Learning, Commitment, and Completeness in Real-Time Heuristic Search

Devin Wild Thomas and Wheeler Ruml

University of New Hampshire
Durham, NH USA
{dwt, ruml} @ cs.unh.edu

Abstract

In real-time heuristic search, an agent must select its next action within a prespecified time bound. Amazingly, it has been possible to prove that, despite this myopia, certain real-time search algorithms are complete (guaranteed to eventually reach a goal) under a small set of assumptions. This hinges on the algorithms learning updated heuristic values. In order to elucidate the space of complete real-time algorithms, previous work has provided general algorithm frameworks, into which components that meet certain criteria can be plugged, and proved them complete. Recently, a cost-algebraic real-time framework was proposed that makes strong assumptions on its heuristic learning component but only weak assumptions on its action commitment component. In this paper, we show that the opposite is also possible: we present and prove complete an alternative framework that makes weaker assumptions about heuristic learning but stronger assumptions on action commitment. We demonstrate in the domain of Real-time SIPP how the new framework implies the completeness of previously-published algorithms This work expands our knowledge of the design space of complete real-time search algorithms.

Introduction

The problem setting of real-time heuristic search (Korf 1990) requires that the planner compute the agent's next action within a specified time bound. For example, this bound may be set to the duration of the previous action, so that the next action will have been selected by the time that the previous one is complete. Many real-time heuristic search algorithms proceed through three stages while selecting the next action: lookahead search, where the planner explores the local state space of the agent's current state; heuristic learning, where the planner updates the heuristic function based on that exploration; and commitment, where the agent actually selects the next action of the agent.

In these algorithms, it is the combination of heuristic learning and the commitment strategy that results in the algorithm being complete (guaranteed to eventually reach a goal). The learning allows the agent to plan its way out of local minima in the heuristic function, even if the local minima are too large to explore in any single lookahead search, while the commitment strategy defines how the agent decides to act. For example, LSS-LRTA* (Koenig and Sun 2009) searches with a time-limited A*, learns an updated heuristic for every state expanded by the lookahead search, and commits to the entire partial plan leading to a least f frontier state. LSS-LRTA* has been shown to be complete, however it is not obvious how free we are to modify the approach without endangering that result.

We call a set of algorithms with a shared structure that solve a particular problem a framework. Existing frameworks for real-time heuristic search include Learning Real-Time Search (LRTS, Bulitko and Lee 2006) and Cost-algebraic Real-time Local Search Space Learning A* (CaRLA, Thomas and Ruml 2025). LRTS is a framework of approaches in the style of LRTA* (Korf 1990) that can incorporate weighted heuristic learning and backups, while CaRLA is a framework of LSS-LRTA*-like approaches that generalizes over a variety of lookahead searches. Both frameworks prove that the algorithms they cover are complete.

CaRLA applies to a general notion of costs called a cost algebra (Edelkamp, Jabbar, and Lluch-Lafuente 2005). In a cost algebra, rather than action costs and g-values necessarily being numbers that accumulate through summation along a plan, they can be from any set of values and accumulate through any operation, so long as those fulfill the requirements of a cost algebra. For example, recent work on the Real-time SIPP problem of planning among dynamic obstacles (Thomas, Ruml, and Shimony 2024) uses a cost algebra in which the cost objects are piecewise linear functions of time and the cost accumulation operator is function composition.

To achieve completeness, CaRLA sets a stringent requirement on the heuristic learning: that it be thorough. This means that, like LSS-LRTA*, it must learn a locally optimal heuristic for every expanded state. Because of this, CaRLA is able to be very general with regard to its commitment strategy, requiring only that the planner commit to a state whose current learned h is lower than the current state's h. However, because the requirement of thorough learning is so stringent, the CaRLA framework does not cover many algorithms. For example, Thomas, Ruml, and Shimony describe three algorithms to solve Real-time SIPP: MaxATFS, MedATFS, and RTAS. Of those three, only MaxATFS is in-

cluded in CaRLA, because of CaRLA's stringent requirements on learning that the others do not meet.

In this paper, we prove the completeness of a framework of LRTA*-style approaches that, with minor corrections to the heuristic learning, includes all three of these approaches. We call this framework Cost-algebraic Real-time A* (CaRTA) to highlight that it considers RTA*-style (Korf 1990) searches that learn updated values only for the root node but commit towards the least f frontier state, rather than just a better frontier state. In contrast to CaRLA, we weaken the thorough requirement on the heuristic learning, but retain completeness by strengthening the requirement on the commitment strategy. This trade off can be particularly important in domains where thorough learning may not always pay off empirically. Unlike LRTS, we do not consider weighted heuristic learning because the multiplication operation does not generalize to cost-algebras. This work expands the set of real-time heuristic search approaches that are known to be complete.

Background

Our work builds upon prior research in cost-algebraic heuristic search, particularly real-time search. We begin with some definitions.[1]

Cost-algebraic Heuristic Search

A cost algebra is a 4-tuple $\langle \Omega, \bullet, \preceq, \mathbf{1} \rangle$ where Ω is the set of possible edge and path costs, $\bullet$ is the operator for computing a path's cost from the costs of its edges, $\preceq$ is the ordering on costs, and $\mathbf{1}$ is the cost of the empty path. The use of $\bullet$ rather than the usual $+$, and $\mathbf{1}$ rather than the usual 0 highlights that the assumptions of the usual setting do not necessarily hold.

Definition 1 (Monoid)**.** *Let Ω be a set and $\bullet : \Omega \bullet \Omega \to \Omega$ a binary operator. A **monoid** is a triple $\langle \Omega, \bullet, \mathbf{1} \rangle$ such that $\mathbf{1} \in \Omega$ and for all a, b, c $\in \Omega$:*

associativity $a \bullet (b \bullet c) = (a \bullet b) \bullet c$
identity $\mathbf{1} \bullet a = a \bullet \mathbf{1} = a$

Monoids are similar to groups, except without the requirement of inverses.

Definition 2 (Total order)**.** *A total order $\preceq$ is a binary relation on set Ω where, $\forall a, b, c \in \Omega$:*

1. $a \preceq a$,
2. $a \preceq b \wedge b \preceq c \implies a \preceq c$,
3. $a \preceq b \wedge b \preceq a \implies a = b$, *and*
4. $a \preceq b \vee b \preceq a$.

If Ω is a set and $\preceq$ is a total order on Ω then a $\prec$ b denotes $(a \preceq b) \wedge (a \neq b)$, and $a \succeq b$ and $a \succ b$ are alternative ways of writing $b \preceq a$ and $b \prec a$ respectively.

Definition 3 (Cost product)**.** *If $\langle \Omega, \bullet, \mathbf{1} \rangle$ is a monoid and $\langle a_1, \cdots, a_n \rangle \in \Omega^n$ is a sequence of length n, then*

$$\prod_{i=1}^{n} a_i = \begin{cases} \mathbf{1}, & n = 0 \\ a_1, & n = 1 \\ a_1 \bullet \cdots \bullet a_n, & n > 1. \end{cases}$$

[1]Definitions 1, 2, 3, 5, and 6 are from Holte and Zilles (2019). Definition 4 is from Edelkamp, Jabbar, and Lluch-Lafuente (2005).

Definition 4 (Least)**.** *The least operator $\bigsqcup$ on a set of costs A with ordering $\preceq$ is defined: $\bigsqcup A = c$ with $A \subseteq \Omega$ and $c \in A$ such that $\forall a \in A : c \preceq a$.*

Definition 5 (Isotonicity)**.** *A monoid $\langle \Omega, \bullet, \mathbf{1} \rangle$ with total order $\preceq$ on Ω is **isotone** iff $a \preceq b$ implies both $(a \bullet c) \preceq (b \bullet c)$ and $(c \bullet a) \preceq (c \bullet b)$ for all a, b, c $\in \Omega$.*

Definition 6 (Cost algebra)**.** *A **cost algebra** is a 4-tuple $\langle \Omega, \bullet, \preceq, \mathbf{1} \rangle$ such that:*

- $\langle \Omega, \bullet, \mathbf{1} \rangle$ *is a monoid,*
- $\preceq$ *is a total order on Ω,*
- $\mathbf{1} = \bigsqcup \Omega$, *and*
- $\langle \Omega, \bullet, \mathbf{1} \rangle$ *with $\preceq$ is isotone.*

For example, the typical shortest path optimization setting, for which A* has been shown to be optimally efficient (Dechter and Pearl 1985; Holte and Zilles 2019), is the cost-algebra $\langle \mathbb{R}^{\geq 0}, +, \leq, 0 \rangle$ defined by the monoid $\langle \mathbb{R}^{\geq 0}, +, 0 \rangle$ of non-negative real-valued costs accumulated through summation and the total order $\leq$ over these costs. Domain-independent Dynamic Programming (Kuroiwa and Beck 2026) is another framework that considers cost-algebraic dynamic programming.

Real-time Heuristic Search

A real-time heuristic search problem is represented by a seven-tuple $\langle S, \sigma, \delta, s_o, h_0, \mathit{isGoal}, b \rangle$, where the state space S is the set of possible states, σ is the mapping of states to the set of their successors, $\delta : S \times \sigma(S) \to \mathbb{R}^{\geq 0}$ is the cost function that defines a finite cost for every edge in σ, $s_o \in S$ is the start state, $h_0 : S \to \mathbb{R}^{\geq 0} \cup \{\infty\}$ is the initial heuristic function that is non-negative for all states and 0 for all goal states, $\mathit{isGoal} : S \to \{\mathit{true}, \mathit{false}\}$ is the goal predicate defining which states are goals, and b is the time bound. Generally, the time bound is specified in units of time and an estimate of the expansion rate of the search is used to calculate a budget in expansions, e.g., a time budget of 0.1 second and an expansion rate of 100 Hz would correspond to a budget of 10 expansions. For this work, as we are not grounded in a particular application, we will assume that b is a budget given directly in expansions.

A common objective of a real-time heuristic search algorithm is to minimize the agent's goal achievement time. Because the agent may be able to plan and execute concurrently, this is often less than the sum of the time spent planning and the time spent executing the plan. A plan $p = \langle s_0, ...s_i..., s_n \rangle$ is a sequence of states, and δ denotes the cost of a plan with

$$\delta(p) = \delta(s_0, s_1) \ + \ \cdots \ + \ \delta(s_{n-1}, s_n) = \sum_{i=1}^{n} \delta(s_{i-1}, s_i).$$

The set of plans $P_{u,v}$, for states $u, v \in S$, consists of all plans from u to v through states in S connected by σ. To denote the cost of a least-cost plan in S from u to v we use:

$$\delta(P_{u,v}) = \bigsqcup_{p \in P_{u,v}} \delta(p). \tag{1}$$

Note that while $P_{u,v}$ may contain an infinite number of plans, if costs are isotonic we do not need to consider plans

with loops, so if we assume a finite sized state space the least operation in Equation 1 need only consider the finite set of non-looping plans.

Two paradigmatic algorithms for this setting are Learning Real-time A* (LRTA*, Korf 1990) and Local Search Space Learning Real-time A* (LSS-LRTA*, Koenig and Sun 2009). Both proceed through three stages: lookahead search, heuristic learning, and commitment, and they differ in how each stage is conducted. We will refer to each execution of these three stages as an iteration of the search, with the current state referring to the state that is both the root state of the lookahead search and the source state of the action emitted by the commitment.

Search Lookahead The search of LRTA* is a fixed-depth search, while the search of LSS-LRTA* is a node-limited A* that is the same as a typical A* search except that the search halts either if a goal is selected for expansion or the expansion budget is exhausted. A search node is a tuple, $\langle s, g, h, p \rangle \in N$, corresponding to the state s and containing the extra information needed by the search: g the cost to s, h the heuristic estimate of the cost to reach a goal from s which is $h_0(n)$ if n has never before been generated or the learned $h(n)$ otherwise, and p a parent pointer to enable reconstructing the plan once a solution is found. We use N to refer to the space of possible search nodes.

As in classic heuristic search, real-time lookahead maintains two sets: *OPEN* $\subset N$ is the set of nodes on the search frontier, those that have been generated but not yet expanded, and *CLOSED* : $S \rightarrow N$ is a mapping of states to their corresponding nodes that have been expanded. In a real-time search, the agent will commit to changing its actual state at the end of each iteration. For this reason, we differentiate between s_o, the start state of the real-time search problem as a whole, and s_r, the current state of the agent, and the root of an individual iteration of the lookahead search. For example, in LRTA* the fixed-depth search can proceed either depth-first or breadth first, and in LSS-LRTA* the node-limited A* search repeatedly expands the lowest $f(n) = g(n) + h(n)$ node $n \in OPEN$, generating the children of n and adding them to *OPEN*.

The search space of the lookahead search is called the local search space (LSS):

Definition 7 (Local search space)**.** *The LSS of an iteration of a real-time heuristic search is the search tree of nodes expanded by that iteration's search. When the search phase of an iteration of the search finishes, LSS = CLOSED.*

Definition 8 (Fringe)**.** *The fringe of a set $X \subseteq S$ is the subset of the state space that contains exactly the successors of all states in X that are not themselves in X, i.e, $\bigcup_{x \in X} \sigma(x) \setminus X$.*

We will generally refer to nodes by their corresponding states, e.g., 'the state s was expanded' rather than 'a node n corresponding to the state s was expanded'. The reason for this is that we will be looking at states that have been expanded by several different iterations of search, and thus may have had several different nodes referring to them, with different g-values corresponding to different starting states

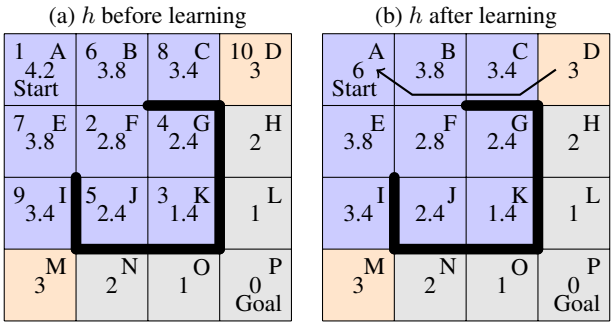

Figure 1: An example of LRTA* learning

for that iteration of search and possibly different h values depending on the heuristic learning.

Heuristic Learning The heuristic learning of LRTA* updates the heuristic value of the current state of the agent s_r from a least-f frontier state denoted s_f:

$$h(s_r) \leftarrow \min_{s_f \in o} g(s_f) + h(s_f). \tag{2}$$

In contrast, the learning of LSS-LRTA* backs up h from the frontier throughout the LSS by backing up the heuristic values from child $c \in$ *LSS* $\cup$ *fringe*(*LSS*) to parent $p \in$ *LSS*, with the backup:

$$h(p) \leftarrow \min_{c \in \sigma(p)} \delta(p, c) + h(c). \tag{3}$$

This backup is applied to each state in the LSS in a Dijkstra-style traversal from the search frontier into the interior of the search, such that Equation 3 holds with equality for all states in the LSS. Some approaches, such as RTA* (Korf 1990) and wbLRTA* (Bulitko and Sampley 2016) learn an inadmissible h with the intuition that this will speed up the agent learning to escape local minima.

Commitment Strategy The commitment strategy of both LRTA* and LSS-LRTA* is to commit towards a least-f frontier state. Though both are commonly implemented as committing to only the first step along the plan, LSS-LRTA* as originally described executes the entire partial plan to a best looking state on the search frontier, which is in contrast to the LRTA* strategy of committing only to the first action along that path.

Heuristic Learning Example Figure 1 and Figure 2 show examples of LRTA* and LSS-LRTA* learning an improved h in 2D grid navigation with 8-way motion. Cells are annotated with the initial heuristic (h_0), which is 8-way distance to the goal, ignoring obstacles. A is the start state. The LSS is shown in blue, the fringe (i.e. the search frontier) is shown in orange, and unexplored states in gray.

For LRTA*, the lookahead search expands three layers (including the root) and tie breaking prefers right before down. Figure 1a shows the expanded nodes in blue and the search frontier in orange, consisting of M and D. Expansion order is noted at the top left of cells. LRTA* learns an updated heuristic for only the current state, which in this case

results in $h(A) = g(D) + h(D)$ $(6 = 3 + 3)$. LRTA* then commits to the first step on the plan to D, which is B.

For LSS-LRTA*, the expansion budget is 10, tie breaking prefers low h and then right before down. Figure 2b is annotated with the learned heuristic, with an arrow from each generated state to the predecessor that inherits its h value. The states in the local minimum (F, G, J, and K) learn they must go away from the goal to escape. For example, state C learns its heuristic from state D, state B from state C, state F from state B, and states G, J, and K from state F. The starting state learns its heuristic from state B. State D retains the same h value, however its provenance has changed, from the original starting heuristic, $(h(D) = h_0(D) = 3)$ to a heuristic resulting from the path to state H on the frontier, $(h(D) = \delta(D, H) + h_0(H) = 1 + 2 = 3)$. While numerically this does not matter, it illustrates that the learning backup of LSS-LRTA* works by exactly propagating the frontier heuristic values throughout the interior of the search.

Cost-algebraic Real-time Heuristic Search

The generalization of the real-time heuristic search problem to the cost-algebraic setting is the tuple $\langle S, \sigma, \omega, s_o, h_0, isGoal, b\rangle$, identical to before, except the scalar edge cost function δ has been replaced by the cost-algebraic equivalent $\omega : S \times S \to \Omega$ and h_0 is cost-algebraic, i.e., $h_0 : S \to \Omega$, and we assume that $h_0(s_g) = \mathbf{1}$ for all goal states $\{s_g \in S : isGoal(s_g)\}$. We further overload ω to denote the cost of a plan with:

$$\omega(p) = \omega(s_0, s_1) \bullet \cdots \bullet \omega(s_{n-1}, s_n) = \prod_{i=1}^{n} \omega(s_{i-1}, s_i).$$

The pseudocode for Cost-algebraic Real-time Heuristic Search (CaRL) (Thomas and Ruml 2025) is shown in Algorithm 1. A CaRL search is instantiated by specifying the search strategy (Search), the heuristic learning strategy (Learn), and the commitment strategy (Commit). Once instantiated, for a specific problem instance CaRL is given the domain D (used by CaRL to compute σ and ω), s_o the origin of the search, h_0 the original heuristic, *isGoal* the goal predicate, and b the search budget. The current state and heuristic are initialized in Line 2. The lookahead search is performed (Line 4), originating at the current state s_r and limited by budget b, and returns the closed list c representing the Local Search Space (LSS), which is the set of states expanded during the search, and the open list o representing the frontier, containing the nodes generated but not expanded during the search. If the search returns an empty frontier, then it has found itself in a dead end and CaRL terminates (line 5) (the completeness of CaRLA relies on the assumption that there are no dead ends, but this pseudocode does not). Then the heuristic learning is performed (Line 7), using the LSS and frontier to update h. Finally, the commitment strategy is used to select the next state s_r for the agent (Line 8).

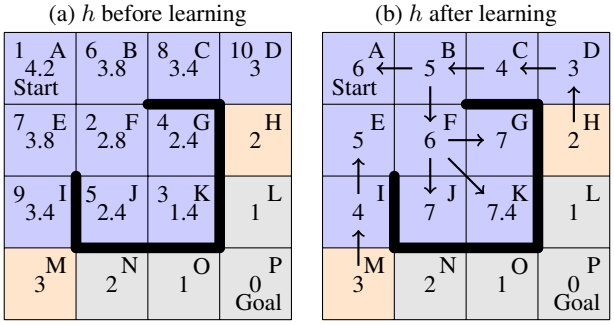

Figure 2: An example of LSS-LRTA* learning

Algorithm 1: Cost-algebraic Real-time Heuristic Search

```
1: function CaRL(D, s_o, h_0, isGoal, b)
2:     s_r ← s_o, h ← h_0
3:     while ¬isGoal(s_r) do
4:         o, c ← Search(D, s_r, h, isGoal, b)
5:         if |o| = 0 then
6:             return DeadEnd
7:         h ← Learn(D, h, o, c, b)
8:         s_r ← Commit(D, h, o, c)
9:     return Success
```

The set of CaRL algorithms is defined in terms of the constraints on the lookahead search, heuristic learning, and commitment strategy employed.

Cost-algebraic LSS-LRTA*

Cost-algebraic Real-time Local Search Space Learning A* (CaRLA, Thomas and Ruml 2025) is a framework of LSS-LRTA*-style CaRL search algorithms that has been shown to be complete subject to the following assumptions:

A1 all actions costs are positive, i.e., $\omega(a, b) \succ \mathbf{1}$;

A2 the state space is finite, i.e., $|S| < \infty$; and

A3 a goal is reachable from every state.

While the agent is not at a goal, CaRLA alternates through the three stages of a real-time search. The nature of the search used by CaRLA is very flexible, in contrast with most prior work that relied on a specific search behavior. CaRLA requires that the lookahead search has a budget sufficient to expand at least one node. If the search has generated a goal then the frontier will not be empty, as it contains at least that state.

Definition 9 (Thorough)**.** *A heuristic learning approach is thorough iff the following Bellman-like condition holds after the heuristic learning stage of a search iteration is complete:*

$$\forall p \in LSS : h(p) = \bigsqcup\{\omega(p, c) \bullet h(c) : c \in \sigma(p)\}$$

CaRLA requires that the heuristic learning is thorough; this means that the information from the search frontier is backed up to every state in the LSS. This mimics the behavior of LSS-LRTA* (Koenig and Sun 2009) in real-time heuristic search, in contrast with approaches such as LRTA* (Korf 1990) where each round of search learns the heuristic of only the root of that search. These two approaches can be seen as opposite ends of a spectrum of methods, with

the thorough learning of CaRLA or LSS-LRTA* representing a 'maximal-learning' approach, and LRTA* representing a more 'minimal-learning' approach (which frees up more time for the lookahead search, which given the same time budget may pay off depending on the problem).

CaRLA requires that its commitment strategy is goal-aware and productive.

Definition 10 (Goal aware)**.** *A commitment strategy is goal aware iff when at least one goal state is present in the LSS it commits to a state along a path to a least cost goal.*

Definition 11 (Productive)**.** *A commitment strategy is productive iff when at current state s_r it commits to a state along the path to a frontier state s_f with $h(s_f) \prec h(s_r)$ unless there is a goal in the LSS.*

Goal aware considers the case where the goal is in the LSS (and does not apply otherwise), and Productive considers when the goal is not in the LSS. LRTA* and LSS-LRTA* commit towards a least-f frontier state; CaRLA relaxes this to allow any state with a lower h, allowing the planner to, for example, commit to use an unbiased estimate of path cost when deciding where to go (subject to being productive) instead of the admissible f.

CaRLA has been shown to be complete.

Cost-algebraic Real-time A*

This paper introduces CaRTA, a complementary, overlapping set of CaRL search algorithms to CaRLA that take a more minimal-learning approach in comparison to CaRLA. In return for the extra flexibility with regards to the learning method, CaRTA makes a less general assumption on the commitment strategy. CaRTA searches must commit towards a least-f frontier state, which though more restrictive includes approaches such as LRTA* and LSS-LRTA*. The cost-algebraic heuristic search problems considered by CaRTA are the same as for CaRLA, and they have identical pseudocode (Algorithm 1). CaRTA shares the LRTA* subset with LRTS, though CaRTA does not consider weighted heuristics or backtracking like LRTS.

The requirements on heuristic learning are looser than CaRLA. A straightforward translation of LRTA* would relax the 'thorough' requirement to require learning only to the search root. However, as we describe in the following section, LRTA* as originally described is not complete, and likewise this approach would not be complete. Instead, CaRTA performs a slightly more complicated learning method than LRTA*, incorperating a pathmax-style backup along the least-f plan, rather than the Bellman-style backup used by of LRTA*.

In order to describe a cost-algebraic version of pathmax, we first define the most operator, similar to the least operator.

Definition 12 (Most)**.** *The least operator $\bigsqcap$ on a set of costs A with ordering $\preceq$ is defined: $\bigsqcap A = c$ with $A \subseteq \Omega$ and $c \in A$ such that $\forall a \in A : a \preceq c$.*

Now, we can describe the cost-algebraic pathmax operator similar to the common scalar-h version:

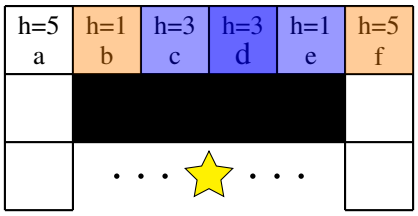

Figure 3: Example problem instance where LRTA* can fail.

Definition 13 (Cost-algebraic Pathmax)**.** *For plan p, the cost-algebraic pathmax of plan p is*

$$\bigsqcap\{(\prod_{i=1}^{n} \omega(p_{i-1}, p_i)) \bullet h(p_n) : n \in [1, |p|)\}$$

Definition 14 (pathmax)**.** *A heuristic learning approach is pathmax if, 1) for any updated state $s \in LSS$, the learned heuristic h' is increased with respect to the prior heuristic h ($h(s) \preceq h'(s)$) and h' remains admissible, and 2) for the current state s_r, the learned heuristic is the cost-algebraic pathmax to a least-f frontier state $s_f \in fringe(LSS)$.*

Definition 15 (minf)**.** *A commitment strategy is minf if it commits to the first step of the same least f plan used by the heuristic learning.*

We define CaRTA as the set of cost-algebraic real-time heuristic search algorithms with pathmax learning and minf commitment strategies.

LRTA* is Incomplete

In this section, we give an example of why the LRTA* heuristic learning backup (Equation 2) from the frontier to the root state of the search is insufficient to guarantee completeness for LRTA*. Specifically, this counterexample uses an inconsistent (but admissible) heuristic and a search depth $d > 1$. If $d = 1$, then the learning is thorough and the search is covered by the CaRLA framework (and therefore complete). If the heuristic is consistent, LRTA* is complete, but it may still scrub for a long time (Sturtevant and Bulitko 2016). The original proof of completeness for LRTA* (Korf 1990) claims the same proof for RTA* applies. However, that proof requires that "the value written into the old state must be strictly greater than the value of the new state". This is correct for RTA*, but as this example demonstrates, it does not always hold for LRTA* as originally described.

Figure 3 shows a grid with 4-way unit cost movement. The states of interest are labeled $a - f$, the initial state is d, and the goal is reachable from all states via a or f (represented abstractly by the star in Figure 3). The heuristic h is admissible, and the initial state d is within a small, symmetric local minimum between a and f. From the starting state d, LRTA* with a search depth of 2 conducts a fixed-depth search that results in expanding d, c, and e, and generating (but not expanding) the frontier $\{b, f\}$. The heuristic update of $h[d]$ is $\min(g[b] : 2 + h[b] : 1 = 3, g[f] : 2 + h[f] : 5 = 7) = 3$, which does not change the heuristic for d. LRTA* then commits to the first action of the least f frontier plan (b, with $f(b) = 3$), so the next state will be c. From

c, we are in an identical situation as we started, just mirrored ($a, b, c \leftrightarrow d, e, f$ respectively), so LRTA* will scrub forever and is not complete.

The solution we propose for CaRTA (and for LRTA* for that matter) is to instead learn a cost-algebraic pathmax. The LRTS framework (Bulitko and Lee 2006), which also generalizes LRTA*, uses this heuristic learning approach (which they call "max of min") because it learns a tighter heuristic that the original "mini-min" rule. Bulitko and Lee show that LRTS with the max of min learning is complete, however, to our knowledge we are the first to point out that LRTA* is incomplete with the original "mini-min" heuristic learning approach for search depths $d > 1$.

For example, consider the same example in Figure 3, except using the pathmax backup (Definition 13) rather than the LRTA* backup (Equation 2). From the starting state d, this CaRTA search with a search depth of 2 conducts a fixed-depth search that results in expanding d, c, and e, and generating (but not expanding) the frontier $\{b, f\}$. The heuristic update of $h[d]$ is:

$$\min(\max(\omega(d,c) + h[c], \omega(d,c) + \omega(c,b) + h[b]),\\ \max(\omega(d,e) + h[e], \omega(d,e) + \omega(e,f) + h[f]))$$

which evaluates to: $\min(\max(1+3, 1+1+1) : 4, \max(1+1, 1+1+5) : 7) = 4$ which increases the heuristic for d, eventually increasing it enough to escape this local minima.

Completeness of CaRTA

In this section we show that CaRTA is complete, i.e., it eventually reaches a goal. We make the same assumptions (A1-A3) as CaRLA. Our proof of completeness for CaRTA is similar to previous real-time completeness proofs: it features the notion of a circulating set of states that the search must cycle through forever in order to remain incomplete.

Definition 16 (Circulating set)**.** *A circulating set is a subset of the state space, $S_\circ \subseteq S$, such that there exists a finite time $t_\circ$ after which, 1) the current state is always in $S_\circ$ and 2) every state in $S_\circ$ is the current state an infinite number of times.*

In contrast with CaRLA, because CaRTA's learning is not thorough, the circulating set contains the nodes actually visited (i.e, that serve as the root of a round of lookahead search) by the agent, rather than the set of all nodes expanded by the lookahead searches (which may be the same, but will generally be a superset).

A sweep is to $S_\circ$ what an iteration of search is to the LSS.

Definition 17 (Sweep)**.** *A **sweep** of a set $X \subseteq S$ is a finite and contiguous sequence of iterations of a real time search, taking place over the interval of time $i = [t_b, t_e]$, such that every state in X is expanded at least once during i.*

Typically, we will refer to a sweep of the circulating set, after which each $s \in S_\circ$ has been visited at least one more time. As each state is visited infinitely often, there must be a sweep infinitely often. $S_\circ$ does not contain a goal state, as CaRTA is goal aware.

For example, Figure 4 shows a cartoon state space S with a donut shaped circulating set $S_\circ$. The agent might expand

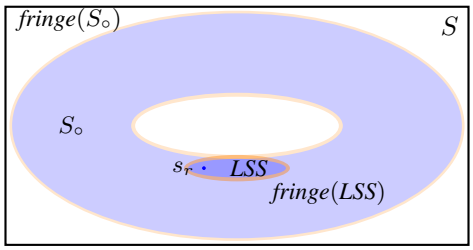

Figure 4: Illustration of the subspaces of state space S: $S_\circ$ and *LSS* are shown in blue, and their fringes in orange.

each state in $S_\circ$ as it loops around the circle. Also shown is the LSS of a search originating at state s_r with search frontier *fringe*(*LSS*). States in the fringe of the LSS may be in the fringe of $S_\circ$ or they may be in $S_\circ$ itself.

We begin by assuming in contradiction that CaRTA is incomplete.

Lemma 18. *If CaRTA is incomplete, it must have a circulating set.*

Proof. CaRTA only terminates when a goal is reached (Algorithm 1 Line 9), because by **A3** no dead-ends exist. So if it is incomplete it must not terminate. By **A2** the search space is finite. So label each visited state $s \in S$, with the time t_s when it is last visited, with $t_s = \infty$ if s is always visited again after all times $t \in \mathbb{R}$. Leave unvisited states unlabeled. As the search space is finite, if CaRTA does not terminate some labeled states must have $t_s = \infty$, so define $S_\circ \leftarrow \{s \in S : t_s = \infty\}$ and $t_\circ \leftarrow \max_{s \in S \setminus S_\circ} t_s$. □

Lemma 19. *In each round of search, either (or both):*

1. *The current state s_r learns an increased heuristic ($h(s_r) \prec h'(s_r)$, where h is the old heuristic and h' the new), or*
2. *the next committed-to state s'_r has a lower h ($h(s'_r) \prec h(s_r)$).*

Proof. In contradiction, assume $h'(s_r) \preceq h(s_r) \wedge h(s_r) \preceq h(s'_r)$. Because CaRTA's learning is pathmax, $\omega(s_r, s'_r) \bullet h(s'_r) \preceq h'(s_r)$, so $\omega(s_r, s'_r) \bullet h(s'_r) \preceq h(s'_r)$, a contradiction due to A1. □

Figure 5 shows three examples illustrating the different possible cases of for Lemma 19. In all three, the heuristic is consistent, a is the starting state, $\{b\}$ is the search frontier, and the star is a cartoon representation of the goal which is reached via b. Figure 5a shows the case where 1) holds, but 2) does not. Precisely: $s_r = a, s'_r = b$ so $h(s_r) = 4, h'(s_r) = 6, h(s'_r) = 5$ therefore: 1) $h(s_r) : 4 < h'(s_r) : 6$, and not 2) $h(s'_r) : 5 < h(s_r) : 4$. Figure 5b shows the case where 1) and 2) both hold: 1) $h(s_r) : 5.5 < h'(s_r) : 6$, and 2) $h(s'_r) : 5 < h(s_r) : 5.5$. Finally, Figure 5c shows the case where 1) does not hold, but 2) does: not 1) $h(s_r) : 6 < h'(s_r) : 6$, but 2) $h(s'_r) : 5 < h(s_r) : 6$.

Theorem 20. *With assumptions A1-A3, CaRTA is complete.*

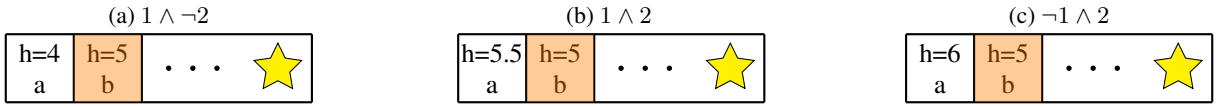

Figure 5: Lemma 19 examples with a consistent heuristic.

Proof. In every state visited by the search, either the heuristic is increased or we move to a state with a lower heuristic (Lemma 19). However, at least once per sweep of the circulating set, we must be at a state with least h among states in the circulating set. There we can only raise h, because there is no state with a lower h that could be moved to. Eventually, h will be raised sufficiently within the circulating set that the min-f frontier node selected by some round of search will be in $fringe(S_\circ)$, because the circulating set is finite and so after a finite number of sweeps the lowest h in the circulating set will grow beyond any fixed heuristic in the fringe, and at least one such fringe state must exist because the goal is reachable from all states and not in the circulating set. The search will then commit to leaving the circulating set, which contradicts its existence. □

Convergence of h with CaRTA

We consider the repeating real-time search setting. Repeating CaRTA is identical to a CaRTA search except that when a goal is reached, the agent returns to the starting state and begins the search anew, but keeping the learned heuristic. In this section, we show that through repeatedly solving the same problem CaRTA algorithms will arrive at the optimal heuristic h^* for all states along at least one optimal plan. We follow an approach similar to the convergence proof for LRTA* (Korf 1990). (We rely on the stricter nature of the CaRTA commitment strategy, so this proof does not apply directly to CaRLA algorithms that are not also CaRTA algorithms, though we also do not rule out the possibility that CaRLA has a similar property.) We will call each subproblem, between starting at the start state and reaching a goal, an epoch of the search. Our line of reasoning to show convergence uses the same assumptions (**A1**-**A3**) as the proof of completeness, with the additional assumption:

A4 the initial heuristic is admissible.

This assumption is required because the combination of CaRTA's non-thorough learning and limited requirements on the behavior of the looakahead would allow too-large h values to linger without being corrected.

Similar to Korf 1990 we assume a set of starting states, S_{st}, with a corresponding set of all optimal plans beginning at those states (P_{st}). If the repeating search is guaranteed to revisit each of those states infinitely often (for example, due to random tiebreaking), then our result applies to those states. The pathmax update can be disjoint between successive rounds of search. However, because the lookahead search halts when a goal is expanded there is a shared base case for us build upon. From this we can define the level of a state:

Definition 21 (Level)**.** *The level of a state s is the smallest number of actions in an optimal plan from s to a goal:*

$$level(s) = \min_{p \in P_s^*} |p|, \tag{4}$$

where P_s^* is the set of optimal plans from s.

We will show that in finite time the optimality of the heuristic will propagate level to level.

What slows this process is the possibility for there to be a state in the search frontier with an underestimating h that is never visited by the search. However, the combination of pathmax and minf alleviate this:

Lemma 22. *A repeating CaRTA search will in finite time learn an increased heuristic for any state causing pathmax to learn a too-low heuristic for a state on an optimal plan.*

Proof. Consider optimal plan p, with ith state p_i which has $h(p_i) < h^*(p_i)$, but with $h(p_j) = h^*(p_j)$ for all j such that $i < j < |p|$. This means there must exist some partial plan p' that starts at p_i and where $\forall k[0, |p'|) : (\prod_{x=0}^{k} \omega(p'_x, p'_{x-1}))\bullet h(p'_k) \preceq h(p_i)$, that is that pathmax learns a too low heuristic because each state until the search frontier along p' is too low. However, this means that minf will commit to such a p'_1 as the next state to visit, and so on for p'_k until the lookahead search expands beyond the low point of the local minimum and $h(p'_k)$ is raised. This chain of states must be finite, as the search space is finite and there can be no loops as $h(p'_{k+1})$ must decrease monotonically in k otherwise $h(p'_k)$ would have been raised already. We know that p_i will be visited again: this is because the pathmax update maintains admissibility so, by Lemma 19 we always either learn an increased heuristic or move to a state with a lower h, eventually p_i will have the lowest h among states in its layer, and the search must pass through that layer to reach a goal, so p_i will be visited when its corresponding start state is randomly chosen. □

With this result, we are ready to show that h converges to optimal:

Theorem 23. *In Repeating CaRTA with assumptions A1-A4 the heuristic values will eventually converge to optimal for every state in every optimal plan from a starting state to a goal state.*

Proof. We prove by induction on levels of the set of states S_c that are in optimal plans P_{st} from states in S_{st}. We will call S_c^i the subset of S_c that is at level i. The base case is S_c^0, which is the set of goal nodes. Because our heuristic is admissible, always $\forall s \in S_c^0 : h(s) = 0 = h^*(s)$. Now for the inductive case, we will show that in finite time $\forall s \in S_c^i : h(s) = h^*(s)$, given that $\forall k \in [0, i) : \forall s^* \in S_c^k : h(s^*) =$

$h^*(s^*)$. By the definition of levels, we know that there is a successor to s, called p_x, with $h(p_x) = h^*(p_x)$ and that $s, p_x, \ldots$ is the prefix of an optimal plan p. By Lemma 22, in finite time any other source of too-low heuristic learning will be increased, so in finite time $h(s) = \omega(s, p_x) \bullet h^*(p_x)$, which is optimal because $s, p_x, \ldots$ is an optimal plan. □

Lemma 22 suggests that h will also converge to optimal for some states beyond those in optimal plans. If learning is thorough then all states expanded infinitely often should converge, and if the search can randomly start from every state then the learned h converges to h^*. However, beyond these special cases, specifying exactly which additional states converge would require specifying the nature of the lookahead search.

Case Study: Real-time SIPP

Real-time SIPP (Thomas, Ruml, and Shimony 2024) is an example of a heuristic search whose analysis relies crucially on a cost-algebraic framework. In this case study we describe how the three Real-time SIPP algorithms described by Thomas, Ruml, and Shimony (2024) fit into the CaRLA and CaRTA frameworks.

Safe Interval Path Planning (SIPP) (Phillips and Likhachev 2011) considers planning problems where states and actions have 'safe intervals' in continuous time, where they are safe or unsafe to occupy. The objective in SIPP is to minimize the arrival time of the agent at a goal. In the offline setting, SIPP can be solved optimally using an A* search where the g-values are the real-valued earliest possible arrival time at each state. This is because it is always at least as good to arrive earlier at a state.

However, in Real-time SIPP the heuristic h is not a scalar; it is instead a function of time. For example, if we are planning to cross a street and the crosswalk is safe, the cost-to-go h might be the time it takes to cross the street: about 10 seconds. But later, when it is not safe, h might be 45 seconds as the agent is forced to wait for the light to change. Thomas, Ruml, and Shimony (2024) search using arrival-time functions (ATFs) for g and h. These functions map the departure time to the earliest arrival time of the agent. This Augmented SIPP (Thomas et al. 2023) search uses a cost algebra consisting of the monoid: $\langle \Omega_{ATF}, \circ, I_{ATF} \rangle$, where Ω_{ATF} is the set of valid ATFs (which are simple piecewise linear functions for SIPP), $\circ$ is function composition, which is a straightforward operation for ATFs because they are piecewise linear, and I_{ATF} is the identity function, where the arrival time is the departure time. This forms a cost-algebra with the ordering preferring earliest arriving nodes (for the current time) and tie breaking to prefer plans that are guaranteed optimal and then applicable for longer (which is possible to determine because of the extra information provided by ATFs).

Using this cost-algebra, Thomas, Ruml, and Shimony (2024) describe three algorithms to solve Real-time SIPP, which in order of decreasing learning thoroughness are: MaxATFS, MedATFS, and RTAS. The three approaches were each found to empirically perform best for different time budgets (more time to search means less thorough learning is more efficient). These approaches partition the heuristic learning into two parts: the static component h_s is admissible for all time (if the crosswalk is ever safe h_s will be the cost to cross it when it is safe), while the dynamic ATF component of the heuristic is h_d. All three use minf commitment, so their commitment strategies are compatible with both CaRLA and CaRTA.

Algorithm	RTAS	MedATFS	MaxATFS
Static	LRTA*	LSS-LRTA*	LSS-LRTA*
Dynamic	LRTA*	LRTA*	LSS-LRTA*

Table 1: Real-time SIPP Algorithm Learning Characteristics

Table 1 shows the learning characteristics of the three approaches. Because MaxATFS' static and dynamic learning are both LSS-LRTA*-style, the learning is thorough so MaxATFs is a CaRLA search. Because RTAS does LRTA*-style learning for both components of the heuristic and MedATFS is only thorough for h_s, neither of them are thorough, so they are not CaRLA searches. But, if we modify them to use the pathmax backup: $\lceil \{h(p_n) \circ g(p_n) : n \in [1, |p|)\} \rceil$, where $\lceil x \rceil$ denotes the upper hull of the ATFS, then they will be in CaRTA and therefore complete.

In summary, we did not previously have an analysis of the completeness for these Real-time SIPP algorithms. Using these frameworks, we have shown that MaxATFS is complete. Through this analysis we have found bugs in MedATFS and RTAS (as in LRTA*). Pathmax is crucial to correct these and make them complete.

Conclusions

We presented CaRTA, a set of cost-algebraic heuristic search algorithms that is complementary and overlapping with CaRLA. CaRLA algorithms must have thorough heuristic learning and can merely have a productive commitment strategy. CaRTA is more general in the learning approach (encompassing pathmax learning to only the search root) and in exchange, a stricter requirement on commitment strategy. This stricter commitment strategy is common in practice and prior work on cost-algebraic real-time heuristic search has found that less thorough learning can empirically outperform more thorough approaches in some settings (Thomas, Ruml, and Shimony 2024). Though CaRTA is intended to generalize LRTA* (Korf 1990), we have shown through counterexample that LRTA* as originally described is incomplete. We show that a pathmax-style approach, similar to the 'max of min' approach used by LRTS , makes CaRTA (and therefore LRTA*) complete. CaRTA strengthens our understanding of the sufficient conditions for completeness for real-time heuristic search algorithms, in conjunction with complementary frameworks such as CaRLA (Thomas and Ruml 2025) and LRTS (Bulitko and Lee 2006). However, there still exist many complete real-time heuristic search algorithms that fall outside of these frameworks, in particular, those that are not 'agent-centered' and thus do not necessarily center their lookahead search around the agent's current state (Björnsson, Bulitko, and Sturtevant 2009; Rivera et al. 2014; Gall, Cserna, and Ruml 2020).

References

Björnsson, Y.; Bulitko, V.; and Sturtevant, N. 2009. TBA*: Time-Bounded A*. In *Proceedings of IJCAI*.

Bulitko, V.; and Lee, G. 2006. Learning in real-time search: a unifying framework. *Journal of Artificial Intelligence Research*, 25(1): 119–157.

Bulitko, V.; and Sampley, A. 2016. Weighted lateral learning in real-time heuristic search. In *Proceedings of SoCS*.

Dechter, R.; and Pearl, J. 1985. Generalized best-first search strategies and the optimality of A*. *Journal of the ACM*.

Edelkamp, S.; Jabbar, S.; and Lluch-Lafuente, A. 2005. Cost-algebraic heuristic search. In *Proceedings of AAAI*.

Gall, K.; Cserna, B.; and Ruml, W. 2020. Envelope-based approaches to real-time heuristic search. In *Proceedings of AAAI*.

Holte, R. C.; and Zilles, S. 2019. On the Optimal Efficiency of Cost-Algebraic A*. *Proceedings of AAAI*.

Koenig, S.; and Sun, X. 2009. Comparing real-time and incremental heuristic search for real-time situated agents. *Journal of Autonomous Agents and Multi-Agent Systems*.

Korf, R. E. 1990. Real-time Heuristic Search. *Artificial Intelligence*.

Kuroiwa, R.; and Beck, J. C. 2026. Domain-independent dynamic programming. *Artificial Intelligence*, 354: 104506.

Phillips, M.; and Likhachev, M. 2011. SIPP: Safe interval path planning for dynamic environments. In *Proceedings of ICRA*, 5628–5635.

Rivera, N.; Illanes, L.; Baier, J. A.; and Hernández, C. 2014. Reconnection with the ideal tree: A new approach to real-time search. *Journal of Artificial Intelligence Research*, 50: 235–264.

Sturtevant, N. R.; and Bulitko, V. 2016. Scrubbing During Learning In Real-time Heuristic Search. *Journal of Artificial Intelligence Research*.

Thomas, D. W.; and Ruml, W. 2025. Real-time Cost-algebraic Heuristic Search. In *Proceedings of SoCS*, volume 18, 162–170.

Thomas, D. W.; Ruml, W.; and Shimony, S. E. 2024. Real-time Safe Interval Path Planning. In *Proceedings of SoCS*.

Thomas, D. W.; Shimony, S. E.; Ruml, W.; Karpas, E.; Shperberg, S. S.; and Coles, A. 2023. Any-Start-Time Planning for SIPP. In *Proceedings of the ICAPS Workshop on Heuristics and Search for Domain-Independent Planning*.