Journal of Artificial Intelligence Research 57 (2016) 307–343 Submitted 07/15; published 10/16

Scrubbing During Learning In Real-time Heuristic Search

Nathan R. Sturtevant STURTEVANT@CS.DU.EDU
Department of Computer Science
University of Denver
Denver, Colorado, USA

Vadim Bulitko BULITKO@UALBERTA.CA
Department of Computing Science
University of Alberta
Edmonton, Alberta, T6G 2E8, Canada

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 with simple examples demonstrating worst-case performance in practice. Lower bounds, however, have not been widely studied. In this paper we study best-case performance more generally and derive theoretical lower bounds for reaching the goal using LRTA*, a canonical example of a real-time agent-centered heuristic search algorithm. The results show that, given some reasonable restrictions on the state space and the heuristic function, the number of steps an LRTA*-like algorithm requires to reach the goal will grow asymptotically faster than the state space, resulting in "scrubbing" where the agent repeatedly visits the same state. We then show that while the asymptotic analysis does not hold for more complex real-time search algorithms, experimental results suggest that it is still descriptive of practical performance.

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 & Koenig, 2000; Shimbo & Ishida, 2003; Hernández & Meseguer, 2005; Bulitko & Lee, 2006; Hernández & Baier, 2012) and various theoretical analysis (Ishida & Korf, 1991; Koenig & Simmons, 1993; Koenig, Tovey, & Smirnov, 2003; Bulitko & Lee, 2006; Bulitko & Bulitko, 2009; Sturtevant, Bulitko, & Björnsson, 2010).

Given the broad work on this problem, it is surprising to find that little work has been done on lower bounds. It is clear that in some examples an agent can travel directly to the goal state on an optimal path. Such examples, however, generally require an unreasonably accurate initial heuristic or a favorable domain-specific tie-breaking schema which are hard to guarantee in practice. Other examples (Koenig & Simmons, 1993; Edelkamp & Schrödl, 2012) show that the worst-case bound is tight, but these examples do not generally characterize when the worst-case and best-case performance coincide.

The main contribution of this paper is a non-trivial lower bound on the travel distance required for an agent to reach the goal in the basic real-time heuristic search framework.

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. Our result shows that this phenomenon is unavoidable when an agent has limited 1-step lookahead, the state space is polynomial, all edges have unit costs, and the initial heuristic is consistent and integer valued. We later show that lifting the lookahead and unit edge cost restrictions allows an agent in some cases to travel directly from the start to the goal. However, such counter examples are somewhat contrived and, as our empirical results demonstrate, do not appear to happen frequently in practice. It is an open question whether similar theoretical results can be derived with fewer restrictions on the agent and the state space. We also examine exponential state spaces and learning over multiple trials until convergence. The insights from this theoretical work also provide explanations into why several approaches taken by other recent literature are effective.

This paper is an extended version of our previously published symposium paper (Sturtevant & Bulitko, 2014). The original paper contained the proof of asymptotic state revisitation under the assumptions of 1-step lookahead, polynomial state spaces, unit edge costs, integer initial heuristics, and non-optimal tie-breaking. This journal paper adds counter-examples for exponential state spaces, larger lookahead, and spaces with non-unit edge costs and/or non-integer initial heuristic. Convergence travel is also analyzed, and new experimental results with different tie-breaking rules are added. Finally, we include additional an discussion around all of these issues.

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. We then discuss the applicability of the theory to a broader class of state spaces and algorithms in Section 5 and build concrete counter-examples. The theoretical results are supported by a brief empirical evaluation in Section 6. 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 simple path that does not include duplicate states. Initially we assume that all edge costs are 1 (**Assumption 1**).

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 change its current state, that is, move to any immediately neighboring state in $N(s)$. The traversal incurs a travel cost of $c(s_t, s_{t+1})$. 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* and is the primary performance metric we are concerned with in this paper. The cost of the shortest possible path between states $a, b \in S$ is denoted by $h^*(a, b)$. We abbreviate $h^*(s, s_g)$ as $h^*(s)$. 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 heuristic is integer-valued (**Assumption 2**). The search agent can modify the heuristic as it sees fit as long as it remains admissible $\forall s \in S\,[h(s) \leq h^*(s)]$, consistent $\forall a, b \in S\,[|h(a) - h(b)| \leq h^*(a, b)]$ and integer-valued at all times. (Admissibility and consistency are assumed throughout the paper.) The heuristic at time t is denoted by h_t; $h_0 = h$. The total magnitude of all updates to the heuristic function performed by the agent between $t = 0$ and $t = T$ is called the *total learning* amount.

Algorithm 1: Basic Real-time Heuristic Search

```
   input  : search problem $(S, E, s_0, s_g, h)$, tie-breaking schema $\tau$
   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      $s_{t+1} \leftarrow \arg\min^{\tau}_{s' \in N(s_t)}(1 + h_t(s'))$
5      $h_{t+1}(s_t) \leftarrow 1 + h_t(s_{t+1})$
6      $t \leftarrow t + 1$
7  $T \leftarrow t$
```

Numerous heuristic search algorithms have been developed for the problem described above (e.g., Korf, 1990; Bulitko & Lee, 2006; Koenig & Sun, 2009). Most of them are based on Algorithm 1. 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 computes the immediate neighbor that minimizes the estimated cost of going to that neighbor (always 1 for now) plus the estimated cost of going from that neighbor to the goal. The lookahead is 1 and thus only immediate neighbors (i.e., $N(s_t)$) are considered by the agent (**Assumption 3**). Ties among neighbors that have the same minimal heuristic values are broken with a tie-breaking schema τ that we provide (**Assumption 4**), as detailed later in the paper, which is sufficiently suboptimal to prove our results. Then, in line 5, the agent updates (or learns) its heuristic in the current state by making it consistent with the heuristic of the neighbor picked in the previous line. The agent then changes its current state to the neighbor (i.e., makes a move). While the general problem allows for the heuristic to decrease, this algorithm will never decrease the heuristic value of a state, since the heuristic is always consistent.

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 $T_{\min}(\mathbf{S})$. In doing so we are concerned with *systematic scrubbing*: an agent is said to scrub systematically on a series of growing state spaces iff the number of state visits asymptotically dominates the number of states in the space: $T_{\min}(\mathbf{S}) \in \omega(|S|)$.

Formally, consider a series of seach problems $\{\mathbf{S}_1, \mathbf{S}_2, \dots \mathbf{S}_2, \dots\}$ of progressively increasing state spaces: $|S_i| = g(i)$ (e.g., $|S_i| = i^2$ for an open two-dimensional grid S_i of $i \times i$ cells). Then the agent scrubs systematically iff its travel time is lower-bounded by a function $T_{\min}(\mathbf{S}_i) = f(i)$ which asymptotically dominates the state space size: $f \in \omega(g)$. We use the standard definition of ω as asymptotic dominance: $f(n) \in \omega(g(n))$ iff $\forall k > 0 \exists n_0 \forall n > n_0 \, [kg(n) \leq f(n)]$. Since all our functions are positive, we omit the absolute value.

We initially work under the assumption that the agent is only trying to reach the goal once (**Assumption 5**), for which we measure *first-trial* travel, some other work has considered *convergence* travel. In convergence travel the agent is teleported back to the start state after reaching the goal, beginning a new trial, now with the updated heuristic. The agent's learning has converged when a trial does not result in further updates to the heuristic function (Bulitko & Lee, 2006). If a systematic tie-breaking schema is used, convergence entails that the agent will follow the same path to the goal on all subsequent trials. First-trial travel is trivially a lower bound on the convergence travel; we will discuss this connection in more detail later in the paper.

3. Related Work

Previous work has focused 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 & Korf, 1991). This follows from the analysis of MTS (Ishida & 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 & Schrödl, 2012).[1] The analysis has been extended to a class of reinforcement learning algorithms, of which LRTA* is a special case (Koenig & Simmons, 1992, 1993, 1996). For state-based value-iteration algorithms, such as the search algorithm listed above, the upper bound is still $O(|S|^2)$.

There has also been substantial work on analyzing algorithms with backtracking moves, introduced with SLA* (Shue & Zamani, 1993a, 1993b) and the more general SLA*T (Shue, Li, & Zamani, 2001; Zamani & 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 by 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 it.

A problem-specific analysis of the minimum learning required to prove the optimal path was described by Sturtevant et al. (2010) but they did not consider the first-trial performance. They provided a proof sketch that agents that lookahead farther will not converge to an optimal solution more quickly. Empirical results suggested that algorithms that look farther ahead have better convergence performance primarily because they perform less learning over and above the minimum required.

1. The chapter containing these results is authored by Koenig.

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 illustrates how tie-breaking can influence best- and worst-case performance, using two simple examples. We present a high-level overview of our 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 4.9 and discuss its implications on state revisits (i.e., "scrubbing"). These results require all of our assumptions (1-5). After this presentation is completed, in Section 5 we discuss extensions of this work to state spaces with non-unit edge costs and larger lookahead. In Appendix A we consider exponential state spaces.

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 heuristics of its immediate neighbors, to the left and right, and goes to the neighbor with the lowest heuristic. If the neighbors' heuristic values are identical then, for the time being, we assume that ties are broken to the neighbor on the right. This is a *favorable* or *fortunate* tie-breaking rule for this example because it moves the agent towards the goal when the heuristic is not sufficient to do so by itself. A tie-breaking schema that breaks ties away from the goal, in this example, would be *unfavorable* or *unfortunate*. In this example we temporarily break Assumption 4 of an agent with unfortunate tie breaking. We use several variations on this problem to illustrate the importance of tie-breaking on solution travel cost.

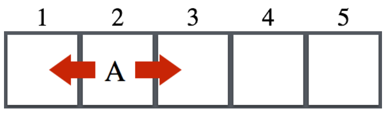


Figure 1: A one-dimensional grid world of five states numbered at the top. The agent 'A' is located in state 2 and has two available actions shown in the figure.

In Figure 2 we plot the initial heuristic values of each state in a ten-state version of this grid, 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 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 and no systematic scrubbing is observed (i.e., $T_{\min}(\mathbf{S}) \notin \omega(|S|)$).

However, tie breaking can adversely affect the agent's performance, and in general there is no guidance available in a state space to allow favorable tie-breaking. Consider the search problem in Figure 3 with 13 states. In this example, each tie is broken away from the goal

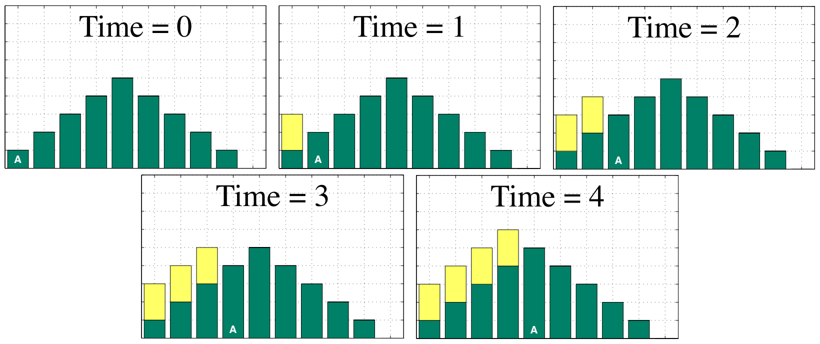


Figure 2: Heuristic learning with favorable tie breaking. The ten states of the problem are along the horizontal axis. The vertical axis shows the value of the heuristic function per time step. The darker bars are the initial heuristic values. The lighter bars are the increases due to learning. The initial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

(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 state space size. Thus, $T_{\min}(\mathbf{S}) \in \omega(|S|)$ as $|S|^2 \in \omega(|S|)$ and we have systematic scrubbing. The difference between these two cases is asymptotically significant. While favorable tie-breaking can be achieved in specific examples, it cannot be always guaranteed in general.

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 a favorable tie-breaking schema for a given problem (or series of problems), we consider the agent's performance with sufficiently suboptimal tie-breaking schemas (Assumption 4) 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 then detail each step individually in the subsequent sections. First, due to consistency of the heuristic and a bounded lookahead, 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 show that with a sufficiently suboptimal tie breaking schema an agent traveling from s_a to s_b must raise the heuristic of s_a to at least that of

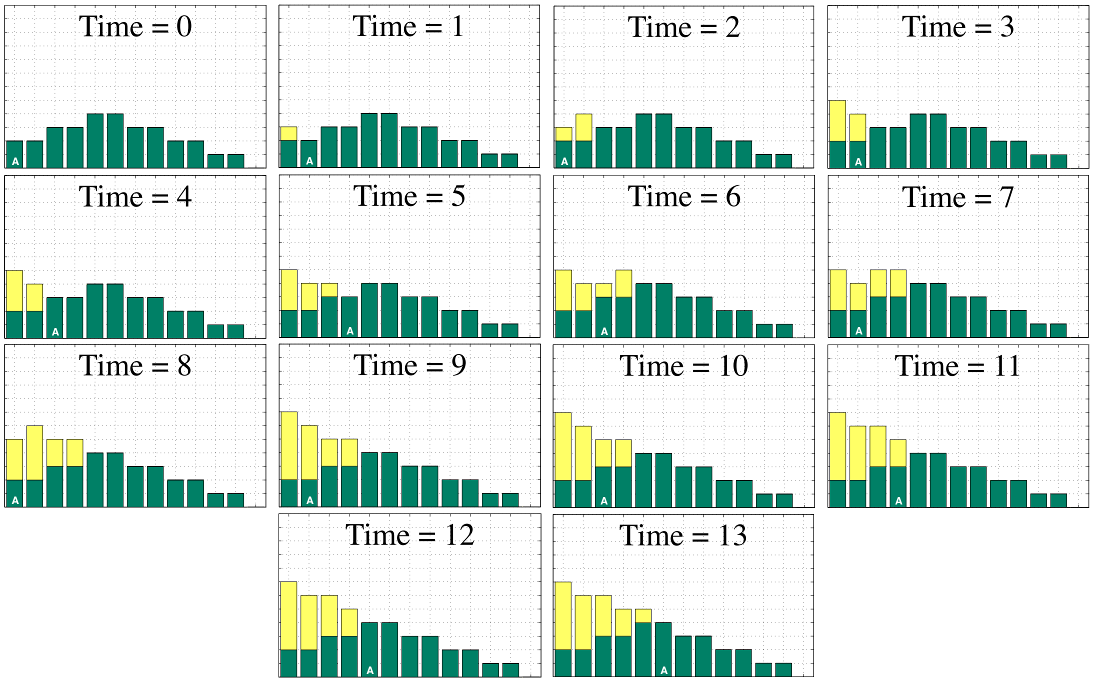


Figure 3: Heuristic learning with unfortunate tie breaking.

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 consistent at all times we use $\hat{h}$ to compute the minimum amount of learning (Section 4.6).

We then apply the argument to polynomial state spaces where the number of states within distance r from a given state grows as $\Theta(r^d)$ (e.g., quadratically on commonly used two-dimensional navigation maps with $d = 2$).[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 asymptotically dominates the number of states $\Theta(r^d)$. Since learning per step is constant-bounded, the amount of travel will also asymptotically dominate the number of states. This result indicates that any real-time heuristic search agent of the type introduced above will necessarily scrub systematically (Corollary 2).

4.3 Learning per Step is Constant Bounded

Here we provide a bound on the learning required to solve a problem the first time — the cumulative updates to the heuristic function from $t = 0$ until $t = T$. In this section we begin by showing that the learning per step is constant-bounded. This will imply that a lower bound on learning is also a lower bound on movement.

2. We use the standard definition of Θ as "bounded above and below": $f(n) \in \Theta(g(n))$ iff $\exists k_1 > 0 \exists k_2 > 0 \exists n_0 \forall n > n_0 \, [k_1 g(n) \leq f(n) \leq k_2 g(n)]$, of Ω as "bounded below": $f(n) \in \Omega(g(n))$ iff $\exists k > 0 \exists n_0 \forall n > n_0 \, [k g(n) \leq f(n)]$ and of O as "bounded above": $f(n) \in O(g(n))$ iff $g(n) \in \Omega(f(n))$.

Lemma 1 The total change in heuristic values during each learning step of Algorithm 1 is constant bounded independently of the number of states in the state space.

Proof. Algorithm 1 updates the heuristic of a state s_t in line 5 of the pseudo code based on the heuristic of a neighboring state s_{t+1}. Because s_{t+1} is an immediate neighbor of s_t and because the heuristic is consistent, $|h_t(s_t) - h_t(s_{t+1})| \leq h^*(s_t, s_{t+1}) = 1$. By definition, 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 in heuristic values at each time step is 2, which is constant bounded. $\square$

We now measure the learning necessary to move between different locations in the world.

4.4 Maintaining a Non-increasing Heuristic Slope

In this section we will prove that there exists a tie-breaking schema, τ, that forces the agent to maintain non-increasing heuristic values for states along its route, which we call the *non-increasing heuristic slope property.* While better and worse tie-breaking schemes may exist for specific problems, τ is sufficiently suboptimal to prove our main result.

As defined earlier, the agent's route is a simple path 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. If the heuristic profile along this route is $\{4, 3, 2, 2\}$, the non-increasing property holds. If the heuristic profile is $\{2, 3, 4, 3\}$, the non-increasing properly does not hold.

We will construct a tie-breaking schema such that whenever the non-increasing property is violated by raising the heuristic at the current state, the agent will be forced to backtrack, removing the offending state from its route. To illustrate, consider Figure 3. At time step 5 the agent raises the heuristic of its current state and violates the non-increasing property (the new, larger heuristic value is shown at time step 6). The agent therefore backtracks, making a move to the left, which removes the offending state from its route. Only after reaching the state at time step 8 can the agent once again move forward while satisfying the property.

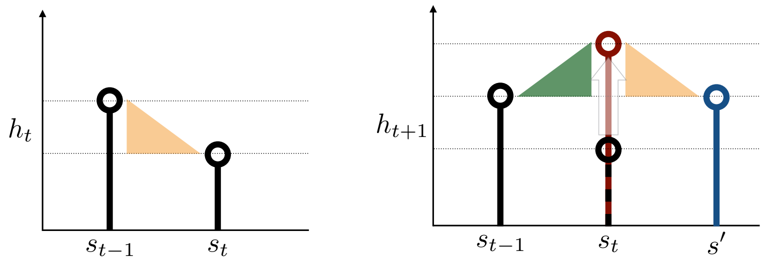


Figure 4: Raising $h(s_t)$ above $h(s_{t-1})$ allows a tie-breaking schema to force the agent to backtrack.

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}:

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

Proof. First, as the agent moved from s_{t-1} to s_t, it must hold that $h_t(s_{t-1}) = 1 + h_t(s_t)$, due to line 5 of Algorithm 1. This is shown in the left side of Figure 4. Due to a consistent, integer-valued heuristic and unit edge costs, the only way the heuristic can be increasing along the route after learning is if $h_{t+1}(s_t) = h_t(s_{t-1}) + 1$, shown on the right side of Figure 4. In this case, the update must have come from some state $s' \in N(s_t)$ which is a state with the lowest heuristic among s_t's neighbors (Figure 4) with $h_t(s') = h_t(s_{t-1})$. Even if s' is not the same as s_{t-1}, it has the same heuristic value. 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} and maintaining the non-increasing heuristic. This tie-breaking schema is τ. $\square$

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 $R = (r_1, \ldots, r_n)$ is non-increasing:

$$h_t(r_1) \geq h_t(r_2) \geq \cdots \geq h_t(r_n). \tag{2}$$

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 (otherwise the agent would backtrack and the route would not grow). $\square$

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. This means, according to Corollary 1, that $h_t(s_a) \geq h_t(s_b)$ for any state s_a already in the route. Informally, before the agent passes through a state with a high heuristic, it must first raise all previous states in its route to have at least equally high heuristics. Since heuristic values in Algorithm 1 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 have already raised the heuristic of s_a by at least $h_0(s_b) - h_0(s_a)$, assuming this term is positive. We are interested in identifying the states in a particular problem which maximize the difference $h_0(s_b) - h_0(s_a)$, as they can be used to bound learning. 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. Informally, we are providing a general definition of a local minima and then proving that the agent must *learn* its way out of the local minima by increasing heuristic values.

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.

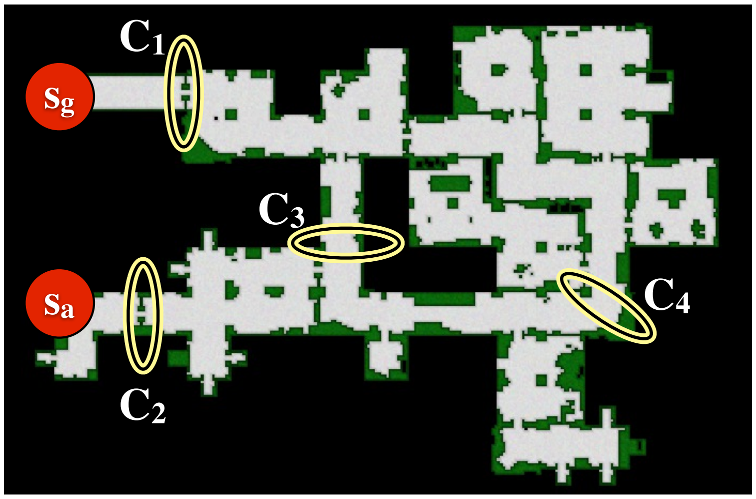


Figure 5: A two-dimensional pathfinding search problem with the goal state labeled s_g and the start state labeled as s_a. C_1, C_2, and $C_3 \cup C_4$ are all cut sets.

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 possible routes $R = (s_a, ..., s_g)$ have a non-empty intersection with C. 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 $\mathbf{C}(s_a, s_g)$. Thus $C_1 \in \mathbf{C}(s_a, s_g)$, $C_2 \in \mathbf{C}(s_a, s_g)$ and $(C_3 \cup C_4) \in \mathbf{C}(s_a, s_g)$.

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 initial heuristic values will be larger. $C_3 \cup C_4$ is an even better cut set as it will have the largest minimum initial heuristic values among $C_1, C_2, C_3 \cup C_4$.

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

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

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.

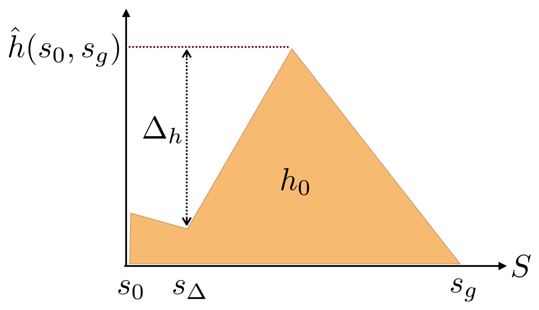


Figure 6: An illustration of s_Δ, Δ_h and $\hat{h}(s_0, s_g)$.

Note that the state s_a can be an arbitrary state 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:

$$\Delta_h = \max_{s \in R(\mathbf{S})} \left[\hat{h}(s, s_g) - h_0(s)\right] \tag{4}$$

$$s_\Delta = \arg \max_{s \in R(\mathbf{S})} \left[\hat{h}(s, s_g) - h_0(s)\right]. \tag{5}$$

If there are several states for which Δ_h is maximized, s_Δ can be set to any of them.

From here it follows that s_Δ is a state on the agent's route to goal whose heuristic value has to be raised by at least Δ_h:

$$\Delta_h \leq h_T(s_\Delta) - h_0(s_\Delta). \tag{6}$$

Figure 6 illustrates this with the heuristic profile of a simple one-dimensional state space where every state between the start state s_0 and the goal state s_g forms a single-state cut set. Hence, $\hat{h}(s_0, s_g)$ is just 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.

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, given (4), means that:

$$\Delta_h \geq \hat{h}(s_0, s_g) - h_0(s_0). \tag{7}$$

Furthermore, it is often possible to identify a state in $R(\mathbf{S})$ which yields a lower bound on Δ_h that is higher than $\hat{h}(s_0, s_g) - h_0(s_0)$. To illustrate: the Δ_h shown in Figure 6 is strictly greater than $\hat{h}(s_0, s_g) - h_0(s_0)$. In a more practical example, shown in Figure 8, the octile distance h_0 on the eight-connected grid leads to $\hat{h}(s_0, s_g) - h_0(s_0) = 0$. But, in practice, an agent will move to the corner first, allowing us to identify a different state as s_Δ. We analyze this example in detail in Section 4.8.

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, so the number of visits to the state s_Δ is $\Omega(\Delta_h)$.

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.

Specifically, since at any time $t \in \{0, \dots, T\}$ the heuristic h_t is consistent, the heuristic value of any state $n \in S$ is upper and lower bounded with respect to an arbitrary state $m \in S$, according to its distance from it:

$$h_t(m) - h^*(m,n) \le h_t(n) \le h_t(m) + h^*(m,n). \tag{8}$$

Thus, when the agent reaches the goal at time T, the heuristic of any state in the state space is lower-bounded according to the distance from s_Δ:

$$\forall n \in S \left[h_T(n) \ge h_T(s_\Delta) - h^*(s_\Delta, n) \right]. \tag{9}$$

The initial heuristic is consistent and hence upper-bounded:

$$\forall n \in S \left[h_0(n) \le h_0(s_\Delta) + h^*(s_\Delta, n) \right]. \tag{10}$$

For each state $n \in S$, the difference between the left sides of (9) and (10) is at least as large as the difference between their right sides:

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

which, due to (6), becomes:

$$\ge \quad \Delta_h - 2h^*(s_\Delta, n) \tag{11}$$

We sum the right side of (11) 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:

$$L_{\min}(\mathbf{S}) \quad \ge \quad \sum_{n \in S} \max\{0, \Delta_h - 2h^*(s_\Delta, n)\}. \tag{12}$$

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 (12). 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:

$$T_{\min}(\mathbf{S}) \in \Omega\left(L_{\min}(\mathbf{S})\right). \tag{13}$$

Note that in this one-dimensional example the non-increasing heuristic slope property allows to derive an even higher lower bound on the necessary amount of learning. Specifically, by the time the agent reaches the goal s_g, all heuristic values to the left of the peak have to be raised to at least the level of the dotted line. The filled-in volume is clearly greater than the area of the diamond in the figure. This indicates a possible direction for future work — finding a lower bound more aggressive than the one in inequality (12) that we use in the rest of the paper.

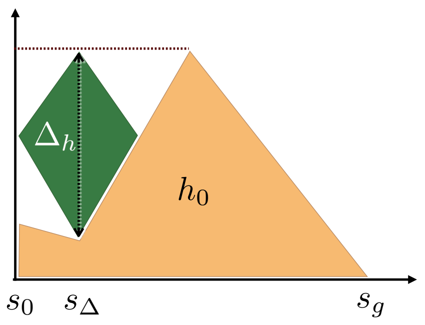


Figure 7: A lower bound on $L_{\min}$ determined by s_Δ, Δ_h.

4.7 General Conditions for Systematic Scrubbing

Recall that our definition of scrubbing is that the number of state visits asymptotically dominates the number of states in the space: $T_{\min}(\mathbf{S}) \in \omega(|S|)$. Thus, one way to establish systematic scrubbing on a series of search problems is to compute or, at least, estimate the sum in Equation 12 and show that it asymptotically dominates the number of states $|S|$. Together with the link between $T_{\min}$ and $L_{\min}$ given by Equation 13, this would establish systematic scrubbing.

Generally speaking, the sum may depend intricately on the search problem structure and the initial heuristic. Thus, from here on we will proceed by analyzing two special cases.

4.8 Special Case: The Corner Map

We cannot make general statements about single problem instances, so instead we parameterize problem instances by their size, allowing us to describe how they scale as the map size grows. In this case we measure the size by the radius or the length of one edge of the map. Consider a series of search problems known as the corner map (Figure 8).

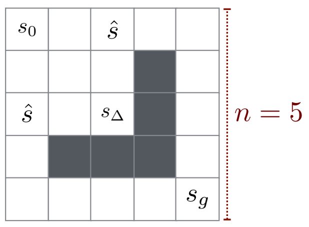


Figure 8: The "corner" map analyzed by Sturtevant et al. (2010).

This map is parameterized by n: a problem instance $\mathbf{S}_n$ for $n > 3$ has $|S_n| \in O(n^2)$ states. This is a two-dimensional grid where the agent can occupy any vacant cell (white in the figure). The agent can move to any of its four cardinal neighbors unless they are blocked by an obstacle (dark grey cells in the figure), all edges have unit cost, and we assume the Manhattan distance as the initial heuristic h_0. Later in the paper we revisit this example with diagonal, non-unit-cost moves and an octile distance heuristic.

If the agent starts in the corner created by the obstacles, that corner state will be s_Δ. Using the analysis in the previous sections, $\hat{h}$ in Equation 3 has the value of $n+1$ in the two states labeled $\hat{s}$ in the figure. Correspondingly, the state s_Δ defined in Equation 5 and also shown in the figure has the initial heuristic value of 4 (for any n); this will be raised to $\hat{h}$ before the goal state is reached. Thus, $\Delta_h = n+1-4 = n-3$. The lower bound on $L_{\min}$ (Inequality 12) becomes:

$$L_{\min}(\mathbf{S}_n) \geq \sum_{s \in S_n} \max\{0, n-3-2h^*(s_\Delta, s)\}. \tag{14}$$

Looking at the structure of the corner, we can re-write the right side of (14) by summing over $i = h^*(s_\Delta, s)$ and multiplying by the number of states with each value of $h^*(s_\Delta, s)$. There is one state (s_Δ) with $h^*(s_\Delta, s) = 0$, two states with $h^*(s_\Delta, s) = 1$, and $i+1$ states with $h^*(s_\Delta, s) = i$. For odd values of n, we re-write the sum in (14) as:[3]

$$L_{\min}(\mathbf{S}_n) \geq \sum_{i=0}^{\frac{n-3}{2}} (i+1)(n-3-2i) = \frac{(n-3)(n-1)(n+1)}{24}. \tag{15}$$

For even values of n the result is:

$$L_{\min}(\mathbf{S}_n) \geq \sum_{i=0}^{\frac{n-4}{2}} (i+1)(n-3-2i) = \frac{(n-2)(n-1)n}{24}. \tag{16}$$

In either case, the sum, as a function of n, is in the class $\Theta(n^3)$ which puts the learning amount $L_{\min}$ in $\Omega(n^3)$. As the amount of learning per move is constant bounded, $T_{\min} \in \Omega(n^3)$. The number of states on a two-dimensional $n \times n$ corner map is $O(n^2)$ which is asymptotically dominated by $T_{\min}$ proving that the agent will scrub systematically on $\{\mathbf{S}_1, \mathbf{S}_2, \dots\}$.

Specific properties of the corner maps allowed us to derive a complexity class for the sum in (12) as well as the total number of states on each map. In the following section we analyze a broader class of search spaces.

4.9 Special Case: Locally Isotropic Polynomial State Spaces

The lower bounds on the amount of learning (12) and total travel (13) hold for a search problem $\mathbf{S}$. However, the bounds do not explicitly reference the number of states in S and thus do not immediately allow us to correlate the amount of travel to the number of states in order to make a claim about state revisitation. To do so we investigate ways of computing the number of states $\eta(r, S, E)$ which are precisely distance r away from the state s_Δ:

$$\eta(r, S, E) = |\{s \in S \mid h^*(s, s_\Delta) = r\}|. \tag{17}$$

Generally speaking, $\eta(r, S, E)$ depends on the topology of the search graph (S, E) and can be arbitrarily complex. However, if the state space is *isotropic* around the state s_Δ

3. This is still a lower-bound for this example because we are not considering that the full path to the goal has to have its heuristic values raised to $h(\hat{s})$. Our analysis assumes that only the first state will have its heuristic raised.

then the number of states distance r away from s_Δ does not depend on the direction and can be described simply as $\eta(r)$. The term isotropic is usually used to refer to the entire state space; we use the term *locally isotropic* to refer to states spaces that are isotropic in a region within radius r of s_Δ.

Then the sum in inequality (12) can be alternatively computed as an integral over the shortest distance r between the state s_Δ and all other states:

$$\sum_{n \in S} \max\{0, \Delta_h - 2h^*(s_\Delta, n)\} = \int_0^{\frac{\Delta_h}{2}} \eta(r)(\Delta_h - 2r)dr. \tag{18}$$

We can further simplify the computation of the integral (18) for *polynomial* state spaces which extend from s_Δ for a distance of at least $\Delta_h/2$; that is, are locally isotropic. These are state spaces in which

$$\eta(r) = \gamma r^{d-1} \tag{19}$$

for $r \in [0, \Delta_h/2]$. For instance, in two-dimensional navigational maps which are also locally isotropic to at least radius $\Delta_h/2$ around the state s_Δ, the degree of the polynomial is $d = 2$ and $\eta(r) = \gamma r^{2-1} = \gamma r, r \in [0, \Delta_h/2]$. Note that this is a theoretical abstraction because states in real-life maps (e.g., Figure 5) are not always locally isotropic; the maps have an asymmetric structure and may not stretch far enough around s_Δ. Also, they may not be polynomial as the obstacles may non-uniformly reduce the number of available states distance r away from any state.

Substituting $\eta(r) = \gamma r^{d-1}$ in (18), we get:

$$\begin{aligned}
\int_0^{\frac{\Delta_h}{2}} \eta(r)(\Delta_h - 2r)dr &= \\
\int_0^{\frac{\Delta_h}{2}} \gamma r^{d-1}(\Delta_h - 2r)dr &= \\
\Delta_h\gamma \int_0^{\frac{\Delta_h}{2}} r^{d-1}dr - 2\gamma \int_0^{\frac{\Delta_h}{2}} r^{d-1}rdr &= \\
\frac{\Delta_h\gamma}{d} r^d \Big|_{r=0}^{r=\frac{\Delta_h}{2}} - \frac{2\gamma}{d+1} r^{d+1} \Big|_{r=0}^{r=\frac{\Delta_h}{2}} &= \\
\frac{\Delta_h\gamma}{d}\left(\frac{\Delta_h}{2}\right)^d - \frac{2\gamma}{d+1}\left(\frac{\Delta_h}{2}\right)^{d+1} &= \\
\frac{\gamma}{d2^d}(\Delta_h)^{d+1} - \frac{\gamma}{(d+1)2^d}(\Delta_h)^{d+1} &= \\
\frac{\gamma}{2^d}\left(\frac{1}{d} - \frac{1}{d+1}\right)(\Delta_h)^{d+1} &= \\
\frac{\gamma}{2^d d(d+1)}(\Delta_h)^{d+1} &\in \Theta((\Delta_h)^{d+1}), \Delta_h \to \infty.
\end{aligned} \tag{20}$$

Combining (12), (18) and (20) and we conclude that for locally 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:

$$L_{\min}(\mathbf{S}) \in \Omega\left((\Delta_h)^{d+1}\right), \Delta_h \to \infty. \tag{21}$$

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

$$T_{\min}(\mathbf{S}) \in \Omega\left((\Delta_h)^{d+1}\right), \Delta_h \to \infty. \tag{22}$$

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

$$\int_0^r \eta(x)dx = \int_0^r \gamma x^{d-1}dx \in \Theta(r^d), r \to \infty. \tag{23}$$

Now, consider an infinite series of growing search problems $\{\mathbf{S}_1, \mathbf{S}_2, \dots\}$. Suppose each search problem $\mathbf{S}_i$ is such that the corresponding state space S_i is locally isotropic and polynomial to radius r_i around the corresponding state s_Δ. Assume also that the degree of the polynomial is $d \geq 1$ and that the radii of the search problems monotonically and unboundedly increase with i:

$$r_1 < r_2 < \dots \tag{24}$$

$$r_i \to \infty \text{ when } i \to \infty. \tag{25}$$

From (23), it follows the state space size is asymptotically lower-bounded by r_i^d:

$$|S_i| \in \Omega(r_i^d), i \to \infty. \tag{26}$$

Suppose also that the state space S_i is asymptotically upper-bounded by r_i^d as well:

$$|S_i| \in O(r_i^d), i \to \infty. \tag{27}$$

Further, suppose the initial heuristic for search problem $\mathbf{S}_i$ is such that the corresponding Δ_h grows linearly with r_i. Finally suppose that the local isotropicity expands far enough from s_Δ: $r_i \geq \Delta_h/2$. Together these two conditions are:

$$\Delta_h/2 \leq r_i \leq \xi\Delta_h \tag{28}$$

where $\xi \geq 1/2$ is a fixed constant.

Corollary 2 (Systematic Scrubbing). Given the assumptions in equations 24-28, any real-time heuristic search algorithm that fits the basic framework formulated earlier in this paper (Assumptions 1-5) will necessarily scrub systematically.

Proof. As the state space radius r_i increases, Δ_h for the problem $\mathbf{S}_i$ will increase linearly with it due to (28). The minimum amount of travel $T_{\min}$ asymptotically grows as a polynomial of degree $d+1$:

$$T_{\min}(\mathbf{S}_i) \in \Omega\left(\Delta_h^{d+1}\right), i \to \infty \tag{29}$$

$$T_{\min}(\mathbf{S}_i) \in \Omega\left(r_i^{d+1}\right), i \to \infty \tag{30}$$

due to (22), (25) and the linear relation between Δ_h for $\mathbf{S}_i$ and its radius r_i (28).

At the same time, the number of states in the state space will asymptotically grow as a polynomial of degree d of the radius:

$$|S_i| \in \Theta(r_i^d), i \to \infty \tag{31}$$

due to (26) and (27). Combining (30) and (31) we get that the necessary amount of travel assymptotically dominates the state space size:

$$T_{\min}(\mathbf{S}_i) \in \omega(|S_i|), i \to \infty \tag{32}$$

and the agent will scrub systematically. □

4.10 Discussion

To informally summarize the results thus far, we have shown that we can systematically measure the heuristic learning that must be performed in a single state by looking at the maximum heuristic difference encountered between that state and the goal. Under our assumptions the total learning required around this state asymptotically dominates the number of states in the state space. Since the learning per step is constant-bounded, agents will be forced to scrub.

This analysis, in its most general form (Section 4.9), is based on several important assumptions. First, we assume a conservative tie-breaking rule that prefers to re-visit states over exploring new ones. Second, we assume that the state space is locally isotropic around s_Δ. Third, we assume that, in general problems, the heuristic error is growing proportional to the radius of the state space. Finally, we assume that the agent has 1-step lookahead, that edges have unit cost, and that heuristic values are integer valued.

We now consider these assumptions in more detail. First, in our experimental results we will explore several tie-breaking rules, as well as aggressive tie-breaking that prefers to move to larger g-costs over small g-costs. These results will show that, while a better tie-breaking rule can improve best-case performance, it can also have a large impact on worst-case performance.

Second, our definition of systematic scrubbing holds for any problem for which the total learning (20) grows asymptotically faster than the state space (23). We only proved that this must occur in polynomial state spaces that are locally isotropic around s_Δ; it would be interesting future work to develop a broader range of models that have this property. Experimental results are provided giving evidence that this does occur in practice.

Third, if the heuristic error (Δ_h) in a series of growing state spaces is constant, then no scrubbing behavior will be guaranteed by our proofs. This requires a highly accurate heuristic and cannot be assumed in general. It is an interesting open question how the heuristic error grows on different classes of problems, whether it is linear, logarithmic, or some other function of the size or radius of the state space. Such analysis is clearly domain and problem dependent, but even if the heuristic consistently underestimates by just 10% in locations that are locally isotropic, and is perfect in others, our analysis holds (Corollary 2).

Finally, we will address larger lookahead and non-unit edge and heuristic costs in Section 5.

4.11 Special Case: Exponential State Spaces

It is not common for real-time agents to traverse exponential state spaces, so we place our analysis of exponential state spaces in Appendix A. The techniques we used to show systematic scrubbing in polynomial spaces that are locally isotropic are insufficient to do so in exponential spaces that are locally isotropic. However, we also do not have a proof that systematic scrubbing will be absent. Thus, more analysis is needed to make a conclusive statement on scrubbing in exponential state spaces.

5. Generalizing the Theory

So far, we have proven that systematic scrubbing (Corollary 2) holds for a basic agent design and a simple state space. In this section we investigate the extent to which the result is generalizable. We independently relax Assumptions 1 & 2 (unit edge costs and integer heuristic values) and Assumption 3 (one-step lookahead) showing, with counter-examples, that Corollary 2 does not directly generalize. Some of these counter-examples contrast with experimental results which suggest that scrubbing does occur in practice. It is an open problem to succinctly describe conditions for asymptotic scrubbing in these more complicated scenarios.

Relaxing the single trial assumption (Assumption 5), however, shows that scrubbing is expected when learning until convergence.

5.1 Non-unit Edge Costs and Non-Integer Initial Heuristics

In this section we address whether Corollary 2 generalizes to search problems with non-unit edge costs and/or heuristic values. In particular, we relax the assumption of unit edge costs (Assumption 1), replacing it with an assumption of constant-bounded edge costs. We also relax the assumption of integer heuristic values (Assumption 2).

We provide a set of counter-examples to systematic scrubbing: specifically constructed search problems where an agent running LRTA* can move from a region of low heuristic values to higher heuristic values without maintaining a non-increasing heuristic slope, which was the key assumption to our previous proofs.

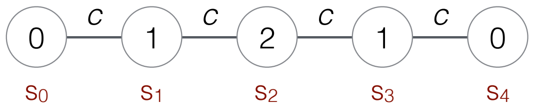


Figure 9: Example chain of states for $n = 2$.

Consider a chain of $2n + 1$ states, illustrated in Figure 9 for $n = 2$, $\{s_0, s_1, s_2, \ldots, s_n, \ldots, s_{2n}\}$ with s_0 the start state and s_{2n} the goal. Suppose the initial heuristic values are monotonically increasing along the chain until state s_n ($h(s_1) < h(s_2) < \ldots h(s_n)$), and monotonically decreasing afterwards. Let $h(i) = i$ for $i \leq n$ and $h(i) = 2n - i$ for $i > n$. This set of growing state spaces is well defined for all $n > 0$ and, if $c = 1$, meets the conditions of Corollary 2. Thus, the example will cause our previously described agent to scrub.

However, if we allow the heuristic values to be non-integer or the edge costs to be non-unit then scenarios exists in which the agent will climb the heuristic grade without scrubbing. For instance, if we change the edge costs to be $c = 1.5$ instead of 1.0, we get the behavior shown Figure 10. Here the agent starts at state 0; the heuristic of this state is updated to be 2.5 which is the edge cost (1.5) plus the h-cost of the neighbor (1.0). The agent then moves to state 1. Notice here that the heuristic to the left (2.5 in state 0) is slightly larger than the heuristic to the right (2.0 in state 2). After learning, the heuristic in state 1 is updated to 3.5, and the agent continues to state 2. In the last step shown here, the agent will begin to follow the gradient to the goal in state 4. Thus, with these edge costs the agent has no choice but to move between the start and the goal without scrubbing.

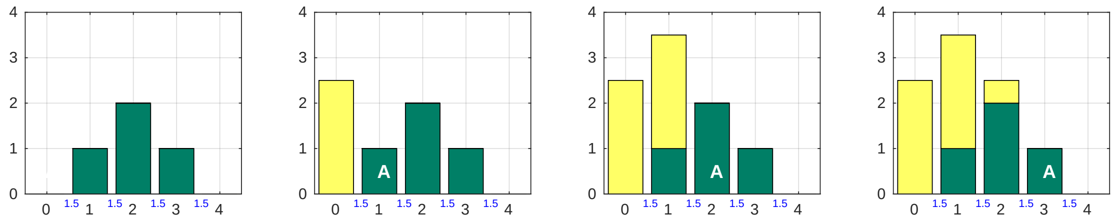


Figure 10: The agent climbs a heuristic grade without scrubbing.

The intuition can be formalized for a chain of $2n$ states as follows:

Lemma 3 Consider a chain of states $\{s_0, s_1, s_2, \dots, s_{2n}\}$ with s_0 being the starting state and s_{2n} being the goal. Suppose the initial heuristic values are monotonically increasing along the chain until some intermediate state $s_n : h(s_1) < h(s_2) < \dots h(s_n)$ forming a heuristic grade of n states. Assume that after s_n the heuristic is monotonically decreasing. Suppose the heuristic grade is of a constant slope $\forall i \in \{0, \dots, 2n-1\} [|h(s_i) - h(s_{i+1})| = \alpha]$ and all edge costs are uniform $\forall i \in \{0, \dots, 2n-1\} [c(s_i, s_{i+1}) = c]$. If the slope α is strictly below the edge cost c then the agent with a lookahead of 1 will climb the grade to state n without any scrubbing, regardless of the tie-breaking schema, and reach the goal in $\Theta(n)$ steps.

Proof. The proof is by induction on the state number k. We will show that whenever the agent is in the state $0 \le k \le n-1$ it will set the heuristic of s_k to be greater than the heuristic of s_{k+2} (i.e., $h_{\text{new}}(s_k) > h(s_{k+2})$) and move to state s_{k+1}. We call this the slope-edge property: the slope growing slower than the edges causes the agent to climb the slope without backtracking.

Base case: $k = 0$. The agent will increase $h(s_0)$ to $h(s_1) + c$ and move to s_1 as its only choice. Starting with $c > \alpha$ we derive:

$$c > \alpha \tag{33}$$
$$h_{\text{new}}(s_0) - h(s_1) > h(s_2) - h(s_1) \tag{34}$$
$$h_{\text{new}}(s_0) > h(s_2). \tag{35}$$

which proves the slope-edge property for $k = 0$.

Inductive step. Suppose the slope-edge property holds for $k = j, j \le n-3$. Then it follows that the agent is now in the state s_{j+1} and $h_{\text{new}}(s_j) > h(s_{j+2})$. From this we immediately conclude that the agent will make its next move to s_{j+2}. Before the move it will set $h_{\text{new}}(s_{j+1})$ to $\min\{h_{\text{new}}(s_j) + c, h(s_{j+2}) + c\} = h(s_{j+2}) + c$. Since $h(s_{j+3}) = h(s_{j+2}) + \alpha$ and $c > \alpha$, we conclude that $h_{\text{new}}(s_{j+1}) > h(s_{j+3})$ which makes the slope-edge property hold for $k = j + 1$.

Once the agent reaches state n, the heuristic is monotonically decreasing towards the goal, and the agent will follow the slope downward until reaching the goal. □

The lemma hinges on the interplay between α and c. The interplay can be satisfied by either non-unit edge costs or non-integer valued initial heuristic. Our example above uses $c = 1.5$ and $\alpha = 1.0$, but the lemma holds for $c = 1.0$ and $\alpha = 0.5$, as well as many other values of c and α.

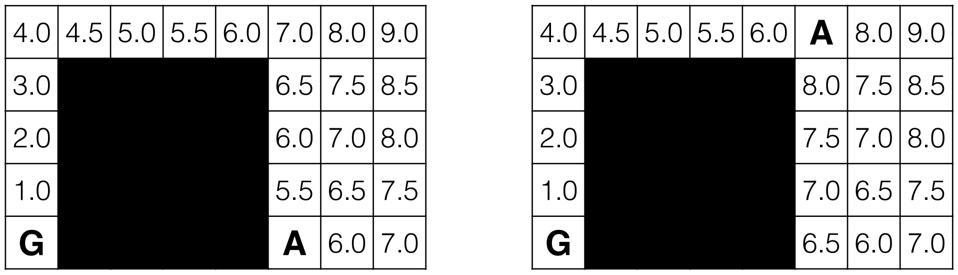


Figure 11: The agent climbs a heuristic grade without scrubbing (intermediate steps omitted).

This situation can happen in practice, as illustrated in Figure 11. The state with the agent is marked **A** and the state with the goal is marked **G**. All other states are labeled with their initial heuristic value. Diagonal edges have cost 1.5 and the heuristic is octile distance, which is the shortest path with no obstacles and only diagonal and cardinal movements allowed. A real-time heuristic search agent with a lookahead of 1 will walk up along the wall because the heuristic increases by only $\alpha = 0.5$ per step while the edge costs along the agent's path are 1. As a result, the agent will walk directly out of the local heuristic minimum and continue to the goal, traversing a shortest path with no scrubbing.

The example, however, is somewhat fragile. If we extend the map downward, the agent will no longer necessarily walk directly to the goal. We illustrate this in Figure 12. In this case the agent begins walking upwards as before, but when it reaches the diagonal line shown in the right side of the figure, the heuristic no longer changes by 0.5 per step. Because the slope-edge property is no longer maintained, the agent can no longer move against the heuristic slope. At this point the agent will have four choices of where to move next (shown as four gray cells in the figure) and, with unfavorable tie breaking will backtrack. Running this example shows that, in this particular case, typical scrubbing behavior then follows.

5.2 Larger Lookahead

In this section we show that equipping the agent with a larger lookahead, breaking Assumption 3, in the style of LSS-LRTA* (Koenig & Sun, 2009) can eliminate systematic scrubbing even in a search problem with unit edge costs.

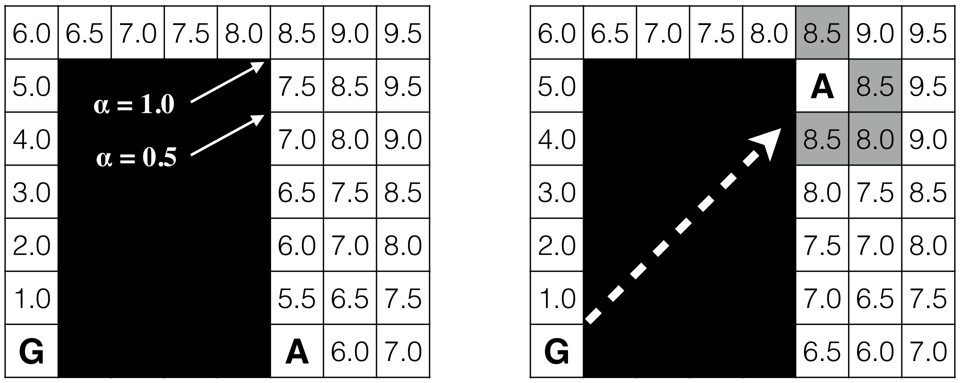


Figure 12: The agent cannot climb the heuristic grade without scrubbing.

First, we redefine our agent algorithm to allow for larger lookahead from Algorithm 1 to Algorithm 2. In the terminology of A* the algorithm works as follows: as long as the goal is not reached (line 3) the agent expands the search space from its current state s_t. The expansion is carried out via an OPEN set which is seeded with the current state and at each moment of time contains all states that have been generated but not expanded. To expand a state means to generate its neighbors. Such neighbors are placed on the OPEN set unless they are already in the CLOSED set (line 9). The CLOSED set contains all states that have been expanded (line 8). States in the OPEN set are expanded in the order of their minimum $f = g + h$ cost, where the g-cost is the distance from the agent's current state s_t (line 7). The number of expansions per step is at most l, an algorithm parameter (line 10). When the expansion process is finished the agent updates the heuristic value for all states on the CLOSED list (line 12). We call these states the local learning space. Then it changes its current state to the most promising state on the OPEN list (line 13). In lines 12 and 13 we use a different cost function, $c_{LS}(s, s')$. This is the cost of the shortest path between s and s' using only the states inside OPEN and CLOSED.

We now provide an example of a set of problems where, using larger lookahead, an agent can walk directly out of a local minima and follow an optimal path to the goal without scrubbing. Similar to the example in Section 5.1, a heuristic barrier will be created behind the agent that drives it forward and precludes backtracking/scrubbing. Our example is in Figure 13. In this example all edge costs are 1. Each state is marked with the current heuristic value, and states are labeled from (b) to (n). The state with the agent is marked with an **A** in the lower left corner. The agent has a lookahead of $l = 5$.

In the first row, the agent will have the gray states in its local learning space. The darker/red states indicate the OPEN list, which is used as the basis for updating heuristic values; the agent will also move to one of these states after learning. No matter how ties are broken (in the f-cost metric used for ordering state expansions), the same five states will be in the local learning space; the gray states must be expanded. But, because state (h) has a lower heuristic than state (b), the agent necessarily moves to state (h). Upon reaching state (h) the space process repeats, except with higher heuristic values. This process is shown in a similar manner as past examples in Figure 14.

Algorithm 2: Real-time Heuristic Search with Lookahead

```
    input  : search problem $(S, E, s_0, s_g, h)$, lookahead $l$
    output: path $(s_0, s_1, \ldots, s_T), s_T = s_g$
 1  $t \leftarrow 0$
 2  $h_t \leftarrow h$
 3  while $s_t \neq s_g$ do
 4      OPEN $\leftarrow \{s_t\}$
 5      CLOSED $\leftarrow \emptyset$
 6      repeat
 7          $n \leftarrow$ best state in OPEN
 8          CLOSED $\leftarrow$ CLOSED $\cup \{n\}$
 9          OPEN $\leftarrow$ OPEN $\cup (N(n) \setminus$ CLOSED$)$
10      until OPEN $= \emptyset \vee |$CLOSED$| = l$
11      for each state $s \in$ CLOSED do
12          $h_{t+1}(s) \leftarrow \min_{s' \in \text{OPEN}} (c_{LS}(s, s') + h_t(s'))$
13      $s_{t+1} \leftarrow \arg\min_{s' \in \text{OPEN}} (c_{LS}(s_t, s') + h_t(s'))$
14      $t \leftarrow t + 1$
15  $T \leftarrow t$
```

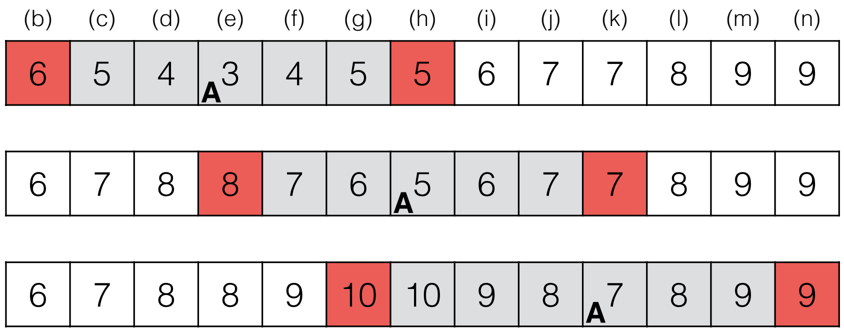

(b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l) (m) (n)
6 5 4 A 3 4 5 5 6 7 7 8 9 9
6 7 8 8 7 6 A 5 6 7 7 8 9 9
6 7 8 8 9 10 10 9 8 A 7 8 9 9

Figure 13: The agent climbs a heuristic grade without scrubbing.

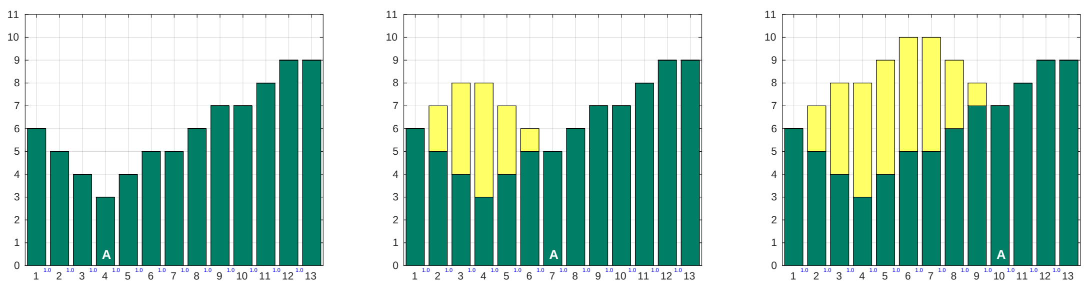


Figure 14: The agent climbs a heuristic grade without scrubbing.

This example can be generalized in two ways: first to any value for the lookahead (l), and second to arbitrary long sequences of states. A given problem instance, however, may not generalize to a different lookahead or start state. We build an example that shows that with lookahead $l = 3$ we can build arbitrarily long sequences of states which a lookahead agent can traverse without scrubbing, reaching the goal in no more than $|S|$ steps.

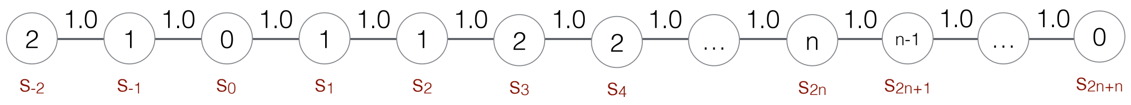


Figure 15: General example for $l = 3$.

Lemma 4 Consider a chain of states $\{s_{-2}, s_{-1}, s_0, \ldots, s_{3n}\}$ with s_0 being the starting state and $s_{3n} = s_{2n+n}$ being the goal. The heuristic of state s_i is $-i$ for $i < 0$. The heuristic of state s_i is $\lceil i/2 \rceil$ for $0 \leq i \leq 2n$. The heuristic of state s_i is $3n - i$ for $i > 2n$ as shown in Figure 15. An agent with a lookahead of 3 will climb the heuristic grade to state $2n$ without any scrubbing, regardless of the tie-breaking schema, and reach the goal in $3n$ steps.

Proof. We prove this by inductively showing an invariant on the heuristic values in the two neighbors on each side of the agent. This invariant will hold until the agent reaches each state up to s_{2n}, after which the agent can walk directly to the goal. The invariant is that the current state has heuristic v, both neighbors of the current state have the same heuristic value $v + 1$, the next neighbor to the right has the value $v + 1$, and the next neighbor to the left has the value $v + 2$. Furthermore, along the path the agent never backtracks to the left, stopping to learn only on the even states $s_0, s_2, s_4, \ldots, s_{2n}$.

Base case: The agent starts at s_0 with heuristic 0. By definition, states s_1 and s_{-1} both have heuristics of 1, state s_2 has a heuristic of 1, and state s_{-2} has a value of 2. After learning, $h(s_{-1}) = 3$, $h(s_0) = 3$ and $h(s_1) = 2$. The agent thus moves to state s_2 with heuristic 1. States s_1, s_3, and s_4 have heuristic 2, and s_0 has heuristic 3. The agent has only moved to the right thus far, and is in an even-numbered state, so invariant holds for the agent's new location.

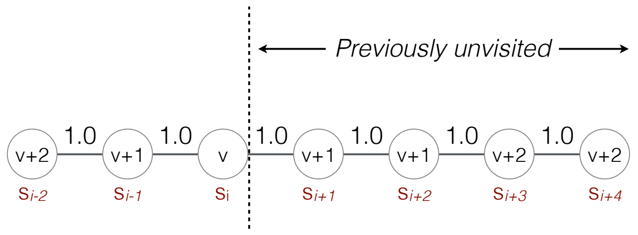


Figure 16: Current heuristic values at the induction step.

Inductive step. The current state of the graph at the beginning of the induction step is shown in Figure 16. We assume the invariant holds for the current agent state s_i with heuristic $v = \lceil i/2 \rceil$ which is $i/2$ since i is even by the invariant. Because states

$s_{i+1}, s_{i+2}, s_{i+3}, s_{i+4}$ have yet to be visited or included in the learning, their heuristic values will be $v+1, v+1, v+2, v+2$ respectively. This follows from the initial heuristic values and the fact that i is even. Furthermore, according to the invariant, states s_{i-1} and s_{i+1} both have heuristics of $v+1$, state s_{i+2} has a heuristic of $v+1$, and state s_{i-2} has a value of $v+2$.

After learning, $h(s_{i-1}) = v+3$, $h(s_i) = v+3$ and $h(s_{i+1}) = v+2$. The agent then must move to s_{i+2} with heuristic $v+1$, which meets the even state condition. $h(s_{i+1})$ was just updated to $v+2$ and $h(s_i)$ to $v+3$. States and s_{i+3} and s_{i+4} have yet to be visited or be part of the local learning space, so they still have their original heuristic values. In particular, since i is even, they will have the heuristic value $\lceil (i+3)/2 \rceil$ and $\lceil (i+4)/2 \rceil$ respectively, which are both equal. Thus $h(s_{i+3}) = v+2$ and $h(s_{i+4}) = v+2$. The invariant still holds, generalizing the example.

The remaining question is what happens after the agent reaches state s_{2n}. At this point the heuristic values to the left of the agent are higher than those to the right where they decrease directly with the edge cost until they reach zero. Thus, the agent will always prefer to move right until it reaches the goal. The agent never backtracks and moves directly from state s_0 to s_{3n}, thus reaching the goal in $3n$ steps. $\square$

This example is for a fixed, odd-valued lookahead. When the structure here is well-understood, it is not difficult to construct similar examples for other odd-valued lookaheads. If the lookahead is even, an additional state with one neighbor is needed next to each state visited by the agent (i.e., a single new state connected to each state s_{2i}). This extra state will never be visited, because it is a dead end, and essentially reduces the lookahead by one, reducing back to the odd-value case for the lookahead.

5.3 Discussion

The examples presented above require particular starting conditions and can be broken by slightly changing their properties such as the starting state, agent lookahead, and the heuristic slope. Thus, while these examples confirm that, under our assumptions, the agents can, under certain circumstances, reach the goal without scrubbing, they do not guarantee that this will happen in practice on a wide range of real-world problem instances (Section 6).

As it may not always be *a priori* clear which parameters an LSS-LRTA* style agent needs to use to avoid scrubbing on a particular problem, we propose several general approaches to combat scrubbing. They work by removing one or more of the assumptions used in our theoretical analysis.

The first approach is to learn faster, trying to violate the bound on constant learning per step. There are several ways to do this. Algorithms such as f-LRTA* (Sturtevant & Bulitko, 2011) and LSS-LRTA* with swamps (Sharon, Sturtevant, & Felner, 2013) can prune states from the state space, effectively raising their heuristics to infinity in a single step. The second approach is to use a better tie-breaking rule. Hernández and Baier (2012) have developed tie-breaking rules that work very well on grid worlds, but on a small fraction of problems result in very poor performance. FALCONS also learns g-costs to do better tie-breaking (Furcy & Koenig, 2000).

The third approach is to decrease Δ_h to decrease the size of the local minima from which the agent must escape. If we use the 0 initial heuristic there will be no local minima, resulting in $\Delta_h = 0$ and rendering the theoretical analysis in this paper inapplicable, but the agent would have no guidance and would wander aimlessly. A better approach is to multiply the heuristic by some constant $0 < \epsilon < 1$, reducing the magnitude of Δ_h. This is in contrast to raising the initial heuristic multiplicatively in an attempt to reduce its heuristic error (Shimbo & Ishida, 2003). The latter method is equivalent to putting a weight on the g-cost which was introduced in real-time heuristic search by Bulitko (2004) with the equivalence proven by Bulitko and Lee (2006). The question of how to perform weighting was recently revisited by Rivera, Baier, and Hernández (2015).

Finally, we could avoid the value-iteration based heuristic search approach altogether, using algorithms such as RIBS (Sturtevant et al., 2010) and EDA* (Sharon, Felner, & Sturtevant, 2014).

5.4 Convergence Travel

The results derived thus far apply just to travel on the first-trial. If one runs Algorithm 1 or Algorithm 2 repeatedly, preserving the learning and using the same start and goal, some of the heuristic values will eventually converge to the true distance to the goal along at least one optimal path from the start to the goal (breaking Assumption 5) (Bulitko & Lee, 2006). We can thus build a more general bound for the learning required for convergence.

To start, we derive an analogous result to Lemma 1, showing that the learning per step is constant-bounded when using larger lookahead.

Lemma 5 Assuming constant-bounded edge costs, constant-bounded branching factor, and constant-bounded lookahead, the total change in heuristic values during each learning step of Algorithm 2 is constant bounded.

Proof. Algorithm 2 updates the heuristic of a state s in the closed list in line 12 of the pseudo code based on the heuristic of a state s' in the open list and the distance between s and s'. The number of states on the closed list is constant bounded by the constant-bounded lookahead, l. The shortest path (c_{LS}) between a state s on the open list and a state s' on the closed list is bounded from above by the maximum edge cost times l times the branching factor. These three values are all constants, so the shortest path is constant bounded. Additionally, the change in heuristic is constant bounded due to consistency. Thus, the change in heuristic for any state on the closed list is also constant bounded. Put together, the total change in heuristic values for all states in CLOSED is also constant bounded. □

Previously, we used a cut set analysis to determine the minimum learning required for some state in the state space. Here, we bypass that analysis and look at the optimal heuristic that must be learned on at least one optimal path between the start to the goal. Let $\mathbf{P}^*(s_0, s_g)$ be the set of all optimal paths between the start and the goal, and let P_i be the ith such path. We define P^*_Δ to be the optimal path that requires the minimum learning - that is, it has the smallest difference between the initial and final heuristic after convergence. Then, we can define alternate versions of Δ_h and s_Δ for convergence.

$$P^*_\Delta = \arg \min_{P_i \in \mathbf{P}^*(s_0,s_G)} (\max_{s \in P_i} [h^*(s,s_g) - h_0(s)]) \quad (36)$$

$$\Delta^*_h = \max_{s \in P^*_\Delta} [h^*(s,s_g) - h_0(s)] \quad (37)$$

$$s^*_\Delta = \arg \max_{s \in P^*_\Delta} [h^*(s,s_g) - h_0(s)] \, . \quad (38)$$

In this definition, Δ^*_h depends only on the error between the perfect heuristic and the initial heuristic and the fact that the algorithm converges to the perfect heuristic on one path. It does not depend on tie-breaking, lookahead, or the edge costs in the environment. In order to make the connection between $L^*_{\min}$ and $T^*_{\min}$ below, we do, however, assume that edge costs are constant bounded to ensure that the learning per step is also constant bounded. Thus, we can provide a convergence form of Equation 12:

$$L^*_{\min}(\mathbf{S}) \geq \sum_{n \in S} \max\{0, \Delta^*_h - 2h^*(s^*_\Delta, n)\}. \quad (39)$$

We can also show that the convergence travel is bounded from below by the convergence learning:

$$T^*_{\min}(\mathbf{S}) \in \Omega\left(L^*_{\min}(\mathbf{S})\right). \quad (40)$$

Thus, on a series of polynomial state spaces where Δ^*_h grows with the map radius, systematic scrubbing will necessarily occur for convergence travel.

Corollary 3 (Systematic Scrubbing for Convergence). Consider a series of search problems $\{\mathbf{S}_1, \mathbf{S}_2, \dots\}$ with locally isotropic polynomial state spaces $\{S_1, S_2, \dots\}$ of the dimension d (i.e., the number of states at radius r from a given state is γr^{d-1}). Each state space S_i has radius $r_i = \max_{a,b \in S_i} h^*(a,b)$. Suppose the initial heuristic for search problem $\mathbf{S}_i$ has heuristic error $\Delta^*_h \geq \mu r_i$ where μ is a positive constant. Also, suppose that the heuristic state space S_i extends for at least $\Delta^*_h/2$ around the state s^*_Δ. Then *any* real-time heuristic search algorithm that runs to convergence will necessarily scrub systematically.

Proof. As the state space radius r_i increases, the heuristic error Δ^*_h will increase at least as μr_i. The minimum amount of travel $T^*_{\min}$ will asymptotically grow at least as r_i^{d+1} (Equation 22 holds trivially for Δ^*_h as well as Δ_h given that they both grow as μr_i). At the same time, the number of states in the state space ($|S_i|$) will asymptotically grow no faster than r_i^d (follows from (23)). This means that $T^*_{\min}(\mathbf{S}_i) \in \omega(|S_i|)$ and thus the agent will scrub systematically. $\square$

Note that we have not analyzed here any influence of number of trials performed by the agent. We leave this analysis for future work.

6. Experimental Results

The goal of this paper is to investigate conditions for scrubbing to occur. We showed, under certain assumptions, that, given the heuristic learning Δ_h required in state s_Δ, the agent must perform asymptotically at least Δ_h^{d+1} moves ($T_{\min}$) to solve a problem in polynomial state spaces of dimension d. If Δ_h is linear in the radius r of the state space where $|S| \in \Theta(r^d)$ then $T_{\min}(\mathbf{S}) \in \omega(|S|)$ and systematic scrubbing will occur.

In the experimental section of this paper we focus primarily on elements of the agent construction, such as lookahead and tie-breaking rules, but also relax the assumption of unit edge costs. We do not address the questions of whether Δ_h is linear in the radius or whether $|S| \in \Theta(r^d)$ in general.

Our initial proof required an unfavorable tie-breaking rule, so in our first set of experiments we will explore what happens as we vary the tie-breaking rule and scale the size of the state space, showing that scrubbing still occurs in practice. Our initial proof also did not generalize when we relaxed the unit-cost and 1-step lookahead assumptions. So, in our second experiment we relax these assumptions and measure performance on game maps, again showing that scrubbing occurs in practice.

To further validate these results and the practical value of our theoretical claims, we implemented a version of LSS-LRTA* that uses essentially the opposite tie-breaking rule from τ used in our proofs. In particular, it learns g-costs like FALCONS (Furcy & Koenig, 2000) and f-LRTA* (Sturtevant & Bulitko, 2011), and breaks ties by moving towards the state with the maximum g-cost. We call this algorithm gLSS-LRTA*. Our theoretical tie-breaking rule is conservative, choosing to move back towards the start state (towards states with minimum g-cost) when possible. gLSS-LRTA* is aggressive, moving away from the start state when possible.

Evidence both here and elsewhere (Sturtevant, 2012) suggests that the game maps are two dimensional. To conclude, we look at maze maps taken from the moving AI repository, which are estimated to be one-dimensional (Sturtevant, 2012). We repeat our experiments on these maps showing that scrubbing is also occurring.

6.1 Experiments on the Corner Map

Consider the corner map used earlier in the paper (Figure 8). 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. We now consider that the map is 8-connected; diagonal movement costs 1.5. An octile distance heuristic will mislead the agent into traveling to the state labeled s_Δ (Equation (5)) while trying to reach the goal.

Under the tie-breaking schema τ constructed earlier in this paper, the final heuristic value of that state, $h_T(s_\Delta)$, will be raised to at least $\hat{h}(s_\Delta, s_g) = h_0(\hat{s})$, based on the states marked $\hat{s}$ in the figure. On this map $\hat{h}(s_\Delta, s_g) = h_0(\hat{s}) = n$ where n is the number of cells along the side of the map. So under τ it will be the case that $h_T(s_\Delta) \geq n$.

By recording $h_T(s_\Delta) - \hat{h}(s_\Delta, s_g) = h_T(s_\Delta) - n$ we can see how different tie-breaking schema compare to τ. Specifically, a measurement of $h_T(s_\Delta) - n < 0$ indicates that the agent did less learning in the state s_Δ than it would have under τ. A measurement of $h_T(s_\Delta) \geq n$ indicates that at least as much learning was performed as is required by τ.

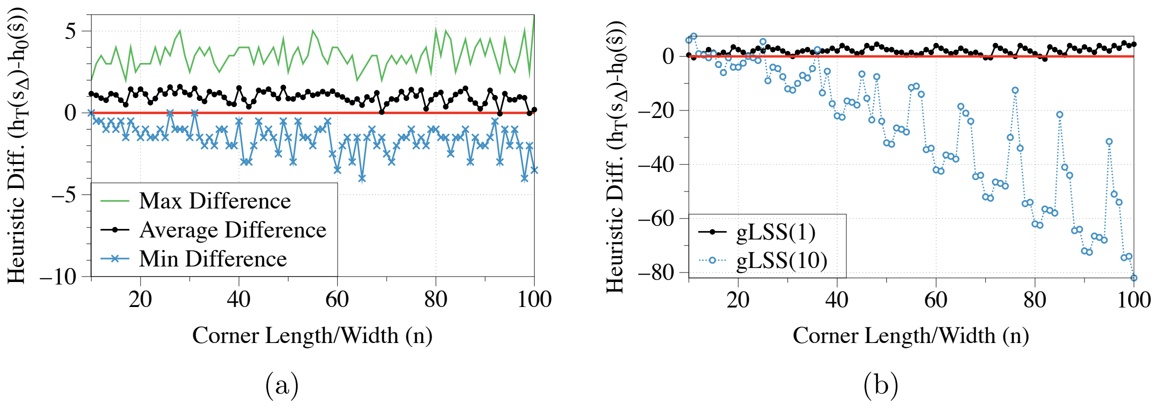


Figure 17: Values of $h_T(s_\Delta) - n$ recorded by running LSS-LRTA* and gLSS-LRTA* on corner maps of different sizes.

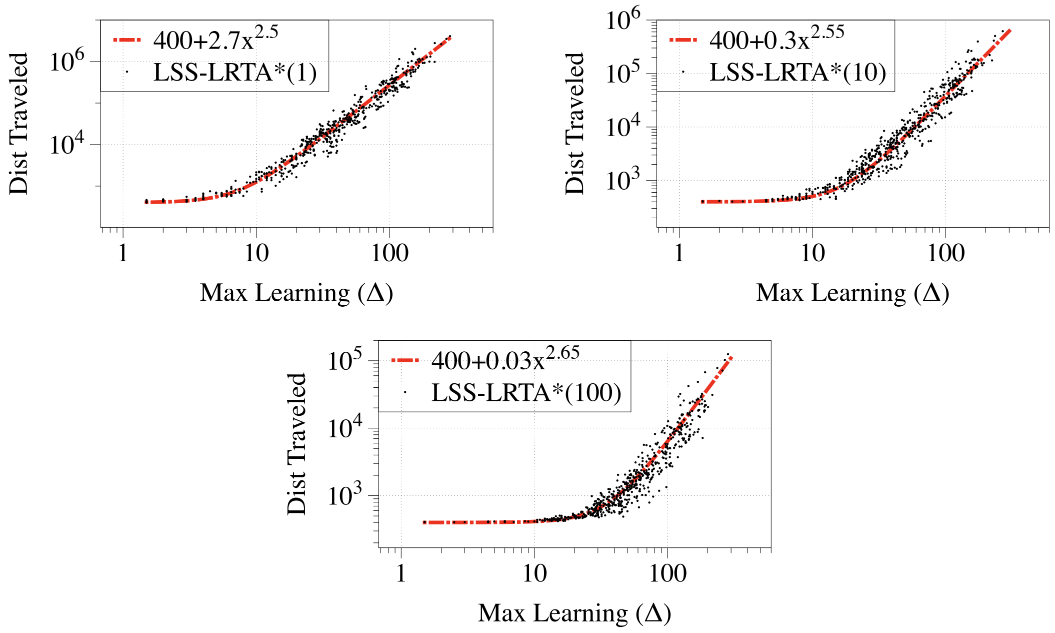


Figure 18: Distance traveled versus maximum learning for any state when solving a problem instance using LSS-LRTA* with lookahead depths (l) of 1 (top left), 10 (top right) and 100 (bottom).

As long as $h_T(s_\Delta) - n$ remains constant as the map radius grows, then scrubbing is occurring. This follows from our analysis in Section 4.8 - subtracting a constant in Equation (14) will not change the asymptotic complexity of the result.

For each value of $n \in \{10, 11, \ldots, 100\}$ (i.e., $n \times n$ map with $\Theta(n^2)$ states), we ran 20 trials of LRTA* with lookahead of 1 and random tie breaking (instead of τ). We also experimented with fixed tie-breaking (chosen deterministically by operator orderings and the internal data structures) and got similar results. The average value, maximum value and minimum value of $h_T(s_\Delta) - n$ on the 20 trials as a function of n are plotted in Figure 17(a). Analyzing the underlying data, we find that LRTA* with randomized tie-breaking performed less than n learning in s_Δ in 18% of the trials. That is, 18% of the data points fall below the zero line. We used linear regression to fit a line to the max and min values over different segments of the data to test if they were growing. Beyond low values of n, the min and max differences do not seem to grow significantly as the problem size increases. Thus, we can conclude that scrubbing is occurring in these experiments.

In Figure 17(b) we look at the performance of gLSS-LRTA* on the same corner map with lookahead 1 and 10. With lookahead 1, gLSS-LRTA* exhibits the same scrubbing behavior as LSS-LRTA* as the radius of the map scales. However, with the lookahead of 10, gLLS-LRTA* is able to escape the corner map without scrubbing. On the largest map tested, where the radius of the local minima is 100, gLSS-LRTA*(10) only increased the heuristic value of s_Δ by 15

We suspect that gLSS-LRTA* does not need to raise to the heuristic value of the corner state as much because on the corner map moving away from the start state is correlated to moving closer to the goal.

In the next section we look at more complex maps from the game *Dragon Age: Origins* to see how LSS-LRTA* and gLSS-LRTA* perform there.

6.2 Pathfinding on Video-Game Maps

In this section we look at pathfinding on video-game maps. Our experiments not only violate assumptions 1-3 (unit edge costs, integer heuristics, and lookahead of one), but the maps are also not guaranteed to be locally isotropic around any state. Despite this, we see that a measure of the heuristic learning correlates with movement that grows asymptotically faster than the state space. Furthermore, experiments with gLSS-LRTA* show that it has worse average-case performance that LSS-LRTA*.

We performed these experiments on the Moving AI benchmark set (Sturtevant, 2012), on all *Dragon Age: Origins* maps with the optimal solution cost in $[400, 404)$. A total of 600 problems were used over 60 maps. We ran these problems with LSS-LRTA* (Koenig & 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 in a fixed way in each state, according to the operator ordering and data structures, without randomization. Note that the optimal solution cost is not predictive of the distance traveled when solving the problem, so this setup gives a wide range of problem difficulties that are constrained to take at least 400 steps to solve.

We measured the maximum learning $\Delta = \max_{s \in R(\mathbf{S})} h_T(s) - h_0(s)$ that occurred in any state as well as the total distance traveled before reaching the goal, and then plotted one point for each problem instance in our test set. If scrubbing is not occurring in practice, then all the values of Δ will be constant-bounded; otherwise we expect a range of values for

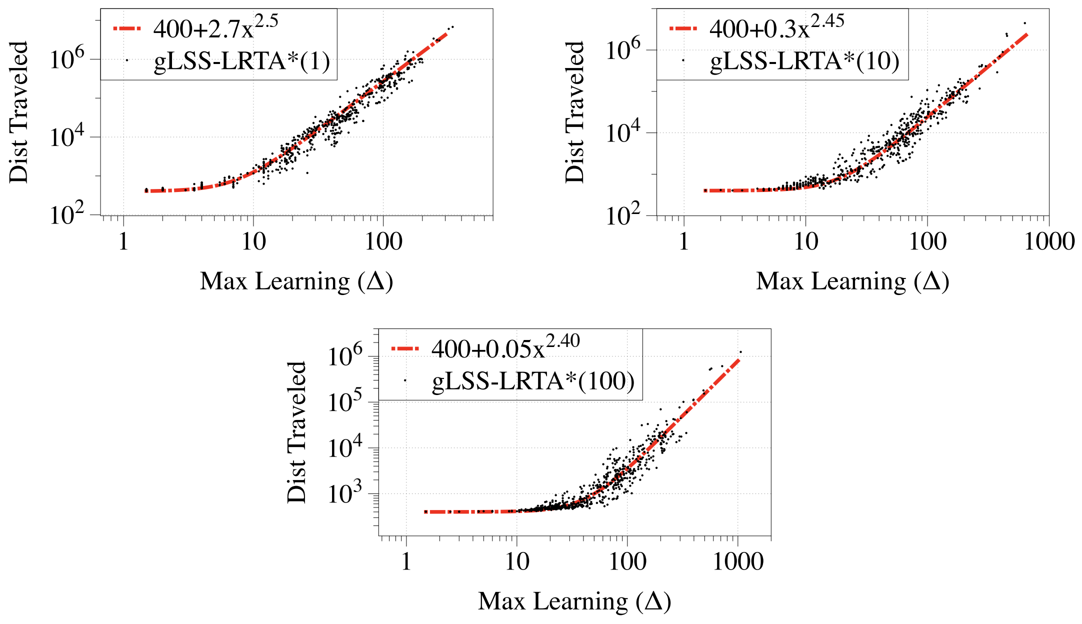


Figure 19: Distance travelled versus maximum learning for any state when solving a problem instance using gLSS-LRTA* with lookahead depths (l) of 1 (top left), 10 (top right) and 100 (bottom).

Δ. If the number of states at a given radius in a map grows polynomially, then the total movement should grow faster than a polynomial with degree two.

The resulting scatter plots are found in Figure 18. We visually fit a polynomial of the form $y = 400 + c_1 \cdot x^{c_2}$ to the data, as we knew that all problems have the optimal solution cost between 400 and 403. (y is the distance traveled and x is Δ.) The values for c_1 and c_2 for each of the three lookahead depths are shown in the figure, although slightly different values do not significantly change the curves or the residuals.

These two-dimensional video-game maps are not isotropic around the state s_Δ and are not exactly polynomial due to their topology, have non-unit-cost edges, have non-unit edge costs, and and may not extend far enough from s_Δ to be locally isotropic (Section 4.9). As a result, our theory does not offer the asymptotic bound of $T_{\min} \in \Omega((\Delta_h)^{d+1}) = \Omega((\Delta_h)^3)$ on the travel to hold. Furthermore, the maximum learning amount Δ we measure is not Δ_h defined earlier in the paper. Yet, the manually fit polynomial curves appear to come close with the degree of the polynomial being between 2.5 and 2.65. Note that the degree of the polynomial is greater than two, the maximum dimensionality of the maps. The data suggests that Δ is predictive of the total movement required to reach the goal and that scrubbing is occurring in practice.

Results for the same experiments with gLSS-LRTA* are found in Figure 19. On these problems the algorithm sometimes raises the heuristic substantially higher than LSS-LRTA* and also travels substantially further.

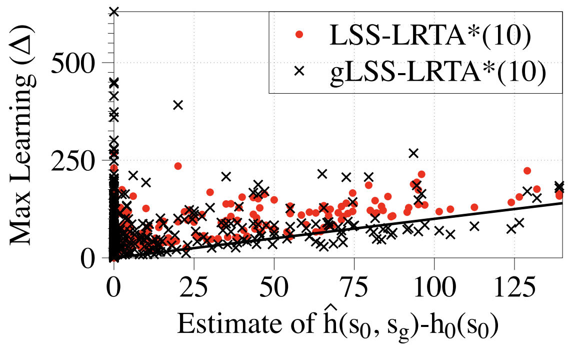


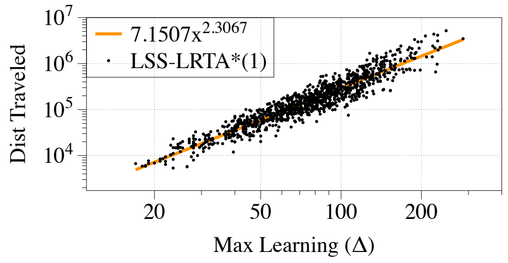
Figure 20: A comparison of the maximum learning in practice (Δ) to an estimate of the learning required between the start and goal states ($\hat{h}(s_0, s_g) - h_0(s_0)$).

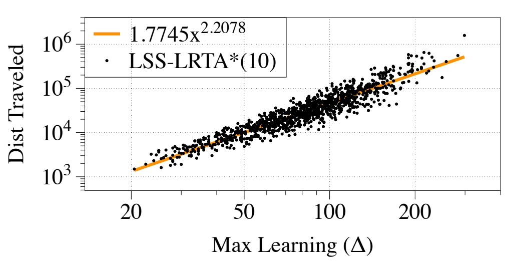
If the different tie-breaking rule achieved better performance, we would expect a smaller range of values on the x-axis (as compared to Figure 18), because there would be fewer states with large amounts of learning. Instead, with lookahead 10 and 100 the range of values is significantly increased, with some states having their heuristic raised by almost 1000. A two-sample Kolmogorov-Smirnov test comparing the LSS-LRTA* and gLSS-LRTA* results on the same problems suggests that the differences are significant with lookahead of 10 or 100, but not with lookahead 1. That is gLSS-LRTA* provides worse performance that just LSS-LRTA*.

Looking deeper into the data with lookahead 10 we find that, although the mean maximum learning is similar (55.29 for LSS-LRTA*(10) versus 59.01 for gLSS-LRTA*(10)), there is a higher standard deviation for gLSS-LRTA*(10) (68.35 versus 45.79 for LSS-LRTA*). The data shows that there are more instances where tie-breaking towards states with higher g-cost can help the agent reach the goal faster (324 instances better versus 240 worse). On the other hand, when moving towards higher g-costs results in worse performance (distance and learning), it outweighs the gains on the other problems, leading to worse average performance. With lookahead 100 the difference in instance counts with better and worse performance is almost equal, but again, the loss of performance from this tie-breaking rule outweighs the gains. The data suggests that moving away from the start state quickly can, in the best case, improve performance, but in the worst case it can have significantly detrimental effects. These results suggest that the corner map is not representative of the video game maps.

We now compare the learning required between the start and the goal ($\hat{h}(s_0, s_g) - h_0(s_0)$), a lower-bound on Δ_h, to Δ, the actual maximum learning performed by the agent. We estimate $\hat{h}(s_0, s_g) - h_0(s_0)$ by measuring over a single shortest path instead of over all shortest paths. If our lower-bound on Δ_h holds for LSS-LRTA*, then all points in Figure 20 would be above the straight line $x = y$ in the figure.

On our 600 pathfinding problems this was indeed the case for LSS-LRTA*(10) on all but one problem. On that one problem, the algorithm had the maximum amount of learning





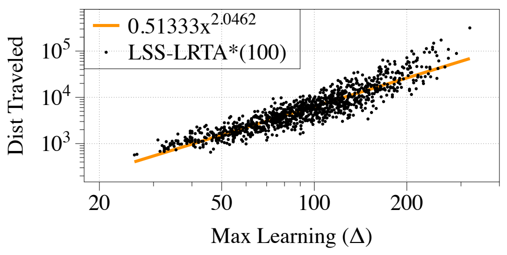


Figure 21: Distance travelled versus maximum learning for any state when solving a problem instances on maze maps with corridor size 8 using LSS-LRTA* with lookahead depths (l) of 1 (top left), 10 (top right) and 100 (bottom).

(Δ) of 54.5 yet the maximum difference between initial heuristic values along a particular optimal solution was 55. Repeating the experiments with gLSS-LRTA*(10), the number of such problems rose from 1 to 36 as illustrated by more markers below the line in the Figure. Simultaneously, gLSS-LRTA*(10) raised heuristic values of some states substantially higher than LSS-LRTA*(10).

6.3 Maze Maps

In the previous section we looked at maps from the game *Dragon Age: Origins*. Analysis of these maps using regression over the number of states at each level in a breadth-first search suggests that they are nearly two-dimensional (Sturtevant, 2012). Specifically, polynomial regression for the equation $a + b \cdot x + c \cdot x^2$ gave an average value of 0.31 for the constant c. According to this same measure, mazes in the benchmark set appear to be approximately one-dimensional, as regression gives a value of 0.01 to 0.03 for the constant c in this equation (Sturtevant, 2012). Thus, mazes represent a different class of problems on which we can re-run our experiments. As before, our theoretical assumptions on the state spaces do not hold in these experiments.

We duplicated our settings from the experiments in Section 6.2 on 10 maze maps with corridor width 8 from the Moving AI benchmark set. Because we had fewer maps, we increased the number of problems solved by solving all probelsm with optimal solution between 400 and 439, resulting in 1100 total problems solved. The results for LSS-LRTA* are in Figure 21. We computed the best-fit curve using least squares for the data to the equation $c_1 \cdot x^{c_2}$, which is also shown.

The fits have correlation of 0.94, 0.94, and 0.91 respectively for lookaheads 1, 10, and 100. On these maps we see that the degree of the fit polynomial is two or greater. As the maps are one-dimensional and Δ is not constant, these results suggest that scrubbing is occurring in practice.

In the initial trials on the maze problems, gLSS-LRTA* had worst-case travel distance several orders of magnitude higher than LSS-LRTA*. This precluded us from running the experiments in a reasonable amount of time and suggests that gLSS-LRTA* is not a practical algorithm on such maps.

7. Conclusions

The primary contribution of this paper is the development of a non-trivial lower bound on the minimum travel that an LRTA*-like real-time heuristic search agent may have to perform to reach the goal state. While previous work has provided examples of problems where state revisitation (i.e., scrubbing) would occur, we provide general conditions that can be used for analysis. In idealized polynomial state spaces the lower bound grows *asymptotically* faster than the state space. This means that the agent will necessarily scrub – an undesirable behavior in many applications such as real-time pathfinding in video games. These theoretical results are supported experimentally on real-world search problems.

This result may appear discouraging, as it suggests that common real-time heuristic search algorithms may not, on their own, be able to avoid scrubbing. While the proofs rely on a several restrictive assumptions, we expect that our results hold more broadly.

From the proof we suggest four directions for future work trying to improve asymptotic performance. This include (1) increasing the amount of learning performed in each step, (2) using different tie breaking rules, (3) decreasing the size of the heuristic local minima and (4) developing algorithms that do not use value-iteration as the core technique driving agent behavior. We hope that future researchers will be able to point to these four directions to explain why their approaches improve performance, and they will be able to identify underlying assumptions about the state space that makes their approaches successful.

While we have shown that our lower bound does not hold when some of our assumptions are broken, we continue to look for properties that we could use to extend the lower bounds to a larger class of problems and agents.

Acknowledgements

The second author received support for this work from the National Research and Engineering Council (NSERC).

Appendix A. Special Case: Locally Isotropic Exponential State Spaces

In this section we consider the case of exponential state spaces. We limit our analysis to *locally isotropic exponential state states spaces* where the number of states exactly cost r away (up to $\Delta_h/2$) from a given state s_Δ is:

$$\eta(r) = \lambda b^r, r \in [0, \Delta_h/2] \tag{41}$$

where b is the branching factor of the space. As in Corollary 2, we will assume that the state space size does not expand substantially beyond $\Delta_h/2$ so the total state space size is asymptotically the same as the size of the state space within the radius $\Delta_h/2$ from the state s_Δ (assumption $\star$).

In such a case we can repeat the derivation in Section 4.9 by substituting $\eta(r) = \lambda b^r$ in (18):

$$\begin{aligned}
\int_0^{\frac{\Delta_h}{2}} \eta(r)(\Delta_h - 2r)dr &= \\
\int_0^{\frac{\Delta_h}{2}} \lambda b^r(\Delta_h - 2r)dr &= \\
\lambda\Delta_h \int_0^{\frac{\Delta_h}{2}} b^r dr - 2\lambda \int_0^{\frac{\Delta_h}{2}} b^r r dr &= \\
\lambda\Delta_h \left. \frac{b^r}{\ln b} \right|_{r=0}^{r=\frac{\Delta_h}{2}} - 2\lambda \int_0^{\frac{\Delta_h}{2}} b^r r dr. & \qquad (42)
\end{aligned}$$

To take the integral $\int_0^{\frac{\Delta_h}{2}} b^r r dr$ in (42) we introduce the function $g(r) = \frac{b^r}{\ln b}$ so that $g'(r) = b^r$ and integrate by parts:

$$\begin{aligned}
\int b^r r dr = \int g'(r) r dr &= \\
g(r)r - \int g(r) r' dr &= \\
g(r)r - \int g(r)dr = r\frac{b^r}{\ln b} - \int \frac{b^r}{\ln b} dr &= \\
r\frac{b^r}{\ln b} - \frac{b^r}{\ln^2 b} + C &= \frac{b^r}{\ln b}\left(r - \frac{1}{\ln b}\right) + C. \qquad (43)
\end{aligned}$$

With (43) equation (42) becomes:

$$\begin{aligned}
\lambda\Delta_h \left. \frac{b^r}{\ln b} \right|_{r=0}^{r=\frac{\Delta_h}{2}} - 2\lambda \int_0^{\frac{\Delta_h}{2}} b^r r dr &= \\
\lambda\Delta_h \left. \frac{b^r}{\ln b} \right|_{r=0}^{r=\frac{\Delta_h}{2}} - 2\lambda \left. \frac{b^r}{\ln b}\left(r - \frac{1}{\ln b}\right) \right|_{r=0}^{r=\frac{\Delta_h}{2}} &= \\
\lambda \left. \frac{b^r}{\ln b}\left(\Delta_h - 2r + \frac{2}{\ln b}\right) \right|_{r=0}^{r=\frac{\Delta_h}{2}} &= \\
\frac{2\lambda}{\ln^2 b} b^{\frac{\Delta_h}{2}} - \frac{\lambda}{\ln b}\left(\Delta_h + \frac{2}{\ln b}\right) &\in \Theta\left(b^{\frac{\Delta_h}{2}}\right), \Delta_h \to \infty \qquad (44)
\end{aligned}$$

Combining (12), (18) and (44) we conclude that for locally isotropic exponential spaces of branching factor b that extend for distance at least $\Delta_h/2$ around the state s_Δ, the minimum amount of total learning is lower-bounded as:

$$L_{\min}(\mathbf{S}) \in \Omega\left(b^{\frac{\Delta_h}{2}}\right), \Delta_h \to \infty. \qquad (45)$$

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

$$T_{\min}(\mathbf{S}) \in \Omega\left(b^{\frac{\Delta_h}{2}}\right), \Delta_h \to \infty. \tag{46}$$

In locally isotropic exponential spaces the number of states within radius $\Delta_h/2$ of the state s_Δ grows as:

$$\int_0^{\frac{\Delta_h}{2}} \eta(r)dr = \int_0^{\frac{\Delta_h}{2}} \lambda b^r dr \in \Theta\left(b^{\frac{\Delta_h}{2}}\right), \Delta_h \to \infty \tag{47}$$

which is asymptotically the same as the total state space according to the assumption ($\star$) and asymptotically the same as the lower bound on the minimum amount of travel (46). This means that in locally isotropic exponential spaces our lower bound $\Omega\left(b^{\frac{\Delta_h}{2}}\right)$ is not sufficient to show systematic scrubbing (i.e., to prove an equivalent of Corollary 2).

Note that (46) is a lower asymptotic bound on the amount of travel whereas (47) is both an upper and lower asymptotic bound on the state space growth. Thus, it may be possible that $T_{\min}$ grows asymptotically faster than the state space size but our lower bound derived above is insufficient to claim so.

References

Bulitko, V. (2004). Learning for adaptive real-time search. Tech. rep. http://arxiv.org/abs/cs.AI/0407016, Computer Science Research Repository (CoRR).

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

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

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

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

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

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

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

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

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

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

Koenig, S., & 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., & 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., & Smirnov, Y. (2003). Performance bounds for planning in unknown terrain. *Artificial Intelligence*, *147*, 253–279.

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

Rivera, N., Baier, J. A., & Hernández, C. (2015). Incorporating weights into real-time heuristic search. *Artificial Intelligence*, *225*, 1–23.

Sharon, G., Felner, A., & Sturtevant, N. (2014). Exponential deepening A* for real-time agent-centered search. In *AAAI Conference on Artificial Intelligence*, pp. 871–877.

Sharon, G., Sturtevant, N. R., & Felner, A. (2013). Online detection of dead states in real-time agent-centered search. In Helmert, M., & Röger, G. (Eds.), *Proceedings of the Sixth Annual Symposium on Combinatorial Search*. AAAI Press.

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

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

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

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

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

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

Sturtevant, N. R., & 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)*, pp. 365–370. AAAI Press.

Sturtevant, N. R., & Bulitko, V. (2014). Reaching the goal in real-time heuristic search: Scrubbing behavior is unavoidable. In *Proceedings of the Symposium on Combinatorial Search (SoCS)*, pp. 166–174.

Zamani, R., & Shue, L.-Y. (2001). A heuristic learning algorithm and its application to project scheduling problems. Tech. rep., Department of Information Systems, University of Wollongong.