E$ (*ie.* there is an edge from n to m), m is called a *successor* or *child* of n. $\kappa(n, m)$ will denote the cost of this edge, and we assume that for any $(n, m) \in E$, $\kappa(n, m) \geq 0$. All edges will also be assumed to be directed edges. Note that we will often omit references to G when the graph in question is clear from the context.

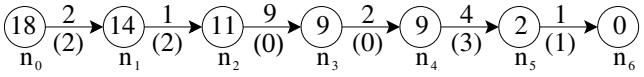

Figure 1: Example path with INC_H values shown below the edge costs in parentheses.

Paths and path costs. A *path* is a sequence of nodes $P = n_0, ..., n_k$ in a given problem such that $(n_i, n_{i+1}) \in E$ for all $0 \leq i < k$. The *cost* of a path is given by the sum of the costs of the edges along the path. P will be called a *solution path* to a task if $n_0 = n_{start}$ and $n_k \in V_{goals}$, and an *optimal solution path* if it has the lowest cost of all solution paths. C^* will denote the cost of all optimal solution paths. We also use $g^*(n)$ to denote the cost of the optimal path from n_{start} to n, $h^*(n)$ as the cost of the lowest cost path from n to any node in V_{goals} (where $h^*(n) = \infty$ if no such path exists), and $g^*(n, m)$ to denote the cost of the optimal path from n to m. This means that $C^* = g^*(n_g) = g^*(n_{start}, n_g)$ where n_g is the goal node with the lowest cost path to it.

Heuristics and heuristic properties. We define a *heuristic function* $H : V \rightarrow \mathbb{R}^{\geq 0}$ as any function from V to the non-negative real numbers. The only restrictions we place on H are that for any n, $H(n) \geq 0$, $H(n) = 0$ if $n \in V_{goals}$, and $H(n) \neq \infty$ if $h^*(n) \neq \infty$. These assumptions correspond to requiring that H never returns a negative value, always returns 0 for goal nodes, and does not incorrectly identify any node from which a goal node is reachable as a dead-end, respectively.

A heuristic H is *admissible* if for any node n, $H(n) \leq h^*(n)$. Unless otherwise stated, we do not assume admissibility, which is why we use H to refer to heuristics instead of h which has most often been used in the literature to denote an admissible heuristic. H is said to be *consistent* if for any edge $(p, c) \in E$, $H(p) \leq H(c) + \kappa(p, c)$, and *inconsistent* if there exists an edge $(p, c) \in E$ such that $H(p) > H(c) + \kappa(p, c)$. We define the *inconsistency of a heuristic H on an edge* from p to c as how far away the heuristic H is from being consistent on the edge from p to c. This value, denoted by $INC_H(p, c)$, is given by

$$INC_H(p, c) = \max(H(p) - H(c) - \kappa(p, c), 0)$$

Notice $INC_H(p, c) > 0$ if and only if H is inconsistent on edge (p, c), and 0 otherwise. As such, the standard definition of consistency is equivalent to requiring that $INC_H(p, c) = 0$ for any $(p, c) \in E$. We also define the *inconsistency of a heuristic H along a path P* of nodes as the sum of the inconsistency of H on each of the edges along P.

As an example of how these metrics are calculated, consider the path $n_0, ..., n_6$ given in Figure 1. In this figure, the heuristic values are shown inside the nodes, the edge costs are shown above the edges, and the value of the inconsistency of H on each edge is shown in parentheses below the edges. For example, H is inconsistent on edge (n_4, n_5), which is why $INC_H(n_4, n_5) = 9 - 2 - 4 = 3$, while the value of $INC_H(n_2, n_3)$ and $INC_H(n_3, n_4)$ are both 0 since H is consistent on these edges.

The inconsistency of H along the path in Figure 1 is then given by the sum of the numbers in the parentheses. In this case, the result is 8. The path in Figure 1 also demonstrates

Algorithm 1 Best-First Search

```
1: g(n_start) = 0, parent(n_start) = NONE
2: OPEN ← {n_start}, CLOSED ← {}
3: while OPEN is not empty do
4:    n ← arg min_{n'∈OPEN} g(n') + H(n')
5:    if n is a goal node then
6:       return solution path extracted from CLOSED
7:    for all child n_c of n do
8:       if n_c ∈ OPEN then
9:          if g(n) + κ(n, n_c) < g(n_c) then
10:            g(n_c) = g(n) + κ(n, n_c), parent(n_c) ← n
11:      else if n_c ∈ CLOSED then
12:         if g(n) + κ(n, n_c) < g(n_c) then
13:            g(n_c) = g(n) + κ(n, n_c), parent(n_c) ← n
14:            CLOSED← CLOSED−{n_c}
15:            OPEN← OPEN ∪{n_c}
16:      else
17:         g(n_c) = g(n) + κ(n, n_c), parent(n_c) ← n
18:         OPEN ← OPEN ∪{n_c}
19:   CLOSED ← CLOSED ∪{n}
20: return no solution exists
```

the importance of the max function for ensuring that INC_H is non-negative. If $INC_H(p, c)$ were calculated as $H(p) - H(c) - \kappa(p, c)$ without taking the maximum with 0, the sum along this path would be -1, and this value does not capture that there is inconsistency along this path.

INC_H is also related to an existing metric for measuring heuristic inconsistency called the *inconsistency rate of an edge (IRE)* (Felner et al. 2005; Zahavi et al. 2007). This metric, which is calculated as $|H(p) - H(c)|$, was intended for graphs with only unit-cost, undirected edges. The directed version of IRE would be $H(p) - H(c)$, though this is still only suitable for unit-cost edges. To see this, notice that the IRE of edges (n_2, n_3) and (n_5, n_6) in Figure 1 are both 2, even though H is consistent on (n_2, n_3) and inconsistent on (n_5, n_6). Since IRE can be negative, we also cannot use the sum of the IRE values along a path P as a measure of the inconsistency along P for the same reason that INC_H is forced to be non-negative. As such, INC_H was found to be a more convenient metric than IRE for this paper.

Best-First Search (BFS). Best-first search is a well-known algorithm framework that includes A* and WA*. Pseudocode for this framework is shown in Algorithm 1 though we assume the reader's familiarity with the concepts of OPEN and CLOSED lists, and $parent$ pointers as typically used in such algorithms.

BFS algorithms maintain a *g-cost* for each node n, denoted $g(n)$. In an A* search which uses a consistent heuristic, $g(n)$ is the cost of the path from n_{start} to n stored implicitly using $parent$ pointers. In other BFS algorithms, $g(n)$ can only be said to be an upper bound on the cost of this path. This is because an earlier portion of the path to n may be improved through re-expansions and this improvement will not be immediately propagated to $g(n)$, thus leading to g-cost inaccuracy. Since the value of $g(n)$ can change, we use $g_t(n)$ when we need to refer to the g-cost of n after t expansions. Using this notation, if the g-cost of some n is not updated during the $t+1$-st iteration, then $g_{t+1}(n) = g_t(n)$.

The algorithms in this framework differ in the *evaluation function* used to order nodes. In this paper, we only consider evaluation functions of the form $g(n) + H(n)$ where H is some heuristic function (see line 4 in Algorithm 1). This restriction still allows for the framework to include A* (which corresponds to the use of an admissible H), WA* (which corresponds to $H = w \cdot h$ where h is admissible), and the use of many other evaluation functions. However, this framework does not include Greedy Best-First Search since that algorithm does not include g in its evaluation function.

NR-BFS. We use *NR-BFS* to refer to a BFS which does not re-expand nodes. NR-BFS is identical to BFS except in lines 12 to 15 of Algorithm 1. These lines are responsible for moving a node from CLOSED to OPEN when a lower cost path has been found to it. By simply removing these lines and leaving this branch of the conditional empty, the result is a search that can never re-expand a node. Alternatively, we can leave the updates made in line 13, while removing only lines 14 and 15 which are responsible for actually making a node a candidate for expansion by moving it back to OPEN. This approach allows for the path to a node in CLOSED to be improved without allowing that node to be re-expanded. We will refer to this technique as *parent pointer updating*. Note that the theorems and proofs below will all apply regardless of whether this technique is used or not.

g-cost error. In the remainder of this paper, it will be convenient to let $g^\delta_t(n) = g_t(n) - g^*(n)$, which we refer to as the *g-cost error* of node n after t node expansions. Notice that like $g(n)$, $g^\delta_t(n)$ can never increase as the search progresses. This is because $g(n)$ is only updated if it will decrease. Formally, this means that if n is first generated by the t-th node expansion then for any t' and t'' where $t \leq t' \leq t''$, $g_{t'}(n) \geq g_{t''}(n)$ and $g^\delta_{t'}(n) \geq g^\delta_{t''}(n)$.

4 Worst-Case Solution Quality Analysis

Having established the needed notation, we now show that for any problem, the inconsistency of H along any optimal path is an upper bound on the suboptimality of any solution found by NR-BFS. To show this bound, we first require the following lemma and a resulting observation.

Lemma 4.1. *Let P_{opt} be an optimal solution path to a given problem. At any time prior to the expansion of a goal node by NR-BFS, there will be a node from P_{opt} which is in OPEN.*

The proof of this lemma has been omitted as it is almost identical to the proof given for Lemma 1 of Hart, Nilsson, and Raphael (1968) which is specific to A*.

We can now use this lemma to show how the evaluation of a node from P_{opt} that is in OPEN can be related to C^*. To do so, suppose that the $t+1$-st node expanded by NR-BFS is a goal node $n_g \in V_{goals}$, and let n be the node from P_{opt} that is guaranteed to be in OPEN after t expansions according to Lemma 4.1. Consider the following derivation of the evaluation of n:

$$\begin{aligned} g_t(n) + H(n) &= g^*(n) + g^\delta_t(n) + H(n) \\ &= g^*(n) + g^\delta_t(n) + h^*(n) - h^*(n) + H(n) \\ &= C^* + g^\delta_t(n) + (H(n) - h^*(n)) \end{aligned} \quad (1)$$

This shows that the evaluation of n differs from C^* by no more than the sum of the g-cost error and $(H(n) - h^*(n))$ which, if positive, is how inadmissible H is on n. Since n_g was selected for the $t+1$-st node expansion instead of n, $g_t(n_g) + H(n_g) \leq g_t(n) + H(n)$ by the definition of NR-BFS. Where C is the cost of the solution found by NR-BFS, this inequality can be simplified to $C \leq g_t(n) + H(n)$ since $n_g \in V_{goals}$ implies that $H(n_g) = 0$ and since $C \leq g(n_g)$ (as explained when discussing g-cost above). Substituting this into equation 1 allows for the following:

Observation 4.2. *If the $t+1$-st node expanded by NR-BFS is a goal node n_g, then there is a node n from some optimal path which is in OPEN after t expansions such that the cost of the solution found to n_g will be no more than*

$$C^* + g_t^\delta(n) + H(n) - h^*(n)$$

Below we will show that $g_t^\delta(n) + H(n) - h^*(n)$ is bound by the inconsistency of H along the optimal path that n is on. To do so, we will first show in Section 4.1 that the inadmissibility of H on any n_i which is on an optimal solution path $P_{opt} = n_0, ..., n_k$ is bound by the inconsistency of H along the portion of P_{opt} from n_i to n_k. We will then show in Section 4.3 that there is at least one node n_i from P_{opt} which is in OPEN after t expansions such that the g-cost error of n_i is no larger than the inconsistency of H along the portion of P_{opt} from n_1 to n_i. These two results will then be used to derive the main theorem given in Section 4.3.

4.1 Bounding Inadmissibility with Inconsistency

In this section, we will show that the inadmissibility of the heuristic value of a node n on any optimal path P_{opt} is bounded by the inconsistency along P_{opt} from n on. We begin by showing the following more general statement:

Lemma 4.3. *If $P = n_0, n_1, ..., n_k$ is an optimal path from n_0 to n_k and $H(n_i) \neq \infty$ for all $0 \leq i \leq k$, then*

$$H(n_0) - H(n_k) - g^*(n_0, n_k) \leq \sum_{i=0}^{k-1} INC_H(n_i, n_{i+1})$$

Proof. Notice that $INC_H(p, c) \geq H(p) - H(c) - \kappa(p, c)$ by the definition of INC_H. As such, the following is true:

$$\sum_{i=0}^{k-1} H(n_i) - H(n_{i+1}) - \kappa(n_i, n_{i+1}) \leq \sum_{i=0}^{k-1} INC_H(n_i, n_{i+1})$$

In the left side of this inequality, $-H(n_{i+1})$ appears in the i-th term of the sum while $H(n_{i+1})$ appears in the $i+1$-st term. As such, when evaluating the sum, all such terms will cancel out except for $H(n_0)$ and $-H(n_k)$. Since P is an optimal path from n_0 to n_k, the sum of the $-\kappa(n_i, n_{i+1})$ terms will be $-g^*(n_0, n_k)$. Therefore, the left side of the equation above is equal to $H(n_0) - H(n_k) - g^*(n_0, n_k)$, and the statement follows. □

This lemma immediately leads to the following bound on the inadmissibility of the heuristic value of any node from which the goal is reachable:

Theorem 4.4. *If $P = n_0, n_1, ..., n_k$ is a path from n_0 to some $n_k \in V_{goals}$ such that $h^*(n_0) = g^*(n_0, n_k)$, then*

$$H(n_0) - h^*(n_0) \leq \sum_{i=0}^{k-1} INC_H(n_i, n_{i+1})$$

This follows from Lemma 4.3 since $n_k \in V_{goals}$ implies that $H(n_k) = 0$, and since $g^*(n_0, n_k) = h^*(n_0)$.

4.2 Bounding g-cost Error with Inconsistency

In this section, we will show that the g-cost error of any node n from an optimal solution path P_{opt} can be bound by the inconsistency from n_{start} to n along P_{opt}, at any time after n is put in CLOSED. For this theorem, we will first need the following definition. Where $P_{opt} = n_0, ..., n_k$, a node n_i will be said to be the *shallowest* node from P_{opt} that is in OPEN after t expansions if all predecessors of n_i from P_{opt} are not in OPEN after t expansions. Formally, this means that for any j where $0 \leq j < i$, n_j is not in OPEN. If $t > 1$, then we can also guarantee that the parent of n_i on P_{opt} will be in CLOSED. This is because for n_i to be the shallowest node from P_{opt} in OPEN after t expansions, all the predecessors of n_i must be in CLOSED. To see this, notice that n_0 will be in CLOSED for any $t > 1$ since n_0 is the first node expanded. Since n_1 must have been generated by this expansion, n_1 must either be in OPEN or CLOSED after t expansions. If n_1 is in OPEN, then $n_1 = n_i$ since n_i is the shallowest node from P_{opt} in OPEN. If n_1 is in CLOSED, then n_2 was generated when n_1 was previously expanded. As such, n_2 is either in OPEN (in which case $n_i = n_2$) or n_2 is in CLOSED. By this inductive argument, it is easy to see that for any j where $0 \leq j < i$, n_j is in CLOSED. Therefore, the parent of the shallowest node from P_{opt} which is in OPEN after t node expansions must be in CLOSED.

We now use this property to prove the following theorem:

Theorem 4.5. *Let $P_{opt} = n_0, n_1, ..., n_k$ be an optimal solution path. If n_i is a node from P_{opt} that is in CLOSED after $t \geq 0$ node expansions of NR-BFS, then*

$$g_t^\delta(n_i) \leq \sum_{j=1}^{i-1} INC_H(n_j, n_{j+1})$$

Proof. The proof is by induction on the number of node expansions, denoted by t. If $t = 0$, the statement is vacuously true since CLOSED is empty. If $t = 1$, only $n_0 = n_{start}$ is in CLOSED and $g(n_0) = g^*(n_0) = 0$. Since this means that $g_1^\delta(n_0) = 0$, the statement is also true for this base case.

Suppose the statement is true after all of the first $t \geq 1$ node expansions. This means that any n from P_{opt} which is in CLOSED after t node expansions must satisfy the bound above. Since $g_{t''}^\delta(n) \leq g_{t'}^\delta(n)$ for any $t'' \geq t'$, the statement will still apply to any such n after $t+1$ node expansions.

Now consider the node selected to be the $t+1$-st expansion, since this is the only node that is newly added to CLOSED on this iteration. If this node is not on P_{opt}, then the statement is true after $t+1$ node expansions. If this node is on P_{opt}, we denote it as n_i for some $i > 0$. We now consider two cases: when the parent of n_i, n_{i-1}, is in CLOSED after t node expansions, and when it is not.

First suppose that n_{i-1} is in CLOSED. This requires that n_{i-1} was the t'-th node expanded where $t' \leq t$. This expansion will have necessarily generated n_i and so $g_{t'}(n_i) \leq g_{t'}(n_{i-1}) + \kappa(n_{i-1}, n_i)$. As n_{i-1} is the parent of n_i on P_{opt}, this means that $g^\delta_{t'}(n_i) \leq g^\delta_{t'}(n_{i-1})$. If $i > 1$, the fact that $g^\delta_t(n_i) \leq g^\delta_{t'}(n_i)$ allows for the following derivation:

$$g^\delta_t(n_i) \leq g^\delta_{t'}(n_{i-1}) \tag{2}$$

$$\leq \sum_{j=1}^{i-2} INC_H(n_j, n_{j+1}) \tag{3}$$

$$\leq \sum_{j=1}^{i-1} INC_H(n_j, n_{j+1}) \tag{4}$$

Line 3 holds by the induction hypothesis on $g^\delta_{t'}(n_{i-1})$, while the final line holds since $INC_H(n_{i-1}, n_i) \geq 0$. If $i = 1$ (*ie.* $n_i = n_1$), then $g_t(n_i) = g^*(n_1) = \kappa(n_0, n_1)$ and $g^\delta_t(n_1) = 0$. Therefore, the inequality in line 4 also applies in this case since the right hand side evaluates to 0. As such, the statement holds for n_i after $t+1$ node expansions if n_{i-1} is in CLOSED after t expansions. Notice that this argument also shows that for any n_i from P_{opt} that is in OPEN, if n_{i-1} is in CLOSED and its g-cost error is bound by the inconsistency of H along P_{opt} from n_1 to n_{i-1}, then $g^\delta_t(n_i)$ is bound by the inconsistency from n_1 to n_i. This fact will be used later in this proof and when proving Theorem 4.6.

Now suppose that n_{i-1} is not in CLOSED after t expansions. Since $t \geq 1$ this means that $i \neq 1$ since n_0 will always be in CLOSED. Let $n_{i'}$ be the shallowest node from P_{opt} which is in OPEN after t node expansions. As the parent of $n_{i'}$, $n_{i'-1}$, is in CLOSED it must be the case that $n_i \neq n_{i'}$ since n_{i-1} is not in CLOSED. Since n_i was selected for expansion instead of $n_{i'}$, the evaluation of n_i must be no larger than that of $n_{i'}$. This allows for the following:

$$g_t(n_i) + H(n_i) \leq g_t(n_{i'}) + H(n_{i'}) \tag{5}$$

$$g^\delta_t(n_i) \leq g^*(n_{i'}) + g^\delta_t(n_{i'}) - g^*(n_i) + H(n_{i'}) - H(n_i) \tag{6}$$

$$\leq g^\delta_t(n_{i'}) + H(n_{i'}) - H(n_i) - g^*(n_{i'}, n_i) \tag{7}$$

Line 6 is derived by expanding $g_t(n)$ to $g^*(n) + g^\delta_t(n)$, and rearranging the terms. Line 7 then holds since $n_{i'}$ is along an optimal path to n_i and so $g^*(n_i) = g^*(n_{i'}) + g^*(n_{i'}, n_i)$.

Since $n_{i'-1}$ is in CLOSED, $g^\delta_t(n_{i'-1})$ is bound according to the induction hypothesis. As such, $g^\delta_t(n_{i'})$ is bound by the inconsistency of H along P_{opt} from n_1 to $n_{i'}$ by the same argument used to derive inequality 4. By substituting this, along with the upper bound on $H(n_{i'}) - H(n_i) - g^*(n_{i'}, n_i)$ given by Lemma 4.3, we can continue from line 7 as follows:

$$g^\delta_t(n_i) \leq \sum_{j=1}^{i'-1} INC_H(n_j, n_{j+1}) + \sum_{j=i'}^{i-1} INC_H(n_j, n_{j+1}) \tag{8}$$

$$\leq \sum_{j=1}^{i-1} INC_H(n_j, n_{j+1}) \tag{9}$$

where the final line holds by combining the sums. Therefore the statement holds for n_i after $t+1$ expansions in the case that n_{i-1} is not in CLOSED after t expansions. Having handled all cases for n_i, the statement is true by induction. □

Notice that the bound given in this theorem does not depend on the inconsistency of the edge from n_0 to n_1. As described in the proof, this is because n_1 will always have an optimal g-cost since n_0 is necessarily expanded.

4.3 Bounding Solution Quality with Inconsistency

In this section, we will use theorems 4.4 and 4.5 to show that the inconsistency of H along any optimal solution path (not including the inconsistency of H on the first edge) can be used to bound the quality of any solution found. This is formalized by the following theorem:

Theorem 4.6. *Where $P_{opt} = n_0, n_1, ..., n_k$ is an optimal solution path to a given problem, any solution found by NR-BFS will have a cost C such that:*

$$C \leq C^* + \sum_{j=1}^{k-1} INC_H(n_j, n_{j+1})$$

Proof. If k is 0 or 1, NR-BFS will necessarily find the optimal solution and thus will satisfy the statement. If $k > 1$, assume that a goal node n_g is first found with the $t+1$-st expansion for some $t \geq 0$. By Lemma 4.1 there will be a node from P_{opt} which is in OPEN after t expansions. Let n_i be the shallowest such node. By Observation 4.2, we can now perform the following derivation:

$$C \leq C^* + g^\delta_t(n_i) + H(n_i) - h^*(n_i) \tag{10}$$

$$\leq C^* + \sum_{j=1}^{i-1} INC_H(n_j, n_{j+1}) + \sum_{j=i}^{k-1} INC_H(n_j, n_{j+1}) \tag{11}$$

$$\leq C^* + \sum_{j=1}^{k-1} INC_H(n_j, n_{j+1}) \tag{12}$$

Line 11 is achieved by applying the two earlier theorems as follows. First, since n_i is the shallowest node from P_{opt} in OPEN, n_{i-1} is in CLOSED and so $g^\delta_t(n_{i-1})$ is bound by the inconsistency of H from n_1 to n_{i-1} by Theorem 4.5. As a result, $g^\delta_t(n_i)$ is bound by the inconsistency from n_1 to n_i by the argument used to derive inequality 4. Secondly, $H(n_i) - h^*(n_i)$ is replaced by the inconsistency from n_i to n_k as given by Theorem 4.4. The final line is then achieved by combining the two summations. □

This theorem shows a formal relationship between the inconsistency of the heuristic and solution quality. We now consider this bound in the special case that H is admissible.

Admissible heuristics. If heuristic H is admissible, then $INC_H(n_i, n_{i+1}) \leq h^*(n_i) - \kappa(n_i, n_{i+1})$ since $H(n_i) \leq h^*(n_i)$ and $H(n_{i+1}) \geq 0$. Where $P_{opt} = n_0, ..., n_k$ is an

optimal solution path, this statement allows for the following derivation on the sum of the inconsistency of H along P_{opt}:

$$\sum_{j=1}^{k-1} INC_H(n_j, n_{j+1}) \leq \left[\sum_{j=1}^{k-1} h^*(n_i)\right] - C^* \quad (13)$$

$$\leq (k-2) \cdot C^* \quad (14)$$

In line 13, the $-C^*$ term arises from the sum of the edge costs along P_{opt}. The final line follows since $h^*(n_i) \leq C^*$ for all nodes on P_{opt}. If for any $(m, n) \in E$, $\kappa(m, n) \geq c$ for some constant $c > 0$, then $k \leq C^*/c$ since there are at most C^*/c edges along P_{opt}. Therefore, by expression 14 and Theorem 4.6, the worst-case solution cost when using NR-BFS with an admissible heuristic such that all edges cost at least $c > 0$ is $(C^*/c - 1) \cdot C^*$.

Since $h(n_i) < C^*$ for all $i > 0$, it is not possible to construct graphs on which an NR-BFS using an admissible heuristic will find solutions that cost exactly $(C^*/c-1)\cdot C^*$. However, it is still possible to construct graphs in which the solutions found by NR-BFS will have a cost that is quadratic in C^*. We describe such graphs in the next section.

4.4 Tightness of the Worst-Case Bound

In this section, we describe a process for constructing graphs that demonstrate the tightness of the bound in Theorem 4.6. We will then show that this worst-case behaviour is not always happening in practice.

Worst-case graphs. Given any path P such that the nodes are assigned any possible heuristic values (admissible or inadmissible), we can construct a graph in which P is the optimal solution path and the cost of the solution found by NR-BFS will be arbitrarily close to the sum of the cost of P and the inconsistency along P (not including the inconsistency of the heuristic on the first edge). For example, Figure 2 shows such a graph that has been built around the path $P = n_0, ..., n_6$ given previously in Figure 1. To this path, we have added two types of edges, both shown in red. The solid red edges create suboptimal routes around inconsistent portions of P. The dashed red edge from n_0 to n_6 is only needed when using parent pointer updating, as an NR-BFS using this technique will outperform the bound if this edge is not included. The costs of the new edges include a constant $\epsilon \geq 0$ whose purpose is to ensure that the worst-case solution is found regardless of how ties are broken between nodes with equal values for $g + H$. If ties are broken in favour of the node with higher g-cost (or equivalently with lower H-cost), then ϵ can be set to 0. For other tie breaking policies, ϵ can be set as a positive value arbitrarily close to 0. In the resulting graph, NR-BFS will find the solution whose cost is arbitrarily close to the bound in Theorem 4.6. For example, the solution found in Figure 2 will cost $25 - 2\epsilon$, while C^* is 19 and the inconsistency of P from n_1 to n_k is 6.

To construct such graphs around any given path $P = n_0, ..., n_k$ it is necessary to identify the inconsistent portions of P, not including edge (n_0, n_1) since any inconsistency in this edge cannot decrease the quality of solutions found. Formally, we consider every maximally long subpath $n_i, ..., n_j$ of P where $1 \leq i < j \leq k$ such that for all i' where

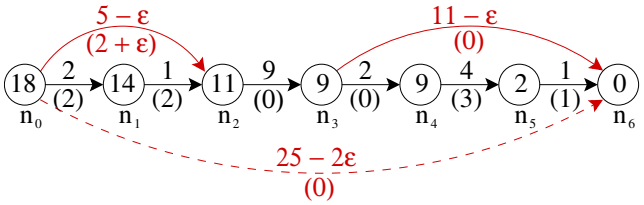

Figure 2: Example worst-case graph.

$i \leq i' < j$, $INC_H(n_{i'}, n_{i'+1}) > 0$. By being of maximal length, we mean that either $i = 1$ or $INC_H(n_{i-1}, n_i) = 0$, and either $j = k$ or $INC_H(n_j, n_{j+1}) = 0$. For each such subpath, we add an edge from n_{i-1} to n_j that has a cost of

$$g^*(n_{i-1}, n_j) - \epsilon + \sum_{i'=i}^{j-1} INC_H(n'_i, n_{i'+1})$$

For example, nodes n_4, n_5, n_6 in Figure 2 form a maximally long inconsistent subpath since H is inconsistent on edges (n_4, n_5) and (n_5, n_6), but not on edge (n_3, n_4). The inconsistency along this subpath is 4. Therefore, an edge is added from n_3 (which is the parent of n_4) to n_6 with a cost of $g^*(n_3, n_6) - \epsilon + 4$. Since $g^*(n_3, n_6) = 7$, the cost of this edge is $11 - \epsilon$. The other maximally long inconsistent subpath of n_1, n_2 is handled similarly, though notice that this subpath does not include n_0 since the inconsistency of (n_0, n_1) cannot impact the quality of solutions found.

When using parent pointer updating, an additional edge is needed from n_0 to n_k whose cost is equal to the sum of C^* and the inconsistency of H along P_{opt} from n_1 to n_k, less $d \cdot \epsilon$ where d is the number of maximally length inconsistent subsequences of P_{opt}. For example, $d = 2$ in Figure 2.

The example in Figure 2 also demonstrates why the value of $H(n_i) - H(n_{i+1}) - \kappa(n_i, n_{i+1})$ for edges on which H is consistent do not factor into the bound. To see this, consider the edge from n_2 to n_3. If the cost of this edge was 10 instead of 9 and the cost of dotted red edge was increased by 1, then NR-BFS would find a solution of cost $26 - 2\epsilon$. This solution is a cost of $6 - 2\epsilon$ larger than C^*, just like the solution found by NR-BFS when $\kappa(n_2, n_3) = 9$. This would also be true for any $\kappa(n_2, n_3) \geq 2$ provided that the dotted red edge was adjusted accordingly. Intuitively, this shows that NR-BFS cannot "make up" on g-cost error that has been accrued along optimal paths using an edge from n_i to n_{i+1} for which $H(n_i) - H(n_{i+1}) - \kappa(n_i, n_{i+1})$ is a negative number.

Worst-case graphs with admissible heuristics. This construction can also be used to generate graphs that have an admissible heuristic on which NR-BFS will find solutions that are quadratic in C^*. To do so, start with any path $n_0, ..., n_k$ that is to be the optimal solution in the completed graph. For each node n_i, set $H(n_i)$ as the cost from n_i to n_k along P if $i < k$ and i is odd, and set $H(n_i)$ as 0 otherwise. To this path add the edges as defined by the construction above. For example, given path $P = n_0, ..., n_6$ such that the edges on P have a cost of 1, the result of this process is the graph with an optimal solution cost of 6 shown in Figure 3. In this graph, NR-BFS will find the path of cost $12 - 2\epsilon$.

In the more general case, consider using this process on the path $n_0, ..., n_k$ such that the edge costs between these

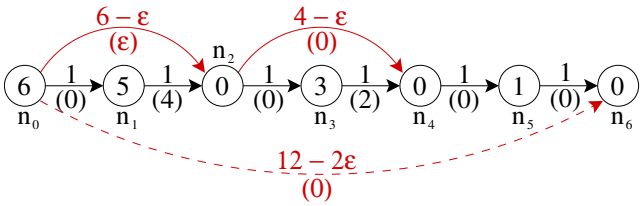

Figure 3: Example worst-case graph when H is admissible

nodes are all 1. The process described above will result in a graph on which NR-BFS will find a solution with a cost that is arbitrarily close to $C^* \cdot (C^* + 2)/4$ if k is even, and arbitrarily close to $(C^* + 1)^2/4$ if k is odd, where $C^* = k$.

Comparing the bound with practice. In the graphs just constructed, g-cost error is accrued by using the added (*ie.* red) edges to bypass inconsistent subpaths on the optimal solution. If the cost of these edges is not as high as given in the construction, the edges are not exactly where defined, or if these edges do not appear at all, NR-BFS will find higher quality solutions. This is why NR-BFS can often outperform the bound in practice. For example, consider the case in which a weight 10 WA* is used on the pathfinding problems described in Section 2. If WA* is set so as to not re-expand nodes, it finds solutions that are an average of 18% more costly than optimal, which is worse than the 12% solution suboptimality seen when nodes are re-expanded. However, this is substantially better than the upper bound of 475% suboptimality that is guaranteed by Theorem 4.6, where this value has been found by calculating the inconsistency of the heuristic over the solution paths found by A*.

The bound is more accurate in Martelli's subgraphs in which the solutions found by NR-BFS are provably no more than 1.5 times larger than optimal. In these subgraphs, the upper bound given by Theorem 4.6 is $C + |V| - 2$ where C is the cost of the solution actually found. Since both C^* and C are $O(2^{|V|})$, this deviation from the bound is insignificant.

Note that the matter of finding tighter domain-specific bounds is left as future work.

4.5 Alternative Heuristic Weighting in NR-BFS

In this section, we will use Theorem 4.6 to find solution quality guarantees for NR-BFS instances that weight a consistent heuristic in a different way than is done by WA*.

For any function $B(x) : \mathbb{R}^{\geq 0} \rightarrow \mathbb{R}^{\geq 0}$ such that $\forall x \geq 0, y \geq 0, B(x + y) \geq B(x) + y$, any solution found by a BFS using a heuristic $H(n) = B(h(n))$ for some admissible h will be no more than $B(C^*)$ (Valenzano et al. 2013). This statement generalizes the well-known result that WA* will find solutions no more costly than $w \cdot C^*$ if the heuristic being weighted is admissible. This is because WA* is a BFS that uses $H(n) = B(h(n))$ where $B(x) = w \cdot x$. Note that in the context of this section, B is called a *bounding function*.

If the heuristic being weighted in WA* is consistent, NR-BFS is also guaranteed to find solutions that cost at most $w \cdot C^*$ (Likhachev, Gordon, and Thrun 2003). By using Theorem 4.6, we can consider other bounding functions for which NR-BFS using $H = B(h)$ where h is consistent will also be guaranteed to return solutions with a cost of no more than $B(C^*)$. To do so, we require the following definition:

Definition 1. *A heuristic H is said to be* B-consistent *for bounding function B if for any nodes p and c for which c is a successor of p, $H(p) - H(c) \leq B(h^*(p)) - B(h^*(c))$.*

If H is B-consistent along $P_{opt} = n_0, ..., n_k$ and $\forall x \geq 0, y \geq 0, B(x + y) \geq B(x) + y$, then $INC_H(p, c) \leq B(h^*(n_i)) - B(h^*(n_{i+1})) - \kappa(n_i, n_{i+1})$ for all $0 \leq i < k$. The inconsistency along P_{opt} will therefore be no more than the sum of these upper bounds, which is $B(h^*(n_0)) - B(h^*(n_k)) - C^*$ by the same argument used to prove Theorem 4.4. Therefore, any solutions found by NR-BFS using a B-consistent heuristic will cost no more than $B(h^*(n_0)) - B(h^*(n_k))$ by Theorem 4.6, which is at most $B(C^*)$ since $h^*(n_0) = C^*$ and $B(h^*(n_k)) \geq 0$.

This result will allow us to identify a set of bounding functions for which solutions found by NR-BFS will cost at most $B(C^*)$ when using the heuristic $H = B(h)$ where h is consistent. Specifically, consider the set of functions B such that for all $x \geq 0, y \geq 0, x \geq 0$, it is true that

$$x \geq y \Rightarrow B(x + z) - B(y + z) \geq B(x) - B(y) \quad (15)$$

It can be easily shown that if B satisfies this condition, then $B(h)$ is B-consistent if h is consistent. If B is twice differentiable, this condition implies that $\forall x \geq 0, B''(x) \geq 0$.

5 Related Work

Previous research into bounding solution quality when not re-expanding nodes has focused on WA* when weighting a consistent heuristic, in which case any solutions found will cost at most $w \cdot C^*$ (Likhachev, Gordon, and Thrun 2003). Ebendt and Drechsler (2009) also proved that when using a consistent heuristic with A^*_ϵ, solutions will be no worse than $(1 + \epsilon)^D \cdot C^*$ where the optimal solution has D edges in it.

Previous investigations into the impact of inconsistency on algorithm performance have focused on admissible heuristics. This began with the work by Martelli (1977) mentioned above, and includes two other techniques which decrease the worst-case runtime of A* from being $O(2^{|V|})$ to being $O(|V|^2)$ while still maintaining optimality (Bagchi and Mahanti 1983; Mero 1984). Felner et al. (2011) then offered a way to improve Mero's technique in undirected graphs and showed that in order for A* to expand $O(2^{|V|})$ nodes, the edge weights must be exponential in $|V|$.

6 Conclusion

This paper presents a formal analysis of the impact that not re-expanding nodes can have on the quality of solutions returned by a best-first search. In particular, the inconsistency of a heuristic along an optimal solution path is shown to bound the suboptimality of the solutions found when not performing re-expansions. This bound, which is tight, was then used to demonstrate that not re-expanding nodes in an A* search that uses an admissible heuristic will have a quadratic effect on solution quality in the worst-case. The bound was also used to find suboptimality guarantees for a large class of best-first search algorithms that weight a consistent heuristic in a different way than WA*.

Acknowledgments

We would like to thank the reviewers for their feedback on this paper, particularly for the reviewer who noticed that there was a simpler proof to the quadratic bound in the case that H is admissible. This research was supported by GRAND and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Bagchi, A., and Mahanti, A. 1983. Search Algorithms Under Different Kinds of Heuristics-A Comparative Study. *Journal of the ACM* 30(1):1–21.

Ebendt, R., and Drechsler, R. 2009. Weighted A* search - unifying view and application. *Artificial Intelligence* 173(14):1310–1342.

Felner, A.; Zahavi, U.; Schaeffer, J.; and Holte, R. C. 2005. Dual Lookups in Pattern Databases. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, 103–108.

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

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* SSC-4(2):100–107.

Likhachev, M.; Gordon, G. J.; and Thrun, S. 2003. ARA*: Anytime A* with Provable Bounds on Sub-Optimality. In *Advances in Neural Information Processing Systems 16 (NIPS-03)*.

Martelli, A. 1977. On the Complexity of Admissible Search Algorithms. *Artificial Intelligence* 8(1):1–13.

Mero, L. 1984. A heuristic search algorithm with modifiable estimate. *Artificial Intelligence* 23(1):13 – 27.

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

Sturtevant, N. 2012. Benchmarks for Grid-Based Pathfinding. *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 *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling*, 355–362.

Valenzano, R. A.; Arfaee, S. J.; Thayer, J. T.; Stern, R.; and Sturtevant, N. R. 2013. Using Alternative Suboptimality Bounds in Heuristic Search. In *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling, ICAPS 2013, Rome, Italy, June 10-14, 2013*.

Zahavi, U.; Felner, A.; Schaeffer, J.; and Sturtevant, N. R. 2007. Inconsistent Heuristics. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, 1211–1216.