Reaching the Goal in Real-Time Heuristic Search: Scrubbing Behavior is Unavoidable

Nathan R. Sturtevant
Department of Computer Science
University of Denver
Denver, Colorado, USA
sturtevant@cs.du.edu

Vadim Bulitko
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada
bulitko@ualberta.ca

Abstract

Real-time agent-centered heuristic search is a well-studied problem where an agent that can only reason locally about the world must travel to a goal location using bounded computation and memory at each step. Many algorithms have been proposed for this problem, and theoretical results have also been derived for the worst-case performance. Assuming sufficiently poor tie-breaking, among other conditions, we derive theoretical best-case bounds for reaching the goal using LRTA*, a canonical example of a real-time agent-centered heuristic search algorithm. We show that the number of steps required to reach the goal can grow asymptotically faster than the state space, resulting in a "scrubbing" when the agent repeatedly visits the same state. This theoretical result, supported by experimental data, encourages recent work in the field that uses novel tie-breaking schemas and/or perform different types of learning.

1 Introduction

The framework of real-time agent-centered heuristic search models an agent with locally limited sensing and perception that is trying to reach a goal while interleaving planning and movement (Koenig 2001). This well-studied problem has led to numerous algorithms (Korf 1990; Furcy and Koenig 2000; Shimbo and Ishida 2003; Hernández and Meseguer 2005; Bulitko and Lee 2006; Hernández and Baier 2012) and various theoretical analysis (Ishida and Korf 1991; Koenig and Simmons 1993; Koenig, Tovey, and Smirnov 2003; Bulitko and Lee 2006; Bulitko and Bulitko 2009; Sturtevant, Bulitko, and Björnsson 2010).

While several worst-case bounds on the time for an agent to reach the goal have been established, there has been little published work on practical lower bounds on the time for an agent to reach the goal the first time. Some work has considered the problem of learning convergence over multiple trials, where an agent travels repeatedly between the start and the goal, which we do not consider here.

The main contribution of this paper is a non-trivial lower bound on the travel distance required for an agent to reach the goal under the basic real-time heuristic search framework. This is important in many applications such as real-time pathfinding in video games. We show theoretically that there exists a tie-breaking schema that forces the agent to revisit its states ("scrub") in polynomially growing state spaces. This is an undesirable property of real-time heuristic search which plagued the field from the beginning. Our result shows that, unfortunately, this phenomenon is unavoidable under certain common conditions. We further support this theoretical result empirically by showing that common tie-breaking schemas exhibit similar undesirable behavior.

The rest of the paper is organized as follows. We begin in Section 2 by formally defining the problem at hand. We will review related work on this problem in Section 3. Our own theoretical analysis is presented in Section 4 and supported by empirical results in Section 5. We then conclude the paper and outline directions for future work.

2 Problem Formulation

In this paper we use a common definition of the real-time heuristic search problem. We define a search problem $\mathbf{S}$ as the n-tuple (S, E, s_0, s_g, h) where S is a finite state of states and $E \subset S \times S$ is a finite set of edges between them. S and E jointly define the search graph which is undirected: $\forall a, b \in S\,[(a, b) \in E \implies (b, a) \in E]$. Two states a and b are *immediate neighbors* iff there is an edge between them: $(a, b) \in E$; we denote the set of immediate neighbors of a state s by $N(s)$. A path P is a sequence of states $(s_0, s_1, \dots, s_n)$ such that for all $i \in \{0, \dots, n-1\}$, $(s_i, s_{i+1}) \in E$. A route R is a path that does not include cycles.

At all times $t \in \{0, 1, \dots\}$ the agent occupies a single state $s_t \in S$. The state s_0 is the start state and is given as a part of the problem. The agent can then change its current state, that is move to any immediately neighboring state in $N(s)$. The traversal incurs a travel cost of 1. The agent is said to solve the search problem at the earliest time T it arrives at the goal state: $s_T = s_g$. The *solution* is a path $P = (s_0, \dots, s_T)$: a sequence of states visited by the agent from the start state until the goal state.

The cumulative cost of all edges in the solution is the *travel cost* – the performance metric we are concerned with in this paper.[1] Since each edge costs 1, the travel cost is the number of edges the agent traversed while solving the prob-

[1]In the language of real-time heuristic search this paper deals strictly with the first trial performance. We do not teleport the agent back to the start state upon reaching the goal and we do not consider learning convergence (Bulitko and Lee 2006).

lem (i.e., T for the solution $(s_0, \dots, s_T)$). The cost of the shortest possible path between states $a, b \in S$ is denoted by $h^*(a, b)$. As usual, $h^*(s) = h^*(s, s_g)$.

The agent has access to a heuristic $h : S \to [0, \infty)$. The heuristic function is a part of the search problem specification and is meant to give the agent an estimate of the remaining cost to go. The search agent can modify the heuristic as it sees fit as long as it remains admissible $\forall s \in S\, [h(s) \leq h^*(s)]$ and consistent $\forall a, b \in S\, [|h(a) - h(b)| \leq h^*(a, b)]$ at all times. The heuristic at time t is denoted by h_t; $h_0 = h$. The total magnitude of all updates to the heuristic function is the *total learning* amount.

Numerous heuristic search algorithms have been developed for the problem described above (e.g., (Korf 1990; Bulitko and Lee 2006; Koenig and Sun 2009)). Most of them are based on Algorithm 1.

Algorithm 1: Basic Real-time Heuristic Search

```
  input  : search problem $(S, E, s_0, s_g, h)$
  output: path $(s_0, s_1, \dots, s_T), s_T = s_g$
1 $t \leftarrow 0$
2 $h_t \leftarrow h$
3 while $s_t \neq s_g$ do
4     $h_{t+1}(s_t) \leftarrow \min_{s' \in N(s_t)}(1 + h_t(s'))$
5     $s_{t+1} \leftarrow \arg\min^{\tau}_{s' \in N(s_t)}(1 + h_t(s'))$
6     $t \leftarrow t + 1$
7 $T \leftarrow t$
```

A search agent following the algorithm begins in the start state s_0 and repeatedly loops through the process of choosing the next state and updating its heuristic until it reaches the goal state s_g for the first time (line 3). Within the loop, in line 4 the agent first updates (or learns) its heuristic by setting it to be the estimated cost to go from the most promising neighbor plus the cost of getting to that neighbor (1). Then in line 5 the agent moves to the next state as to greedily minimize its heuristic. Ties among neighbors that have the same minimal heuristic values are broken with a tie-breaking schema τ as detailed later in the paper.

The problem we consider in this paper is to describe the minimum amount $T_{\min}(\mathbf{S})$ of the travel cost that *any* search agent of the type described above would necessarily incur in finding a solution to $\mathbf{S} = (S, E, s_0, s_g, h)$. While the exact value of $T_{\min}(\mathbf{S})$ can intricately depend on the particulars of a specific search problem $\mathbf{S}$, we will derive a useful asymptotic lower bound on $T_{\min}(\mathbf{S})$.

3 Related Work

Previous work has focused primarily on deriving upper bounds on the agent's travel cost. For instance, LRTA* (Algorithm 1) is guaranteed to reach the goal in $O(|S|^2)$ steps (Ishida and Korf 1991). This follows from the analysis of MTS (Ishida and Korf 1991) if the target's position is fixed. Examples have been provided to show that the exact bound, $|S|^2/2 - |S|/2$, is tight (Edelkamp and Schrödl 2012). The analysis was later extended on a sub-class of reinforcement learning algorithms of which LRTA* is a special case (Koenig and Simmons 1992; 1993; 1996). The results are still upper bounds and, for value-iteration algorithms, are still $O(|S|^2)$.

There has also been substantial work on analyzing algorithms with backtracking moves, introduced with SLA* (Shue and Zamani 1993a; 1993b) and the more general SLA*T (Shue, Li, and Zamani 2001; Zamani and Shue 2001). Such algorithms behave in the same way as our algorithm until the total learning exceeds a certain threshold T. After that the agent backtracks to the previous state whenever it raises a heuristic value. It was shown (Bulitko and Lee 2006) that the travel cost of SLA*T is upper bounded by $h^*(s_0, s_g) + T$ where T is the threshold (learning quota). However, this upper bound holds only if the path being built by SLA*T is processed after every move and all state revisits are removed from the path.

The analysis was later generalized beyond SLA*T onto a wider class of backtracking algorithms which no longer required removing state duplicates from the path after every move. With some modifications to the definition of the travel cost, an upper bound exponential in T was derived for a wide class of algorithms (Bulitko and Bulitko 2009).

A problem-specific analysis of the minimum learning required to prove the optimal path was described in (Sturtevant, Bulitko, and Björnsson 2010), but they do not consider the first trial performance. They also showed that algorithms that look farther ahead have better performance primarily because they perform less learning over and above the minimum required.

4 Our Analysis

Our goal in this paper is to derive a non-trivial asymptotic lower-bound on the amount of travel any search agent that uses Algorithm 1 will need to perform to reach the goal state. We will achieve this goal in stages. Section 4.1 presents our derivation informally, illustrated with a few intuitive examples. We present a map of the derivation in Section 4.2 and then detail individual steps in Sections 4.3 through 4.6. We then apply the analysis to the case of polynomially growing state spaces in Section 4.7 and discuss its implications on state revisits (i.e., "scrubbing").

4.1 Intuition

Consider the five-state grid world in Figure 1. The goal is state 5 on the far right. The agent starts out in state 2. At each state the agent examines the heuristic in its left and right immediate neighbors and goes to one with the lowest heuristic. If the neighbors' heuristic values are identical then the ties are broken to the neighbor on the right. We will use several variations on this problem to illustrate the importance of tie-breaking on solution travel cost.

Now consider heuristic values in a ten-state version of this grid world in Figure 2, where the goal is on the far right and the agent starts out on the far left. The initial heuristic values are shown with dark bars, and the learned updates to the heuristic are light bars stacked on top. Each plate shows a single time step. The agent's current position is indicated with an 'A'. After reaching time step 4, the agent

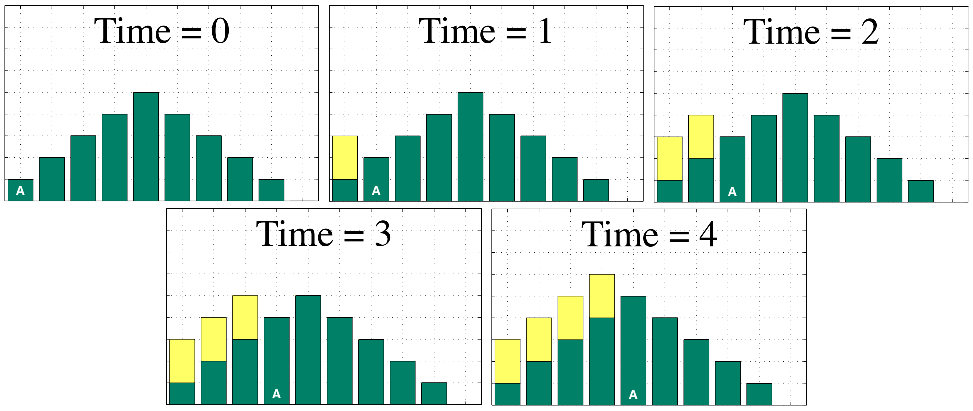

Figure 2: Heuristic learning with best-case tie breaking. The ten states of the problem are along the horizontal axis. The vertical shows the value of the heuristic function per time step. The darker bars are the initial heuristic values. The lighter bars are the increases due to learning. The initial heuristic is at the top. The agent's current state is shown with an 'A'.

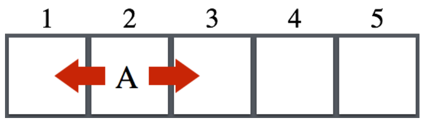

Figure 1: A one-dimensional grid world of five states numbered at the top. The agent 'A' is located in state 2 and has two available actions shown in the figure.

can continue straight to the goal without further heuristic updates.

Due to the fortunate tie-breaking schema in this particular example, the total amount of learning (i.e., the total area of light bars) is 8 and the travel cost is 9. Scaling this example to $2k$ states, the total amount of learning will be $2(k-1)$ and the path produced will have the optimal cost of $2k-1$, with no state revisits. Thus, the total learning in this example grows linearly with the state space size.

However, tie breaking can adversely affect the agent's performance. Consider the search problem in Figure 3 with 13 states. In this example, each tie is broken away from the goal (i.e., to the left). The travel cost is now 20 and the total amount of learning is 18 indicating some revisits of the 13 states by the agent, such as at time steps 3, 7, and 9. Scaling this search problem, the total amount of learning will be asymptotically quadratic in the number of states. The amount of heuristic learning per move is at most 2 which means that the travel cost will also be asymptotically quadratic in the number of states and number of revisits per state will increase with the search space size.

The difference between these two cases is asymptotically significant. While best-case tie-breaking can be achieved in simple examples, it cannot be always guaranteed.

4.2 Analysis Overview

Our goal is to quantify the travel required by any agent of the type described in Section 2 to find a solution. The examples in the previous section demonstrated that the travel cost can depend on the tie-breaking schema used by the agent. Since it may not always be possible to design the best tie-breaking schema for a given problem (or series of problems), we consider the agent's performance with sufficiently suboptimal tie-breaking schemas and develop a meaningful lower bound on the amount of travel. Our lower bound is asymptotic in the number of states and, unfortunately, demonstrates that under common conditions the agent will necessarily have to revisit states many times – an undesirable phenomenon known as "scrubbing" in real-time heuristic search.

We present the high-level argument immediately below and detail each step individually in the subsequent sections. First, due to consistency of the heuristic, the learning performed in each step is constant-bounded. Thus, an asymptotic lower bound on the learning required to reach the goal is also an asymptotic lower bound on travel distance (Section 4.3). We then show that with a sufficiently suboptimal tie breaking an agent traveling from s_a to s_b must raise the heuristic of s_a to at least s_b (Section 4.4). Given a search graph, a current location, and a goal, we identify a lower bound on the maximum heuristic value $\hat{h}$ that the agent must encounter when traveling to the goal (Section 4.5). Since the heuristic is maintained consistent at all time we use $\hat{h}$ to compute the minimum amount of learning (Section 4.6).

We then apply the argument to polynomial search spaces where the number of states within distance r from a given state growth as $\Theta(r^d)$ (e.g., quadratically on the commonly used two-dimensional navigation maps, $d = 2$). In such spaces we show that, given an appropriately inaccurate initial heuristic, the amount of learning necessarily performed by the agent can grow as $\Omega(r^{d+1})$ which is asymptotically higher than the number of states $\Theta(r^d)$. Since learning per step is constant-bounded, the amount of travel will also grow asymptotically faster than the number of states. This result indicates that any real-time heuristic search agent of the type introduced above will necessarily revisit the states ("scrub") with the number of revisits growing as well (Section 4.7).

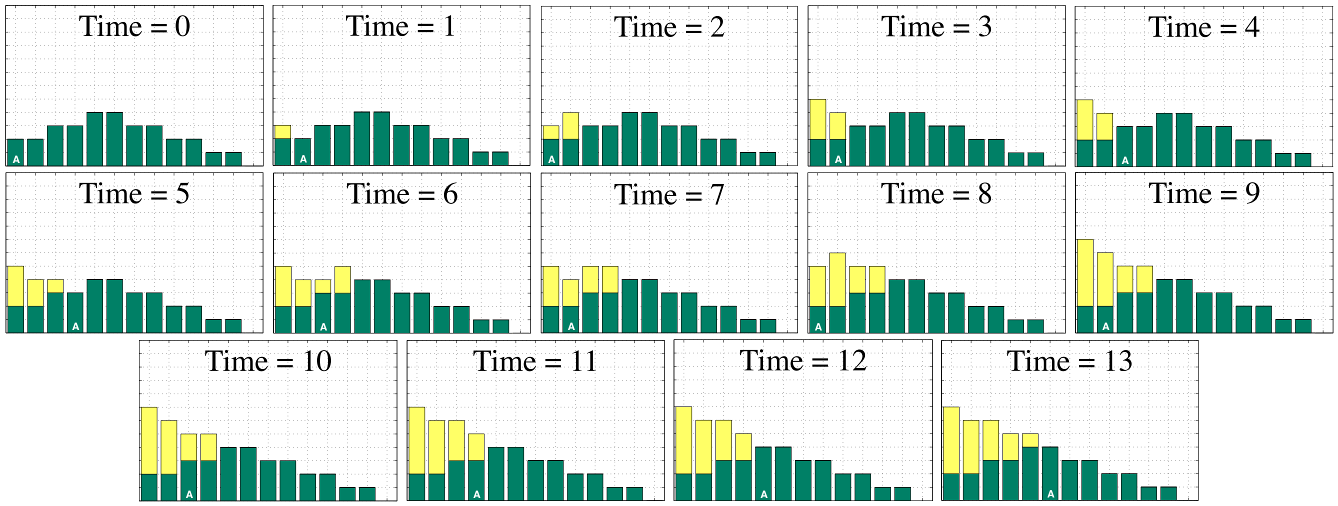

Figure 3: Heuristic learning with unfortunate tie breaking.

This result suggests that if we want to improve the performance of a learning real-time heuristic search agent of the form described here, we need to experiment with novel forms of learning and/or movement rules. Incidentally, that is what recent work in the field has investigated (Sturtevant and Bulitko 2011; Hernández and Baier 2012).

4.3 Learning per Step is Constant Bounded

It is difficult to lower-bound the agent movement directly but we can bound the learning required to solve a problem – the cumulative updates to the heuristic function from $t = 0$ until $t = T$. We show that the learning per step is constant-bounded. This implies that a lower bound on learning is also a lower bound on movement.

Lemma 1 The total change in heuristic values during each learning step of Algorithm 1 is constant bounded.

Proof. Algorithm 1 updates the heuristic of a state s_t in line 4 of the pseudo code based on the heuristic of a neighboring state s_{t+1}. Because s_{t+1} is a neighbor of s_t and because the heuristic is consistent, $|h_t(s_t) - h_t(s_{t+1})| \leq 1$. The new heuristic for s_t is $1 + h_t(s_{t+1})$. Thus, the heuristic of s_t can increase by at most 2. Since s_t is the only state that has its heuristic updated, the maximum change is heuristic values at each time step is 2 which is constant bounded. $\square$

While not shown here, this proof can be generalized to the case of arbitrary finite search graphs with finite edge costs as well as to search algorithms that update the heuristic value in multiple states per move (i.e., with larger local learning spaces such as LSS-LRTA* (Koenig and Sun 2009)).

4.4 Maintaining Heuristic Slope

In this section we will prove that there exists a tie-breaking schema that forces the agent to maintain a non-increasing heuristic along its route. The agent's *route* is defined as a sequence of states from the start state s_0 to the current state s_t with any loops removed. For instance, if by the time $t = 5$ the agent has traversed the path $P = (s_0, A, B, A, C, D)$ then its route $R = (s_0, A, C, D)$. A heuristic *profile* is the vector of heuristic values along the route.

We will construct a tie-breaking schema such that whenever this property is violated due to raising heuristic at the current state, the agent will be forced to backtrack, removing the offending state from its route. To illustrate, consider Figure 3. In the second row the agent increases the heuristic of its current state and violates the non-increasing property. The agent therefore backtracks, making a move to the left thereby removing the offending state from its route. Only after reaching the far-left state can the agent once again move forward while satisfying the inequality (the last plate of the second row and all plates in the third row).

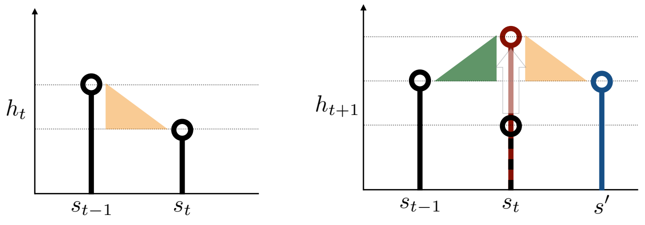

Figure 4: Why raising $h(s_t)$ above $h(s_{t-1})$ allows a tie-breaking schema to force the agent to backtrack.

Lemma 2 (Forced Backtracking) Consider an agent in a non-goal state s_t at time t. Its current route is $R = (s_0, \ldots, s_{t-1}, s_t)$. Then there exists a tie-breaking schema τ such that after updating the heuristic of s_t from $h_t(s_t)$ to $h_{t+1}(s_t)$ if the updated value raises the heuristic profile so that it ceases to be non-increasing, then the agent will backtrack to its previous state s_{t-1}:

$$h_{t+1}(s_{t-1}) < h_{t+1}(s_t) \implies s_{t+1} = s_{t-1}. \qquad (1)$$

Proof. First, as the agent moved from s_{t-1} to s_t, it must hold that $h_t(s_{t-1}) \geq 1 + h_t(s_t)$ which implies that $h_t(s_{t-1}) >$

$h_t(s_t)$. This means that the only way the antecedent of implication (1) can become true is if the agent, being in state s_t, raises its heuristic $h_t(s_t)$ to $h_{t+1}(s_t)$ which, in turn, exceeds $h_t(s_{t-1}) = h_{t+1}(s_{t-1})$. Suppose the raise did happen and $h_{t+1}(s_t) = 1 + h_t(s')$ where $s' \in N(s_t)$ is a state with the lowest heuristic among s_t's neighbors (Figure 4). As all edge costs are 1, all heuristic values are integers[2] which means that $h_t(s') = h_t(s_{t-1})$. This means that after the raise, the state s_{t-1} will be among the ties of $\arg\min_{s \in N(s_t)} h_t(s)$. Thus, a tie-breaking schema will be able to break the ties towards s_{t-1}, making the agent backtrack from s_t to s_{t-1}. This tie-breaking schema is τ. □

Since backtracking removes the offending state from the agent's route, we have the following corollary.

Corollary 1 (Heuristic Slope) At any time during the search, there exists a tie-breaking schema such that the heuristic along the agent's route $R = (r_1, \dots, r_n)$ is non-increasing:

$$h_t(r_1) \geq h_t(r_2) \geq \cdots \geq h_t(r_n). \qquad (2)$$

Proof. We prove this claim by induction on route length n. For $n = 1$ inequality (2) trivially holds. Suppose it holds for the route of length n then when the $(n+1)$*th* state is added to the route, by Lemma 2 it must hold that the heuristic of the added state is less than or equal to the heuristic of the route's previous end (for otherwise the agent would backtrack and the route would not grow). □

4.5 A Lower Bound on the Maximum Heuristic Encountered

Assume that at time t the agent added the state s_b to its route. According to Corollary 1 this means that $h_t(s_a) \geq h_t(s_b)$ for any state s_a already in the route. Since heuristic values never decrease during learning (due to consistency), this also means that $h_t(s_a) \geq h_0(s_b)$ which implies that the agent must raise the heuristic of s_a by at least $h_0(s_b) - h_0(s_a)$. We are interested in identifying the states in a particular problem which maximize the difference $h_0(s_b) - h_0(s_a)$. In this section we show how to find these states. In particular, s_b will be the state with largest heuristic that the agent *must* encounter en route to the goal.

Figure 5 illustrates this concept with pathfinding on a video-game map. Suppose that the agent is in state s_a and is trying to reach state s_g. We observe that the agent must pass through one of the states within the bottleneck C_1 before reaching the goal. Similarly, the agent must also pass through one of the states in C_2 before reaching the goal.

We say that a set of states C is a *cut set* with respect to the states s_a and s_g iff $s_a \notin C$, $s_g \notin C$ and all routes $R = (s_a, ..., s_g)$ have a non-empty intersection with C. To illustrate, in Figure 5 the sets C_1, C_2 and $C_3 \cup C_4$ are three different cut sets with respect to s_a and s_g. Given two states s_a and s_g, we denote the set of all their cut sets as $\mathbf{C}(s_a, s_g)$.

We use the notion of cut sets to derive a lower bound on the maximum heuristic value seen by an agent en route to the goal. For the map in Figure 5, assuming the standard straight-line heuristic, C_1 is not the best cut set because the initial heuristic values of its states will be relatively small. C_2 is better because its heuristic values will be larger. $C_3 \cup C_4$ is an even better cut set as it will have the largest heuristic values among $C_1, C_2, C_3 \cup C_4$.

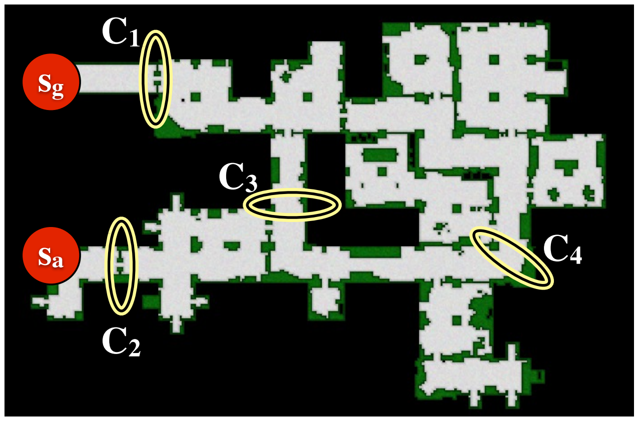

Figure 5: A two-dimensional navigational search problem with the goal state labeled s_g and the start state labeled as s_a. Several cut sets are labeled C_i and explained in the text.

In general, we can find the best lower bound by considering all cut sets and choosing the one with the maximal minimal initial heuristic:

$$\hat{h}(s_a, s_g) = \max_{C \in \mathbf{C}(s_a, s_g)} \min_{s \in C} h_0(s) \qquad (3)$$

From this definition and the definition of a cut set it follows that an agent will necessarily travel through some state s such that $h_0(s) \geq \hat{h}(s_a, s_g)$. Thus, $\hat{h}(s_a, s_g)$ is a useful lower bound on the maximum heuristic encountered along a route from s_a to s_g.

Note that these two states can be arbitrary states in S. Thus, if $R(\mathbf{S})$ is the set of states in a route generated by the agent while solving the problem $\mathbf{S}$ then we can define:

$$\begin{aligned} \Delta_h &= \max_{s \in R(\mathbf{S})} \hat{h}(s, s_g) - h_0(s) \qquad (4) \\ s_\Delta &= \arg\max_{s \in R(\mathbf{S})} \hat{h}(s, s_g) - h_0(s). \qquad (5) \end{aligned}$$

Here s_Δ is a state on the agent's route to goal whose heuristic value has to be (maximally) raised by at least Δ_h:

$$\Delta_h \leq h_T(s_\Delta) - h_0(s_\Delta). \qquad (6)$$

While all states along a route $R(\mathbf{S})$ may not be known *a priori*, $R(\mathbf{S})$ must contain the initial state s_0 which means that $\Delta_h \geq \hat{h}(s_0, s_g) - h_0(s_0)$. Furthermore, for certain classes of problems it may be possible to identify a state in $R(\mathbf{S})$ which yields a lower bound on Δ_h higher than $\hat{h}(s_0, s_g) - h_0(s_0)$.

Figure 6 illustrates this with a simple one-dimensional search space where all states between the start state s_0 and

[2]If they are not integers we can, without loss of generality, take the ceiling to make them integers.

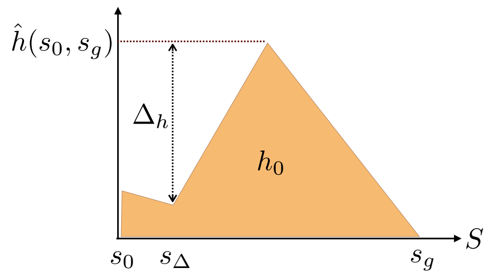

Figure 6: An illustration of s_Δ, Δ_h and $\hat{h}(s_0, s_g)$.

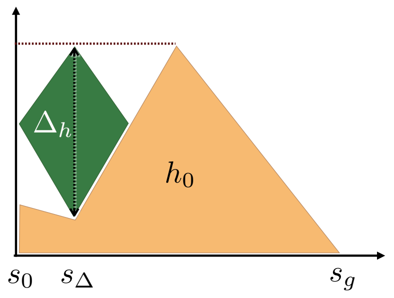

Figure 7: A lower bound on $L_{\min}$ determined by s_Δ, Δ_h.

the goal state s_g belong to a cut set. Hence $\hat{h}(s_0, s_g)$ tracks the highest initial heuristic on the agent's route. Thus, the heuristic of all states to the left of the peak will have to be raised to at least $\hat{h}(s_0, s_g)$. The maximum amount of learning, Δ_h, happens in the state s_Δ.

4.6 Minimum Learning and Minimum Travel Required Overall

We have now established that there exists a state s_Δ along the agent's route to goal whose heuristic the agent will necessarily raise by at least Δ_h. Per Lemma 1, the amount of learning per move is constant-bounded, the number of visits to the state s_Δ is $\Omega(\Delta_h)$ which will cause repeated visits to the state as Δ_h increases.

Consistency of the heuristic allows us to derive even stronger lower bounds on the total amount of learning and travel cost. Indeed, raising the heuristic of s_Δ by Δ_h implies that the heuristic values of many other states have to be raised as well, contributing to the total amount of learning, travel and state re-visits.

When the agent reaches the goal at T, by definition of consistency:

$$\forall n \in S \left[h_T(n) \geq h_T(s_\Delta) - h^*(s_\Delta, n) \right]. \tag{7}$$

The initial heuristic is consistent and hence upper-bounded:

$$\forall n \in S \left[h_0(n) \leq h_0(s_\Delta) + h^*(s_\Delta, n) \right]. \tag{8}$$

For each state $n \in S$, the difference between the left sides of (7) and (8) is at least as large as the difference between their right sides:

$$h_T(n) - h_0(n) \quad \geq \quad h_T(s_\Delta) - 2h^*(s_\Delta, n) - h_0(s_\Delta)$$

which, due to (6), becomes:

$$\geq \quad \Delta_h - 2h^*(s_\Delta, n) \tag{9}$$

We sum the right side of (9) over all states $n \in S$ and, since no heuristic value decreases during learning, we derive the following lower bound on the total amount of learning:

$$L_{\min}(\mathbf{S}) \quad \geq \quad \sum_{n \in S} \max\{0, \Delta_h - 2h^*(s_\Delta, n)\}. \tag{10}$$

This is illustrated in Figure 7 where consistency of the heuristic determines the diamond around Δ_h. The area of the diamond is the right side of inequality (10).

As the amount of learning per move is constant bounded (Section 4.3) we can also derive a lower bound on the amount of travel:

$$T_{\min}(\mathbf{S}) \in \Omega\left(L_{\min}(\mathbf{S})\right). \tag{11}$$

4.7 Special Case: Isotropic Polynomial State Spaces

The lower bounds on the amount of learning (10) and total travel (11) hold for a general search space $\mathbf{S}$. The problem lies with computing the number of states $\eta(r, S, E)$ which are precisely distance r away from the state s_Δ:

$$\eta(r, S, E) = |\{s \in S \mid h^*(s, s_\Delta) = r\}|. \tag{12}$$

Generally speaking, $\eta(r, S, E)$ depends on the topology of the search graph (S, E) and can be arbitrarily complex. However, if the search space is *isotropic* around the state s_Δ then the number of states distance r away from s_Δ does not depend on the direction and can be described simply as $\eta(r)$. Then the sum in inequality (10) can then be alternatively computed as an integral over the shortest distance r between the state s_Δ and all other states of the space:

$$\sum_{n \in S} \max\{0, \Delta_h - 2h^*(s_\Delta, n)\} = \int_0^{\frac{\Delta_h}{2}} \eta(r)(\Delta_h - 2r)dr. \tag{13}$$

We can further simplify the computation of the integral (13) for *polynomial* search spaces which extend from s_Δ for the distance of at least $\Delta_h/2$. Together these assumptions make $\eta(r)$ a $(d-1)$-degree polynomial of r where d is the dimension of the space. While a general $(d-1)$-degree polynomial of r is $\sum_{i=0}^{d-1} k_i r^i$, for the sake of brevity of the following asymptotic analysis and without loss of generality we consider only the dominating term of the polynomial:

$$\eta(r) = \alpha r^{d-1} \tag{14}$$

For instance, for common two-dimensional navigational maps which are also isotropic and extend indefinitely around the state s_Δ, $d = 2$ and $\eta(r) = \alpha r^{2-1} = \alpha r \in \Theta(r)$. Note that this is a theoretical abstraction because real-life maps (e.g., Figure 5) are not isotropic as they have an asymmetric structure. They may also not stretch far enough around the state s_Δ. Finally, they are not polynomial as the obstacles may non-uniformly reduce the number of available states distance r away from any state.

Substituting $\eta(r) = \alpha r^{d-1}$ in (13), we get:

$$\begin{aligned}
\int_0^{\frac{\Delta_h}{2}} \eta(r)(\Delta_h - 2r)dr &= \\
\int_0^{\frac{\Delta_h}{2}} \alpha r^{d-1}(\Delta_h - 2r)dr &= \\
\Delta_h\alpha \int_0^{\frac{\Delta_h}{2}} r^{d-1}dr - 2\alpha \int_0^{\frac{\Delta_h}{2}} r^{d-1}rdr &= \\
\frac{\Delta_h\alpha}{d} r^d \Big|_0^{\frac{\Delta_h}{2}} - \frac{2\alpha}{d+1} r^{d+1} \Big|_0^{\frac{\Delta_h}{2}} &= \\
\frac{\Delta_h\alpha}{d}\left(\frac{\Delta_h}{2}\right)^d - \frac{2\alpha}{d+1}\left(\frac{\Delta_h}{2}\right)^{d+1} &= \\
\frac{\alpha}{d2^d}(\Delta_h)^{d+1} - \frac{\alpha}{(d+1)2^d}(\Delta_h)^{d+1} &= \\
\frac{\alpha}{2^d}\left(\frac{1}{d} - \frac{1}{d+1}\right)(\Delta_h)^{d+1} &= \\
\frac{\alpha}{2^d d(d+1)}(\Delta_h)^{d+1} &\in \\
\Theta((\Delta_h)^{d+1}). \qquad (15)
\end{aligned}$$

Combining (10), (13) and (15) and we conclude that for isotropic polynomial spaces of dimension d that extend for distance at least $\Delta_h/2$ around the state s_Δ, the minimum amount of total learning is lower-bounded as:

$$L_{\min}(\mathbf{S}) \in \Omega\left((\Delta_h)^{d+1}\right). \qquad (16)$$

As the amount of learning per step is constant-bounded, the same asymptotic lower bound applies to the travel cost:

$$T_{\min}(\mathbf{S}) \in \Omega\left((\Delta_h)^{d+1}\right). \qquad (17)$$

Note that under our assumptions, the number of states *within* radius r of the state s_Δ grows as:

$$\int_0^r \eta(x)dx = \int_0^r \alpha x^{d-1}dx \in \Theta(r^d). \qquad (18)$$

Suppose that when an isotropic polynomial search space of dimension d increases, we are able to maintain the heuristic delta Δ_h a linear function of state space diameter $\max_{a,b\in S} h^*(a,b)$ (see example in Section 5.1 and Figure 8). Then the minimum amount of travel $T_{\min}$ will grow at least as a polynomial of degree $d+1$ (Equation (17)) whereas the number of states will grow as a polynomial of degree d (Equation (18)). Consequently, *any* real-time heuristic search algorithm that follows the basic framework formulated earlier in this paper and is unlucky enough with its tie breaking will necessarily revisit its states. The number of revisits scales up with the search space size.

This result is probably not completely surprising to any researcher that has spent time watching these algorithms perform in practice, but there are some limitations to this theory. Most importantly, we have only shown that this holds for a basic agent design. We conjecture that these bounds hold for some of the more advanced agent designs, such as with larger lookahead during learning, but we do not yet have theoretical results to confirm this.

5 Experimental Results

To derive the theoretical lower bounds in the previous sections we have made a number of simplifying abstractions. For instance, we used a particular tie-breaking schema, unit edge costs, and limited the agent's lookahead during learning to 1. The experiments presented in this section support this conjecture empirically by utilizing different tie-breaking rules, non-unit edge costs, and different lookahead depths.

In the first experiment we investigate applicability of Corollary 1 to other tie-breaking schemas on a synthetic map. We then take the next step and use actual video-game maps and a deeper lookahead.

5.1 Other Tie-Breaking Schemas

Consider a previously published and analyzed scalable corner map (Sturtevant, Bulitko, and Björnsson 2010) in Figure 8.

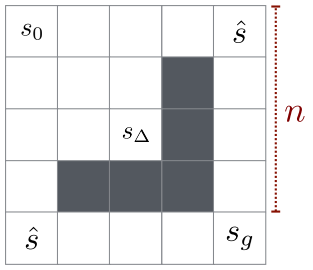

Figure 8: The "corner" map (Sturtevant, Bulitko, and Björnsson 2010).

This example contains a corner-shaped wall with the start s_0 in the upper left corner and the goal s_g behind the wall in the bottom right corner. The map is 8-connected; diagonal movement costs 1.5. An octile distance heuristic will mislead the agent into traveling to s_Δ (Equation (5)) while trying to reach the goal. Under the tie-breaking schema τ constructed in this paper, the final heuristic value of that state, $h_T(s_\Delta)$, will be raised to at least $\hat{h}(s_0, s_g) = h_0(\hat{s})$, based on the states marked $\hat{s}$ in the figure. On this map $\hat{h}(s_0, s_g) = h_0(\hat{s}) = n$ where $n+1$ is the number of cells along the side of the map. By recording $h_T(s_\Delta) - \hat{h}(s_0, s_g) = h_T(s_\Delta) - n$ we can see how different tie-breaking schema affect the theoretical bound. Specifically, a measurement $h_T(s_\Delta) - n < 0$ indicates that the agent had more fortunate tie-breaking schema than τ.

For each value of n, we ran 20 trials of LRTA* with lookahead of 1 and random tie breaking (instead of τ). We also experimented with fixed tie-breaking and got similar results. Averages, maxima and minima of $h_T(s_\Delta) - n$ as a function

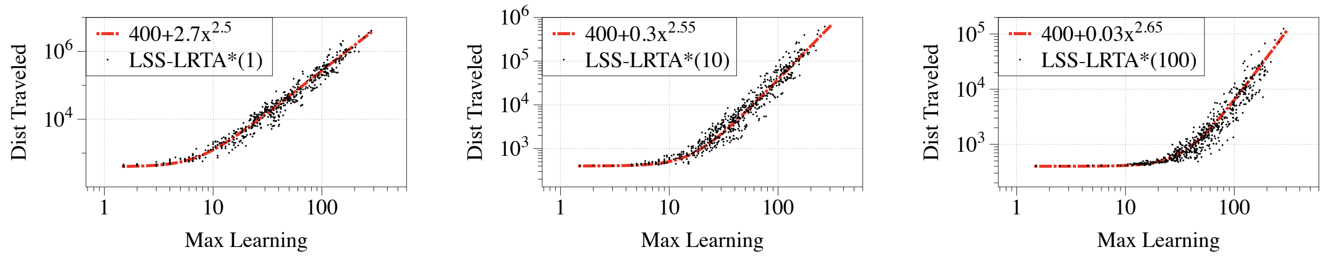

Figure 10: Solution cost versus max learning for any state when solving a problem instance using LSS-LRTA* with lookahead depths of 1, 10 and 100.

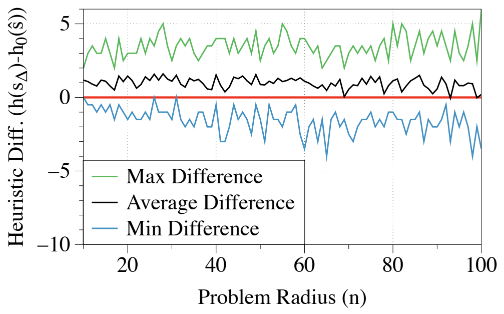

Figure 9: Readings of $h_T(s_\Delta) - n$ recorded by running LRTA* on the corner map.

of n are plotted in Figure 9. Although on 18% of the trials LRTA* with randomized tie-breaking did better than our tie-breaking schema τ would suggest, the difference does not seem to grow significantly as the problem size increases. This suggests that Δ_h is a robust measure of the learning that must occur for the agent to reach the goal.

5.2 Pathfinding on Video-Game Maps

In this experiment we identify the state with the maximum learning after an agent solves a problem. We then attempt to correlate the amount of learning at that state to the cost of the agent's solution.

We performed this experiment on the Moving AI benchmark set (Sturtevant 2012), on all problems in *Dragon Age: Origins* maps with the optimal solution cost in $[400, 404)$. We ran these problems with LSS-LRTA* (Koenig and Sun 2009) with lookahead depths of 1, 10 and 100 (LSS-LRTA* with lookahead 1 is equivalent to Algorithm 1). Diagonal movement was allowed with cost 1.5. Ties were broken systematically in each state without randomization.

A total of 600 problems were used. We measure the maximum learning that occurred in any state as well as the total distance traveled before reaching the goal, and then plot one point for each problem instance in our test set. The results are shown in Figure 10.

We visually fit a polynomial $400 + c_1 \cdot x^{c_2}$ to the data as we knew that all problems have the optimal solution cost between 400 and 403. The values for c_1 and c_2 for each of the three lookahead depths are shown in the figure. The correlation coefficients are 0.94, 0.92 and 0.91 respectively.

These two-dimensional $d = 2$ maps are not isotropic around the state s_Δ due to their asymmetric topology, have non-unit-cost edges, are not fully polynomial due to non-uniform obstacle distribution, and may not extend far enough from s_Δ to satisfy the conditions of the analysis in Section 4.7. As a result, our theory does not require the asymptotic bound of $\Omega((\Delta_h)^3)$ to hold. However, the manually fit polynomial curves do come somewhat close with the degree of the polynomial being between 2.5 and 2.65. The data therefore appears to support our claim that maximum heuristic error (Δ_h) is predictive of the total movement required to reach the goal. We also see that as Δ_h grows across different maps and problem instances, the distance moved appears to grow faster than the state space.

6 Conclusions

The primary contribution of this paper is the development of a non-trivial lower bound on the minimum travel *any* LRTA*-like real-time heuristic search agent may have to perform to reach the goal state. In theoretical polynomial state spaces the lower bound grows asymptotically faster than the state space which means that the agent will necessarily "scrub" (revisit states) – an undesirable behavior in many applications such as real-time pathfinding in video games. These theoretical results are supported experimentally with real-life polynomial search problems.

Our proofs depend on many assumptions, such as unit edge costs and one-step lookahead. While we conjecture that the results hold with arbitrary edge costs and larger lookahead, we do not as yet have a complete proof for it. As such, this is an important area of future work.

This bound may appear discouraging as it suggests that common real-time heuristic search algorithms may not, on their own, be able to avoid "scrubbing". On the positive side, the analysis points to two areas for future research: improved heuristic learning algorithms and improved rules for movement and tie breaking. Work in this direction has already started (Sturtevant and Bulitko 2011; Hernández and Baier 2012) and we hope that our theoretical results will encourage further work in this direction.

References

Bulitko, V. K., and Bulitko, V. 2009. On backtracking in real-time heuristic search. *CoRR* abs/0912.3228.

Bulitko, V., and Lee, G. 2006. Learning in real time search: A unifying framework. *Journal of Artificial Intelligence Research (JAIR)* 25:119–157.

Edelkamp, S., and Schrödl, S. 2012. *Heuristic Search - Theory and Applications*. Academic Press.

Furcy, D., and Koenig, S. 2000. Speeding up the convergence of real-time search. In *National Conference on Artificial Intelligence (AAAI)*, 891–897.

Hernández, C., and Baier, J. A. 2012. Avoiding and escaping depressions in real-time heuristic search. *Journal of Artificial Intelligence Research (JAIR)* 43:523–570.

Hernández, C., and Meseguer, P. 2005. LRTA*(k). In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1238–1243.

Ishida, T., and Korf, R. 1991. Moving target search. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 204–210.

Koenig, S., and Simmons, R. G. 1992. Complexity analysis of real-time reinforcement learning applied to finding shortest paths in deterministic domains. Technical Report CMU–CS–93–106, School of Computer Science, Carnegie Mellon University, Pittsburgh.

Koenig, S., and Simmons, R. G. 1993. Complexity analysis of real-time reinforcement learning. In *National Conference on Artificial Intelligence (AAAI)*, 99–105.

Koenig, S., and Simmons, R. G. 1996. The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms. *Machine Learning* 22(1-3):227–250.

Koenig, S., and Sun, X. 2009. Comparing real-time and incremental heuristic search for real-time situated agents. *Journal of Autonomous Agents and Multi-Agent Systems* 18(3):313–341.

Koenig, S.; Tovey, C.; and Smirnov, Y. 2003. Performance bounds for planning in unknown terrain. *AI* 147:253–279.

Koenig, S. 2001. Agent-centered search. *Artificial Intelligence Magazine* 22(4):109–132.

Korf, R. 1990. Real-time heuristic search. *Artificial Intelligence* 42(2–3):189–211.

Shimbo, M., and Ishida, T. 2003. Controlling the learning process of real-time heuristic search. *Artificial Intelligence* 146(1):1–41.

Shue, L.-Y., and Zamani, R. 1993a. An admissible heuristic search algorithm. In *International Symposium on Methodologies for Intelligent Systems (ISMIS-93)*, volume 689 of *LNAI*, 69–75.

Shue, L.-Y., and Zamani, R. 1993b. A heuristic search algorithm with learning capability. In *ACME Transactions*, 233–236.

Shue, L.-Y.; Li, S.-T.; and Zamani, R. 2001. An intelligent heuristic algorithm for project scheduling problems. In *Annual Meeting of the Decision Sciences Institute*.

Sturtevant, N. R., and Bulitko, V. 2011. Learning where you are going and from whence you came: H- and G-cost learning in real-time heuristic search. In *International Joint Conference on Artificial Intelligence (IJCAI)*, IJCAI'11, 365–370. AAAI Press.

Sturtevant, N.; Bulitko, V.; and Björnsson, Y. 2010. On learning in agent-centered search. In *Autonomous Agents and Multiagent Systems (AAMAS)*, 333–340. International Foundation for Autonomous Agents and Multiagent Systems.

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

Zamani, R., and Shue, L.-Y. 2001. A heuristic learning algorithm and its application to project scheduling problems. Technical report, Department of Information Systems, University of Wollongong.