

Foundations of Artificial Intelligence

B14. State-Space Search: Properties of A*, Part I

Malte Helmert

University of Basel

March 25, 2026

Foundations of Artificial Intelligence

March 25, 2026 — B14. State-Space Search: Properties of A*, Part I

B14.1 Introduction

B14.2 Optimal Continuation Lemma

B14.3 f -Bound Lemma

B14.4 Optimality of A* with Reopening

B14.5 Summary

State-Space Search: Overview

Chapter overview: state-space search

- ▶ B1–B3. Foundations
- ▶ B4–B8. Basic Algorithms
- ▶ B9–B15. Heuristic Algorithms
 - ▶ B9. Heuristics
 - ▶ B10. Analysis of Heuristics
 - ▶ B11. Best-first Graph Search
 - ▶ B12. Greedy Best-first Search, A*, Weighted A*
 - ▶ B13. IDA*
 - ▶ B14. Properties of A*, Part I
 - ▶ B15. Properties of A*, Part II

B14.1 Introduction

Optimality of A*

- ▶ advantage of A* over greedy search:
optimal for heuristics with suitable properties
- ▶ very important result!

↪ next chapters: a closer look at A*

- ▶ A* with reopening ↪ this chapter
- ▶ A* without reopening ↪ next chapter

Optimality of A* with Reopening

In this chapter, we prove that A* with reopening is optimal when using admissible heuristics.

For this purpose, we

- ▶ give some basic definitions
- ▶ prove two lemmas regarding the behaviour of A*
- ▶ use these to prove the main result

Reminder: A* with Reopening

reminder from Chapter B11/B12: A* with reopening

A* with Reopening

```

open := new MinHeap ordered by ⟨f, h⟩
if h(init()) < ∞:
    open.insert(make_root_node())
distances := new HashMap
while not open.is_empty():
    n := open.pop_min()
    if distances.lookup(n.state) = none or g(n) < distances[n.state]:
        distances[n.state] := g(n)
        if is_goal(n.state):
            return extract_path(n)
        for each ⟨a, s'⟩ ∈ succ(n.state):
            if h(s') < ∞:
                n' := make_node(n, a, s')
                open.insert(n')
return unsolvable

```

Solvable States

Definition (solvable)

A state s of a state space is called solvable if $h^*(s) < \infty$.

German: lösbar

Optimal Paths to States

Definition (g^*)

Let s be a state of a state space with initial state s_1 .

We write $g^*(s)$ for the cost of an optimal (cheapest) path from s_1 to s (∞ if s is unreachable).

Remarks:

- ▶ g is defined for nodes, g^* for states (Why?)
- ▶ $g^*(n.state) \leq g(n)$ for all nodes n generated by a search algorithm (Why?)

Settled States in A*

Definition (settled)

A state s is called **settled** at a given point during the execution of A* (with or without reopening) if s is included in *distances* and $distances[s] = g^*(s)$.

German: erledigt

B14.2 Optimal Continuation Lemma

Optimal Continuation Lemma

We now show the first important result for A* with reopening:

Lemma (optimal continuation lemma)

Consider A* with reopening using a **safe** heuristic at the beginning of any iteration of the **while** loop.

If

- ▶ state s is settled,
- ▶ state s' is a solvable successor of s , and
- ▶ an optimal path from s_1 to s' of the form $\langle s_1, \dots, s, s' \rangle$ exists,

then

- ▶ s' is settled or
- ▶ open contains a node n' with $n'.state = s'$ and $g(n') = g^*(s')$.

German: Optimale-Fortsetzungs-Lemma

Optimal Continuation Lemma: Intuition

(Proof follows on the next slides.)

Intuitively, the lemma states:

If no optimal path to a given state has been found yet, open must contain a “good” node that contributes to finding an optimal path to that state.

(This potentially requires multiple applications of the lemma along an optimal path to the state.)

Optimal Continuation Lemma: Proof (1)

Proof.

Consider states s and s' with the given properties at the start of some iteration (“iteration A”) of A*.

Because s is settled, an earlier iteration (“iteration B”) set $\text{distances}[s] := g^*(s)$.

Thus iteration B removed a node n with $n.\text{state} = s$ and $g(n) = g^*(s)$ from *open*.

A* did not terminate in iteration B.
(Otherwise iteration A would not exist.)

Hence n was expanded in iteration B. ...

Optimal Continuation Lemma: Proof (2)

Proof (continued).

This expansion considered the successor s' of s .

Because s' is solvable, we have $h^*(s') < \infty$.

Because h is safe, this implies $h(s') < \infty$.

Hence a successor node n' was generated for s' .

This node n' satisfies the consequence of the lemma.

Hence the criteria of the lemma were satisfied for s and s' after iteration B.

To complete the proof, we show: if the consequence of the lemma is satisfied at the beginning of an iteration, it is also satisfied at the beginning of the next iteration. ...

Optimal Continuation Lemma: Proof (3)

Proof (continued).

- ▶ If s' is settled at the beginning of an iteration, it remains settled until termination.
- ▶ If s' is not yet settled and *open* contains a node n' with $n'.\text{state} = s'$ and $g(n') = g^*(s')$ at the beginning of an iteration, then either the node remains in *open* during the iteration, or n' is removed during the iteration and s' becomes settled. □

B14.3 f-Bound Lemma

f-Bound Lemma

We need a second lemma:

Lemma (f-bound lemma)

Consider A^* with reopening and an *admissible* heuristic applied to a *solvable* state space with optimal solution cost c^* .

Then *open* contains a node n with $f(n) \leq c^*$ at the beginning of each iteration of the **while** loop.

German: f-Schranken-Lemma

f-Bound Lemma: Proof (1)

Proof.

Consider the situation at the beginning of any iteration of the **while** loop.

Let $\langle s_0, \dots, s_n \rangle$ with $s_0 := s_1$ be an optimal solution. (Here we use that the state space is solvable.)

Let s_i be the first state in the sequence that is not settled.

(Not all states in the sequence can be settled: s_n is a goal state, and when a goal state is inserted into *distances*, A^* terminates.)

...

f-Bound Lemma: Proof (2)

Proof (continued).

Case 1: $i = 0$

Because $s_0 = s_1$ is not settled yet, we are at the first iteration of the **while** loop.

Because the state space is solvable and h is admissible, we have $h(s_0) < \infty$.

Hence *open* contains the root n_0 .

We obtain: $f(n_0) = g(n_0) + h(s_0) = 0 + h(s_0) \leq h^*(s_0) = c^*$, where " \leq " uses the admissibility of h .

This concludes the proof for this case.

...

f-Bound Lemma: Proof (3)

Proof (continued).

Case 2: $i > 0$

Then s_{i-1} is settled and s_i is not settled.

Moreover, s_i is a solvable successor of s_{i-1} and $\langle s_0, \dots, s_{i-1}, s_i \rangle$ is an optimal path from s_0 to s_i .

We can hence apply the optimal continuation lemma (with $s = s_{i-1}$ and $s' = s_i$) and obtain:

(A) s_i is settled, or

(B) *open* contains n' with $n'.state = s_i$ and $g(n') = g^*(s_i)$.

Because (A) is false, (B) must be true.

We conclude: *open* contains n' with

$f(n') = g(n') + h(s_i) = g^*(s_i) + h(s_i) \leq g^*(s_i) + h^*(s_i) = c^*$,
where " \leq " uses the admissibility of h . □

B14.4 Optimality of A* with Reopening

Optimality of A* with Reopening

We can now show the main result of this chapter:

Theorem (optimality of A* with reopening)

A with reopening is optimal when using an admissible heuristic.*

Optimality of A* with Reopening: Proof

Proof.

By contradiction: assume that the theorem is wrong.

Hence there is a state space with optimal solution cost c^* where A* with reopening and an admissible heuristic returns a solution with cost $c > c^*$.

This means that in the last iteration, the algorithm removes a node n with $g(n) = c > c^*$ from *open*.

With $h(n.state) = 0$ (because h is admissible and hence goal-aware), this implies:

$$f(n) = g(n) + h(n.state) = g(n) + 0 = g(n) = c > c^*.$$

A* always removes a node n with minimal f value from *open*.

With $f(n) > c^*$, we get a contradiction to the f -bound lemma, which completes the proof. □

B14.5 Summary

Summary

- ▶ **A* with reopening** using an **admissible** heuristic is optimal.
- ▶ The proof is based on the following lemmas that hold for solvable state spaces and admissible heuristics:
 - ▶ **optimal continuation lemma**: The open list always contains nodes that make progress towards an optimal solution.
 - ▶ **f-bound lemma**: The minimum f value in the open list at the beginning of each A* iteration is a lower bound on the optimal solution cost.