Chapter S:V

V. Formal Properties of A*

- Properties of Search Space Graphs
- Auxiliary Concepts
- Roadmap

- Completeness of A*
- Admissibility of A*
- Efficiency of A*
- Monotone Heuristic Functions

Remarks (outline) :

- Task: Find a cheapest path from s to some node $\gamma \in \Gamma$.

- Heuristic methods are often characterized as unpredictable:
  - They work wonders most of the time.
  - They may fail miserably some of the time.

- Using simple tests on the heuristic function h we can guarantee that
  - A* will find an optimum solution path, that
  - a heuristic function h_1 entails a higher efficiency in A* search than another heuristic function h_2, and that
  - A* will never reopen nodes on CLOSED.

Properties of Search Space Graphs

Properties for Node Expansion

Required properties of G for node expansion as basic step, *Prop*$_0(G)$:

1. G is a state-space graph (directed OR graph).
2. G is implicitly defined.
   (a) G has a single start state s and
   (b) G has a computable function *successors*$(.)$ returning successor states of the state given as argument.
3. G is locally finite.
4. A set Γ of goal states in G is given; in general Γ will not be a singleton set. Goal states are terminal states (states without successors) in G.
5. G has a computable function $\star(.)$ returning true if a given state is a goal state.

Properties of Search Space Graphs

Additional Required Properties

Additional required properties for best-first search, *Prop*$_1(G)$:

1. Evaluation function f is defined for G and assigns cost values to paths in G starting in s.
2. f is computable.
3. When f evaluates a solution bases P_{s-n}, the computed value does not depend on the time of computation.
4. When f evaluates a solution bases P_{s-n}, f estimates optimum cost of solution paths that have P_{s-n} as initial part.
5. A most promising solution base has a minimum f-value in a candidate set.

Properties of Search Space Graphs

Additional Required Properties

Additional required properties for best-first search, *Prop*$_1(G)$:

1. Evaluation function f is defined for G and assigns cost values to paths in G starting in s.
2. f is computable.
3. When f evaluates a solution bases P_{s-n}, the computed value does not depend on the time of computation.
4. When f evaluates a solution bases P_{s-n}, f estimates optimum cost of solution paths that have P_{s-n} as initial part.
5. A most promising solution base has a minimum f-value in a candidate set.

Additional required properties for solving optimization problems with BF*, *Prop*$_{BF}(G)$:

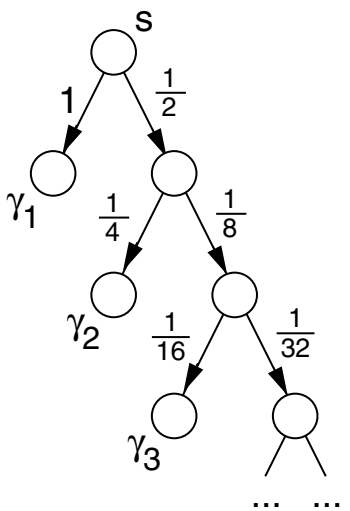
1. f is cycle-avers. (Avoiding corrupted backpointer structures.)
2. f is order-preserving. (Avoiding path discarding problems.)

Properties of Search Space Graphs

Non-Existence of Optimum Solution Paths

A search space graph may have solution paths, but no optimum solution path.

Example:

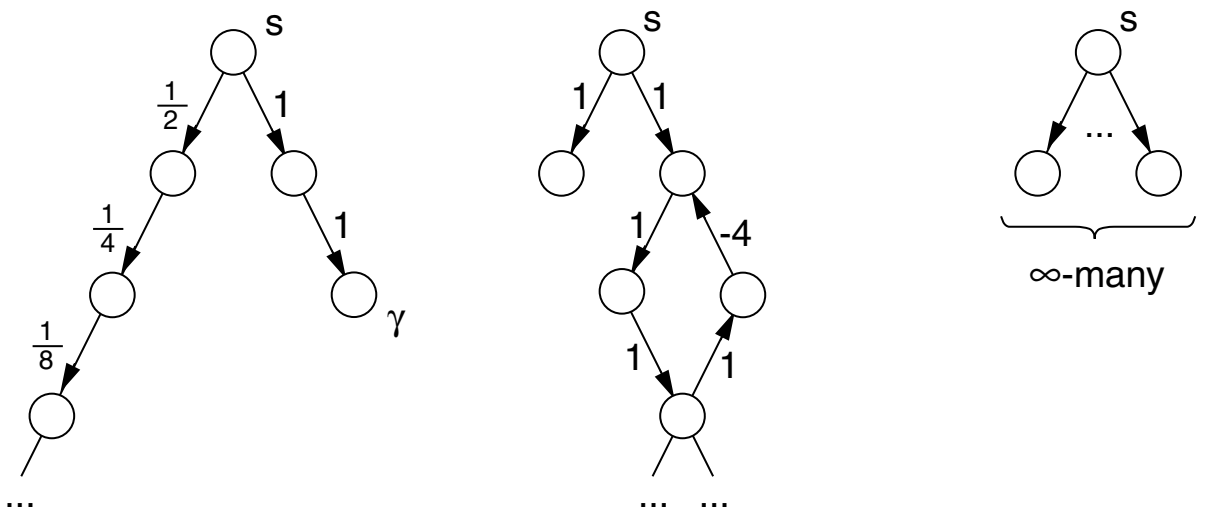


Observe that for each solution path a cheaper solution path can be found.

Properties of Search Space Graphs

Problems in the Search for Optimum Solution Paths

A* search is not guaranteed to find solutions in every problem setting.



Infinite path

Cycle with negative weight

Infinite number of successors

What are sufficient conditions to prove that A* will find an optimum solution?

Properties of Search Space Graphs

Prop$_{A^*}(G)$: Required Properties of G for A* Search

1. G has *Prop*$_0(G)$ properties.
2. The evaluation function f is based on the recursive cost function that computes cost of a path as sum cost of its edges.
3. A computable heuristic function h gives for each node n in G a heuristic estimate $h(n)$ of the cheapest path cost from n to Γ.
4. Every edge (n, n') in G has nonnegative cost $c(n, n')$.
   Therefore, we assume $h(n) \geq 0$ for all nodes $n \in G$ and $h(\gamma) = 0$ for $\gamma \in \Gamma$.
5. G has a positive lower bound δ of edge costs.
   I.e., there is a fixed δ such that for each edge $(n, n') \in G$ holds $c(n, n') \geq \delta > 0$.

2. + 3.	$\Rightarrow$	$f = g + h$ satisfies *Prop*$_1(G)$.
2. + 3.	$\Rightarrow$	$f = g + h$ is order-preserving.
4.	$\Rightarrow$	$f = g + h$ is cycle-avers.
5.	$\Rightarrow$	Existence of optimum solution paths w.r.t. $f = g + h$.
5.	$\Rightarrow$	Completeness of A* w.r.t. $f = g + h$.

Remarks:

- In the context of A* search we do not consider further solution constraints, i.e., each path in G from s to a node $\gamma \in \Gamma$ is a solution path.
- The positive lower bound δ on edge cost values is chosen for the entire graph G.
- The existence of δ in Property 5 implies that the sum cost of a path can exceed any given bound — if the path is long enough.

  Property 5 can be replaced by a requirement that for each bound B there is a length bound $l(B)$ such that all paths longer than $l(B)$ have at least cost B.
- Of course, a positive lower bound of edge costs entails that every edge has nonnegative cost. However, Property 4 already guarantees the pruning of cyclic paths.

Properties of Search Space Graphs

Existence of Optimum Solution Path

Lemma 54 (Path Existence Entails Optimum)

Let G be a search space graph with *Prop*$_{A^*}(G)$. If there is a path in G from node n to node n' in G, then there is also a cheapest path from n to n' in G.

Properties of Search Space Graphs

Existence of Optimum Solution Path

Lemma 54 (Path Existence Entails Optimum)

Let G be a search space graph with *Prop*$_{A^*}(G)$. If there is a path in G from node n to node n' in G, then there is also a cheapest path from n to n' in G.

Proof (sketch)

1. Let P be a path from n to n' in G with path cost C.
2. P has at most $\lceil \frac{C}{\delta} \rceil$ edges since each edge on P contributes at least δ, with $\delta > 0$.
3. Paths with more than $\lceil \frac{C}{\delta} \rceil$ edges have a path cost value higher than C.
4. A path starting from n with more than $\lceil \frac{C}{\delta} \rceil$ edges has higher path cost than P.
5. The number of paths in G starting from n with a given length $l \geq 0$ is finite.
   (Proof by induction using local finiteness of G.)
6. The number of paths starting from n with a length bound by $\lceil \frac{C}{\delta} \rceil$ is finite.
7. From this finite set we can select all those paths from n to n', including P.
8. Among this selection there is a path P^* from n to n' with minimum cost.
9. P^* is a cheapest path from n to n' in G since all path lengths have been considered.

Properties of Search Space Graphs

Existence of Optimum Solution Path (continued)

Corollary 55 (Solution Existence Entails Optimum)

Let G be defined as before. If there is a *solution path* in G, then there is also an optimum solution path in G.

Proof (sketch)

1. The proof is analogous to that of the previous lemma.
2. Starting point is an existing solution path $P_{s-\gamma}$ in G with path cost C.
3. Select in Step 7 of the previous proof all paths from s to nodes in Γ, especially $P_{s-\gamma}$.
4. The cheapest path among these is an optimum solution path in G.

If there is a solution path in G with *Prop*$_{A^*}(G)$, then the cheapest cost C^* of a solution path is uniquely determined. There can be more than one optimum solution path.

Chapter S:V

V. Formal Properties of A*

- Properties of Search Space Graphs
- Auxiliary Concepts
- Roadmap

- Completeness of A*
- Admissibility of A*
- Efficiency of A*
- Monotone Heuristic Functions

Auxiliary Concepts

Search Space Graph versus Traversal Tree

When searching a graph with algorithm BF:

1. Information on the explored part of that graph is maintained by BF in form of a subtree rooted in s, the traversal tree.
2. Discarding paths to a node does not affect completeness if solution paths can be constructed analogously for the remaining paths.
3. Pruning cycles does not affect admissibility if the cost of the path without cycle does not exceed the cost of the path with cycle.
4. Discarding more costly paths to a node does not affect admissibility if order-preserving cost measures are used.

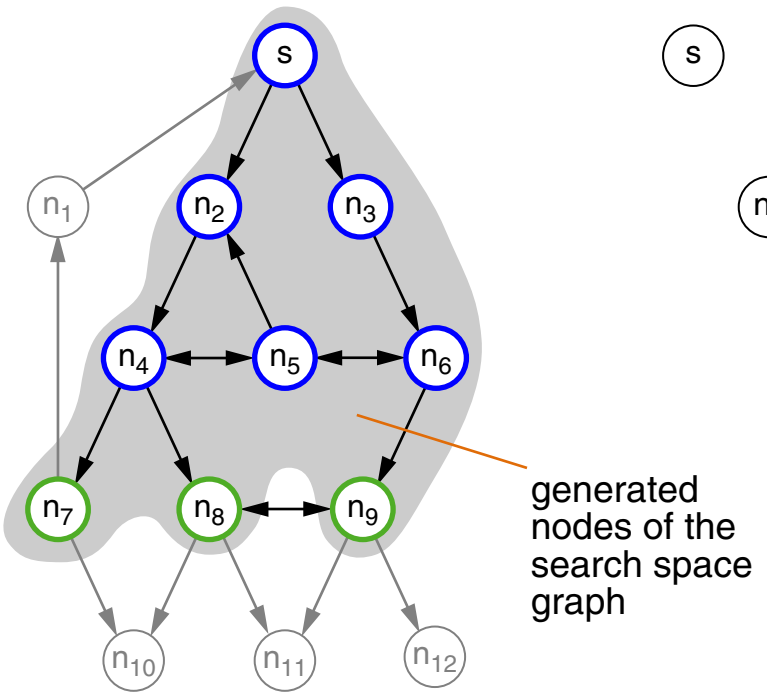
Remarks:

- Usually, a traversal tree will not contain all the information on the portion of G that has been explored by A* so far. By path discarding some of the explored edges will be lost. Additionally, nodes in CLOSED can be discarded if no back-pointer references to these nodes exist. In order to simplify proofs, we will assume that A* does not perform cleanup-CLOSED.
- A search space graph is defined by the problem (i.e., all possible states along with all possible operator applications). While a search space graph is constant, traversal trees develop and change while A* is running.

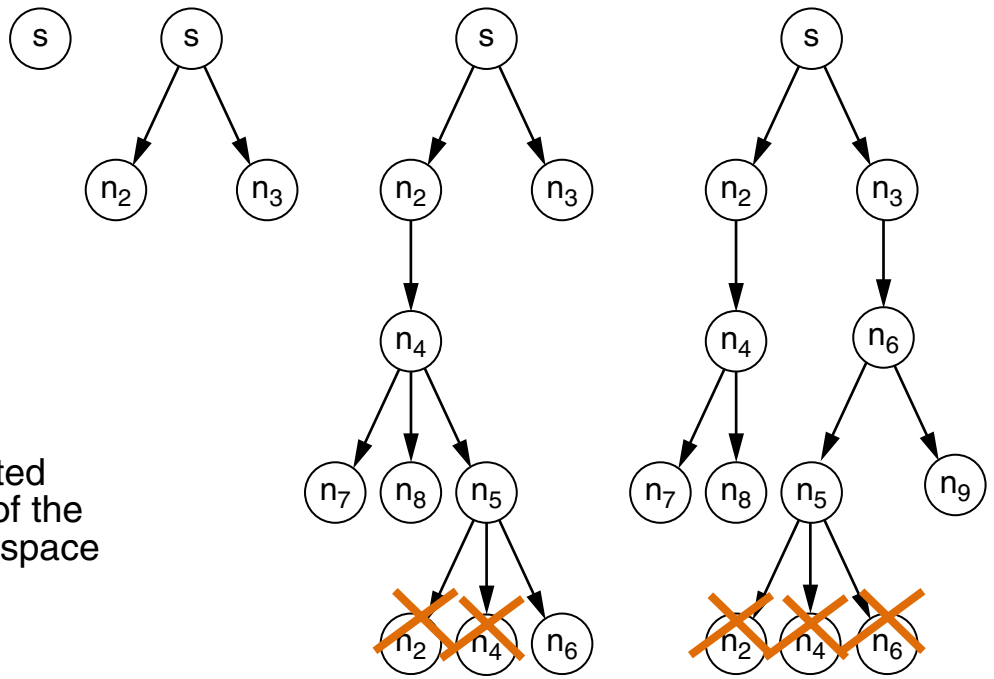
Auxiliary Concepts

Illustration of Traversal Trees

Search space graph:



Traversal trees at different points in time:



Remarks:

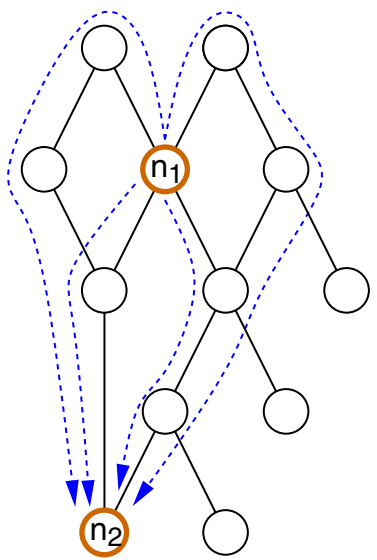
- When reopening a CLOSED node with A*, all its successor nodes in the traversal tree have invalid f-values stored with the nodes. Since A* uses an order-preserving evaluation function, no OPEN node among the successors of a reopened node in the traversal tree will be expanded using its invalid f-value; instead, the f-value will be corrected before expansion by a series of further reopening operations.
- Q. Which edge cost values and which h-values result in the traversal trees given in the illustration?

Auxiliary Concepts

Definition 56 (Specific Paths and Functions)

Let G be a search space graph with *Prop*$_{A^*}(G)$ and start node s, let Γ denote the set of all goal nodes in G, and let n_1, n_2 be nodes in G.

1. $P_{n_1-n_2}$ denotes a path from n_1 to n_2 in G.
2. $\mathbf{P}_{n_1-n_2}$ denotes the set of all paths from n_1 to n_2 in G.



Remarks:

- Notations:
  - P Single (mathematical) objects are usually denoted by simple symbols (normal, italic).
  - $\mathbf{P}$ Sets of (mathematical) objects are usually denoted by bold symbols (bold, normal).
  - $\mathcal{P}$ Systems consisting of (different mathematical) objects are usually denoted by calligraphic symbols.
- Although not shown in the drawing, $\mathbf{P}_{n_1-n_2}$ may contain paths with cycles.
- If node n_2 is not reachable from n_1, the set $\mathbf{P}_{n_1-n_2}$ is empty.
- Valid back-pointer paths are found by any best-first strategy that prunes cyclic paths; if the cost of the path without the cycle is higher than that of the path with the cycle, the resulting back-pointer-defined traversal tree structure built by BF* will be corrupt.
- At any stage of A* search the current traversal tree is the union of all back-pointer paths to nodes on OPEN.

Auxiliary Concepts

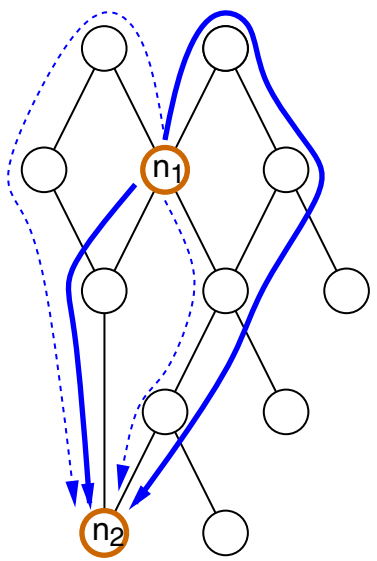
Definition 56 (Specific Paths and Functions (continued)**)**

Let G be a search space graph with *Prop*$_{A^*}(G)$ and start node s, let Γ denote the set of all goal nodes in G, and let n_1, n_2 be nodes in G.

3. $k(n_1, n_2)$ denotes the cost of a cheapest path from n_1 to n_2.

   $k(n_1, n_2) = \min\{C_P(n_1) \mid P$ *path from* n_1 *to* n_2 *in* $G\}$

4. $\mathbf{P}^*_{n_1-n_2}$ denotes the set of cheapest cost paths from n_1 to n_2 in G.



Remarks:

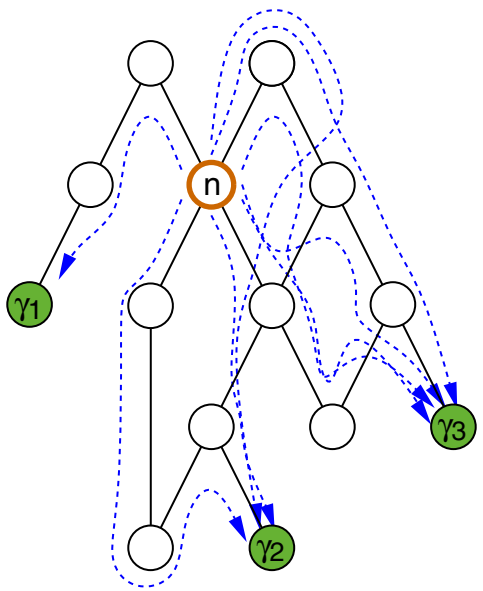
- If there is no path from n_1 to n_2, we define $k(n_1, n_2) := +\infty$.
- In search space graphs without positive lower bound of edge cost values, $k(n_1, n_2)$ has to be defined as the infimum of the cost values for paths in $\mathbf{P}_{n_1-n_2}$. Since we consider only search space graphs G with *Prop*$_{A^*}(G)$, a cheapest cost path exists between all pairs of connected nodes. Hence, $\mathbf{P}_{n_1-n_2} \neq \emptyset$ implies $\mathbf{P}^*_{n_1-n_2} \neq \emptyset$. [Lemma 54]
- For an edge (n, n') in G we obviously have $k(n, n') \leq c(n, n')$: There may be a path cheaper than the edge cost of (n, n').

Auxiliary Concepts

Definition 56 (Specific Paths and Functions (continued)**)**

Let G be ... Let n be a node in G.

5. $\mathbf{P}_{n-\Gamma}$ denotes the set of paths from n to a node in Γ in G.
6. $\mathbf{P}^*_{n-\Gamma}$ denotes the set of cheapest paths in G from n to a node in Γ.
7. $C^* = \min_{\gamma \in \Gamma} k(s, \gamma)$ denotes the cost of a cheapest path from s to a node in Γ.
8. Γ^* denotes the set of goal nodes that can be reached from s with cost C^*.



Remarks:

- If G is a search space graph with *Prop*$_{A^*}(G)$, then for each node $\gamma \in \Gamma$ that is reachable from n in G, there is a path from n to γ with cheapest cost $k(n, \gamma)$.
  Hence, $\mathbf{P}_{n-\Gamma} \neq \emptyset$ implies $\mathbf{P}^*_{n-\Gamma} \neq \emptyset$.
- Obviously, $\mathbf{P}^*_{s-\Gamma} = \mathbf{P}^*_{s-\Gamma^*}$ in all cases.

Auxiliary Concepts

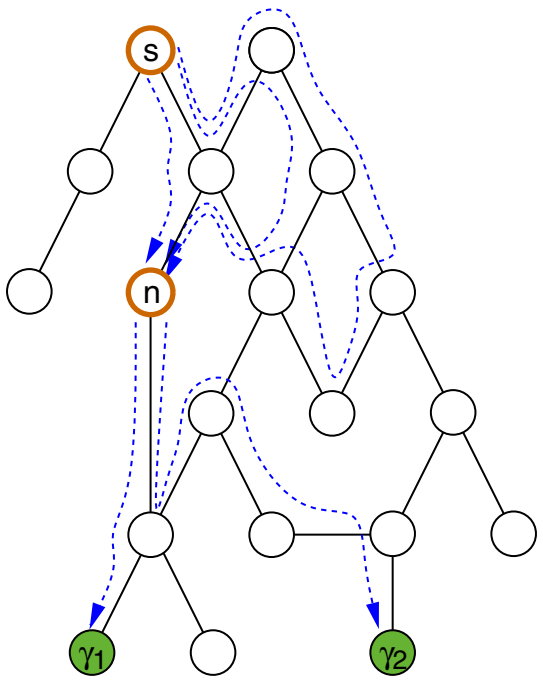
Definition 56 (Specific Paths and Functions (continued)**)**

Let G be ... Let P_{s-n} be a path in G and n' an intermediate node on P_{s-n}.

9. $g_{P_{s-n}}(n')$ denotes the path cost of the initial part of P_{s-n} up to n'.

Let G be ... Let $P_{n-\gamma}$ be a path in $\mathbf{P}_{n-\Gamma}$ and n' an intermediate node on $P_{n-\gamma}$.

10. $h_{P_{n-\gamma}}(n')$ denotes the path cost of the latter part of $P_{n-\gamma}$ from n' to γ.

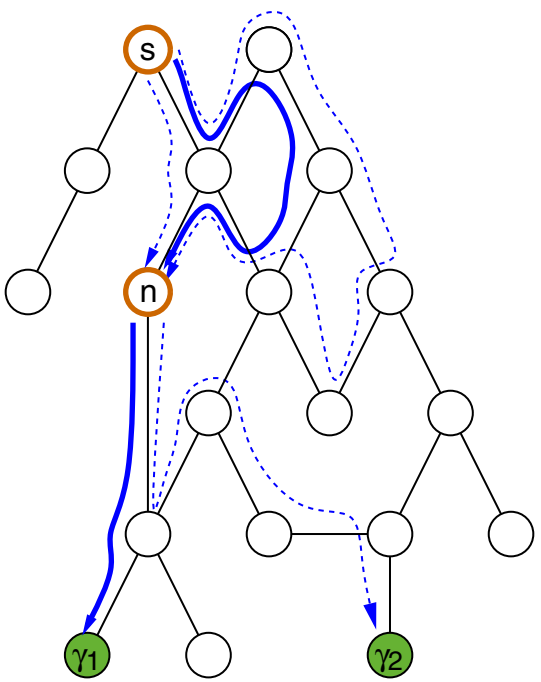


Auxiliary Concepts

Definition 56 (Specific Paths and Functions (continued)**)**

Let G be ... Let n be a node in G.

11. $g^*(n)$ denotes the cost of a cheapest path from s to n, i.e., $g^*(n) = k(s, n)$.
12. $h^*(n) = \min_{\gamma \in \Gamma} k(n, \gamma)$ denotes the cost of a cheapest path from n to Γ.
13. $f^*(n)$ denotes the optimum cost over all solution paths constrained to go through n, i.e., $f^*(n) = g^*(n) + h^*(n)$.



Remarks:

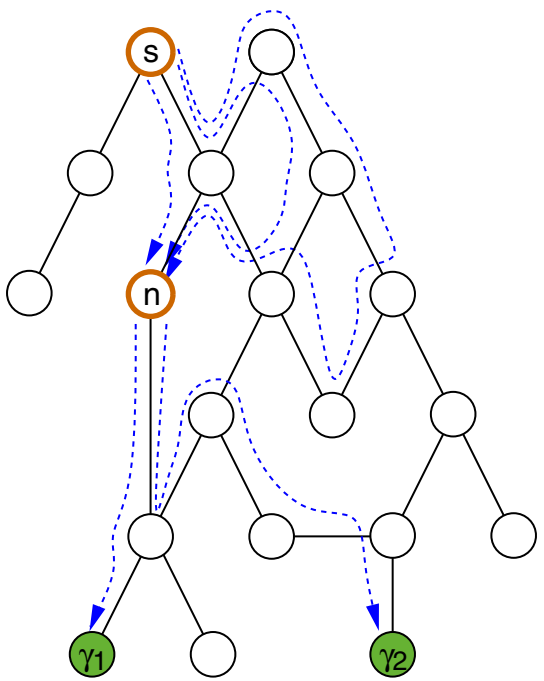
- If there is no path from s to n, then $g^*(n) = k(s,n) = +\infty$.
- $h^*(n)$ is no estimate but the cost value of paths in $\mathbf{P}^*_{n-\Gamma}$. Hence, $C^* = h^*(s)$.
- $\mathbf{P}_{s-n} \neq \emptyset$ implies $\mathbf{P}^*_{s-n} \neq \emptyset$.
- $\mathbf{P}_{n-\Gamma} \neq \emptyset$ implies $\mathbf{P}^*_{n-\Gamma} \neq \emptyset$.
- Paths in $\mathbf{P}^*_{s-n}$ have path cost $g^*(n)$, paths in $\mathbf{P}^*_{n-\Gamma}$ have path cost $h^*(n)$.

Auxiliary Concepts

Definition 56 (Specific Paths and Functions (continued)**)**

Let G be ... Let n be a node in G.

14. $g(n)$ denotes the cost of the current back-pointer path PP_{s-n} in A* search.
15. $h(n)$ denotes the estimated cheapest cost of paths in $\mathbf{P}_{n-\Gamma}$.
16. $f(n) = g(n) + h(n)$ denotes the current value of f for node n in A* search.



Remarks:

- During an A* search the values of $g(n)$ and $f(n)$ may decrease over time, whereas the values $h(n)$ are fixed for each node n.
- The current back-pointer path PP_{s-n} is the cheapest path from s to n that was found by A* so far.
- If PP_{s-n} is the current back-pointer path of a node n and all nodes in PP_{s-n} are in CLOSED except perhaps n, then $f(n) = g_{PP_{s-n}}(n) + h(n)$.
- $h(n)$ is used as an estimate for $h^*(n)$.
- $h(n)$ can be computed for all nodes, even if $\mathbf{P}_{n-\Gamma} = \emptyset$, i.e., if there are no paths from n to nodes in Γ.

Auxiliary Concepts

Lemma 57 (Basic Observations)

Let G be a search space graph with *Prop*$_{A^*}(G)$ and start node s, let Γ denote the set of all goal nodes in G, and let γ be a goal node in Γ.

Then we have:

1. $g(s) = g^*(s) = 0$
2. $h(\gamma) = h^*(\gamma) = 0$

For a solution path $P_{s-\gamma}$ with intermediate node n we have:

3. $g_{P_{s-\gamma}}(n) \geq g^*(n)$
4. $h_{P_{s-\gamma}}(n) \geq h^*(n)$
5. $g_{P_{s-\gamma}}(\gamma) = h_{P_{s-\gamma}}(s)$

For $\gamma \in \Gamma^*$ holds:

6. $f^*(s) = \underbrace{g^*(s)}_{0} + h^*(s) = h^*(s) = C^* = g^*(\gamma) = g^*(\gamma) + \underbrace{h^*(\gamma)}_{0} = f^*(\gamma)$

Remarks:

- Q. Consider Point 6 in Lemma 57. What is the difference between $f^*(n)$ for an intermediate node $n \notin \{s, \gamma\}$, compared to $f^*(s)$ or $f^*(\gamma)$?
- Q. For which nodes $n \in V$ holds $f^*(n) = C^*$?

Chapter S:V

V. Formal Properties of A*

- Properties of Search Space Graphs
- Auxiliary Concepts
- Roadmap

- Completeness of A*
- Admissibility of A*
- Efficiency of A*
- Monotone Heuristic Functions

Roadmap

Important Lemmas and Theorems

Termination of A*

↓

Completeness of A* ← Lemma "Shallowest OPEN Node"

↓

Admissibility of A* ← Lemma "Shallowest OPEN Node on Optimum Path"

Lemma "C*-Bounded OPEN Node"

↓

Efficiency of A*

Chapter S:V

V. Formal Properties of A*

- Properties of Search Space Graphs
- Auxiliary Concepts
- Roadmap

- Completeness of A*
- Admissibility of A*
- Efficiency of A*
- Monotone Heuristic Functions

Completeness of A*

Two important concepts for algorithms with regard to the returned solutions are completeness and admissibility.

Recall:

- Definition of Completeness:
  An algorithm is complete if it terminates with a solution if a solution exists.

- Definition of Admissibility:
  An algorithm is admissible if it terminates with an optimum solution if a solution exists.

For OR-graphs solutions are solution paths, for AND/OR-graphs solutions are solution graphs. The above definitions apply to search algorithms for both types of graphs.

Remarks:

- The definition of admissibility does not consider the existence of an optimum solution. Existence is implied by the search space graph properties $Prop_{A^*}(G)$ for the case that a solution path exists at all. [Corollary 55]
- Instead of “admissible” we may also use the phrase “optimum finding”.

Completeness of A*

Termination

Lemma 58 (Termination on Finite Graph)

A* terminates on finite graphs G that have *Prop*$_{A^*}(G)$.

Completeness of A*

Termination

Lemma 58 (Termination on Finite Graph)

A* terminates on finite graphs G that have *Prop*$_{A^*}(G)$.

Proof (sketch)

1. The number of cycle-free paths in a finite graph is finite.
2. Due to the positive edge costs A* will prune cyclic paths.
3. When A* expands a node, new nodes may be added to OPEN or not.
4. If a node n is added to OPEN, a new back-pointer path from s to n is used.

   (This fact is obvious when a node is reached for the first time. However, a new (back-pointer) path is also considered if a node on CLOSED is reopened or if a node on OPEN is updated. The latter fact is not needed in the proof.)
5. A* never finds a back-pointer path twice, since A* reopens a node on CLOSED (or updates a node on OPEN) only if it finds a strictly cheaper path to it. Discarded back-pointer paths cannot be recovered.
6. At some point the reservoir of back-pointer paths is exhausted or OPEN is empty.
7. Only a finite number of node expansions can be performed by A*.

Remarks:

1. Since G has *Prop*$_{A^*}(G)$, the evaluation function f is cycle-avers. Therefore, we only have to consider cycle-free paths. Cyclic paths might be considered in the FOREACH loop in A*, but such a solution base will never enter OPEN.

2. Obviously, the new back-pointer paths that are used in Point 4 are cycle-free.

3. The statement of the above lemma is true also for search space graphs with non-negative edge cost values, or even more general for search space graphs with non-negative cycle costs. A* will still prune cyclic paths for such graphs.

4. Termination on finite graphs holds for all BF algorithms that prune cyclic paths, i.e., that use an evaluation function f which returns values for cyclic paths that are at least as high as the values for the corresponding acyclic path.

Completeness of A*

Shallowest OPEN Node

Definition 59 (Shallowest OPEN Node)

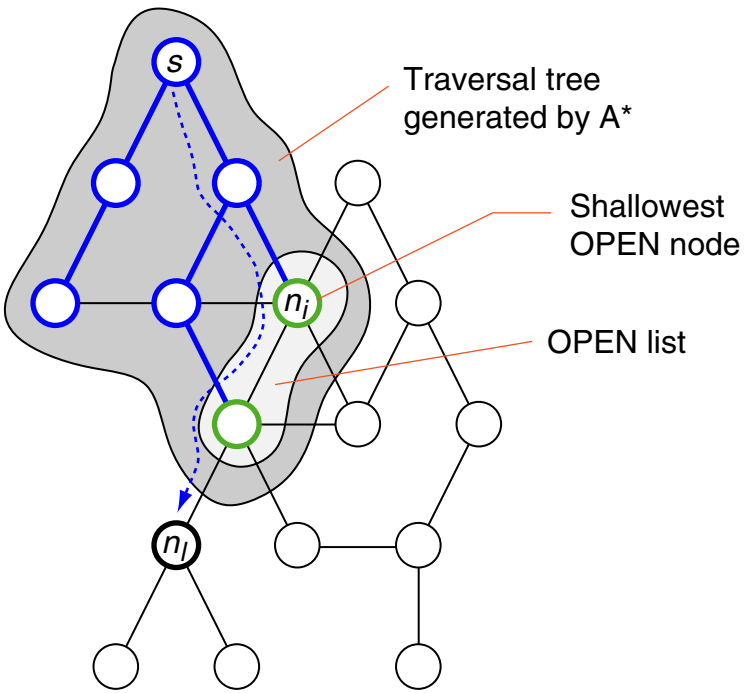
Let G be a search space graph with *Prop*$_{A^*}(G)$, $P = (n_0, n_1, n_2, \ldots, n_l)$ with $n_0 = s$, be an arbitrary path in G, and let G be processed by A*. The node n_i, $0 \leq i \leq l$, is the shallowest OPEN node on P iff ($\leftrightarrow$) n_i is on OPEN and none of the nodes $n_0, \ldots, n_{i-1}$ is on OPEN.

The shallowest OPEN node on a path P is the first OPEN node which we come across when following P starting from s.

To be precise we would need to define the shallowest OPEN node on P at some point in time, before A* terminates. A point in time is whenever A* (re)enters the main loop. At any point in time at least all nodes of an initial part of P are known to A*.

Completeness of A*

Shallowest OPEN Node (continued)



Node on OPEN

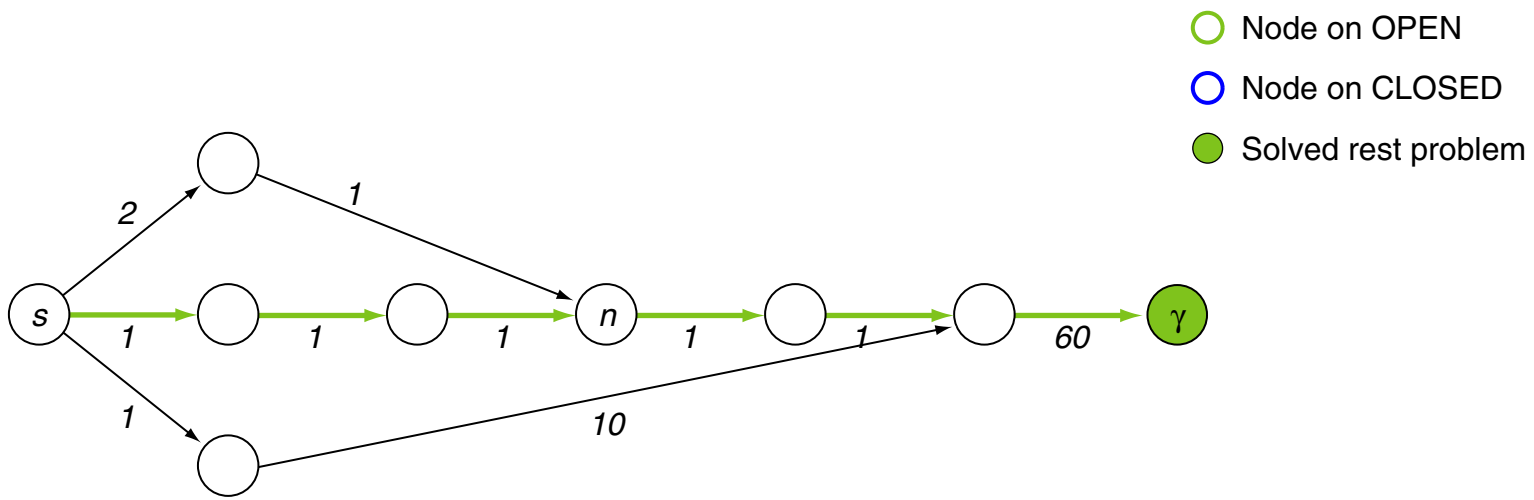
Node on CLOSED

Solved rest problem

Completeness of A*

Illustration of Shallowest OPEN Node

Consider the following search space graph G:



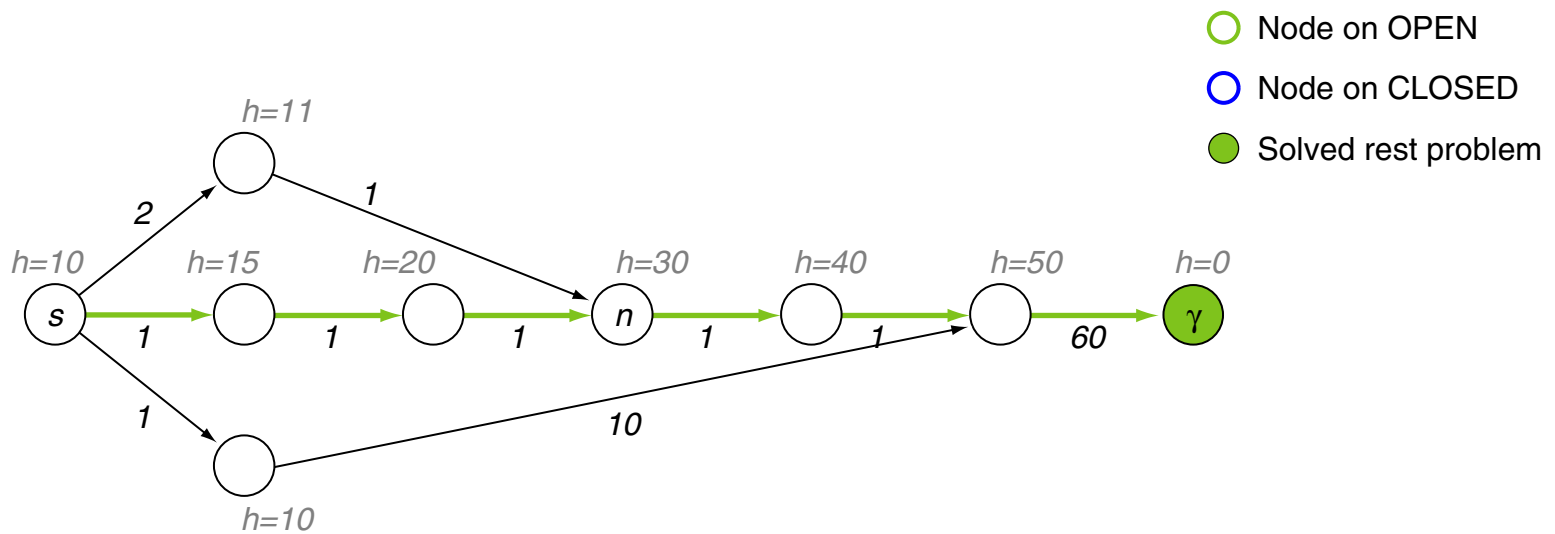
Distinguish between paths in G and back-pointer paths (found by A*) :

- Q. How does A* search this graph?
- Q. Which is the shallowest OPEN node in which path at which point in time?

Completeness of A*

Illustration of Shallowest OPEN Node

Consider the following search space graph G:



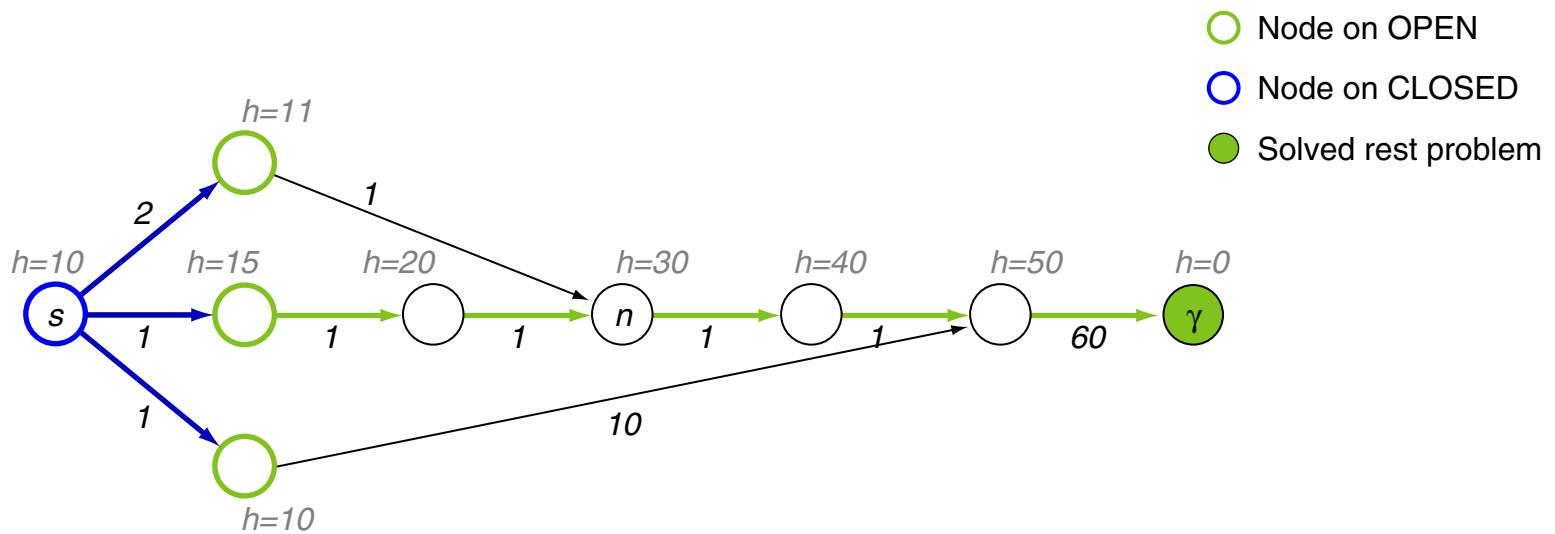
Distinguish between paths in G and back-pointer paths (found by A*) :

- Q. How does A* search this graph?
- Q. Which is the shallowest OPEN node in which path at which point in time?

Completeness of A*

Illustration of Shallowest OPEN Node

Consider the following search space graph G:



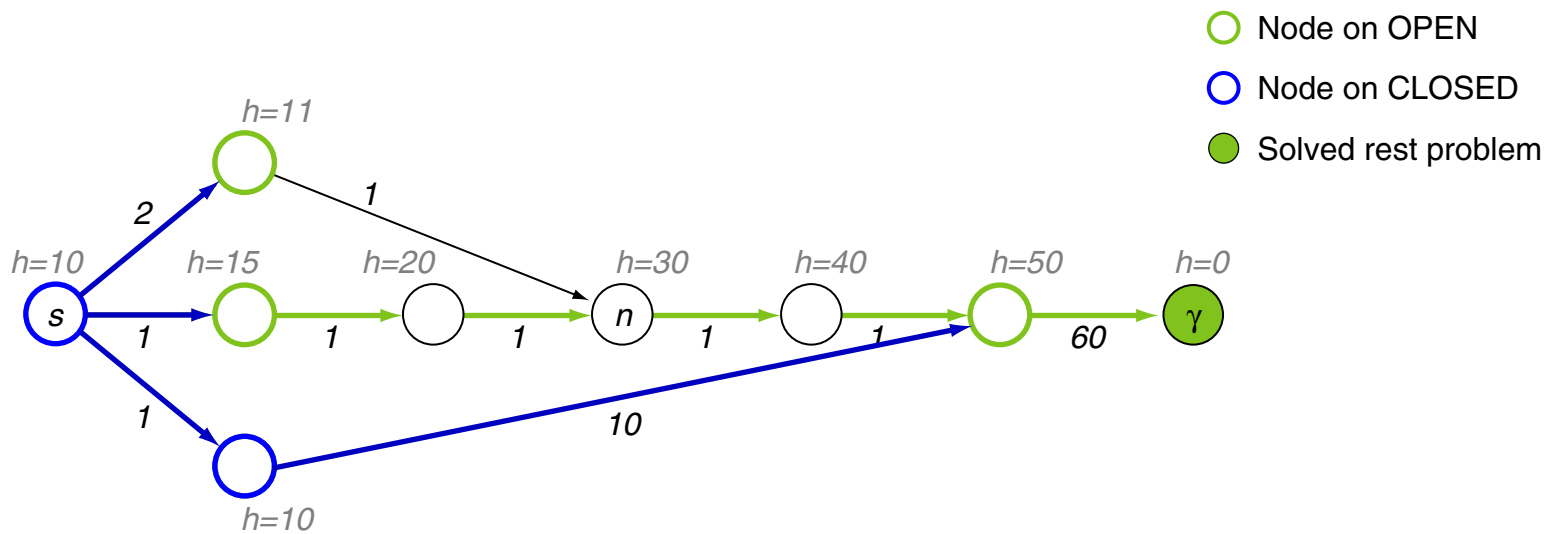
Distinguish between paths in G and back-pointer paths (found by A*) :

- Q. How does A* search this graph?
- Q. Which is the shallowest OPEN node in which path at which point in time?

Completeness of A*

Illustration of Shallowest OPEN Node

Consider the following search space graph G:



Distinguish between paths in G and back-pointer paths (found by A*) :

- Q. How does A* search this graph?
- Q. Which is the shallowest OPEN node in which path at which point in time?

Completeness of A*

Shallowest OPEN Node (continued)

Lemma 60 (Shallowest OPEN Node)★

Let G be a search space graph with *Prop*$_{A^*}(G)$ and let P_{s-n} be a path in G. Then at any point in time before A* terminates it holds:

If not all nodes on P_{s-n} are in CLOSED, then there is a shallowest OPEN node n' on P_{s-n} and all predecessors of n' on P_{s-n} are in CLOSED.

Completeness of A*

Shallowest OPEN Node (continued)

Lemma 60 (Shallowest OPEN Node)★

Let G be a search space graph with *Prop*$_{A^*}(G)$ and let P_{s-n} be a path in G. Then at any point in time before A* terminates it holds:

If not all nodes on P_{s-n} are in CLOSED, then there is a shallowest OPEN node n' on P_{s-n} and all predecessors of n' on P_{s-n} are in CLOSED.

Proof (sketch)

1. The basic observation is that whenever an OPEN node n' on P_{s-n} is moved to CLOSED, a successor node n'' of n' on P_{s-n} is added to OPEN if it was neither in OPEN nor in CLOSED.
2. So, at any point in time a CLOSED node on P_{s-n} (except n) is followed by a node on P_{s-n} which is either in OPEN or in CLOSED.
3. By induction this result can be extended to non-empty sequences of CLOSED nodes.
4. Therefore, at any point in time a CLOSED node is followed in P_{s-n} by a sequence of CLOSED nodes followed by an OPEN node in P_{s-n} or all following nodes in P_{s-n} are in CLOSED.
5. Initially, s on OPEN. s is shallowest OPEN node on P_{s-n} and s has no predecessors.
6. At all following points in time s is in CLOSED and the lemma follows from 4.

Remarks:

- ❑ In order to include the case that a function *cleanup_closed* is integrated into A*, we must use the clumsy formulation "n' has been selected for expansion at some previous point in time" instead of "n' is in CLOSED". Additionally, we have to take into consideration that nodes in CLOSED are discarded not before it has become clear that nodes in OPEN reachable by them are already reached by paths that are at least as cheap.

- ⋆ The elegance of this Lemma becomes obvious at second sight only: It states a property that holds for *all* paths that start with node s and that holds *always*.

- ⋆ The importance of this Lemma results from the fact that it rules out the “Fail”-case for A* if we consider a path $P_{s-\gamma}$: either we select the last node γ of $P_{s-\gamma}$ from OPEN or there “is still work to do” (= a predecessor node of γ on $P_{s-\gamma}$ is on OPEN).

Completeness of A*

Lemma 61 (Completeness for Finite Graph)

A* is complete for finite graphs G with $\mathit{Prop}_{A^*}(G)$.

Completeness of A*

Lemma 61 (Completeness for Finite Graph)

A* is complete for finite graphs G with *Prop*$_{A^*}(G)$.

Proof (sketch)

1. Assume that there is a solution path $P_{s-\gamma}$.
2. At any point before A* terminates there is a shallowest OPEN node on $P_{s-\gamma}$ or all nodes on $P_{s-\gamma}$ have been selected for expansion. [Lemma 60]
3. If there is a shallowest OPEN node on $P_{s-\gamma}$, A* will not terminate with “Fail”.
4. If all nodes on $P_{s-\gamma}$ have been selected for expansion, also γ has been selected for expansion and A* terminates with solution γ.

Remarks:

- Completeness of BF algorithms can be proven analogously if cycles are pruned.
- The proof uses the arguments in the proof of the above lemma for the special case of solution paths. [Lemma 60]

Completeness of A*

Lemma 62 (Shallowest OPEN Node on Path)★

Let G be a search space graph with *Prop*$_{A^*}(G)$ and let P_{s-n} be a path in G. Then at any point in time before A* terminates the following holds: If not all nodes in P_{s-n} are in CLOSED, then we have for the shallowest OPEN node n' on P_{s-n}

$$g(n') \leq g_{P_{s-n}}(n')$$

Completeness of A*

Lemma 62 (Shallowest OPEN Node on Path)★

Let G be a search space graph with *Prop*$_{A^*}(G)$ and let P_{s-n} be a path in G. Then at any point in time before A* terminates the following holds: If not all nodes in P_{s-n} are in CLOSED, then we have for the shallowest OPEN node n' on P_{s-n}

$$g(n') \leq g_{P_{s-n}}(n')$$

Proof (sketch)

1. If n' is the shallowest OPEN node on P_{s-n}, then all its predecessors on that path have been expanded and are in CLOSED.
2. For an arbitrary point in time before A* terminates we have:

   The successor of an initial sequence of CLOSED nodes on path P_{s-n} is reached with $g(n') \leq g_{P_{s-n}}(n')$ (i.e., the back-pointer path $PP_{s-n'}$ is at most as costly as P_{s-n} up to n').

   More formally:
   If $(s, n_1, \ldots, n_i, n_{i+1})$ is the initial part of P_{s-n} and $s, n_1, \ldots, n_i$ are in CLOSED, then it holds $g(n_{i+1}) \leq g_{P_{s-n}}(n_{i+1})$.

   Proof by induction.

Completeness of A*

Theorem 63 (Completeness)

A* is complete for infinite graphs G with $\mathit{Prop}_{A^*}(G)$.

Completeness of A*

Theorem 63 (Completeness)

A* is complete for infinite graphs G with *Prop*$_{A^*}(G)$.

Proof (sketch)

1. Assume that there is a solution path $P_{s-\gamma}$.
2. At any point before A* terminates there is always a shallowest OPEN node on $P_{s-\gamma}$, and hence A* will not terminate with "Fail". [Lemma 60]
3. For all nodes n on $P_{s-\gamma}$ there is a value $g_{P_{s-\gamma}}(n) + h(n)$. Define $M := \max_{n \in P_{s-\gamma}} (g_{P_{s-\gamma}}(n) + h(n))$.
4. At time t, the f-value of the shallowest OPEN node n_t on $P_{s-\gamma}$ is at most M, $f(n_t) \leq g_{P_{s-\gamma}}(n_t) + h(n_t) \leq M$. [Lemma 62]
5. A* will never expand a node n_M with $f(n_M) > M$, since all nodes on $P_{s-\gamma}$ including γ have to be expanded before.
6. The set of paths in G starting in s with path cost of at most M is finite. [Lemma 54, Point 6]
7. After finitely many node expansions, A* will choose a goal node (not necessarily γ) from OPEN and terminate with a solution.

Remarks:

- Within the proof we exploit the fact that path cost values of infinite paths are unbounded, i.e., for each bound B there is a length l_B such that all paths in G starting from s with length l_B have at least path cost B.
- In the proof of Theorem 63 the selection strategy of A* for nodes on OPEN is used. An analogous statement holds for algorithm A*$_\varepsilon$ which is also based on the evaluation function $f = g + h$. [Definition of A*$_\varepsilon$] In Steps 5 and 6 of an analogous proof for A*$_\varepsilon$ we would use $(1+\varepsilon)M$ instead of M.
- In the book of Pearl this theorem is denoted as Theorem 1. [Pearl 1984]

Chapter S:V

V. Formal Properties of A*

- Properties of Search Space Graphs
- Auxiliary Concepts
- Roadmap

- Completeness of A*
- Admissibility of A*
- Efficiency of A*
- Monotone Heuristic Functions

Admissibility of A*

Lemma 64 (Node Cost on Optimum Path)

For a search space graph G with *Prop*$_{A^*}(G)$ and a node n on some optimum path $P^*_{s-\gamma} \in \mathbf{P}^*_{s-\Gamma}$ in G holds:

$$f^*(n) = g^*(n) + h^*(n) = C^*$$

Admissibility of A*

Lemma 64 (Node Cost on Optimum Path)

For a search space graph G with *Prop*$_{A^*}(G)$ and a node n on some optimum path $P^*_{s-\gamma} \in \mathbf{P}^*_{s-\Gamma}$ in G holds:

$$f^*(n) = g^*(n) + h^*(n) = C^*$$

Proof (sketch)

1. Let $P^*_{s-\gamma} \in \mathbf{P}^*_{s-\Gamma}$ be an optimum solution path which contains n.
2. Therefore, $g_{P^*_{s-\gamma}}(n) + h_{P^*_{s-\gamma}}(n) = C^*$
3. Due to the optimality of g^* and h^* it holds that: $g^*(n) \leq g_{P^*_{s-\gamma}}(n)$ and $h^*(n) \leq h_{P^*_{s-\gamma}}(n)$
4. If we had $g_{P^*_{s-\gamma}}(n) > g^*(n)$ or $h_{P^*_{s-\gamma}}(n) > h^*(n)$, we could construct a cheaper path from s to γ, using the cheapest path from s to n and the cheapest path from n to γ. This would be a contradiction to $P^*_{s-\gamma} \in \mathbf{P}^*_{s-\Gamma}$.
5. Hence, $g^*(n) = g_{P^*_{s-\gamma}}(n)$ and $h^*(n) = h_{P^*_{s-\gamma}}(n)$ and we have $g^*(n) + h^*(n) = C^*$.

Remarks:

- In the book of Pearl this lemma is denoted as Equation 3.4. [Pearl 1984]

Admissibility of A*

Corollary 65 (Implications of Lemma 64)

Let G be a search space graph with *Prop*$_{A^*}(G)$.

1. If a node n is not contained in any optimum solution path, then $f^*(n) > C^*$.
2. If a path P is optimum, then every part of P is optimum.

Point 2 states that the search problem exhibits the principle of optimality (or optimum substructure) used in dynamic programming (Bellman).

The principle of optimality is fulfilled due to the fact that f^* can be defined recursively, using an additive (and thus order-preserving) cost measure.

This in turn means, that if f^* guides our search, A* search will never deviate from optimum paths. Unfortunately, f^* is not at our disposal.

Admissibility of A*

Definition 66 (Admissibility of h)

Let G be a search space graph with *Prop*$_{A^*}(G)$. A heuristic function h is called admissible iff ($\leftrightarrow$)

$$h(n) \leq h^*(n) \quad \text{for all } n \in G.$$

Thus an admissible heuristic function h provides an optimistic estimate of the cheapest solution cost for a node in G.

Similarly, the A* evaluation function $f = g + h$ with admissible h provides an optimistic estimate of the cheapest solution cost for s with respect to the current traversal tree.

Admissibility of A*

Corollary 67 (Shallowest OPEN Node on Optimum Path [Lemma 62])★

Let G be a search space graph with *Prop*$_{A^*}(G)$ and let P^*_{s-n} be an optimum path in G. Then at any point in time before A* terminates the following holds: If not all nodes in P^*_{s-n} are in CLOSED, then we have for the shallowest OPEN node n' on P^*_{s-n}

$$g(n') = g^*(n')$$

Admissibility of A*

Corollary 67 (Shallowest OPEN Node on Optimum Path [Lemma 62])★

Let G be a search space graph with *Prop*$_{A^*}(G)$ and let P^*_{s-n} be an optimum path in G. Then at any point in time before A* terminates the following holds: If not all nodes in P^*_{s-n} are in CLOSED, then we have for the shallowest OPEN node n' on P^*_{s-n}

$$g(n') = g^*(n')$$

Proof (sketch)

1. Let n' be the shallowest OPEN node on P^*_{s-n}.
2. Then we have $g(n') \leq g_{P^*_{s-n}}(n')$ [Lemma 62]
3. Since P^*_{s-n} is an optimum path, we have $g_{P^*_{s-n}}(n') = g^*(n')$.
4. Altogether we have $g(n') = g^*(n')$. [Lemma 57]
5. A* found an optimum back-pointer path $PP_{s-n'}$ to n' and $PP_{s-n'}$ will not be changed later.

Remarks:

- ⋆ The Lemma states that nodes on optimum paths are reached by A* with optimum cost when they become shallowest OPEN node on that path for the first time. Put another way, the cost of a shallowest OPEN node on an optimum path is afterwards never changed by A*.
- ❑ To better understand the Corollary, construct a search space graph with a traversal tree such that an optimum path $P^*_{s-\gamma}$ has more than one OPEN node. Hint: a node can be reached on some non-optimum path (long) before its predecessors on the optimum path are expanded.
- ❑ Q. Why does the path $PP_{s-n'}$ not always form a subpath of P^*_{s-n}?
- ❑ In the book of Pearl this lemma is denoted as Lemma 2. [Pearl 1984]

Admissibility of A*

Lemma 68 (C^*-Bounded OPEN Node)

Let G be a search space graph with *Prop*$_{A^*}(G)$ and let A* use some admissible heuristic function h. For each optimum path $P^*_{s-\gamma} \in \mathbf{P}^*_{s-\Gamma}$ and at each point in time before A* terminates there is an OPEN node n' on $P^*_{s-\gamma}$ with $f(n') \leq C^*$.

Admissibility of A*

Lemma 68 (C^*-Bounded OPEN Node)

Let G be a search space graph with *Prop*$_{A^*}(G)$ and let A* use some admissible heuristic function h. For each optimum path $P^*_{s-\gamma} \in \mathbf{P}^*_{s-\Gamma}$ and at each point in time before A* terminates there is an OPEN node n' on $P^*_{s-\gamma}$ with $f(n') \leq C^*$.

Proof (sketch)

1. Let $P^*_{s-\gamma} = (s, n_1, n_2, \ldots, n', \ldots, \gamma)$ be an optimum solution path, i.e., $P^*_{s-\gamma} \in \mathbf{P}^*_{s-\Gamma}$.
2. Since γ is a goal node, there is at any point in time a shallowest OPEN node n' on $P^*_{s-\gamma}$ before A* terminates. [Lemma 60]
3. n' is optimally reached by A*, i.e., $g(n') = g^*(n')$. [Corollary 67]
4. Using the admissibility of h we have
   $f(n') = g(n') + h(n') = g^*(n') + h(n') \leq g^*(n') + h^*(n') = f^*(n')$.
5. Since $n' \in P^*_{s-\gamma}$ we have $f^*(n') = C^*$. [Lemma 64]
6. Altogether we have $f(n') \leq C^*$.

Remarks:

- Since in the proof of Lemma 68 no selection strategy for nodes on OPEN is used, an analogous statement holds for BF* algorithms based on the evaluation function $f = g + h$, using a different selection strategy such as A*$_\varepsilon$. [Definition of A*$_\varepsilon$]
- In the book of Pearl this lemma is denoted as Lemma 1 and Nilsson Result 2 respectively. [Pearl 1984]

Admissibility of A*

Theorem 69 (Admissibility [Hart, Nilsson, Raphael 1968,1972]**)**

A* is admissible when using an admissible heuristic function h on search space graphs G with *Prop*$_{A^*}(G)$.

Admissibility of A*

Theorem 69 (Admissibility [Hart, Nilsson, Raphael 1968,1972]**)**

A* is admissible when using an admissible heuristic function h on search space graphs G with *Prop*$_{A^*}(G)$.

Proof (sketch)

1. Let there be a solution path in G, i.e. $\mathbf{P}^*_{s-\Gamma} \neq \emptyset$.
2. A* is complete and will terminate with a solution path.
3. Assume A* terminates returning a non-optimum goal node $\gamma \in \Gamma$ with $f(\gamma) = g(\gamma) > C^*$.
4. A* selected γ from OPEN.
5. Thus $f(n) \geq f(\gamma) > C^*$ for all $n \in$ OPEN.
6. This contradicts to Lemma 68 which states that there is an OPEN node n' with $f(n') \leq C^*$.

Remarks:

- This result holds for any BF* algorithm that uses an optimistic heuristic evaluation function f that is order-preserving and allows pruning of cyclic paths for search space graphs where path cost values of infinite paths are unbounded.
- In the book of Pearl this lemma is denoted as Theorem 2 and Nilsson Result 4 respectively. [Pearl 1984]