Learning Where You Are Going and from Whence You Came: h- and g-Cost Learning in Real-Time Heuristic Search

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

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

Abstract

Real-time agent-centric algorithms have been used for learning and solving problems since the introduction of the LRTA* algorithm in 1990. In this time period, numerous variants have been produced, however, they have generally followed the same approach in varying parameters to learn a heuristic which estimates the remaining cost to arrive at a goal state. Recently, a different approach, RIBS, was suggested which, instead of learning costs to the goal, learns costs from the start state. RIBS can solve some problems faster, but in other problems has poor performance. We present a new algorithm, f-cost Learning Real-Time A* (f-LRTA*), which combines both approaches, simultaneously learning distances from the start and heuristics to the goal. An empirical evaluation demonstrates that f-LRTA* outperforms both RIBS and LRTA*-style approaches in a range of scenarios.

1 Introduction

In this paper we study the problem of *agent-centered real-time heuristic search* [Koenig, 2001]. The first distinctive property of such search is that an agent must repeatedly plan and execute actions within a constant time interval that is independent of the number of states in the problem being solved. The second property is that in each step the planning must be restricted to the state space around the agent's current state. The size of this part must be constant-bounded independently of the total number of states in the problem.

The goal state is not reached by most local searches, so the agent runs the risk of getting stuck in a dead end. To address this problem, real-time heuristic search algorithms update (or learn) their heuristic function (h-costs) with experience. The classic algorithm in this field is LRTA*, developed in 1990 [Korf, 1990].

While LRTA* is both real-time and agent-centric, its learning process can be slow and result in apparently irrational behavior. The reason is two-fold. First, heuristic costs tend to be inaccurate for states distant from the goal state which is where the agent starts out. Thus, in the learning process, LRTA* updates/learns inaccurate h-costs of its current state from inaccurate h-costs of neighboring states. This makes the learning process slow. Second, because LRTA*'s learning is limited to h-costs, it tends to spend a large amount of time filling in heuristic depressions (i.e., visiting and updating states that are not on an optimal path to the goal state).

These problems were partially addressed by RIBS [Sturtevant *et al.*, 2010], which learns costs from the start state (g-costs), and uses these values to prune the search space. While RIBS is superior to LRTA* in some state spaces, it performs worse than LRTA* in problems where there is a wide diversity of heuristic costs. Additionally, RIBS takes time to find an optimal solution on the first trial, while LRTA* may find a suboptimal solution quickly and then slowly converge.

The drawbacks of LRTA* and RIBS motivate the primary contribution of this paper — the development of f-cost Learning Real-Time A* (f-LRTA*) which learns *both* g- and h-costs. This enables pruning methods which detect and remove states guaranteed not to be on an optimal path. As a result, f-LRTA* can learn faster than both LRTA*-like algorithms and RIBS in a range of scenarios. When an agent knows from whence it came (g-costs), it is able to avoid repeating the mistakes it made in the past by pruning away irrelevant portions of the state space. Learning a better estimate of where it is going (h-costs) allows it to follow more aggressive strategies in seeking the goal.

2 Problem Formulation

We define a heuristic search problem as an undirected graph containing a finite set of states S and weighted edges E, with a state s_{start} designated as the *start state* and a state s_{goal} designated as the *goal state*. At every time step, a search agent has a single *current state*, a vertex in the search graph, and takes an action by traversing an out-edge of the current state. We adopt the standard plan-execute cycle where the agent does not plan while moving and does not move while planning. The agent knows the search graph in its vicinity but not the entire space.

Each edge has a finite positive cost $c(s, s')$ associated with it. The total cost of the edges traversed by an agent from its start state until it arrives at the goal state is called the *solution cost*. The problem specification includes a heuristic function $h(s)$ which provides an *admissible* (non-overestimating) estimate of the cost to travel from the current state, s, to the goal

Algorithm 1 LRTA*($s_{\text{start}}, s_{\text{goal}}, d$)

```
1: $s \leftarrow s_{\text{start}}$
2: while $s \neq s_{\text{goal}}$ do
3:     generate $d$ successor states of $s$, generating a frontier
4:     find a frontier state $s'$ with the lowest $c(s, s') + h(s')$
5:     $h(s) \leftarrow c(s, s') + h(s')$
6:     change $s$ one step towards $s'$
7: end while
```

state. A state is *expanded* when all its neighbors are generated and considered by an algorithm.

When an agent reaches the goal state, a *trial* has completed. The agent is teleported back to the start state, and a new trial begins. An agent is said to *learn* when it adjusts costs of any states. An agent has *converged* when no learning occurs on a trial. Real-time heuristic search literature has considered both first-trial and convergence performance. We follow the practice and employ both performance measures in this paper. To avoid trading off solution optimality for problem-solving time, which is often domain-specific, we only consider algorithms that converge to an *optimal* solution after a finite number of trials.

We require algorithms to be *complete* and produce a path from start to goal in a finite amount of time if such a path exists. To guarantee completeness, real-time search requires *safe explorability* of the search graph, defined as the lack of terminal non-goal states (e.g., agent's plunging to its death off a cliff). The lack of directed edges in our search graph gives us safe explorability as the agent is always able to return from a suboptimally chosen state by reversing its action.

3 Related Work

There are two types of costs commonly used in heuristic search: (i) the remaining cost, or h-cost, defined as an admissible estimate of the minimum cost of any path between the given state and the goal and (ii) the travel cost so far, or g-cost, defined as the minimum cost of all explored paths between the start state and a given state. The f-cost of a state is the current estimate of the total path cost through the state (i.e., the sum of the state's g- and h-costs). The initial h-cost function is part of the problem specification.

3.1 Learning the cost to go (h-costs)

Most work in real-time heuristic search focused on h-cost learning algorithms, starting with the seminal Learning Real-time A* (LRTA*) algorithm [Korf, 1990] shown as Algorithm 1. As long as the goal state is not reached (line 2), the agent follows the plan (lines 3-4), learn (line 5) and execute (line 6) cycle. The planning consists of a lookahead during which d (unique) closest successors of the current state are expanded. During the learning part of the cycle, the agent updates $h(s)$ for its current state s. Finally, the agent moves by going towards the most promising state discovered in the planning stage.

Research in the field of learning real-time heuristic search has resulted in a multitude of algorithms with numerous variations. Most of them can be described by four attributes.

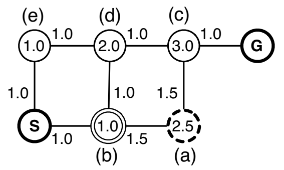

Figure 1: Dead-ends and redundant states.

The *local search space* (LSS) is the set of states whose heuristic values are accessed in the planning stage. The *local learning space* is the set of states whose heuristic values are updated. A *learning rule* is used to update the heuristic values of the states in the learning space. The *control strategy* decides on the actions taken following the planning and learning phases. While no comprehensive comparison of all these variants exists, some strategies were co-implemented and compared within LRTS [Bulitko and Lee, 2006].

In this paper we use LSS-LRTA* [Koenig and Sun, 2009] as a representative h-cost-learning algorithm. LSS-LRTA* expands its LSS in an A* fashion, uses the Dijkstra's algorithm to update h-costs of all states in the local search space and then moves to the frontier of the search space.

The primary drawback of learning h-costs is that the learning process is slow. This appears to be necessarily so, because such algorithms update inaccurate h-costs in their learning space from also inaccurate h-costs on the learning space frontier. The local nature of such updates makes for a small, incremental improvement in the h-costs.

Combined with the fact that the initial h-costs tend to be highly inaccurate for states distant from the goal state, LRTA*-style algorithm can take a long time to escape a heuristic depression. A recent analysis of a "corner depression" shows that in this example LRTA* and its variants must make $\Omega(N^{\frac{3}{2}})$ visits to $\Theta(N)$ states [Sturtevant *et al.*, 2010].

3.2 Learning the cost so far (g-costs)

The first algorithm to learn g-costs was FALCONS [Furcy and Koenig, 2000]. But, FALCONS did not use g-costs for pruning the state space, only as part of the movement rule. RIBS [Sturtevant *et al.*, 2010] is an adaptation of IDA* [Korf, 1985] designed to improve LRTA*'s performance when escaping heuristic depressions. It does so by learning costs from the start state to a given state (i.e., g-costs) instead of the h-costs. When g-costs are learned, they can be used to identify *dead* states and remove them from the search space, increasing the efficiency of search. RIBS classifies two types of dead states, those that are *dead-ends* and those that are *redundant*[1]. States that are not dead are called alive.

A state n is a *dead-end* if, for all of its successor s_i, $g(n) + c(n, s_i) > g(s_i)$. This means that there is no optimal path from start to s_i that passes through n. This is illustrated in Figure 1 where states are labeled with their g-costs. The state marked (a) is dead because (a) is not on a shortest path to either of its successors (b) or (c).

A state n is *redundant* if, for every non-dead successor s_i, there exists a distinct and non-dead parent state $s_p \neq n$ such

[1]We adapt the terminology of RIBS [Sturtevant *et al.*, 2010] by defining a general class of dead states and its two subclasses.

that $g(s_p) + c(s_p, s_i) \leq g(n) + c(n, s_i)$. This means that removing n from the search graph is guaranteed not to change the cost of an optimal path from the start to the goal. This is also illustrated in Figure 1 where exactly one of state (e) or state (b) can be marked redundant on the path to (d).

RIBS can learn quickly in state spaces with many dead-end and/or redundant states. For instance, in the "corner depression" of $\Theta(N)$ states mentioned before, RIBS is able to escape in just $O(N)$ state visits, which is an asymptotic improvement over LRTA*'s $\Omega(N^{\frac{3}{2}})$ [Sturtevant *et al.*, 2010].

On the negative side, RIBS inherits performance characteristics from IDA* and has poor performance when there are many unique f-costs along a path to the goal. For instance, if every state has a unique f-cost, RIBS will take $\Theta(N^2)$ moves to reach the goal state while visiting N states. Additionally, RIBS has a local search space consisting only of the current state and its immediate neighbors. While this does not hurt asymptotic performance, it does introduce a larger constant factor when compared to larger search spaces.

4 f-LRTA*

The primary contribution of this paper is the new algorithm f-LRTA* that learns both g and h costs. f-LRTA* uses the LSS-LRTA* learning rule to learn its h-costs. Like RIBS, f-LRTA* also learns g-costs which it uses to identify dead states thereby attempting to substantially reduce the search space. Unlike RIBS, f-LRTA* has an additional rule for pruning states by comparing the total cost of the best solution found so far and the current f-cost of a state. Also unlike RIBS, f-LRTA* works with an arbitrarily sized LSS.

The pseudo-code for f-LRTA* is listed as Algorithm 2. It takes two main control parameters: d which determines the size of the lookahead and w which controls the movement policy. The algorithm works in three stages detailed below.

4.1 Stage 1: Expanding LSS and Learning g-costs

In the first stage, the local search space (LSS) is expanded (line 3 in the pseudo-code of the function f-LRTA*). As with other similar algorithms, the LSS is represented by two lists: *OPEN* (the frontier) and *CLOSED* (the expanded interior). The states in the LSS have local g-costs, denoted by g_l and defined relative to the current state of the agent, and global g-costs defined relative to the start state.

The LSS is expanded incrementally with the shape of expansion controlled by sorting of the *OPEN* list. All results presented in this paper sort the *OPEN* list by local $f_l = g_l + h$-costs. Dead states count as part of the LSS, but are otherwise ignored in the initial expansion phase. Expansion and g-cost propagation occurs in the function ExpandAndPropagate.

When a state s is expanded (line 1 in function ExpandAndPropagate), its successors are placed on *OPEN* in the standard A* fashion (line 3). If the global g-cost of s can be propagated to a neighbor, the g-cost of the neighbor is reduced. The reduction is recursively propagated as far as possible through *OPEN* and *CLOSED* (ExpandAndPropagate lines 9-13). If the neighboring state was dead, that neighbor is also expanded (ExpandAndPropagate line 10), which helps guarantee that f-LRTA* will be able to converge to the optimal solution. If a shorter path to s is found through a neighbor (ExpandAndPropagate line 15), all previous neighbors of s are reconsidered in case their g-costs can also be reduced.

When global g-costs are updated for a state, the h-cost of that state is reset to its default heuristic value (ExpandAndPropagate lines 8 and 17). This is needed to ensure that f-LRTA* will be able to converge to the optimal solution.

Algorithm 2 f-LRTA*

```
globals: OPEN/CLOSED
f-LRTA*($s_{\text{start}}, s_{\text{goal}}, d, w$)
 1: $s \leftarrow s_{\text{start}}$
 2: while $s \neq s_{\text{goal}}$ do
 3:     ExpandLSS($s, s_{\text{goal}}, d$)
 4:     GCostLearning()
 5:     LSSLRTA-HCostLearning()
 6:     Mark dead-end and redundant states on CLOSED and OPEN
 7:     if all states on OPEN are marked dead then
 8:         move to neighbor with lowest $g$-cost
 9:     else
10:         $s \leftarrow$ best non-dead state $s_i$ in OPEN
11:         (chosen by $g_l(s_i) + w \cdot h(s_i)$ or $g(s_i) + w \cdot h(s_i)$)
12:     end if
13: end while

ExpandLSS($s_{\text{curr}}, s_{\text{goal}}, d$)
 1: push $s_{\text{curr}}$ onto OPEN (kept sorted by lowest $g_l$ or $g_l$+$h(s_{\text{curr}})$)
 2: for i = 1 ... $d$ do
 3:     $s \leftarrow OPEN.pop()$
 4:     place $s$ on CLOSED
 5:     if $s$ is dead then
 6:         continue for loop at line 2
 7:     end if
 8:     ExpandAndPropagate($s, s_{\text{goal}}, true$)
 9:     if $s$ is goal then
10:         return
11:     end if
12: end for

ExpandAndPropagate($s_{\text{curr}}, s_{\text{goal}}, expand$)
 1: for each successor $s_i$ of $s$ do
 2:     if $expand$ then
 3:         perform regular A* expansion steps on $s_i$
 4:     end if
 5:     if $g(s) + c(s, s_i) < g(s_i)$ then
 6:         mark $s_i$ as $alive$
 7:         $g(s_i) \leftarrow g(s) + c(s, s_i)$
 8:         $h(s_i) \leftarrow$ default heuristic for $s_i$
 9:         if $s_i$ is on CLOSED and was $dead$ then
10:             ExpandAndPropagate($s_i, s_{\text{goal}}, true$)
11:         else if $s_i$ is on OPEN or CLOSED then
12:             ExpandAndPropagate($s_i, s_{\text{goal}}, false$)
13:         end if
14:     end if
15:     if $g(s_i) + c(s_i, s) < g(s)$ and $s_i$ is not $dead$ then
16:         $g(s) \leftarrow g(s_i) + c(s_i, s)$
17:         $h(s) \leftarrow$ default heuristic for $s$
18:         Restart FOR loop with $i \leftarrow 0$ on line 1
19:     end if
20: end for

GCostLearning($s_{\text{goal}}$)
 1: for each state $s$ in OPEN do
 2:     ExpandAndPropagate($s$, $s_{\text{goal}}$, $false$)
 3: end for
```

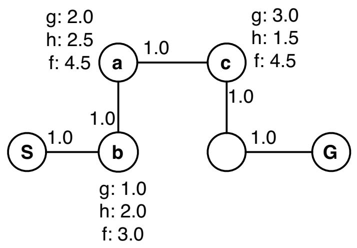

Figure 2: Effects of w on the movement policy.

At this point in the execution, most of the possible g-cost updates have been performed, but updates from states on *OPEN* to other states on *OPEN* have not been performed. These updates can change the g-cost of states on *CLOSED* and are performed inside the GCostLearning procedure.

The parameter d limits the size of the *CLOSED* list, however the g-cost propagation rules can lead to states being re-expanded. All re-expansions take place in the LSS and are bounded by the size of d. Thus, f-LRTA* retains its real-time agent-centered nature.

4.2 Stage 2: Learning h-costs and State Pruning

In the second stage, h-cost learning occurs according to the LSS-LRTA* learning rule (the corresponding pseudo-code is omitted to save space). First, states on the *OPEN* list are sorted by h-cost. Beginning with the lowest h-cost, heuristics values are propagated into the LSS: If a state n on the *OPEN* list is expanded with an immediate neighbor, m, then $h(m) \leftarrow \max(h(m), h(n) + c(m, n))$. As with LSS-LRTA*, after the first such update, subsequent increases to $h(m)$ are not allowed. Dead states on *OPEN* are ignored.

Finally, states within the LSS are examined to see if they can be further pruned. While RIBS rules for marking a state dead were used in Stage 1, we also mark a state dead if its f-cost is greater than the cost of the best solution found on the previous trials.

4.3 Stage 3: Movement

After performing learning, the movement policy moves to a state selected from *OPEN* (line 10 in function f-LRTA*). A common policy is to move to the state with the lowest local f-cost, $f_l(n) = g_l(n) + h(n)$. f-LRTA* can also move to the state with lowest global f-cost, $f(n) = g_l(n) + w \cdot h(n)$. Weighting ($w$) the heuristic also provides alternate movement policies. Low weights result in performance more similar to RIBS, where the focus is on finding accurate g-costs. High weights emphasize the heuristic over the g-cost, and are similar to moving to the state with best f_l.

Unlike weighted A* [Pohl, 1970] and weighted-LRTA* [Shimbo and Ishida, 2003], f-LRTA* will not converge to an inadmissible heuristic for *any* value of $1 \leq w < \infty$. However, the movement policy may not select the optimal path upon convergence if $w \neq 1$. Hence, once convergence with $w \neq 1$ occurs, we reset w to 1 and continue the convergence process.

We demonstrate the difference that the movement policy can have in Figure 2. We assume the agent is at state (a) and the local search space is two states: (b) and (c). With $w = 1$, f-LRTA* will direct the agent backwards to state (b), which has the lowest (global) f-cost. However, with $w = 10$ weighted f-cost of (c) will be $3.0 + 15.0 = 18.0$, while the weighted f-cost of (b) will be $1.0 + 20.0 = 21.0$. Thus, the agent will move to (c).

5 Completeness and Optimality

f-LRTA* is complete and finds an optimal solution upon convergence. For the sake of space we give only a sketch of the proof which also motivates some of the design choices.

Completeness is proved by contradiction. Assume that f-LRTA* is unable to find a solution if one exists. That means that it is either in a state with no moves or it is stuck in an infinite loop. The former is not possible because f-LRTA* makes a move even when all neighbors of a state are dead (line 8 in function f-LRTA*). The latter is impossible because with every iteration of such a loop, the algorithm would keep increasing h-costs of states on the loop by a lower-bounded positive amount. The h-cost of any state has a finite upper bound, so infinite looping is not possible. The h-cost of a state can be reset (lines 8 and 17 in ExpandAndPropagate) whenever the state's g-cost is decreased. However, only a finite number of resets are possible, because every state, besides the start state, has a finite positive g-cost and each reduction is by a lower-bounded positive amount.

Optimality can be proved in the same way as for most LRTA*-style algorithms with the following caveat. f-LRTA* uses the standard dynamic programming update for its h-costs which guarantees admissibility of h. However, dead states are removed from the *OPEN* list when h updates are performed in stage 2. As a result, the learned h-costs are admissible for the search graph with the dead states removed but may be inadmissible with respect to the original graph. This phenomenon motivates two design choices in f-LRTA*. First, when a shorter path to a dead state is discovered, the state is made alive (line 6 in ExpandAndPropagate) and the g-cost is decreased (lines 7 and 16). Second, whenever a g-cost of a state is decreased, the h-cost of that state is reset to the initial heuristic (lines 8 and 17). Thus, it is possible to prove that along at least one optimal paths to the goal, states with optimal g-costs will always have admissible h-costs.

6 Empirical Evaluation

In order to evaluate the strengths of learning both g- and h-costs, we compare f-LRTA* to two contemporary algorithms: LSS-LRTA* and RIBS. We do so in two pathfinding domains using several performance metrics: solution cost (i.e., the total distance traveled), states expanded, and planning time. These are measured on the first trial and over all trials until convergence, along with the number of trials required for convergence. The implementation focus for f-LRTA* has been on correctness, not always speed, although f-LRTA* does have more learning overhead. All experiments were run on a single core of a 2.40GHz Xeon server.

In the following tables and graphs we will list control parameters with the name of the algorithms. The d in LSS-LRTA*(d) is the number of states in the *CLOSED* list of the LSS. The d in f-LRTA*(d, w) has the same function; w is

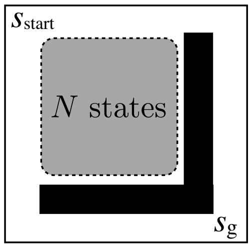

Figure 3: The corner heuristic depression map.

the weight used by the movement policy. f-LRTA$_l$* chooses the best state to move to using local g-costs, while f-LRTA* uses global g-costs. f-LRTA* may expand fewer than d states if many of them are dead, but may perform more than d re-expansions if many g-cost updates are performed. RIBS takes no parameters.

We use two domains; the first, based on octile movement, is a *de facto* standard in the real-time heuristic literature. The second, based on a constrained heading model, is a more practical pathfinding domain for video games and robotics.

The first domain is grid-based pathfinding on a two-dimensional grid with some blocked cells. The agent occupies a single vacant cell and its coordinates form the agent's state. It can change its state by moving in one of the eight directions (four cardinal and four diagonal). The move costs are 1 for cardinal moves and 1.5 for diagonal moves. The initial heuristic is the octile distance, which is a natural extension of the Manhattan distance for diagonal moves. It also happens to be a perfect heuristic in the absence of blocked cells.

The constrained heading domain is also grid-based pathfinding but the states and moves are defined differently. Specifically, the agent's state is the x, y-coordinates of the vacant cell it presently occupies as well as its heading, which is 0...345° by 15° increments. There are twenty actions available to the agent: moving forward or backwards while changing the heading i degrees ($i \in \{-45, -30, -15, 0, 15, 30, 45\}$) (14 actions). The agent can also turn in place without changing location, but may not remain completely stationary (6 actions). The initial heuristic is the Euclidean distance between the coordinates of a state and the goal state, ignoring the current and the goal headings. Movement costs are the heading divided by 15 mod 6 indexed into the array $\{1.0, 3.25, 3.75, 1.50, 3.75, 3.25\}$. Turning in place has cost 1.

We first perform scaling experiments to illustrate asymptotic performance, and then look at data on maps from a recent commercial video game.

6.1 Asymptotic Performance

We begin by duplicating the scaling experiments used to evaluate RIBS [Sturtevant *et al.*, 2010]. These experiments take place on the map in Figure 3, where the number of states in the corner is N. The goal is on the other side of the corner which creates a heuristic depression of N states. Marking dead states helped RIBS escape the depression in $O(N)$ moves whereas LRTA* takes $\Theta(N^{1.5})$ moves.

The results on the corner depression map with octile movement are found in Figure 4. We plot states expanded in the

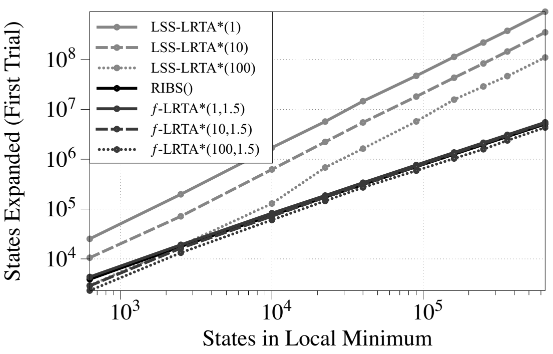

Figure 4: Scaling results with octile movement model.

first trial. When increasing the number of states in the local minimum and measuring the slope of the lines on the log-log plot, we observe that f-LRTA* retains the superior asymptotic performance of RIBS as compared to LSS-LRTA*. RIBS and f-LRTA* show a linear fit with the number of states in the depression with a correlation coefficient of 0.99. LSS-LRTA* fits $N^{1.5}$ with a coefficient of 0.99. While the number of states expanded is roughly the same for RIBS and f-LRTA*, the distance travelled by f-LRTA* on the first trial is $3 - 20$ times less than RIBS, due to larger lookahead. Plots for convergence follow the same trends. In this experiment f-LRTA* performance depends on moving towards the state with best global f-cost using a low heuristic weight. In all other experiments higher weights and/or movement to the state with best (local) f_l-cost shows better performance. This has partially to do with the size of local minima, but there may be other factors at work which have yet to be understood.

We use the same map in the second domain. In the start state, the agent faces to the right, and in the goal state the agent also faces to the right. The results for distance traveled on the first trial is shown in Figure 5. First, we observe that RIBS performance is notably worse than LSS-LRTA* or f-LRTA*. This is due to the fact that there are fewer states that share the same f-cost, meaning that RIBS uses many more iterations to explore the search space, as would IDA*.

The data on this log-log plot fits with a correlation coefficient of 0.99 to the following polynomials. LSS-LRTA* with $d \in \{1, 10, 100\}$ has the convergence cost of $N^{2.13}, N^{2.00}, N^{1.91}$ respectively, where N is the number of states in the heuristic depression. RIBS has the convergence cost of $N^{2.42}$. f-LRTA* with $d \in \{1, 10, 100\}$ and $w = 10.0$ fits $N^{1.62}, N^{1.54}, N^{1.57}$ respectively. The data supports the conjecture that f-LRTA* is asymptotically faster in this domain than both LSS-LRTA* and RIBS.

6.2 Experiments on Game Maps

We use standard benchmark problems for the game Dragon Age: Origins[2] to experiment on more realistic maps.

Under the octile movement model we experimented with the 106 problems which have octile solution length 1020-1024. The average results are in Table 1. The LSS column is the average number of states expanded each time the agent

[2]http://www.movingai.com/benchmarks/

	First Trial			All Trials				
Algorithm	Dist.	States Exp.	Time	Dist.	States Exp.	Time	Trials	LSS Exp.
RIBS	2,861,553	1,219,023	4.46	2,861,553	1,219,023	4.46	1	1.0
LSS-LRTA*(1)	633,377	2,761,163	12.58	23,660,293	98,619,076	266.81	12,376	9.4
LSS-LRTA*(10)	113,604	952,402	2.18	3,844,178	21,970,613	53.30	1951	41.5
LSS-LRTA*(100)	18,976	305,518	0.63	559,853	5,920,709	14.74	295	264.9
f-LRTA$_l$*(1,10.0)	127,418	389,367	2.44	1,848,413	5,321,043	32.14	890	6.7
f-LRTA$_l$*(10,10.0)	52,097	188,144	0.89	663,378	1,808,493	9.18	264	19.8
f-LRTA$_l$*(100,10.0)	10,890	82,414	0.35	136,184	624,262	2.96	69	124.3

Table 1: Average results on Dragon Age: Origins maps with octile movement.

	First Trial			All Trials				
Algorithm	Dist.	States Exp.	Time	Dist.	States Exp.	Time	Trials	LSS Exp.
LSS-LRTA*(1)	22,047	2,845,386	3.60	886,622	98,876,890	121.55	4276	21.4
LSS-LRTA*(10)	4,650	2,005,258	3.58	153,688	43,058,124	73.82	890	129.8
LSS-LRTA*(100)	820	1,155,577	2.93	25,380	22,751,673	55.29	161	879.7
FLRTA$_l$*(1,1.5)	3,779	331,433	1.60	54,488	3,363,426	16.08	88	16.0
FLRTA$_l$*(10,1.5)	1,816	318,176	1.56	21,448	2,522,682	12.27	45	58.4
FLRTA$_l$*(100,1.5)	530	362,386	1.93	5,634	2,223,981	11.58	21	423.4

Table 2: Average results on Dragon Age: Origins maps with constrained movement.

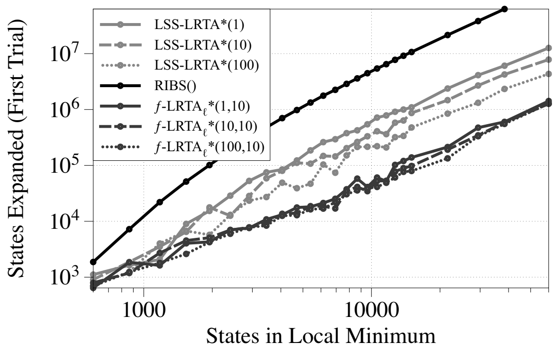

Figure 5: Scaling results with constrained heading model.

plans. f-LRTA* does less work on average because many states in the LSS are dead. On these maps, f-LRTA* traveled less distance and expanded fewer states to solve each problem on both the first trial and all trials. All differences between algorithms with the same d are statistically significant with at least 95% confidence.

In the constrained heading model we ran the 1280 problems with optimal octile solution length 100-104, with the results shown in Table 2. All differences between algorithms with the same d are statistically significant with at least 95% confidence. This movement model is significantly more difficult for LSS-LRTA*, but RIBS is unable to complete the test set in reasonable amounts of time. f-LRTA* can solve the whole problem set faster than RIBS solves just 36 problems. The same trends can be observed in this data as in the octile movement model.

7 Conclusions and Future Work

In this paper we proposed a new algorithm, f-LRTA* which combines traditional h-learning real-time heuristic search with the g-cost learning. As a result, f-LRTA* retains superior asymptotic performance in the corner depression. Additionally, learning h-costs allows f-LRTA* to outperform RIBS in pathfinding on video game maps. Future work will investigate dynamic rules for adapting movement based on the number of state re-expansions performed in order to avoid using different values of w for different problems.

Our hope is that this work will encourage work new directions, not only in real-time agent-centered search, but perhaps in other fields as well. An important question first raised by Furcy and Koenig [2000] is how g-cost learning might apply to reinforcement learning. Ultimately, this work shows that there is value in knowing from whence you came.

References

[Bulitko and Lee, 2006] Vadim Bulitko and Greg Lee. Learning in real time search: A unifying framework. *JAIR*, 25:119–157, 2006.

[Furcy and Koenig, 2000] David Furcy and Sven Koenig. Speeding up the convergence of real-time search. In *AAAI*, pages 891–897, 2000.

[Koenig and Sun, 2009] Sven Koenig and Xiaoxun Sun. Comparing real-time and incremental heuristic search for real-time situated agents. *Autonomous Agents and Multi-Agent Systems*, 18(3):313–341, 2009.

[Koenig, 2001] Sven Koenig. Agent-centered search. *AI Mag.*, 22(4):109–132, 2001.

[Korf, 1985] Richard Korf. Depth-first iterative deepening: An optimal admissible tree search. *AIJ*, 27(3):97–109, 1985.

[Korf, 1990] Richard Korf. Real-time heuristic search. *AIJ*, 42(2-3):189–211, 1990.

[Pohl, 1970] Ira Pohl. Heuristic search viewed as path finding in a graph. *AIJ*, 1:193–204, 1970.

[Shimbo and Ishida, 2003] Masashi Shimbo and Toru Ishida. Controlling the learning process of real-time heuristic search. *AIJ*, 146(1):1–41, 2003.

[Sturtevant *et al.*, 2010] Nathan R. Sturtevant, Vadim Bulitko, and Yngvi Börnsson. On learning in agent-centered search. In *AAMAS*, pages 333 – 340, 2010.