Real-time Cost-algebraic Heuristic Search

Devin Wild Thomas and Wheeler Ruml

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

Abstract

Planning under time pressure arises in many situations. Real-time heuristic search, in which an agent must compute its next action within a prespecified time bound, has proven to be a useful model of real-time planning. However, it is laborious to prove the completeness of new real-time search algorithms. In this paper, we provide a general proof of the completeness of a standard real-time heuristic search algorithm in any problem domain that obeys the axioms of a cost algebra. The proof includes additional detail on how h values change as the algorithm learns. This foundation clarifies the dependence of the proof on domain and algorithm properties and will ease future applications of real-time planning.

Introduction

Real-time heuristic search (Korf 1990) is a common approach to planning under time pressure. It requires that the planner finish computing its next action within a prespecified time bound. For example, a real-time agent might set the time bound based on the duration of the action it is currently executing, so that it will have a next action planned and ready to execute by the time that the current action has finished. Typically, a real-time heuristic search alternates through three stages: lookahead search, where the planner performs a limited search rooted at its current state; heuristic learning, where the developed search space is used to learn updated heuristic values to escape local minima and improve future searches; and commitment, where the planner selects an action or set of actions to commit to executing.

For example, real-time A* (RTA*) (Korf 1990) searches to a fixed depth, learns an updated heuristic value for the root state of the search, and commits to the minimum f top-level action (child of the root). In contrast, local search space learning real-time A* (LSS-LRTA*) (Koenig and Sun 2009) searches using a time-limited A*, learns an updated heuristic for each state expanded during the search by backing up h from the search frontier, and commits to the entire partial plan to the minimum f node on the search frontier.

When designing a new real-time heuristic search algorithm, the central property that researchers often want to achieve is completeness, that the agent reaches a goal state if one exists. Typically, one proves completeness by showing that in order to be incomplete it must have a set of states, which we will call the circulating set, that it revisits over and over forever and then showing that no circulating set can exist. Previous proofs assume we are in the shortest path setting, where cost of a path is the sum of the scalar cost of the edges of the path, and we prefer shortest paths.

In offline heuristic search, A* has been studied in the more general cost-algebraic setting. A cost algebra is a generalization of the typical cost propagation done by a heuristic search, i.e., A* considers the cost of a path to be the sum of the cost of the edges along said path. But, a cost-algebraic heuristic search (Edelkamp, Jabbar, and Lluch-Lafuente 2005) solves a more general set of problems beyond just shortest paths.

For example, in the widest paths problem, path cost accumulates through the operator $\min$ rather than summation, and we prefer larger costs. On the other hand, in any-start-time planning (Thomas et al. 2023), the cost of a plan is a function of time, these costs accumulate through function composition, and we prefer the earliest arriving plan. Both of these are examples of cost algebras. Edelkamp, Jabbar, and Lluch-Lafuente (2005) showed that cost-algebraic A* finds optimal solutions. Further, Holte and Zilles (2019) showed that cost-algebraic A* is optimally efficient in most[1], but not all, of the situations in which A* is optimally efficient (Dechter and Pearl 1985). It would be useful to extend real-time heuristic search similarly.

New variants of real-time search arise regularly. For example, Thomas, Ruml, and Shimony (2024) define a variant of LSS-LRTA* that uses functions to represent h values instead of scalars, and Fickert et al. (2020) define a variant that uses probability distributions for h. Different algorithms can have different strategies for lookahead search, heuristic learning, and commitment. It would be useful to have a general real-time search algorithm framework that could be applied to new cost structures, as cost-algebraic A* can, and for this algorithm to be defined generically so that specific applications have the flexibility to select the best search, learning, and commitment strategies for their setting. This is exactly the role of this paper. We provide a formal definition

[1]When you have a consistent heuristic, but not always when you only have an admissible heuristic.

of cost-algebraic real-time local search space learning A*, which generalizes LSS-LRTA* to the cost-algebraic setting and to a variety of search, learning and commitment strategies, and a proof of completeness for this generalized algorithm. The proof includes novel detail on how the search escapes heuristic depressions. This work clarifies the dependence of the proof on domain and algorithm properties and will ease future applications of real-time planning.

Background

Our work builds upon prior research in cost algebraic heuristic search and real-time heuristic search.

Cost-algebraic Heuristic Search

Cost-algebraic A* is a generalization of A* to a more general notion of cost than the scalar sum of costs used by A* to solve shortest-path problems. We begin by restating the definition[2]: A cost algebra is a 4-tuple $\langle \Omega, \times, \preceq, \mathbf{1} \rangle$ where Ω is the set of possible edge and path costs, $\mathbf{1}$ is the cost of the empty path, $\times$ is the operator for computing a path's cost from the costs of its edges, and $\preceq$ is the ordering on costs.

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

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

Definition 2 (Total order)**.** 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 $\langle \Omega, \times, \mathbf{1} \rangle$ is a monoid and $\langle a_1, \cdots, a_n \rangle \in \Omega^n$ is a sequence of length n, then

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

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

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

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

- $\langle \Omega, \times, \mathbf{1} \rangle$ is a monoid,
- $\preceq$ is a total order on Ω,
- $\mathbf{1} = \bigsqcup \Omega$, and
- $\langle \Omega, \times, \mathbf{1} \rangle$ 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 real-valued costs accumulated through summation and the total order $\leq$ over these costs.

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

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, $\delta : S \otimes \sigma(S) \to \mathbb{R}^{\geq 0}$ 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. (Note that we use $\otimes$ to denote the Cartesian product in order to avoid confusion with the operator $\times$ of a cost algebra.) 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. The most common objective of a real-time heuristic search algorithm is to minimize the agent's total trajectory cost and planning time to reach a goal.

As an example and a starting point for our generalization of real-time heuristic search, we will now go over a specific real-time heuristic search algorithm. Local search space learning real-time A* (LSS-LRTA*) (Koenig and Sun 2009) is a popular real-time heuristic search approach. It alternates between three stages: lookahead search, heuristic learning, and commitment. We will refer to each execution of these three stages as an iteration of the search.

Search Lookahead The search of LSS-LRTA* is a node-limited A*, which 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 A*, the search maintains two sets: $\mathit{OPEN} \subset N$ is the set of nodes on the search frontier, those that have been generated but not yet expanded, and $\mathit{CLOSED} : S \to N$ is the 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 root of an individual iteration of the search. The node-limited A* search repeatedly expands the lowest $f(n) = g(n) + h(n)$ node $n \in \mathit{OPEN}$, generating the children of n and adding them to *OPEN*.

The search space of this node-limited 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

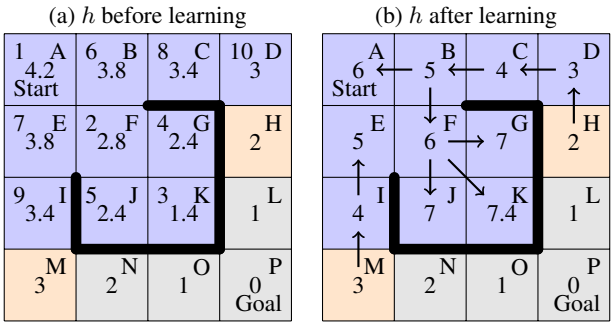

Figure 1: An example of LSS-LRTA* learning

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

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

$$h(p) \leftarrow \min_{c \in \sigma(p)} \delta(p, c) + h(c). \tag{1}$$

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 1 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 LSS-LRTA* is to commit to executing the entire partial plan to a best looking state on the search frontier, which is in contrast to the RTA* strategy of committing only to the first action along that path. The LSS-LRTA* strategy has the benefit of increasing the natural search budget, since there is more time to plan while the agent is executing a whole partial plan rather than a single action.

Heuristic Learning Example Figure 1 shows an example of LSS-LRTA* learning an improved h in 2D grid navigation with 8-way motion. Figure 1a is annotated with the initial heuristic (h_0), which is 8-way distance to the goal, ignoring obstacles. A is the start state, the expansion budget is 10, tie breaking prefers low h and then right before down. Expansion order is noted at the top left of cells. The LSS is shown in blue, the fringe (i.e. the search frontier) is shown in orange, and unexplored states in gray. Figure 1b is annotated with the learned heuristic, with an arrow from each generated state to the predecessor that inherits its h value if any. The states in the local minimum (F, G, J, and K) learn that 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 their heuristic has changed (though the value has not), 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 distinction 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 LSS-LRTA*

We begin by generalizing the real-time heuristic search problem to the cost-algebraic setting. This is straightforward: a cost-algebraic real-time heuristic search problem on the cost algebra $\langle \Omega, \times, \preceq, \mathbf{1} \rangle$ is represented by a tuple $\langle S, \sigma, \omega, s_o, h_0, isGoal, b \rangle$, identical to a real-time heuristic search problem except the scalar edge cost function δ has been replaced by the cost-algebraic equivalent $\omega : S \otimes S \rightarrow \Omega$, and h_0 is cost-algebraic, i.e., $h_0 : S \rightarrow \Omega$, and we assume that $h_0(s_g) = \mathbf{1}$ for all goal states $\{s_g \in S : isGoal(s_g)\}$. A plan $p = \langle s_0, ...s_i..., s_n \rangle$ is a sequence of states, and we overload ω to denote the cost of a plan with:

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

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:

$$\omega(P_{u,v}) = \bigsqcup_{p \in P_{u,v}} \omega(p). \tag{2}$$

Note that though while $P_{u,v}$ may contain an infinite number of plans, thanks to isotonicity we do not need to consider plans with loops, so if we assume a finite sized state space the least operation in Equation 2 need only consider the finite set of non-looping plans.

Next, we describe a cost-algebraic and generalized version of LSS-LRTA*, Cost-algebraic Real-time local search space Learning A* (CaRLA). CaRLA is a generalization of LSS-LRTA* to work on any cost algebra and a more general class of node-limited search methods beyond just A*. We specified CaRLA generically, so that our results may be used by as wide a variety of applications as possible. Additionally, these generalizations allow us to highlight the algorithm properties our results depend on.

Algorithm 1: CaRLA

```
1: function CaRLA(D, s_o, h_0, isGoal, b)
2:     s_r ← s_o, h ← h_0
3:     while ¬isGoal(s) 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 pseudocode for CaRLA is shown in Algorithm 1. A CaRLA 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 CaRLA is given the domain D (used by CaRLA 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. We assume that the search has a budget sufficient to expand at least one node. The current state and heuristic are initialized in Line 2. While the agent is not at a goal, CaRLA alternates through the three stages of a real-time search. 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 LSS and open list o representing the frontier. If the search returns an empty frontier, then it has found itself in a dead end and CaRLA terminates (line 5). If the search has generated a goal then the frontier will not be empty, as it contains at least that state. 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). We now specify the requirements we place on Search, Learn, and Commit for CaRLA.

Search Lookahead The CaRLA search is a node-limited cost-algebraic search. Our only requirement for the search lookahead is that it produces a LSS containing at least the root, and generate the fringe of that LSS. For example, cost-algebraic A*, cost-algebraic GBFS, cost-algebraic focal search, and cost-algebraic breadth-first search would all be compatible with our results, though we would not expect all of these search methods to perform equally well. Methods based on depth-first or beam search could also work if modified to retain a closed list of expanded nodes with parent pointers for heuristic learning. Likewise, in cost-algebras where a weight is applicable a cost-algebra specific weighted A* would work. In fact, it is often possible (though not necessarily helpful) to do a normal shortest path search. For example, consider a real-time agent navigating in the widest path cost-algebra. While likely not as useful as a widest path search, a shortest path search would still generate an LSS that the agent could use.

Heuristic Learning CaRLA's learning is a sequence of cost-algebraic backups from child $c \in \mathit{LSS} \cup \mathit{fringe}(\mathit{LSS})$ to parent $p \in \mathit{LSS}$. Equation 3 is a straightforward generalization of the LSS-LRTA* backup (Equation 1). Summation has been replaced by the $\times$ operator and scalar minimization over the children has been replaced by the cost algebraic least operation ($\bigsqcup$).

$$h(p) \leftarrow \bigsqcup\{\omega(p,c) \times h(c) : c \in \sigma(p)\} \tag{3}$$

We require that CaRLA's learning component implement Equation 3 in a way we will call thorough.

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

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

Similar to its search stage, CaRLA does not prescribe a specific traversal to order the learning, only that the learning is thorough and takes at most time $c = b - runtime(Search)$ (so that it is real-time). For example, Pemberton and Korf (1992) fill a queue with all the states of the LSS and repeatedly pop from the queue, backing up the h value of the popped state s and placing all states with s as a child back on the queue if $h(s)$ changed, while LSS-LRTA* uses a Dijkstra-style traversal, which Edelkamp, Jabbar, and Lluch-Lafuente (2005) prove also finds optimal solutions in a cost algebra. Both of these approaches are thorough because they guarantee to backup to state s after all children $c \in \sigma(s)$ with $h(c) \prec h(s)$. This is because Pemberton and Korf (1992) place all predecessors back on the backup queue for every child updated, and the Dijkstra style traversal performs backups in h-order. The node-limited CaRLA search look ahead produces a LSS with $O(1)$ size, so a Dijkstra-style traversal would be $O(1)$ in time. However, by requiring that our learning be thorough we have excluded approaches (with existing completeness proofs) such as (L)RTA* that only update a single state and wbLRTA* that deliberately scales the learned heuristic. One of the reasons we do not generalize to a weighted inadmissible heuristic is that it is not obvious how the concept of a scalar weight, as used by wbLRTA* and similar algorithms (or in the offline setting, weighted A*) would be translated generally to all cost-algebraic settings.

Commitment Strategy CaRLA commits towards a best frontier state. We require that CaRLA's commitment strategy be goal aware:

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.

Most algorithms terminate the search lookahead when a goal is expanded, and commit to moving towards it. If a goal is not present, it will commit to a 'best' frontier node. As what combination of g, h, or other values make a node the best is an open area of research, we define a general standard to use with CaRLA.

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.

Lemma 12. *Assuming positive action costs ($\omega(a,b) \succ \mathbf{1}$), a CaRLA search iteration, which has completed lookahead search and heuristic learning has at least one frontier state s_f with $h(s_f) \prec h(s_r)$.*

Proof. Because CaRLA's learning completed it must not have detected a dead-end, so the search frontier is not empty. Consider a least child s_c of s_r where (because CaRLA is productive) $h(s_r) = \omega(s_r, s_c) \times h(s_c)$. Because action costs are positive, $h(s_c) \prec h(s_r)$. Follow the same logic for the children of s_c and their successors until the frontier state s_f is reached, $h(s_f) \prec \ldots \prec h(s_c) \prec h(s_r)$. □

We define CaRLA to have a productive commitment strategy. For example: the best top-level action strategy of RTA*, the best frontier node strategy of LSS-LRTA* and the dynamic lookahead strategy using f of Kiesel, Burns, and Ruml (2015) that interpolates between them are all productive because they select the minimum $f = g + h$ frontier node and then move some distance along the path to it. It is also compatible with a greedy style where $f = h$, or a weighted style if the cost algebra supports an operation akin to scaling the weight on h. Strategies such as selecting a frontier node by minimum expected plan cost $\hat{f}$ (Kiesel, Burns, and Ruml 2015) would not be covered by CaRLA without adding a restriction on h to guarantee that the commitment strategy is productive.

In summary, we require that CaRLA conducts a lookahead search that generates a LSS, we require that CaRLA is thorough in backing up h into that LSS, and finally that CaRLA's commitment strategy productively commits to moving towards a state with lower h.

Completeness of CaRLA

In this section we show that CaRLA is complete, i.e., it eventually reaches a goal. We make the following assumptions about the search problem:

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.

Note that $\mathbf{1}$ is defined as the identity (Definition 1) and as the least element of Ω (Definition 6), which means that a cost algebra is already defined to have non-negative edge costs, so A1 is only a slight strengthening.

Our proof is based on the basic structure of the proof of Fickert et al. (2020) for Nancy, which is itself based on the completeness proofs for weighted lateral LRTA* (Bulitko and Sampley 2016) and RTA* (Korf 1990). In outline, we show that the algorithm can only fail to terminate if it becomes stuck in a circulating set of nodes that are visited infinitely often, and under our assumptions, no such set can exist. We will do this with more detail than previous proofs, elucidating the dynamics of heuristic learning.

Definition 13 (Circulating set)**.** A circulating set is a subset of the state space, $S_\circ \subseteq S$, such that there exists a time $t_\circ$ where, for all times $t > t_\circ$, any state expanded by the search is in $S_\circ$ and every state in $S_\circ$ is expanded an infinite number of times after time t.

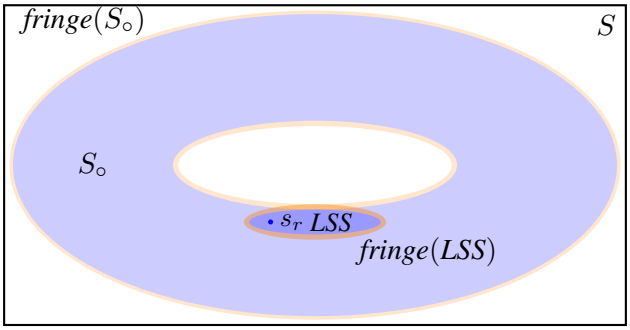

Figure 2: Illustration of the subspaces of state space S: $S_\circ$ and *LSS* are shown in blue, and their fringes in orange.

We denote the fringe of $S_\circ$ as S_f, such that:

$$S_f \coloneqq \{s \in S \setminus S_\circ : (\exists s_\circ \in S_\circ : s \in \sigma(s_\circ)\}.$$

Naturally, as each parent in $S_\circ$ is expanded infinitely often, each child in the fringe is generated infinitely often. $S_\circ$ does not contain a goal state, as CaRLA is goal aware.

For example, Figure 2 shows a cartoon state space S with a donut shaped circulating set $S_\circ$. The agent might expand 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 CaRLA is incomplete.

Lemma 14. *If CaRLA is incomplete, it must have a circulating set.*

Proof. CaRLA 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 expanded state $s \in S$, with the time t_s when it is last expanded, with $t_s = \infty$ if s is always expanded again after all times $t \in \mathbb{R}$. Leave unexpanded states unlabeled. As the search space is finite, if CaRLA does not terminate some labeled states must have $t_s = \infty$, so define $S_\circ \leftarrow \{s \in S : t_s = \infty\}$ and $t_\circ \leftarrow \max_{s \in S \setminus S_\circ} t_s$. □

We next define three important properties of heuristics over sets of states. To our knowledge, the first two (local admissibility and local consistency) are novel, while local optimality has appeared in many prior works under the name local consistency. We have made this change so that local consistency is more similar to global consistency.

Definition 15 (Local admissibility)**.** A heuristic h is **locally admissible** on set $S_L \subseteq S$ if it does not overestimate the cost to reach the goal given the current estimates on the fringe of S_L. That is, with $F \leftarrow \mathit{fringe}(S_L)$ and recalling Equation 2,

$$\forall s \in S_L : \forall s_f \in F : h(s) \preceq \omega(P_{s,s_f}) \times h(s_f).$$

A heuristic is globally admissible when it does not overestimate the cost to reach any goal. With A*, a globally admissible heuristic guarantees that search will return an optimal solution. In contrast, local admissibility is defined with respect to the fringe of a set of states, rather than the goals.

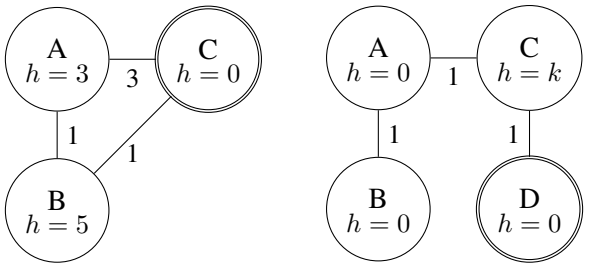

Figure 3: Two examples showing CaRLA's behavior. A is the start, and a double circle indicates a goal.

Definition 16 (Local consistency)**.** A heuristic h is **locally consistent** on set $S_L \subseteq S$ if:

$$\forall p \in S_L : \forall c \in \sigma(p) : h(p) \preceq \omega(p, c) \times h(c).$$

The property of local consistency is identical to the global consistency we are used to, however it is local to the set S_L rather than the entire state space.

Definition 17 (Local optimality)**.** A heuristic h is **locally optimal** on set $S_L \subseteq S$ if it is locally consistent and:

$$\forall p \in S_L : \exists c \in \sigma(p) : h(p) = \omega(p, c) \times h(c).$$

Local optimality is similar to local admissibility, in that it is anchored to the values on the fringe of S_L rather than the goal (when there is not a goal in S_L). For simplicity, in our proofs we will strain these definitions by referring to states that are, for example, 'inadmissible on $S_\circ$'. By this, we mean to identify and select a state that is violating the corresponding relationship, in this case the state $s \in S_\circ$ such that $\exists s_f \in F : h(s) \succ \omega(P_{s,s_f}) \times h(s_f)$.

Observation 18. *Because CaRLA's learning is thorough, after each iteration of search CaRLA learns a locally optimal heuristic on the LSS.*

However local admissibility, or even local optimality do not imply global admissibility. For example, consider the problem shown in Figure 3a, we have 3 states: A, B, and C and three edges: AB, BC, and AC with costs 1, 1, and 3 respectively. The goal is C, and A has $h(A) = 3$. However, B has $h(B) = 5$, and so search with an expansion budget of 1, starting at A will generate B with $f = 1 + 5$ and C with $f = 3+0$ and commit to C. It will never expand B, and thus never learn that $h(B)$ and therefore $h(A)$ are inadmissible. This is despite the fact that $h(A)$ is locally optimal on the LSS ($\{A, C\}$) in this example.

We now explore how heuristic learning propagates in $S_\circ$. Fundamentally, it does so through repeated learning on LSSs that overlap, while the fringe of $S_\circ$ acts as an anchor on the h values. To more clearly understand this relationship, we define two additional concepts: a sweep, which is to $S_\circ$ what an iteration of search is to the LSS; and the layer of a state, which describes how deep it is in $S_\circ$ relative to the $fringe(S_\circ)$.

Definition 19 (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.

In general, we will refer to a sweep of the circulating set, after which each $s \in S_\circ$ has been expanded at least one more time. As each state is expanded infinitely often, there must be a sweep infinitely often.

Definition 20 (Layer)**.** For a set $S_L \subseteq S$, the **layer** of a state $s \in S_L$ is the minimum number of states $s \in S_L$ along any best-looking path through the fringe of S_L. More precisely, where $P_f \leftarrow \{p \in P_{s,s_f} | s_f \in fringe(S_L)\}$:

$$layer(s) = \min_{\{p \in P_f : \omega(p) \preceq \omega(P_f)\}} |p|$$

In the context of the heuristic learning of CaRLA, the backups in the circulating set proceed layer-by layer. Layers are the natural unit of induction for the heuristic learning of CaRLA. We denote the i_{th} layer of S_L as $S_i = \{s \in S_L : layer(s) = i\}$. Note that there may be a path from s to the frontier that contains fewer than $layer(s)$ steps, this is because we restrict layers to only consider minimum cost paths to the frontier.

Lemma 21. *If a heuristic h is locally consistent on set $S_L \in S$, it is also locally admissible on S_L.*

Proof. By induction on layer. In the base case, the fringe of S_1 is a subset of the children of states in S_1, so local admissibility holds. For each layer i, the optimal path to the frontier must be through a state in the layer $i - 1$. At layer $i - 1$ local admissibility holds, so by local consistency of h on S_L, it holds for layer i. □

Observation 22. *If a heuristic h is locally admissible on set S_L, a CaRLA learning backup to parent $p \in S_L$ from child $c \in \sigma(p)$ cannot make h locally inadmissible.*

Now we address the possibility of inadmissible heuristic values within $S_\circ$. One could imagine a situation where a child on the search frontier tricks its parent into learning an artificially high h. However, while that may happen at the start of the search, it will not happen forever.

Lemma 23. *There exists a finite integer k such that, after k sweeps of $S_\circ$, the learned h is locally admissible on $\bigcup_{i=0}^{k} S_i$ forever.*

Proof. By induction on layer i = 1 ... k. In the base case, layer 1, the path to the frontier is only a single edge. And because we are in layer 1, this fringe child must be considered by Equation 3 and either the h will immediately backup from the fringe node, meaning it is locally optimal, or it will backup a lower h from a state in the circulating set, that was in the search frontier for this iteration of CaRLA. Thus after 1 sweep, all states in layer 1 will be locally admissible. Note that in deeper layers, with insufficient sweeps, it could be possible that all the children under consideration are locally inadmissible on $S_\circ$, however this is not the case when the prior layer is guaranteed to be locally admissible. For the inductive step, consider layer i after i sweeps. Every state must have at least one child from layer $i - 1$ that is locally admissible, so they will either backup a locally admissible h from that child or a lower h from some other child. □

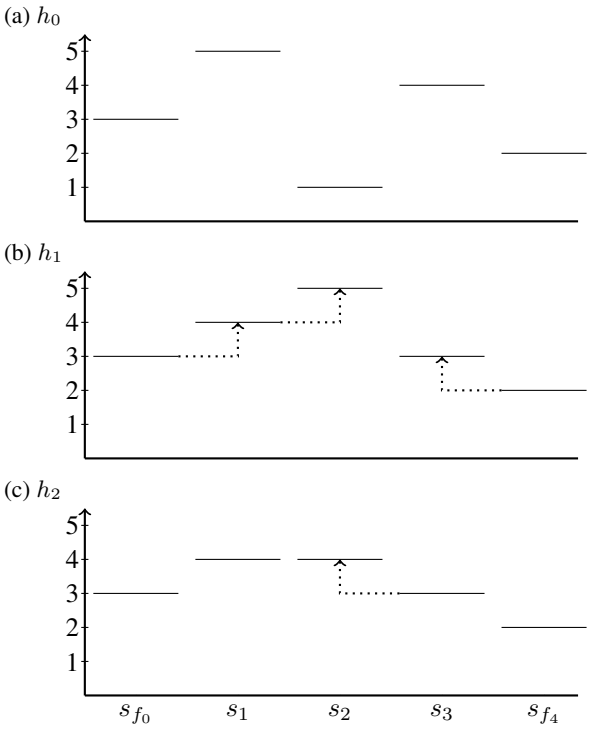

Figure 4: An example illustrating Lemma 23

Admissibility Example Figure 4 illustrates the result of Lemma 23 by showing how the heuristic can evolve over the course of three sweeps, shown as h_0, h_1, and h_2. We have five states. The first and last (s_{f_0}, s_{f_4}) are on the fringe, while the rest (s_1, s_2, s_3) are in the circulating set. Neighboring states on the axis are connected by bidirectional unit edges, and the rest of the circulating set is not shown. Let us assume that, during a sweep a 1 expansion budget search visits each of the three circulating set states in order (s_1 then s_2 then s_3) possibly while visiting other states that are not shown in between. In the beginning (h_0), we have s_1 and s_3 with locally inadmissible heuristics, and s_2 with a not locally optimal heuristic. After one sweep (h_1), we have s_1 learning a lower h from s_{f_0}, s_2 from the newly decreased s_1 (or from s_3) and then s_3 from s_{f_4}. Notice that after 1 sweep we are now locally admissible for the first layer (s_1, s_3) but not the second (s_2). After the next sweep (h_2), by s_2 learning from s_3 the inadmissibility is corrected, and now all shown states have a locally optimal heuristic.

Corollary 24. *After a finite number of sweeps of $S_\circ$, CaRLA has learned a locally admissible h on $S_\circ$.*

Proof. By A2, $|S|$ is finite, so $|S_\circ|$ is as well. If a state $s \in S_\circ$ has *layer*$(s) > |S_\circ|$, the least cost plan to the fringe must involve a cycle. That cycle could be removed, and must have positive cost by A1, which would contradict it being a least cost plan. Therefore, no state can have *layer*$(s) > |S_\circ|$ and so by Lemma 23 we need at most $|S_\circ|$ sweeps. □

Now that we have limited h from above by showing that it is eventually locally admissible on $S_\circ$, we now limit it from below by showing that there is an aspect of h that is monotonically increasing. Recall from the discussion of Figure 1 that an h value learned by CaRLA can be viewed as a tuple: $h(s) = \langle p, h_0(s') \rangle$ where we say $h(s)$ is based on p (which is a shortest path from s to s'), $h_0(s')$ is the original h value of s' at the start of the search, and $h(s) = \omega(p) \times h_0(s')$ is the current heuristic value of s, which is the result of backing up the original heuristic value of s' along p. With this in mind, we are ready to show that CaRLA is complete.

In this proof, the local admissibility of h will provide an upper bound while the monotonic raising of the lowest h value in $S_\circ$ provides a lower bound. The proof has two steps. First, we show that the least h value of not locally optimal state $s_{\min}$ is monotonically increasing. Then, we show that only a finite number of these increases are required to raise the h values to be locally optimal. Our proof of Lemma 25 shares similar elements with the proof of convergence for asynchronous value iteration (Bertsekas 2012, Section 2.6.1), but extended to the cost-algebraic setting.

To understand how $h(s_{\min})$ increases, consider a CaRLA agent with a search budget of 1 expansion in the example shown in Figure 3b. Because C has such a bad heuristic, CaRLA will commit to B and learn $h(A) \leftarrow h(B) + 1$, then from B it will commit to A and learn $h(B) \leftarrow h(A) + 1$. This ping-pong effect (Edelkamp and Eckerle 1997; Sturtevant and Bulitko 2016) is an illustration of how the agent may need to update A $k/2$ times before the heuristic is locally optimal on the set $\{A, B, C\}$ and the agent will commit to C rather than B. Specifically, the set of paths is $P = \{\langle A \rangle, \langle A, B \rangle, \langle A, B, A \rangle \ldots \langle A, B, \ldots, B, A \rangle\}$, and the set of possible paths for learned h values is $P' = \{p \in P : |p| \leq k\}$, which is finite. Each time we ping-pong from A to B and back, the two shortest remaining paths in P' are eliminated and CaRLA can no longer base h values on them.

Lemma 25. *After a finite number of sweeps of $S_\circ$, CaRLA has learned a locally optimal h on $S_\circ$.*

Proof. We know h is admissible after a finite number of sweeps by Corollary 24, so consider those sweeps to have been completed and h to be admissible.

We will consider a state to be labeled *exact* at the end of a learning stage when we have proven that its h value is locally optimal and will not change in the future (similar to the labeling used in Labeled RTDP, (Bonet and Geffner 2003)). We will inductively prove that $S_\circ$ will be correctly labeled exact. The base case for our induction is the fringe of $S_\circ$. The fringe states are not in $S_\circ$ so it is a vacuous truth that they are locally optimal on $S_\circ$. They will never be expanded, so their h will never change and it is correct to label them *exact*.

Now for the inductive step, assume that all states currently marked exact are locally optimal. Consider the subset of $S_\circ$ that has not been marked *exact* $S_{ie} \leftarrow \{s \in S_\circ : \neg \textit{exact}(s)\}$ and a least h state $s_{\min} \in S_{ie}$, i.e., $\forall s \in S_{ie} : h(s_{\min}) \preceq h(s)$. We will show that either $s_{\min}$ will be marked exact and therefore removed from S_{ie} or $h(s_{\min})$ will increase. A least child of $s_{\min}$ is s_c where (because CaRLA's learning stage is thorough) $h(s_{\min}) = \omega(s_{\min}, s_c) \times h(s_c)$. There are

two cases to consider: if s_c is marked *exact* or not. If s_c is marked *exact*, then $s_{\min}$ can also be marked *exact* because there is no state in S_{ie} with a lower h for $s_{\min}$ to learn from and $h(s_c)$ is locally optimal and will not change. However, if s_c is not labeled *exact*, we still know that for all descendants d of $s_{\min}$ not marked *exact*, $h(s) \preceq h(d)$, and by **A1** action costs are positive so no h based on paths from these descendants can lower $h(s_{\min})$. Therefore, $h(s_{\min})$ is monotonically increasing if its least child is not marked *exact*.

What remains is to show that it requires only a finite number of $s_{\min}$ raises from states not marked *exact* until h is locally optimal on $S_\circ$. Consider the set of all possible h values that can be learned by states in $S_\circ$. Each $h(s)$ for state s corresponds to an original $h_0(s')$ for some state s' and a path p from s to s', giving $h(s) = \omega(p) \times h_0(s')$. By **A2** S is finite so the set of states and the set of possible paths $p \in P_{adm}$ (with cost low enough to be the basis of an admissible h) are finite. Each time we increase $h(s_{\min})$, we are eliminating the path that $h(s_{\min})$ was based on from P_{adm}, along with any other paths with lower cost (because there is no state with lower h that could be based on them). Because P_{adm} is of finite size, if we continue to remove elements from it, it will eventually be empty. This implies that there are no more least cost children not marked *exact*, because we would have monotonically increased $h(s_{\min})$ so many times that there are no possible paths in P_{adm} to base a not locally optimal heuristic on. Because all least cost children are now marked *exact*, the first case applies and we can correctly mark $s_{\min}$ exact. This completes the inductive step. Therefore, after a finite number of sweeps, h is locally optimal on $S_\circ$. □

In summary, we have shown that CaRLA will learn a locally admissible heuristic on the circulating set, with the local admissibility growing layer by layer from the fringe of the circulating set. Once the heuristic is admissible, the remaining task is to fill in heuristic depressions. Generally this will progress bottom up in each local minimum as illustrated in the ping-pong example (Figure 3b). Eventually this will lead to a locally optimal heuristic on the circulating set, and because CaRLA is productive it will eventually leave the circulating set in search of lower h.

Theorem 26. *CaRLA is complete.*

Proof. Consider, in contradiction, a CaRLA search that fails to find a goal. It must have a circulating set $S_\circ$ (Lemma 14), and after a finite number of steps the h will be locally optimal on $S_\circ$ (Lemma 25). Consider the root of an iteration of search. Because the heuristic is locally optimal and because the commitment strategy is productive, the best looking path found by the search will lead towards a lower h state s'. Any state $s \in S_\circ$ with $h(s') \prec h(s)$ will not be visited again until its h is lowered, something that can not happen unless the agent leaves the circulating set because h is locally optimal on $S_\circ$. Therefore, the set of states $\{s \in S_\circ : h(s) \prec h(s')\}$ with h sufficiently low to visit is monotonically decreasing in size, until it is empty. CaRLA then must commit to (or expand) a fringe state because the fringe and its descendants will be the only way to commit to a state with a lower h, which must exist (Lemma 12). This is a contradiction with the definition of the circulating set, because CaRLA will expand a state outside the circulating set. □

The contradiction arises because the circulating set can not exist as more than a transient phenomenon. Along with the generalization to the cost-algebra setting, the novelty of our approach to proving completeness is in the generality of the CaRLA framework, and in the clearer description of the process the heuristic goes through, first becoming admissible, then optimal by increasing the lowest states.

Prior work has either 1) assumed an admissible h_0 (Korf 1990; Koenig and Sun 2009; Edelkamp and Eckerle 1997), or 2) like Fickert et al. (2020) and Pemberton and Korf (1992) observed that each state is updated infinitely often and appealled to the convergence of asynchronous value iteration (Bertsekas 2012), or 3) not learned admissible heuristics (e.g. RTA* (Korf 1990) and weighted lateral LRTA* (Bulitko and Sampley 2016)).

Related Work

Heuristic depressions (Ishida 1992; Sturtevant and Bulitko 2016) are a key challenge for real-time heurisitc search. Two of the strategies to escape them more quickly are to do more search, and to learn more aggressively. Hernández and Baier (2012) describe how real-time heuristic search algorithms can be guided to avoid heuristic depressions, Hernandez et al. (2017) describe a pruning strategy to avoid re-exploring the same states in subsequent episodes of the search. In the related non-deterministic planning setting, Bonet and Geffner (2003) describe how to label and avoid re-exploring states whose value has already converged. Other approaches learn an inadmissible weighted heuristic that more quickly fills in the heuristic depression (Shimbo and Ishida 2003; Bulitko and Sampley 2016). Bulitko and Lee (2011) describe an alterrnative general framework of real-time heuritic search algorithms, however a weighted heuristic is a core component of their framework that does not generalize easily to all cost-algebras.

Conclusion

CaRLA is a generic framework for cost-algebraic real-time heuristic search. When used with a thorough learning strategy and a goal-aware and productive commitment strategy, CaRLA is complete. Our proofs illustrate how the search escapes heuristic depressions, first by learning an admissible heuristic, then by raising that admissible heuristic until the agent is forced out. This foundation clarifies the dependence of the proofs on domain and algorithm properties and will ease future applications of real-time planning.

Acknowledgments

Our thanks to Malte Helmert for suggesting this topic to us. We are grateful for support from the United States National Science Foundation via grant 2008594.

References

Bertsekas, D. 2012. *Dynamic Programming and optimal control: Volume II; Approximate Dynamic Programming*. Athena Scientific. ISBN 9781886529441.

Bonet, B.; and Geffner, H. 2003. Labeled RTDP: Improving the Convergence of Real-Time Dynamic Programming. In *Proceedings of ICAPS*.

Bulitko, V.; and Lee, G. 2011. Learning in Real-Time Search: A Unifying Framework. *Journal of Artificial Intelligence Research*.

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.; and Eckerle, J. 1997. New strategies in learning real time heuristic search. In *Proceedings of the AAAI Workshop on On-Line Search*.

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

Fickert, M.; Gu, T.; Staut, L.; Ruml, W.; Hoffmann, J.; and Patrik, M. 2020. Beliefs We Can Believe In: Replacing Assumptions with Data in Real-Time Search. In *Proceedings of AAAI*.

Hernández, C.; and Baier, J. A. 2012. Avoiding and escaping depressions in real-time heuristic search. *Journal of Artificial Intelligence Research*.

Hernandez, C.; Botea, A.; Baier, J. A.; and Bulitko, V. 2017. Online Bridged Pruning for Real-Time Search with Arbitrary Lookaheads. In *Proceedings of IJCAI*.

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

Ishida, T. 1992. Moving target search with intelligence. In *Proceedings of AAAI*.

Kiesel, S.; Burns, E.; and Ruml, W. 2015. Achieving goals quickly using real-time search: experimental results in video games. *Journal of Artificial Intelligence Research*.

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. *Journal of Artificial Intelligence*.

Pemberton, J.; and Korf, R. 1992. Making locally optimal decisions on graphs with cycles (Technical Report 920004). *Computer Science Department, University of California at Los Angeles*.

Shimbo, M.; and Ishida, T. 2003. Controlling the learning process of real-time heuristic search. *Journal of Artificial Intelligence*.

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

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*.