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

Efficiency of A*

Efficiency of Search Algorithms

The basic steps in the loop of a search algorithm are:

1. Select a most promising solution base.
2. Select a node in that solution base.
3. Expand that node.

→ Efficiency is related to the number of node expansions.

Heuristics influence efficiency by

1. excluding nodes from expansion entirely (pruning), and by
2. preventing nodes from being expanded more than once (no reopening)

Efficiency of A*

Efficiency of Search Algorithms (continued)

Definition 70 (Dominance, Optimality)

1. A search algorithm A_1 dominates a search algorithm A_2 if each node that is expanded by A_1 is also expanded by A_2.

2. A_1 strictly dominates A_2 if A_1 dominates A_2 and A_2 does not dominate A_1.

3. A search algorithm is optimum regarding a class of search algorithms if it dominates all members of this class.

Remarks:

- By "node expanded by algorithm A" we mean that all direct successor nodes are generated and processed. This is not the same as "node selected for expansion by algorithm A". Although algorithm A_1 dominates A_2, the latter algorithm may terminate returning a different goal node.

- Dominance defines a *partial* ordering on search algorithms.

- Dominance relations are often proved with regard to some fixed tie breaking rule.

- Instead of the term “dominates” we may also use the phrase “is more efficient than”.

Efficiency of A*

Efficiency of Search Algorithms

By not-expanding nodes, parts of the search space graph are pruned. The efficiency of A* depends on the accuracy of the heuristic estimate h.

Consider two extreme cases:

1. If estimates are perfect ($h = h^*$), A* will follow cheapest paths to goal nodes. In this case $f(n) = C^*$ holds for each expanded node n.
2. If no heuristic is used ($h = 0$), A* degenerates to a uniform cost search.
   In this case each node n reachable from s with path cost of at most C^* will be expanded, i.e., $f(n) = g(n) \leq C^*$.

Hypothesis:

The closer h is to h^* (as long as $h \leq h^*$), the more powerful it is with respect to pruning.

Efficiency of A*

Conditions for Node Expansion I

Theorem 71 (Necessary Condition for Node Expansion I)

Let A* process a search space graph G with $\mathit{Prop}_{A^*}(G)$, using an admissible heuristic function h. For each node n expanded by A* holds:

$$f(n) \leq C^*$$

Efficiency of A*

Conditions for Node Expansion I

Theorem 71 (Necessary Condition for Node Expansion I)

Let A* process a search space graph G with *Prop*$_{A^*}(G)$, using an admissible heuristic function h. For each node n expanded by A* holds:

$$f(n) \leq C^*$$

Proof (sketch)

1. If there is no solution path in G, then $C^* = +\infty$ and this theorem is obviously true.
2. If there is a solution path in G, then there is also an optimum solution path with cost C^*.
   [Corollary "Solution Existence Entails Optimum"]
3. At any time before A* terminates, there exists an OPEN node n' with $f(n') \leq C^*$.
   [Lemma "C^*-bounded OPEN Node"]
4. Since A* expands n, its f-value is less or equal to the f-value of all nodes on OPEN. Thus we have $f(n) \leq f(n')$ and therefore $f(n) \leq C^*$.

Remarks:

- Theorem 71 defines a set S of nodes that will not be expanded: $S = \{n \mid f(n) > C^*\}$.
  Q. Why might this knowledge be helpful and how could it be applied?

- The application of Theorem 71 requires knowledge on C^*.
  Q. Is this a problem?
  Q. If yes, how to solve it?

- In the book of Pearl this theorem is denoted as Theorem 3 and Nilsson Result 5 respectively.
  [Pearl 1984]

Efficiency of A*

Conditions for Node Expansion I

Theorem 72 (Sufficient Condition for Node Expansion I)

Let A* process a search space graph G with $\mathit{Prop}_{A^*}(G)$, using an admissible heuristic function h. A* will expand each node n on OPEN with $f(n) < C^*$.

Efficiency of A*

Conditions for Node Expansion I

Theorem 72 (Sufficient Condition for Node Expansion I)

Let A* process a search space graph G with *Prop*$_{A^*}(G)$, using an admissible heuristic function h. A* will expand each node n on OPEN with $f(n) < C^*$.

Proof (sketch)

1. Let OPEN contain a node n with $f(n) < C^*$.
2. If there is no solution path in G, then g-values determined for newly generated nodes are higher over time (because of lower bound δ on edge cost values). Therefore, n will be expanded at some point in time, even if G is an infinite graph.
3. Let A* terminate with goal node γ.
4. Due to the admissibility of A*, $f(\gamma) = C^*$.
5. From $f(n) < C^*$ and $C^* = f(\gamma)$ follows that $f(n) < f(\gamma)$.
6. The value of $f(n)$ can only decrease ($f(\gamma)$ remains constant).
7. A* expands the node n before it terminates with solution γ.

Remarks:

- In the book of Pearl this theorem is denoted as Theorem 4. [Pearl 1984]

Efficiency of A*

Conditions for Node Expansion I [Conditions II, Conditions III]

Let $n \in$ OPEN. For admissible heuristic functions h hold:

$$\text{node expansion by A*} \;\Rightarrow\; \underbrace{f(n) \leq C^*}_{\text{necessary}}$$

$$\underbrace{f(n) < C^*}_{\text{sufficient}} \;\Rightarrow\; \text{node expansion by A*}$$

Observe that the Theorems 71 and 72 do not give a condition for node expansion that is both necessary and sufficient:

In particular, for OPEN nodes with $f(n) = C^*$ some tie breaking rule has to determine which of these nodes are expanded or not.

Efficiency of A*

Conditions for Node Expansion I

Corollary 73 (Re-Expansion of Expanded Nodes)

Let A* process a search space graph G with *Prop*$_{A^*}(G)$, using an admissible heuristic function h. If a node n on CLOSED is reopened, then A* will expand node n again.

Proof (sketch)

1. For each node n on CLOSED, $f(n) \leq C^*$ is true (Theorem 71).
2. Due to reopening, the value of $f(n)$ decreases, $f(n) < C^*$.
3. Hence, A* expands the node n (Theorem 72).

Even worse, any node n' in CLOSED, which has n in its back-pointer path, i.e. $n \in PP_{s-n'}$, when n is reopened, will be reopened and expanded as well.

Efficiency of A*

Conditions for Node Expansion II

The expansion conditions I are *algorithm-centered*:

- The value of $g(n)$ is no property of node n but depends on the path to n that A* has discovered.
- Theorem 72 requires that n resides on OPEN before it is selected for expansion. Whether a given node enters OPEN depends not on n itself but on the behavior of A* while exploring the paths leading to n.

Instead of arguing about algorithmic concepts (discovered paths, OPEN list), expansion conditions should be formulated with regard to the search space graph G only.

Efficiency of A*

Conditions for Node Expansion II

Definition 74 (Cost-Bounded Paths)

A path P is called C-bounded iff ($\leftrightarrow$) for each node n on P holds

$$g_P(n) + h(n) \leq C.$$

P is called strictly C-bounded iff ($\leftrightarrow$) for each node n on P holds

$$g_P(n) + h(n) < C.$$

C-boundedness can be checked for each finite path in G—not only for paths considered by A*.

Efficiency of A*

Conditions for Node Expansion II

Theorem 75 (Necessary Condition for Node Expansion II)

Let A* process a search space graph G with *Prop*$_{A^*}(G)$, using an admissible heuristic function h. For each node n expanded by A*, the back-pointer path of n at the time of expansion is a C^*-bounded path from s to n in G.

Efficiency of A*

Conditions for Node Expansion II

Theorem 75 (Necessary Condition for Node Expansion II)

Let A* process a search space graph G with *Prop*$_{A^*}(G)$, using an admissible heuristic function h. For each node n expanded by A*, the back-pointer path of n at the time of expansion is a C^*-bounded path from s to n in G.

Proof (sketch)

1. Since A* expands n, the equation $f(n) = g(n) + h(n) \leq C^*$ holds. [Theorem 71]
2. When n is on OPEN, all predecessors n' of n on the current back-pointer path PP_{s-n} have been expanded before.
3. At time of expansion of such a node n' we also had $g'(n') + h(n') \leq C^*$, where g' denotes the path cost at the time when n' was expanded.
4. Pointer-paths are only changed when a cheaper path is found, i.e., g-values can only decrease.
5. Thus we have $f(n') = g(n') + h(n') \leq g'(n') + h(n') \leq C^*$.

Efficiency of A*

Conditions for Node Expansion II

Theorem 76 (Sufficient Condition for Node Expansion II)

Let A* process a search space graph G with *Prop*$_{A^*}(G)$, using an admissible heuristic function h. A* will expand each node n for which we have a strictly C^*-bounded path from s to n in G.

Efficiency of A*

Conditions for Node Expansion II

Theorem 76 (Sufficient Condition for Node Expansion II)

Let A* process a search space graph G with *Prop*$_{A^*}(G)$, using an admissible heuristic function h. A* will expand each node n for which we have a strictly C^*-bounded path from s to n in G.

Proof (sketch)

1. Let P be a strictly C^*-bounded path from s to n in G.
2. Let A* terminate with goal node γ.
3. Due to admissibility of A*, $f(\gamma) = C^*$.
4. Let n' be the shallowest OPEN node on P when A* terminates.
5. Since all predecessors of n' on P are on CLOSED, the path cost $g(n')$ currently maintained by A* cannot be higher than $g_P(n')$: $f(n') = g(n') + h(n') \leq g_P(n') + h(n')$
6. Since P is strictly C^*-bounded, we have $g_P(n') + h(n') < C^*$ and thus $f(n') < f(\gamma)$.
7. A* expands the node n' and all nodes on P before it terminates with solution γ.

Remarks:

- If there is no solution path in G, we have $C^* = +\infty$ and any node reachable from s will be expanded by A* using h.
- An optimum path to a node n may be C^*-bounded only, although a strictly C^*-bounded path exists.
- In the book of Pearl, the two previous theorems are denoted as Theorem 6 and Theorem 5 respectively. [Pearl 1984]

Efficiency of A*

Conditions for Node Expansion II [Conditions I, Conditions III]

For admissible heuristic functions h hold:

$$\text{node expansion by A*} \;\Rightarrow\; \underbrace{\exists C^*\text{-bounded path } P_{s-n}}_{\text{necessary}}$$

$$\underbrace{\exists \text{ strictly } C^*\text{-bounded path } P_{s-n}}_{\text{sufficient}} \;\Rightarrow\; \text{node expansion by A*}$$

Again, none of the Theorems 75 and 76 gives a condition for node expansion that is both necessary and sufficient:

The theorems do not treat C^*-bounded paths with $f(n') = C^*$ for some intermediate node n'.

Efficiency of A*

Informedness and Dominance

Definition 77 (Informedness)

A heuristic function h_2 is called more informed than a heuristic function h_1 iff ($\leftrightarrow$) both functions are admissible and if holds

$$h_2(n) > h_1(n), \qquad \forall n \in G, n \notin \Gamma$$

An A* algorithm that uses such a heuristic function h_2 is called more informed than an A* algorithm that uses h_1.

Efficiency of A*

Informedness and Dominance (continued)

Corollary 78 (Dominance)

Let G be a search space graph with *Prop*$_{A*}(G)$. If $h_2(n) > h_1(n)$ holds for any n in G with $n \notin \Gamma$, and if $h_2(n)$ and $h_1(n)$ both are admissible, then A*$_2$ (A* informed by $h_2(n)$) dominates A*$_1$ (A* informed by $h_1(n)$).

Efficiency of A*

Informedness and Dominance (continued)

Corollary 78 (Dominance)

Let G be a search space graph with *Prop*$_{A^*}(G)$. If $h_2(n) > h_1(n)$ holds for any n in G with $n \notin \Gamma$, and if $h_2(n)$ and $h_1(n)$ both are admissible, then A*$_2$ (A* informed by $h_2(n)$) dominates A*$_1$ (A* informed by $h_1(n)$).

Proof (sketch)

1. Let A*$_2$ expand a node n in G with $n \notin \Gamma$.
2. Thus the current back-pointer path P of n is a C^*-bounded path P from s to n with respect to h_2, i.e., $g_P(n') + h_2(n') \leq C^*$ for each n' on P. [Theorem 75]
3. Since no goal node is contained in P and h_2 is more informed than h_1, for each node n' on P holds $h_2(n') > h_1(n')$.
4. Hence $g_P(n') + h_1(n') < C^*$ for each n' on P .
5. Thus P is strictly C^*-bounded with respect to h_1.
6. A*$_1$ will expand n. [Theorem 76]

Remarks:

- The requirement of Definition 77 (Informedness), namely, the strict inequality between h_2 and h_1, is rarely satisfied—even in situations where h_2 is clearly superior to h_1.
  Example 8-Puzzle: h_1 (number of misplaced tiles) often equals h_2 (sum of Manhattan distances of misplaced tiles).

- In the book of Pearl this theorem is denoted as Theorem 7 and Nilsson Result 6 respectively.
  [Pearl 1984]

Efficiency of A*

Informedness and Dominance: Discussion

Q. If $h_2 > h_1$ cannot be satisfied in practice, then under which conditions does A*$_2$ (using h_2) dominate A*$_1$ (using h_1) when $h_2 \geq h_1$?

A1: If both algorithms use the same tie breaking rule and if the rule is purely structural (i.e., independent of the values of g and h).

A2: In situations where we want to find *all* goal nodes that can be reached with cheapest cost, Theorem 75 gives both a necessary and sufficient condition for node expansion.
In this case, Definition 77 (Informedness) can permit equalities without affecting Corollary 78.

A3: Theorem 85 demonstrates that under reasonable assumptions equalities between h_1 and h_2 only violate the dominance of A*$_2$ on a small set of nodes for which $g^*(n) + h_2(n) = C^*$ holds.

Remarks:

- For search space graphs G with *Prop*$_{A^*}(G)$ it is possible to find *all* goal nodes that can be reached with cheapest cost. But for search space graphs G with *Prop*$_{A^*}(G)$ it is in general not possible to find *all* optimum cost solution paths (path discarding for multiple optimum cost paths to the same node).

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

Monotone Heuristic Functions

Motivation

Previous consideration:

The efficiency of A* can be measured by the number of nodes it manages to exclude from expansion.

$$\text{maximize } |\underbrace{\{n \mid f(n) > C^*\}}_{S}|$$

More reasonable:

The number of expansion operations should be analyzed. Why?

Monotone Heuristic Functions

Motivation

Previous consideration:

The efficiency of A* can be measured by the number of nodes it manages to exclude from expansion.

$$\text{maximize } |\underbrace{\{n \mid f(n) > C^*\}}_{S}|$$

More reasonable:

The number of expansion operations should be analyzed. Why?

Starting point:

Under certain conditions A* will never reopen a node from CLOSED. What does that mean?

Monotone Heuristic Functions

Motivation

Recap: Cheapest paths to a goal, which are constrained to pass through a node n, cannot be cheaper than unrestricted cheapest paths to a goal.

Formally:

$$\forall\, n : \;\; h^*(s) \leq g^*(n) + h^*(n)$$

or: $$h^*(s) \leq k(s,n) + h^*(n), \quad \text{since } g^*(n) = k(s,n)$$

Monotone Heuristic Functions

Motivation

Recap: Cheapest paths to a goal, which are constrained to pass through a node n, cannot be cheaper than unrestricted cheapest paths to a goal.

Formally:

$$\forall\, n: \;\; h^*(s) \leq g^*(n) + h^*(n)$$

or: $$h^*(s) \leq k(s,n) + h^*(n), \quad \text{since } g^*(n) = k(s,n)$$

In general, the following "triangle inequality" holds:

$$\forall\, n, n': \quad h^*(n) \leq k(n,n') + h^*(n')$$

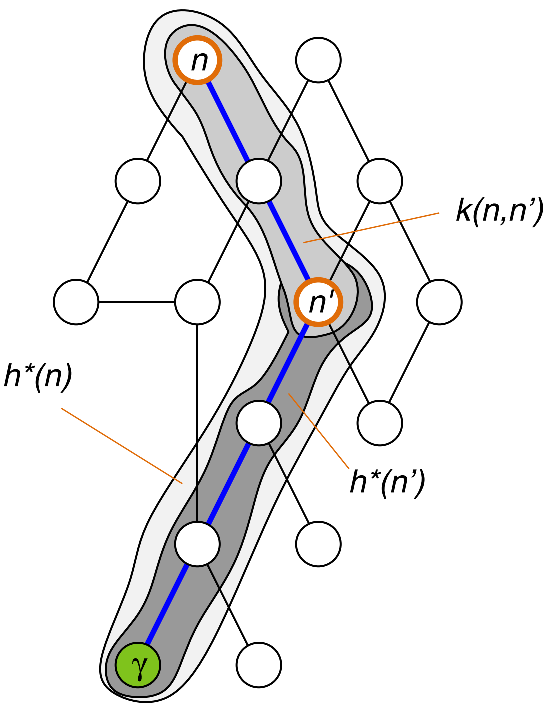
It is reasonable to expect that if the process of estimating $h(n)$ is "consistent", it should inherit the triangle inequality from h^*.

Monotone Heuristic Functions

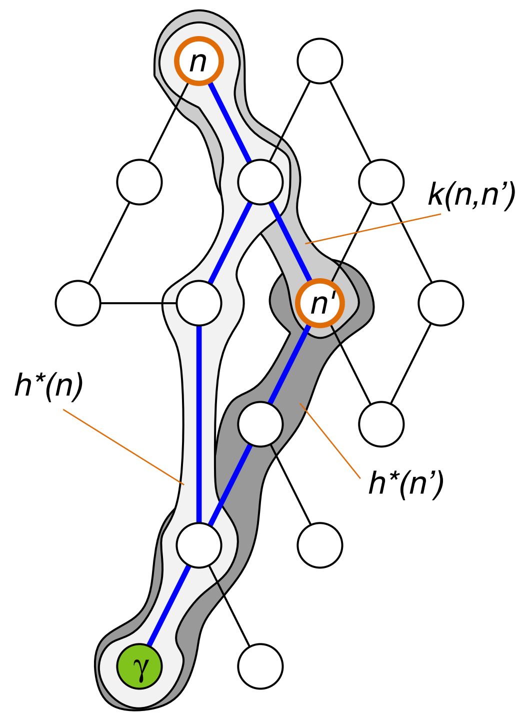
Illustration of the Global Triangle Inequality

$$\forall\ n, n' : \quad h^*(n) \leq k(n,n') + h^*(n')$$

Identical path:



Different paths:

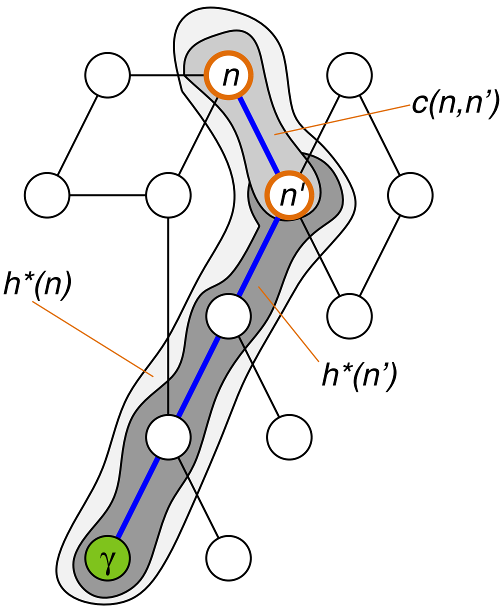


Monotone Heuristic Functions

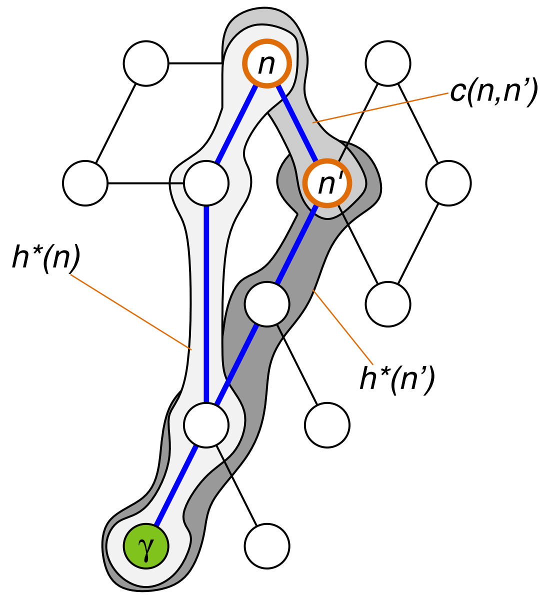
Illustration of the Local Triangle Inequality

$$\forall\ n, n':\quad h^*(n) \leq c(n, n') + h^*(n')$$

Identical paths:



Different paths:



Monotone Heuristic Functions

Definition 79 (Consistency Condition)

Let G be a search space graph with *Prop*$_{A^*}(G)$.
A heuristic function h is called consistent, iff ($\leftrightarrow$) for all nodes n, n' in G holds:

$$h(n) \leq k(n, n') + h(n')$$

Monotone Heuristic Functions

Definition 79 (Consistency Condition)

Let G be a search space graph with *Prop*$_{A^*}(G)$.
A heuristic function h is called consistent, iff ($\leftrightarrow$) for all nodes n, n' in G holds:

$$h(n) \leq k(n, n') + h(n')$$

Definition 80 (Monotonicity Condition)

Let G be a search space graph with *Prop*$_{A^*}(G)$.
A heuristic function h is called monotone, iff ($\leftrightarrow$) for all nodes n, n' in G with $n' \in succ(n)$ holds:

$$h(n) \leq c(n, n') + h(n')$$

Remarks:

- The consistency condition corresponds to the fulfillment of the global triangle inequality.
- The monotonicity condition corresponds to the fulfillment of the local triangle inequality.
- The global triangle inequality is obviously fulfilled for nodes n, n' where n' ist not reachable from n in G. Recall that we defined $k(n, n') := +\infty$ for such nodes.
- If $c(n, n') = +\infty$ for nodes n, n' with $n' \notin succ(n)$, the local triangle inequality holds for all nodes n, n' in G.

Monotone Heuristic Functions

Theorem 81 (Consistency Equivalent to Monotonicity)

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

$$\text{Consistency} \quad \Leftrightarrow \quad \text{Monotonicity}$$

$$h(n) \leq k(n, n') + h(n') \qquad h(n) \leq c(n, n') + h(n')$$

Monotone Heuristic Functions

Theorem 81 (Consistency Equivalent to Monotonicity)

Let G be a search space graph with $Prop_{A^*}(G)$. A heuristic function h is consistent iff ($\leftrightarrow$) h is monotone.

$$\text{Consistency} \quad \Leftrightarrow \quad \text{Monotonicity}$$

$$h(n) \leq k(n,n') + h(n') \qquad h(n) \leq c(n,n') + h(n')$$

Proof (sketch)

1. "$\Rightarrow$"
   Monotonicity follows from consistency, since consistency states the triangle inequality for any pair of nodes n, n' with n' reachable from n considering cost of a cheapest path. Monotonicity considers special pairs of nodes $n' \in succ(n)$, and per definition holds: $k(n,n') \leq c(n,n')$

2. "$\Leftarrow$"
   Let n_l be reachable from n_0 and let $P = (n_0, n_1, \ldots, n_l)$ be a cheapest path from n_0 to n_l. Using the monotonicity of h it can be proven by induction over the path length that

   $$h(n_0) \;\leq\; \sum_{i=1}^{l} c(n_{i-1}, n_i) + h(n_l)$$

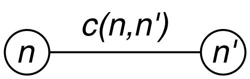
   Since a cheapest path P was considered, we have $k(n_0, n_l) = \sum_{i=1}^{l} c(n_{i-1}, n_i)$.

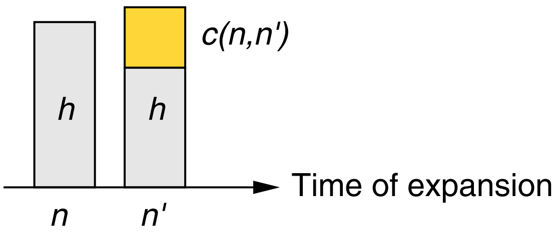
Monotone Heuristic Functions

Illustration of a Monotone h [non-monotone]

Monotonicity defines a restriction on $h(n')$: When moving from n to n' along the edge (n, n'), the h-value decreases at most by $c(n, n')$.

- For an edge:



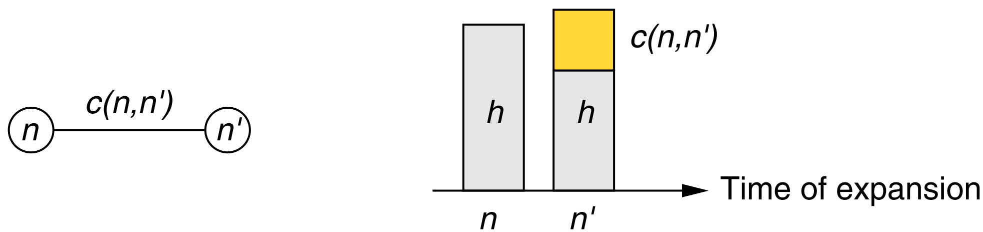


Monotone Heuristic Functions

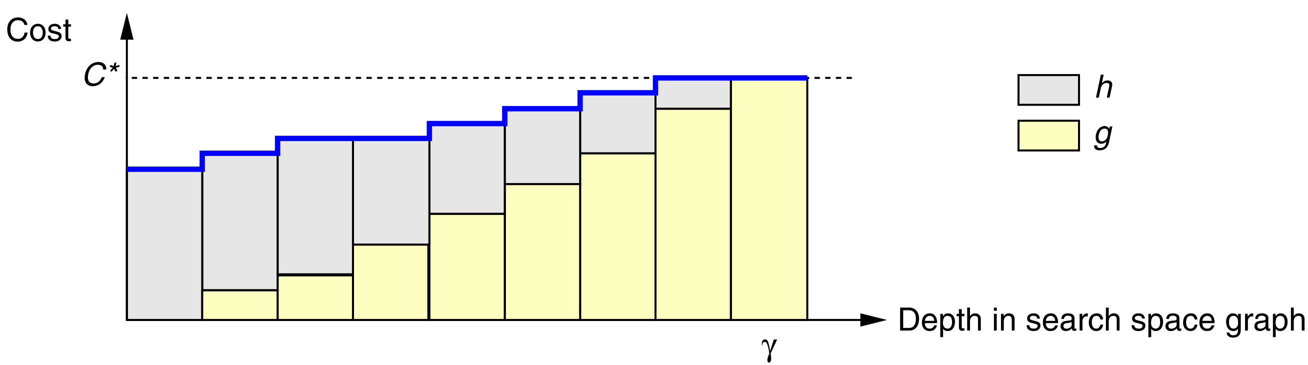
Illustration of a Monotone h [non-monotone]

Monotonicity defines a restriction on $h(n')$: When moving from n to n' along the edge (n, n'), the h-value decreases at most by $c(n, n')$.

- For an edge:



- For an optimum solution path:



Monotone Heuristic Functions

Theorem 82 (Monotone Heuristic Functions)

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

Monotone Heuristic Functions

Theorem 82 (Monotone Heuristic Functions)

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

Proof (sketch)

1. Let h be consistent, i.e., $h(n) \leq k(n, n') + h(n')$ for all nodes n, n' in G.
2. Consider an arbitrary node n.
3. If no goal node is reachable from n, then $h^*(n) = +\infty$ and thus $h(n) \leq h^*(n)$.
4. If some goal node is reachable from n, there is also a goal node γ reachable from n with cheapest cost. [Corollary "Solution Existence Entails Optimum"]
5. Using $n' = \gamma$, we have $h(n) \leq \underbrace{k(n, \gamma)}_{h^*(n)} + \underbrace{h(\gamma)}_{0}$.
6. Since n is arbitrary chosen, $h(n) \leq h^*(n)$ holds for all nodes. Hence h is admissible.

Remarks:

- Consider the special case $h(n) = 0$. For a graph with positive edge cost $h(n) = 0$ is monotone.
- Q. Compare A* with $h(n) = 0$ to the shortest-path algorithm of Dijkstra. Does Dijkstra's shortest-path algorithm reopen nodes?

Monotone Heuristic Functions

Theorem 83 (No Reopening)

Let G be a search space graph with *Prop*$_{A^*}(G)$. An A* algorithm that uses a monotone heuristic function h will expand only nodes to which it has found cheapest paths:

$$g(n) = g^*(n), \quad \forall\, n \in \text{CLOSED}$$

Monotone Heuristic Functions

Theorem 83 (No Reopening)

Let G be a search space graph with *Prop*$_{A^*}(G)$. An A* algorithm that uses a monotone heuristic function h will expand only nodes to which it has found cheapest paths:

$$g(n) = g^*(n), \quad \forall\, n \in \text{CLOSED}$$

Proof (sketch)

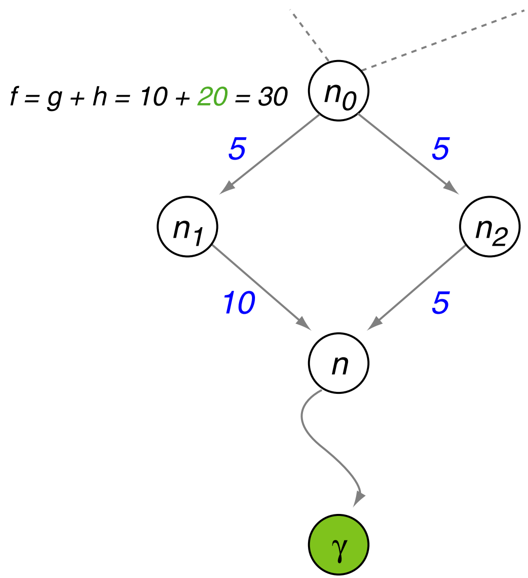
1. Assume that A* selects a node n for expansion with $g(n) > g^*(n)$.
2. Let P^*_{s-n} be a cheapest path from s to n.
3. If $P^*_{s-n} \cap \text{OPEN} = \{n\}$, then all predecessors of n on P^*_{s-n} have been expanded and $g(n) = g^*(n)$, contradicting the assumption. [Lemma “Shallowest OPEN Node on Optimum Path”]
4. If $P^*_{s-n} \cap \text{OPEN} \neq \{n\}$, let n' be the shallowest OPEN node on P^*_{s-n}.
5. Using $g(n') = g^*(n')$ [Lemma “Shallowest OPEN Node on Optimum Path”] and the monotonicity of h we have $f(n') = g(n') + h(n') = g^*(n') + h(n') \;\leq\; g^*(n') + k(n', n) + h(n)$.
6. Since $n' \in P^*_{s-n}$, we have $g^*(n') + k(n', n) = g^*(n)$ and thus $f(n') \leq g^*(n) + h(n)$.
7. According to the assumption $g(n) > g^*(n)$ we have $f(n') < g(n) + h(n)$ and thus $f(n') < f(n)$.
8. A* selects n' for expansion instead of n, contradicting the assumption.

Remarks:

- Note that a heuristic function can be admissible without being monotone: admissibility is necessary for monotonicity.
- In the book of Pearl this theorem is denoted as Theorem 10 and Nilsson Result 7 respectively. [Pearl 1984]

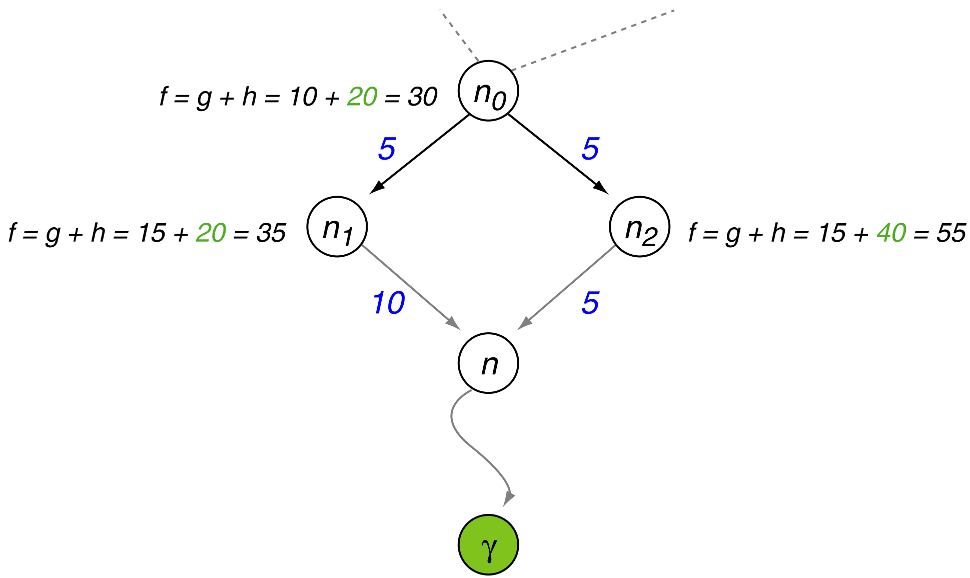
Monotone Heuristic Functions

Illustration of a Non-Monotone h [monotone]



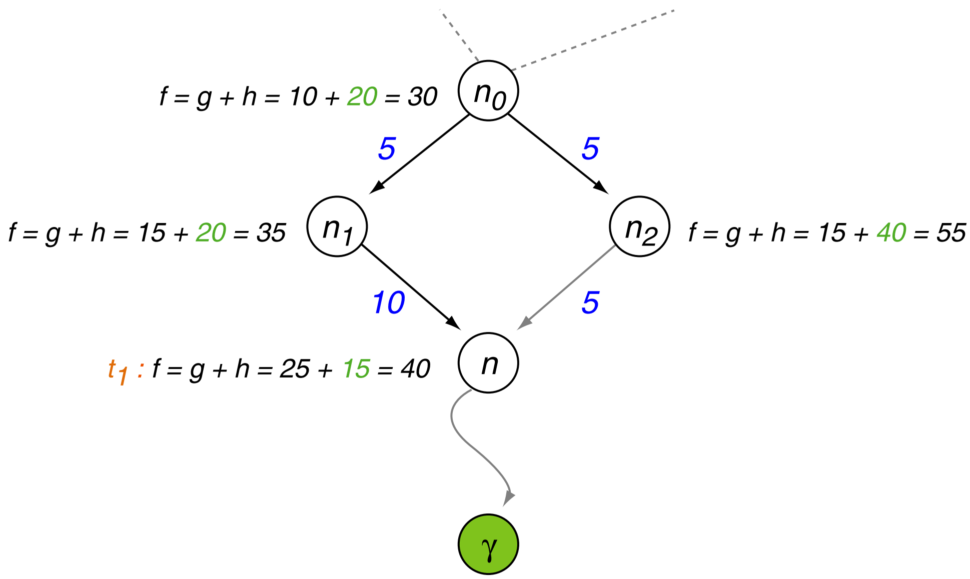
Monotone Heuristic Functions

Illustration of a Non-Monotone h [monotone]



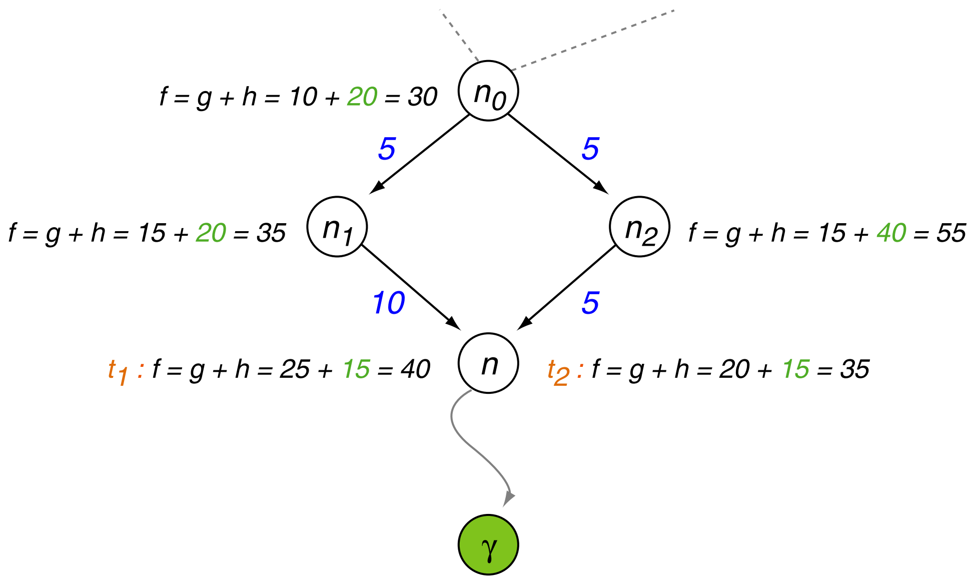
Monotone Heuristic Functions

Illustration of a Non-Monotone h [monotone]



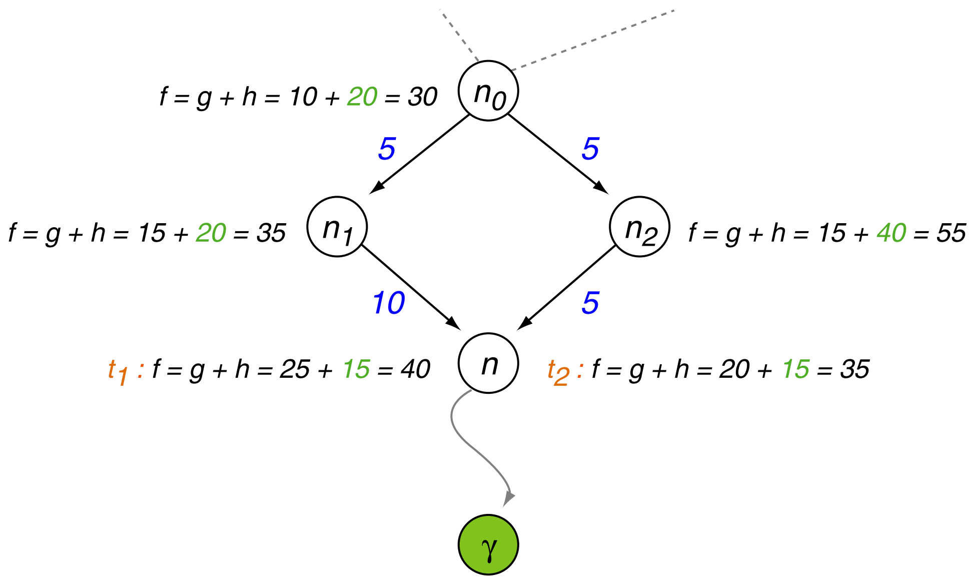
Monotone Heuristic Functions

Illustration of a Non-Monotone h [monotone]



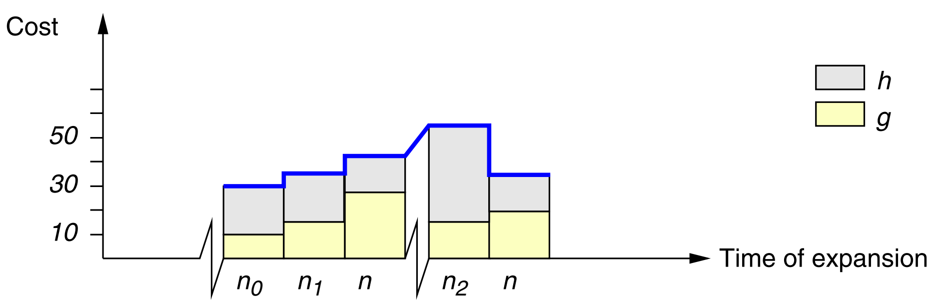
Monotone Heuristic Functions

Illustration of a Non-Monotone h [monotone]



The monotonicity condition, $h(n_2) \leq c(n_2, n) + h(n)$, is not satisfied for n_2: $40 > 5 + 15$

Sequence of node expansions: $n_0, n_1, n, \ldots, n_2, n$



Monotone Heuristic Functions

Theorem 84 (Non-Decreasing f-Values)

Let G be a search space graph with *Prop*$_{A^*}(G)$. When using a monotone heuristic function h the f-values of the sequence of nodes expanded by an A* algorithm will be non-decreasing.

Monotone Heuristic Functions

Theorem 84 (Non-Decreasing f-Values)

Let G be a search space graph with *Prop*$_{A^*}(G)$. When using a monotone heuristic function h the f-values of the sequence of nodes expanded by an A* algorithm will be non-decreasing.

Proof (sketch)

1. Let n_2 be expanded directly after n_1 in the sequence of node expansions.
2. If n_2 is no successor of n_1 in G, then n_2 is on OPEN as well and $f(n_2) \geq f(n_1)$.
3. If n_2 is successor of n_1 in G, then a new path to n_2 was found when n_1 was expanded.

   For the f-value of n_2 holds:

   (a) If n_2 was newly generated, then, since h is monotone,
   $f_{\text{new}}(n_2) = g(n_1) + c(n_1, n_2) + h(n_2) \;\geq\; g(n_1) + h(n_1) = f(n_1)$. [Illustration]

   (b) If n_2 was already on OPEN then holds:
   - If $f(n_2)$ was improved, then Case (a) defines the new f-value of n_2.
   - If $f(n_2)$ was not improved, then $f(n_2) \geq f(n_1)$ must still have hold.

   Finally, n_2 was not on CLOSED, since it was expanded. [Theorem 83]

Remarks:

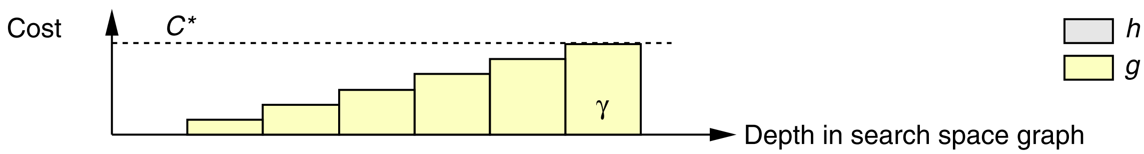
- In the book of Pearl this theorem is denoted as Theorem 11 and Nilsson Result 8 respectively. [Pearl 1984]

Monotone Heuristic Functions

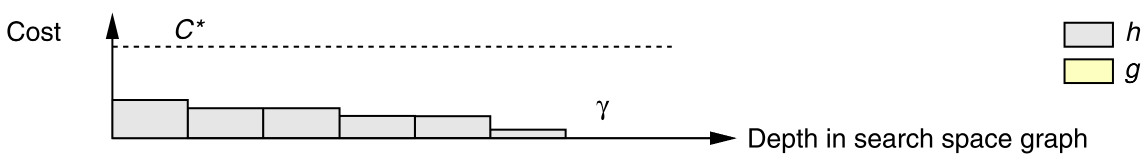
Illustration of Non-Decreasing f-Values

1. Along an optimum solution path (in the search space graph):

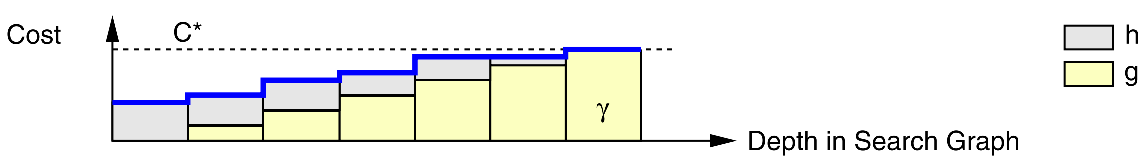
   - Path cost values $g(n)$ are (strictly) increasing since $c(n, n') > 0$.



   - Usually, estimated cheapest remaining cost values $h(n)$ are decreasing. The monotonicity condition $h(n) \leq c(n, n') + h(n')$ restricts the possible changes in $h(n)$.



➜ This ensures that the estimated cheapest total cost values $f(n)$ are non-decreasing.



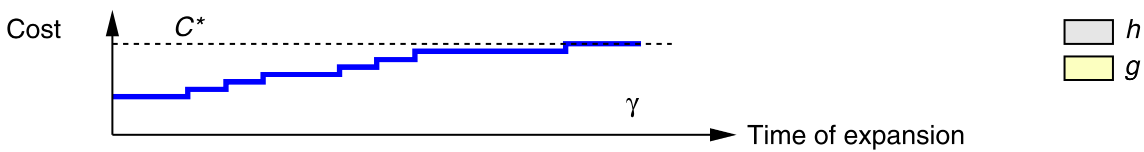
Monotone Heuristic Functions

Illustration of Non-Decreasing f-Values (continued)

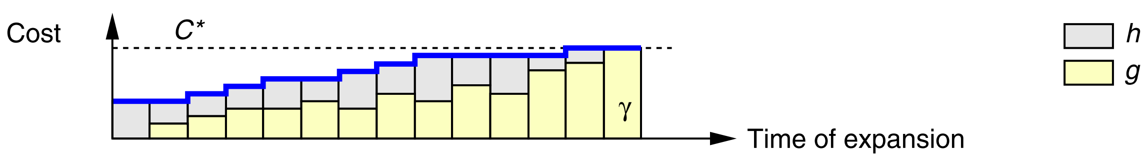
2. In the sequence of nodes expanded by A* (in order of expansion):

   We assume there is a solution path.

   - Again we have: Estimated cheapest total cost values $f(n)$ of nodes expanded by A* are non-decreasing. [Theorem 84]

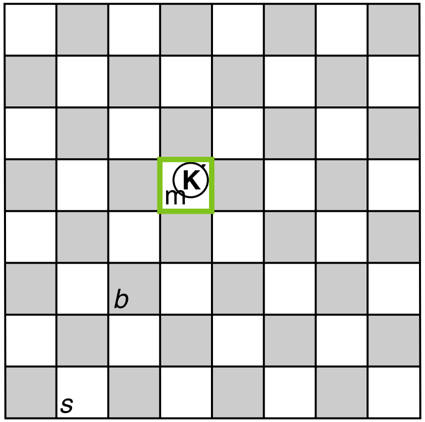


   - But: For the sequences of path cost values $g(n)$ or estimated cheapest remaining cost values $h(n)$ for node expanded by A* no monotonicity can be stated.
     Q. Why not?



Monotone Heuristic Functions

Example: Knight Moves (revisited)



OPEN	CLOSED
$\{c, e, l, n,$ $f, g, h, i, j, k, o, p\}$	$\{m, b, d, a, s\}$

n	$g(n)$	$h_1(n)$	$f(n)$
s	0	2	2
a	1	1	2
b	1	1	2
c	1	2	3
d	2	0	2
e	2	1	3
f	2	2	4
g	3	1	4
h	3	1	4
i	3	1	4
j	3	1	4
m	2	0	2
l	2	1	3
n	2	1	3
k	2	2	4
o	2	2	4
p	2	2	4

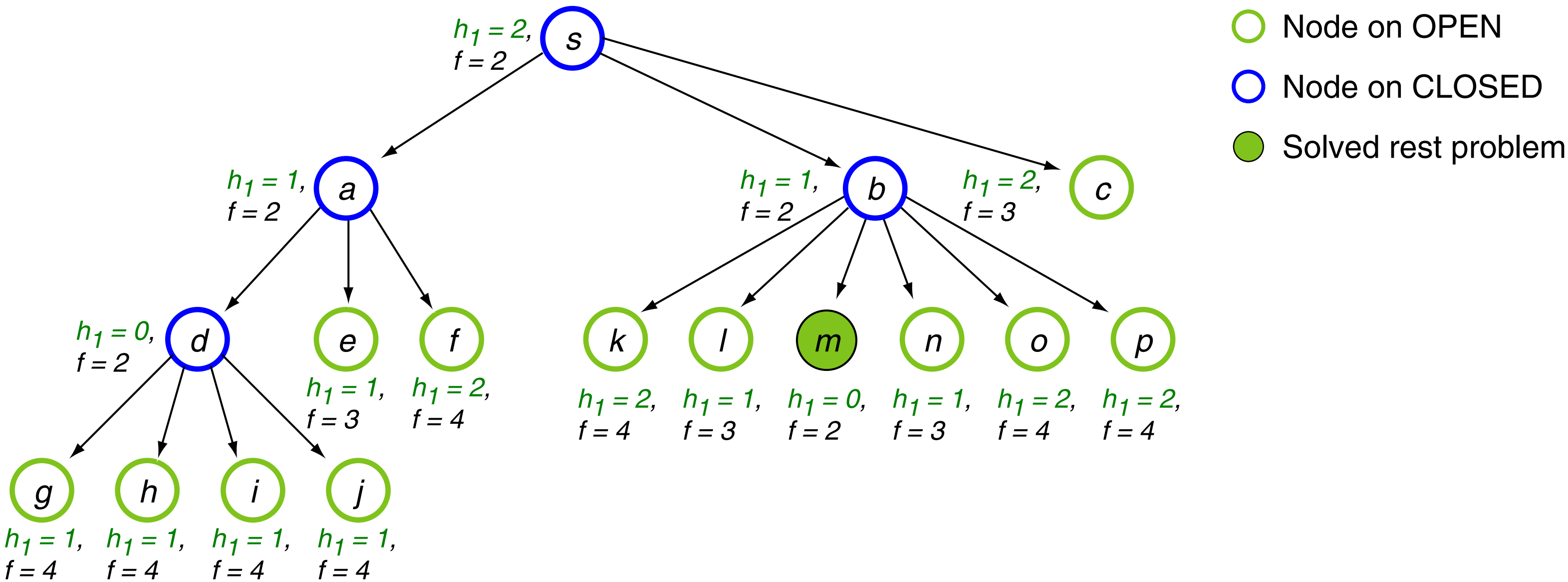
f-values never decrease.

$$h = \left\lceil \frac{\#rows}{2} \right\rceil$$

Monotone Heuristic Functions

Example: Knight Moves (revisited)

Analyzed part of the search space graph:

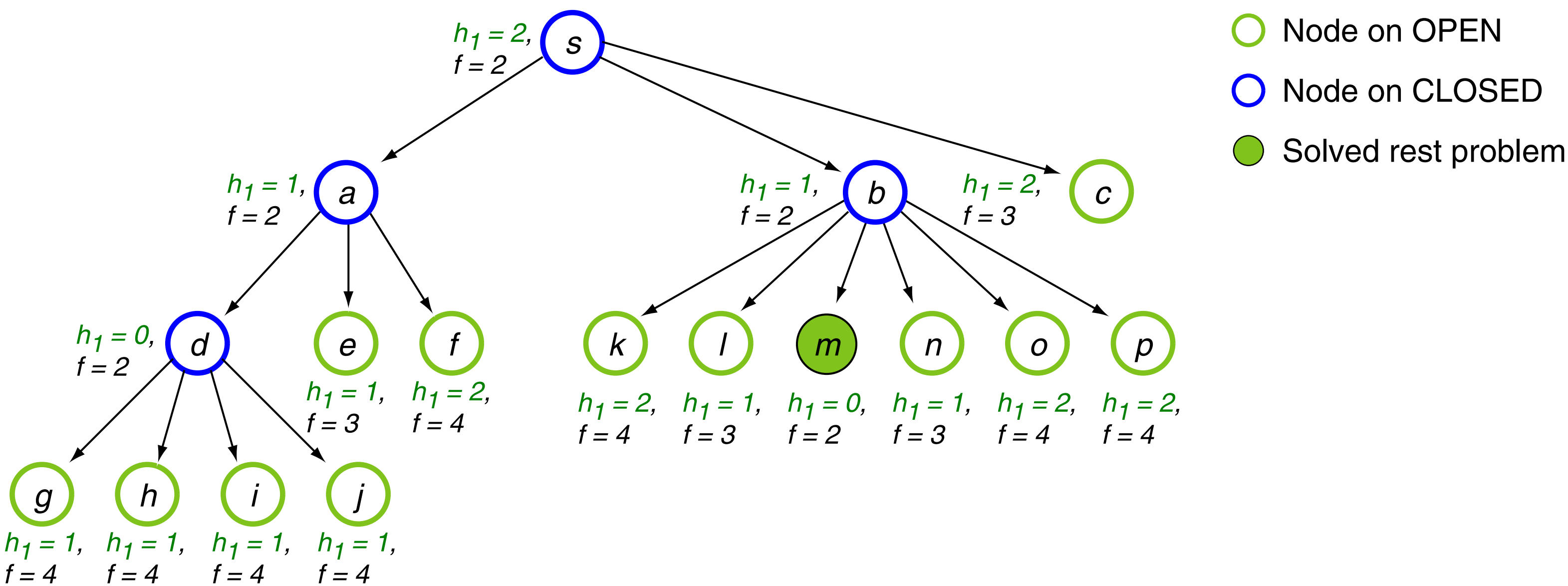


f-values never decrease. Q. Is h monotone—i.e., does $h(n) \leq c(n, n') + h(n')$ hold?

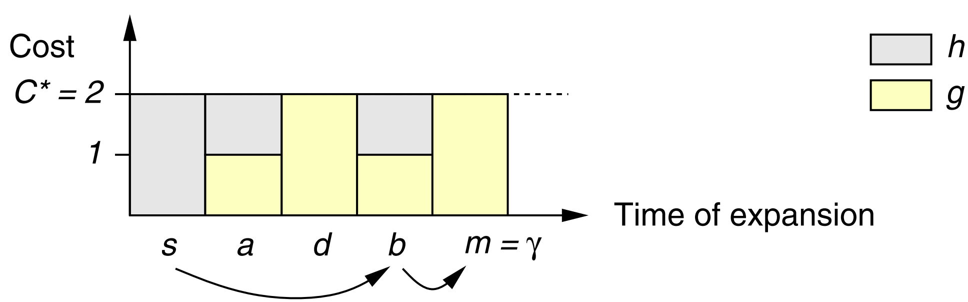
Monotone Heuristic Functions

Example: Knight Moves (revisited)

Analyzed part of the search space graph:



f-values never decrease. Q. Is h monotone—i.e., does $h(n) \leq c(n, n') + h(n')$ hold?



Monotone Heuristic Functions

Coping with non-monotonicity [Martelli 77]

Starting point: Decreasing f-values indicate non-monotonicity in estimations.

Martelli suggests algorithm B with a node selection strategy based on storing a value F that is the biggest f-value of nodes expanded so far:

- Initially, assign $F \leftarrow 0$.
- When entering the main loop:
  - If there is a node n' on OPEN with $f(n') < F$,
    then select a node n from OPEN with $f(n) < F$ and minimum g-value.
  - Otherwise, select a node n with minimum f-value from OPEN and let $F \leftarrow f(n)$.

Algorithm B is A* with a different node selection strategy.

➜ Algorithm B is admissible. B dominates A*.

The exponential number of node expansions example problem for A* will be solved with a quadratic number by algorithm B.

Monotone Heuristic Functions

Coping with non-monotonicity [Mero 84]

Starting point: Non-monotone behavior of h is observed in node expansions.

Mero suggests algorithm B' with an additional adaption of h-values in node expansions in Martelli's algorithm:

- Initially, assign $h'(s) \leftarrow h(s)$ for the start node s.
- When entering the main loop:
  - Select a node n with minimum value $g(n) + h'(n)$ from OPEN.
  - If n is expanded, then let

$$h'(n') \leftarrow \begin{cases} \max\{h(n'), (h'(n) - c(n,n'))\} & \text{for all successor nodes } n' \text{ of } n \\ & \text{that were not in OPEN or CLOSED,} \\ \max\{h'(n'), (h'(n) - c(n,n'))\} & \text{for all other successor nodes } n' \text{ of } n. \end{cases}$$

Algorithm B' is A* with heuristic values h' changing during runtime.

➜ Algorithm B' is admissible. B' dominates B and A*.

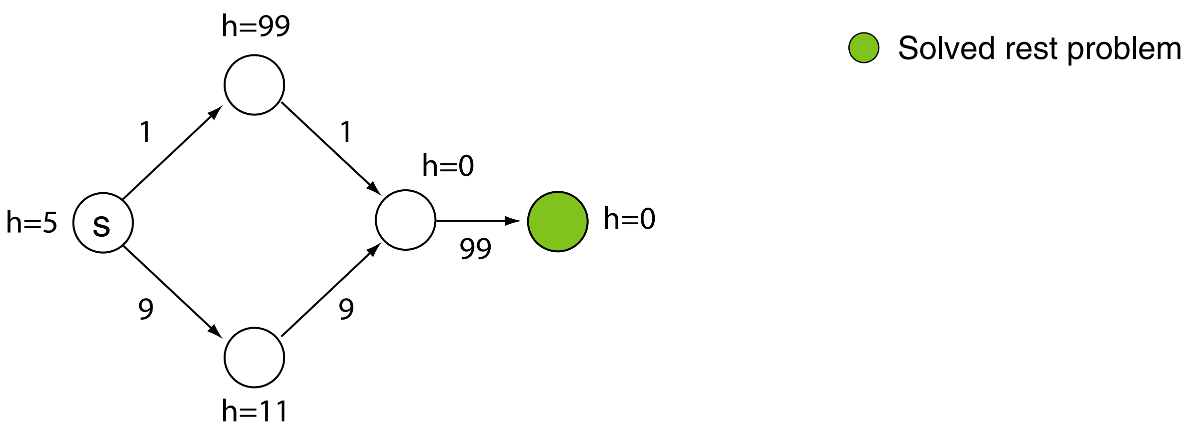
Remarks:

- If only the adaption of h-values for successor nodes is used, the resulting method is known as *pathmax*:

  $$h'(n') = \max\{h(n'), h'(n) - c(n,n')\}\text{ for all successors } n' \text{ of } n.$$

  Let A* use h'-values in the node selection step instead of h-values.

  In this way, f-values in back-pointer paths increase along the path. However, using this method does not prevent CLOSED nodes from being reopened. The following example, which shows this behavior, is based on an idea of Holte [Holte 2010]:



  Remarks on Mero's full method can be found in [Zhang 2009].

Monotone Heuristic Functions

Conditions for Node Expansion III

Theorem 85 (Necessary and Sufficient Conditions for Node Expansion III)

Let G be a search space graph with *Prop*$_{A^*}(G)$ and let A* use a monotone heuristic function h. The condition $g^*(n) + h(n) \leq C^*$ is a necessary condition for expanding node n. A sufficient condition for expanding n is $g^*(n) + h(n) < C^*$.

Monotone Heuristic Functions

Conditions for Node Expansion III

Theorem 85 (Necessary and Sufficient Conditions for Node Expansion III)

Let G be a search space graph with *Prop*$_{A^*}(G)$ and let A* use a monotone heuristic function h. The condition $g^*(n) + h(n) \leq C^*$ is a necessary condition for expanding node n. A sufficient condition for expanding n is $g^*(n) + h(n) < C^*$.

Proof (sketch)

1. The necessary condition follows by combining Theorem 71 and Theorem 83.
2. The sufficient condition follows from the non-decreasing behavior of f-values along optimum paths, i.e.,
$$g^*(n_1) + h(n_1) \leq g^*(n_1) + c(n_1, n_2) + h(n_2) = g^*(n_2) + h(n_2)$$
for nodes n_1, n_2 on an optimum path P^*_{s-n} to a node n with n_2 being a successor of n_1.
3. $\Rightarrow$ From condition $g^*(n) + h(n) < C^*$ then follows that P^*_{s-n} is strictly C^*-bounded.
4. $\Rightarrow$ Hence, n will be expanded. [Theorem 76]

Monotone Heuristic Functions

Conditions for Node Expansion III [Conditions I, Conditions II]

For monotone heuristic functions h hold:

$$\text{node expansion by A*} \;\Rightarrow\; \underbrace{g^*(n) + h(n) \leq C^*}_{\text{necessary}}$$

$$\underbrace{g^*(n) + h(n) < C^*}_{\text{sufficient}} \;\Rightarrow\; \text{node expansion by A*}$$

Again, Theorem 85 does not give a condition for node expansion that is both necessary and sufficient:

The theorem does not treat nodes with $g^*(n) + h(n) = C^*$.

Remarks:

- If monotone heuristic functions are used, then the condition $g^*(n) + h(n) < C^*$ has to be tested only for a node n—but the computation of $g^*(n)$ may be costly. If searching a strictly C^*-bounded path to n, any node on that path has to be considered.

- Since f-values for the sequence of expanded nodes are non-decreasing, the advantage of using different tie breaking rules along with the same monotone heuristic h is limited to the number of nodes with $g^*(n) + h(n) = C^*$. Recall that such nodes may occur on any path, not only on solution paths. An inferior tie breaking rule may select a node with $g^*(n) + h(n) = C^*$ that is not on an optimum solution path.

  Nodes with $g^*(n) + h(n) < C^*$ may be expanded at different points in time due to different tie breaking rules but they will be expanded before termination.

- Theorem 85 gives an explanation why the difference between the relation $f(n) \leq C^*$ and $f(n) < C^*$ is often insignificant: If $f(n)$ can be modeled as a continuous random variable, the equality $f(n) = C^*$ becomes a rare event.

- In the book of Pearl this theorem is denoted as Theorem 12. [Pearl 1984]

Monotone Heuristic Functions

Definition 86 (Largely Dominating Algorithms)

An algorithm A_1 (informed by h_1) largely dominates A_2 (informed by h_2) if every node expanded by A_1 is also expanded by A_2 except, perhaps, some nodes for which $h_1(n) = h_2(n) = C^* - g^*(n)$ holds.

Corollary 87 (of Theorem 85)

Let G be a search space graph with *Prop*$_{A^*}(G)$. If $h_1(n) \geq h_2(n)$ holds for any n and if $h_1(n)$ and $h_2(n)$ both are monotone, then A*$_1$ (A* informed by $h_1(n)$) largely dominates A*$_2$ (A* informed by $h_2(n)$).

Monotone Heuristic Functions

Definition 86 (Largely Dominating Algorithms)

An algorithm A_1 (informed by h_1) largely dominates A_2 (informed by h_2) if every node expanded by A_1 is also expanded by A_2 except, perhaps, some nodes for which $h_1(n) = h_2(n) = C^* - g^*(n)$ holds.

Corollary 87 (of Theorem 85)

Let G be a search space graph with *Prop*$_{A^*}(G)$. If $h_1(n) \geq h_2(n)$ holds for any n and if $h_1(n)$ and $h_2(n)$ both are monotone, then A*$_1$ (A* informed by $h_1(n)$) largely dominates A*$_2$ (A* informed by $h_2(n)$).

Proof (sketch)

1. Let n be a node expanded by A*$_1$ but not by A*$_2$.
2. It holds: $g^*(n) + h_1(n) \leq C^*$ and $C^* \leq g^*(n) + h_2(n)$ [Theorem 85]
3. Since $h_1(n) \geq h_2(n)$ holds: $C^* \leq g^*(n) + h_2(n) \leq g^*(n) + h_1(n) \leq C^*$
4. $\Rightarrow \; h_1(n) = h_2(n) = C^* - g^*(n)$

Remarks:

- Note that in the absence of monotonicity, the advantage of A*$_1$ over A*$_2$ would be much less certain: every descendant of n that is reachable from n by a—wrt. h_1 strictly C^*-bounded path—will also be expanded by A*$_1$ and possibly not by A*$_2$.
- If h is monotone (or consistent), then A* largely dominates every admissible algorithm having access to the same h. [Dechter/Pearl 1983]
- If h is admissible but not monotone (or consistent), then there are admissible algorithms that, using the same h, will grossly outperform A* in some problem instances, regardless of what tie breaking rule A* invokes. [Pearl 1984]
- Monotonicity is not an exceptional property but can be often established for an admissible heuristic. For instance, all the heuristic functions discussed in the introduction are monotone.