The Optimality of A*

Presented by
Jarrell Waggoner
&
Jimmy Cleveland

12/09/08

Overview

Informed Search

Uninformed search: no information about the goal state, other than if it has been reached yet

Informed search: some heuristic information about the goal state is available at each node in the graph

Uninformed Search	Informed Search
Depth-first	Best-first
Breadth-first	A^*
Dijkstra	

Historical Notes

A^*:

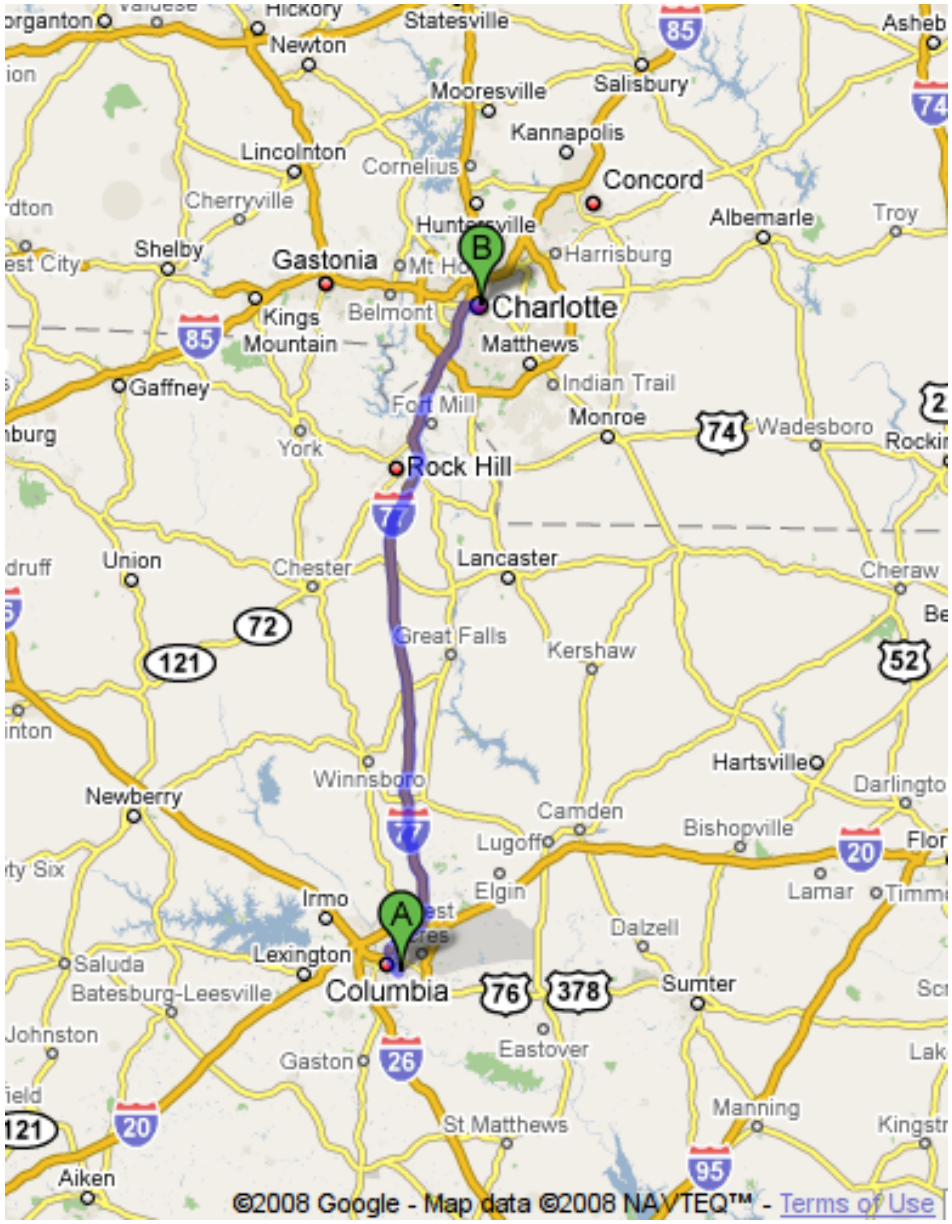
- Created in 1968
- "A Formal Basis for the Heuristic Determination of Minimum Cost Paths"
- Authored by Peter Hart, Nils Nilsson, and Bertram Raphael

In the original paper, the algorithm we now know as A^* was simply labeled "Algorithm A." Since the star $(*)$ is used to denote optimality, and Algorithm A was optimal when given an admissible heuristic, it became known as A^*.

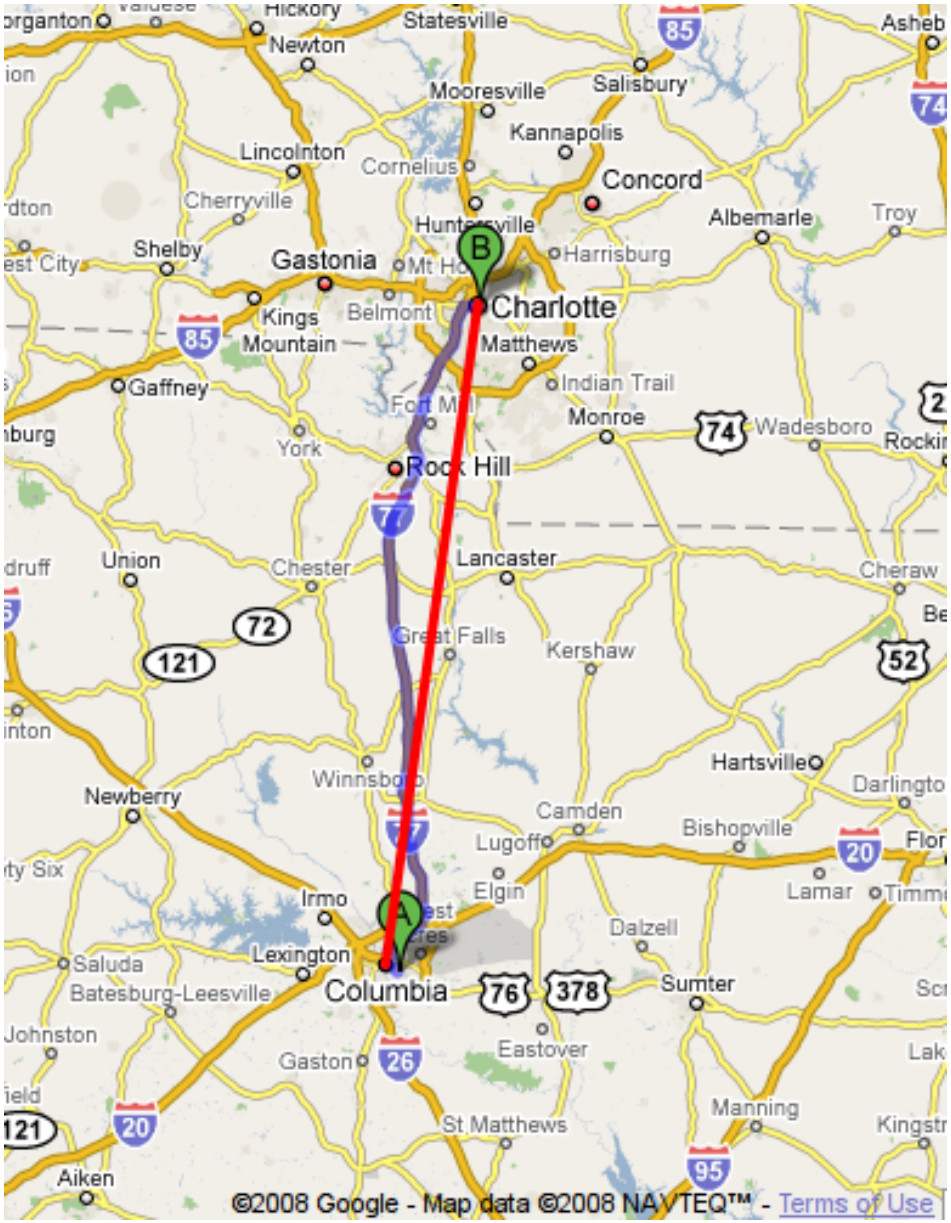
Introduction to A*

1: Put the start node, s, on a list called $OPEN$ of unexpanded nodes
2: **if** $|OPEN| = 0$ **then**
3: Exit—no solution exists
4: Remove a node n from $OPEN$, at which $f = g + h$ is minimum and place it on a list called $CLOSED$
5: **if** n is a goal node **then**
6: Exit with solution
7: Expand node n, generating all its successors with pointers back to n
8: **for all** successor n' of n **do**
9: Calculate $f(n')$
10: **if** $n' \notin OPEN$ AND $n' \notin CLOSED$ **then**
11: Add n' to $OPEN$
12: Assign the newly computed $f(n')$ to node n'
13: **else**
14: If new $f(n')$ value is smaller than the previous value, then update with the new value (and predecessor)
15: If n' was in $CLOSED$, move it back to $OPEN$
16: Go to (2)

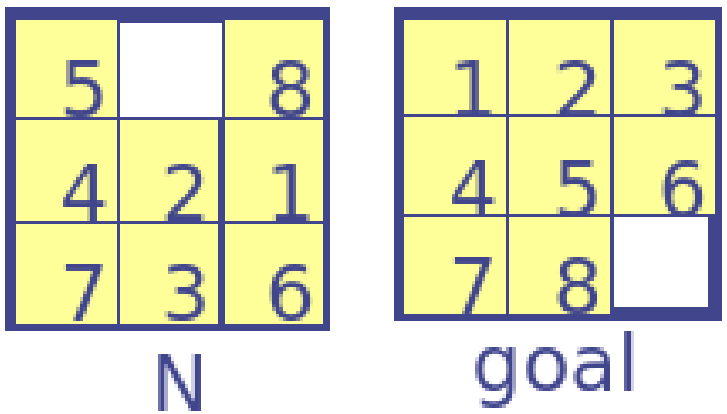
Map Example


Statesville
Newton
Mooresville
Salisbury
Kannapolis
Lincolnton
Cornelius
Concord
Cherryville
Albemarle
Troy
Shelby
Gastonia
Harrisburg
B
Charlotte
Kings Mountain
Belmont
Matthews
Gaffney
Indian Trail
Fort Mill
Monroe
Wadesboro
York
Rock Hill
Union
Chester
Lancaster
Great Falls
Kershaw
Cheraw
Winnsboro
Hartsville
Newberry
Camden
Lugoff
Bishopville
Irmo
Elgin
Lamar
A
Dalzell
Lexington
Saluda
Batesburg-Leesville
Columbia
Sumter
Johnston
Gaston
Eastover
St Matthews
Manning
Aiken
©2008 Google - Map data ©2008 NAVTEQ™ - Terms of Use

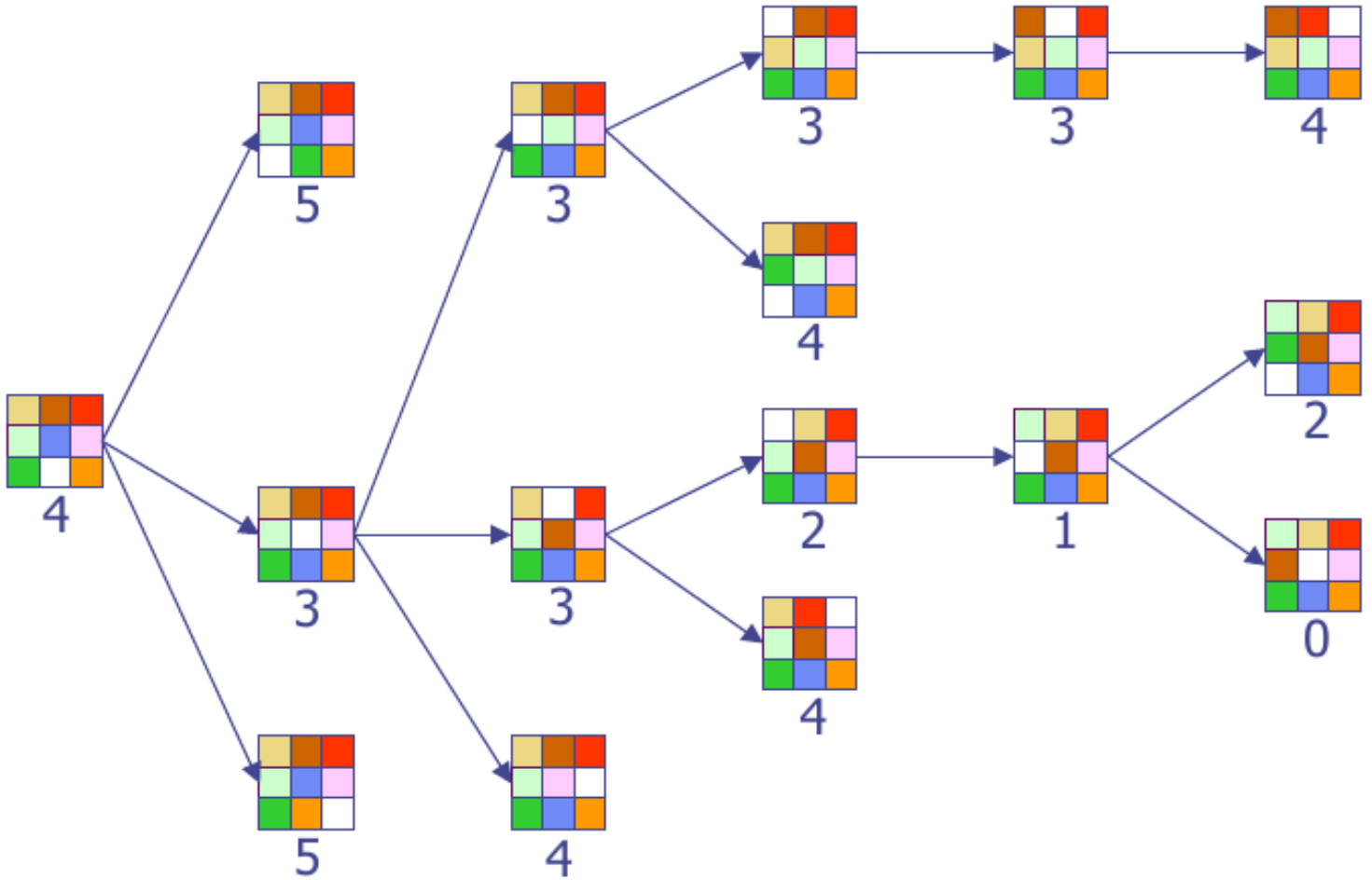
Map Example


Statesville
Newton
Mooresville
Salisbury
Kannapolis
Lincolnton
Cornelius
Concord
Cherryville
Albemarle
Shelby
Gastonia
Harrisburg
Charlotte
Kings Mountain
Belmont
Matthews
Gaffney
Indian Trail
Monroe
Wadesboro
York
Rock Hill
Union
Chester
Lancaster
Great Falls
Kershaw
Cheraw
Newberry
Winnsboro
Hartsville
Camden
Bishopville
Lugoff
Elgin
Irmo
Lexington
Columbia
Dalzell
Sumter
Saluda
Batesburg-Leesville
Johnston
Gaston
Eastover
St Matthews
Manning
Aiken
©2008 Google - Map data ©2008 NAVTEQ™ - Terms of Use

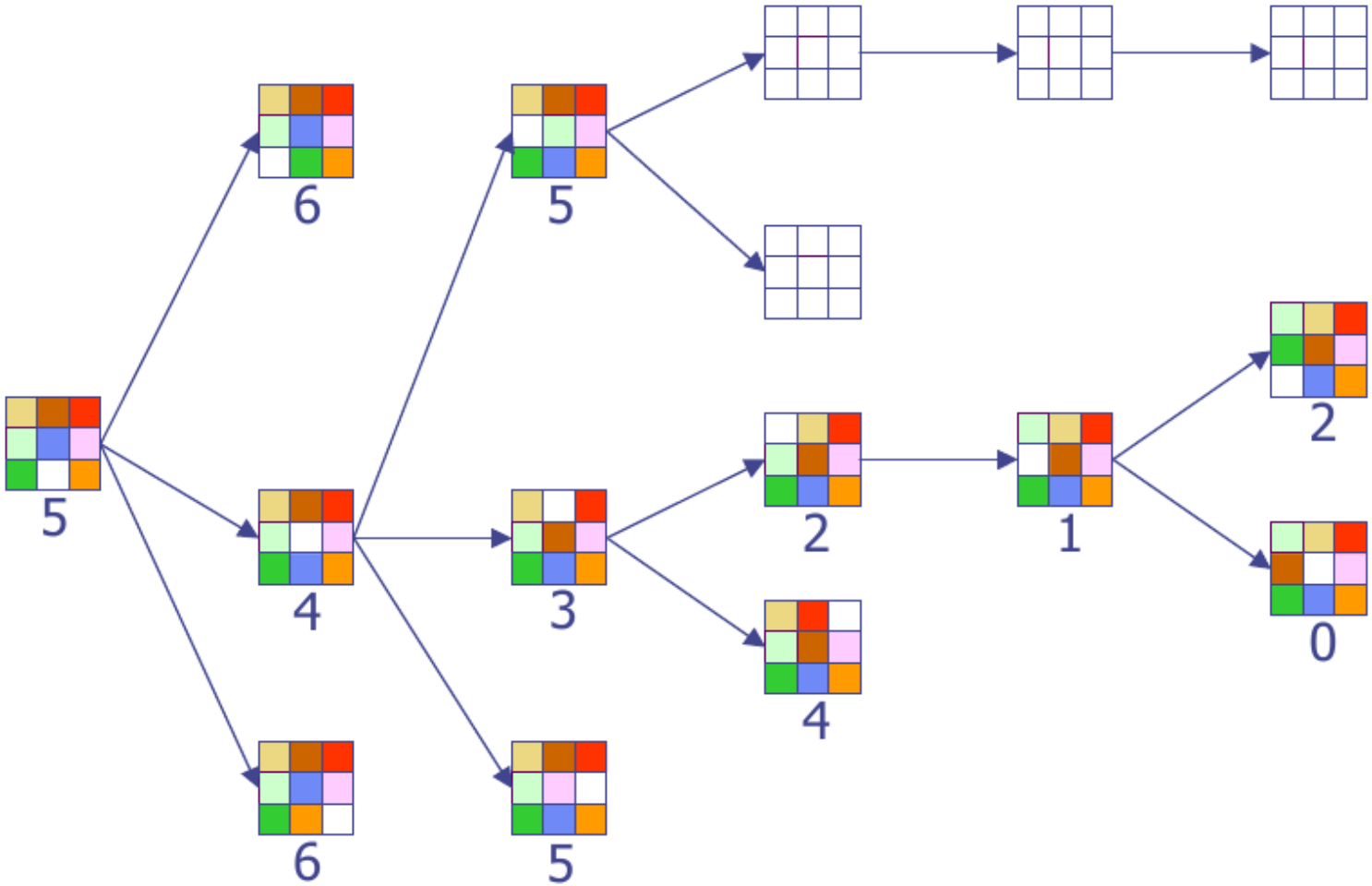
8-Puzzle Example



8-Puzzle Example



8-Puzzle Example



Introduction to A* (cont.)

Some notes about A^*:

- A^* is a class of algorithms, not a single, set algorithm
- Depth-first, breadth-first, uniform-cost, and Dijkstra's algorithm are specific instances of A^* and vary only by the choice of heuristic and tie-breaking rule
- A^* is a specific case of best-first search, where the heuristic evaluation function $f(x)$ is defined as $f(n) = g(n) + h(n)$.

$$\text{Best-first} \supseteq A^* \supseteq \begin{cases} \text{Depth-first} \\ \text{Breadth-first} \\ \text{Dijkstra} \end{cases}$$

Optimality

Optimality (finding a shortest path): Provided A^* is given an admissible heuristic, it will always find a shortest path because the "optimistic" heuristic will never allow it to skip over a possible shorter path option when expanding nodes

Optimality (number of node expansions): Specifically, the number of node expansions verses other algorithms with the same heuristic information (as $A*$ is clearly more optimal than algorithms that lack heuristic information)

It is this second definition that we seek to prove.

Notation

G	directed locally finite graph $G = (V, E)$
C^*	the cost of the cheapest solution path
$C(.)$	the cost function defined over all solution paths
Γ	a set of goal nodes, $\Gamma \subseteq V$
$P_{n_i - n_j}$	a path in G between node n_i and n_j
P^S	a solution path, i.e., a path in G from s to some goal node $\gamma \in \Gamma$
$c(n, n')$	cost of an arc between n and n', $c(n, n') \geq \delta > 0$, where δ is a constant
$f(.)$	evaluation function defined over partial paths, i.e., to each node n along a given path $P = s_1, n_1, n_2, \ldots, n$ we assign the value $f_P(n)$ which is shorthand notation for $f(s, n_1, n_2, \ldots, n)$
$g(n)$	the sum of the branch costs along the current path of pointers from n to s
$g^*(n)$	the cost of the cheapest path going from s to n
$g_P(n)$	the sum of the branch costs along path P from s to n
$h(n)$	a cost estimate of the cheapest path between n and Γ
$h^*(n)$	the cost of the cheapest path going from n to Γ
$k(n, n')$	cost of the cheapest path between n and n'

Definitions

Admissible Heuristic: A heuristic function $h(n)$ is said to be **admissible** on (G, Γ) iff $h(n) \leq h^*(n)$ for every $n \in G$

Consistent Heuristic: A heuristic function $h(n)$ is said to be **consistent** (or monotone) on G iff for any pair of nodes, n' and n, the triangle inequality holds:

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

Surely Expanded: Nodes that **must** be expanded to reach a goal node (regardless of the tie-breaking rule or algorithm used)

Definitions (cont.)

Admissible Algorithm: An algorithm that, given a problem space where $h(n) \leq h^*(n)$ for every $n \in G$, will return least-cost solutions

Dominance: Algorithm A **dominates** some other algorithm B if the set of nodes that A expands is a **subset** of the nodes expanded by B (not just fewer nodes)

Strictly Dominate: A strictly dominates B iff A dominates B and B does not dominate A

Optimal (strong definition): Algorithm A is optimal over a class $\underline{A}$ of algorithms iff A dominates every member of $\underline{A}$

Weakly Optimal: No member of $\underline{A}$ strictly dominates A

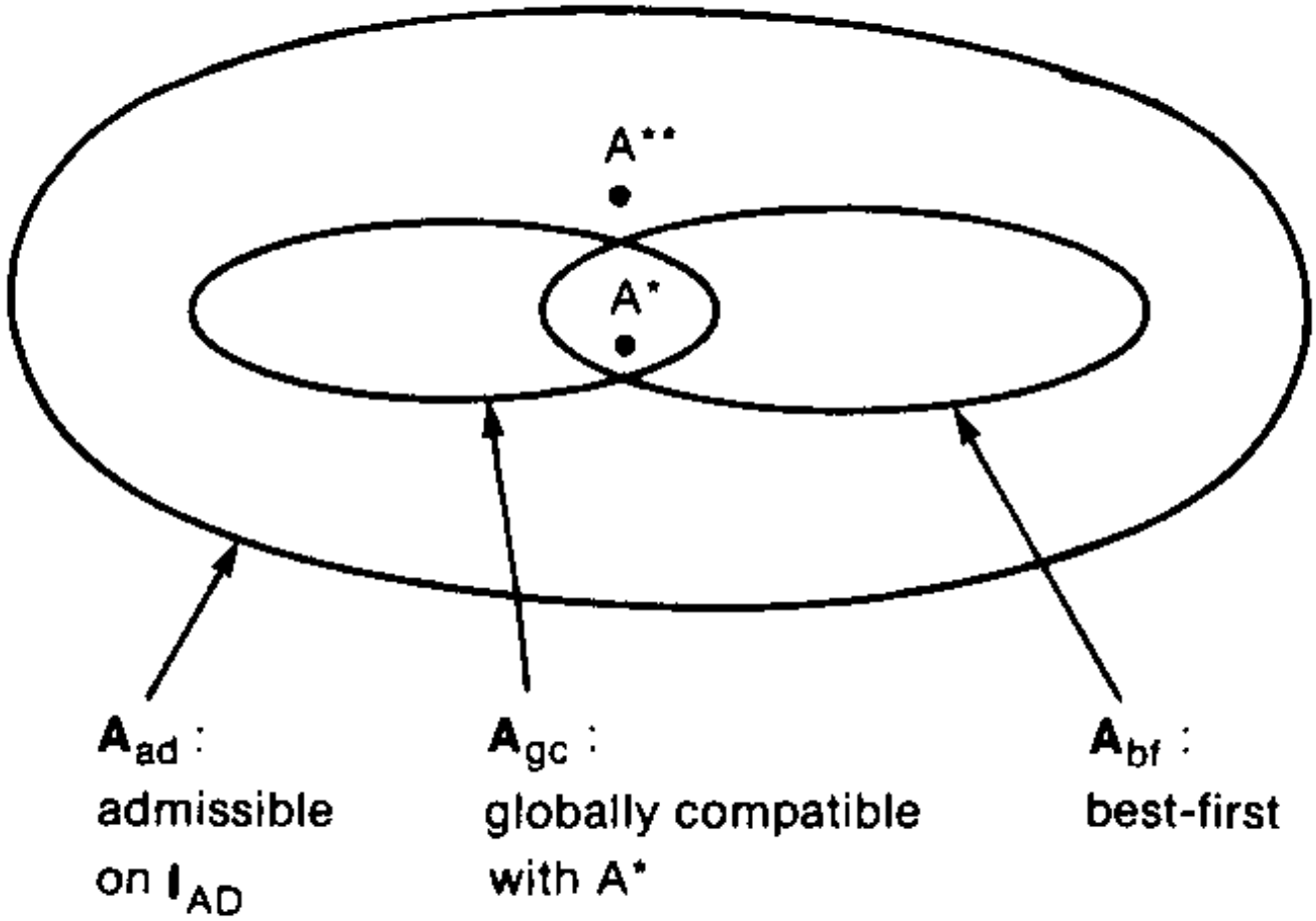
Outline

Algorithm Classes

To determine the optimality of A^* over other algorithms with the same heuristic information, we group these algorithms into classes (since A^* is already a class of algorithms, each with a different tie-breaking rule):

- $\underline{A}_{ad}$: Algorithms that return least-cost solutions when given an admissible problem space (though not necessarily an admissible heuristic to run on that problem space)
- $\underline{A}_{bf}$: Subclass of $\underline{A}_{ad}$ that accepts any path-dependent evaluation function, but operates in a best-first manner
- $\underline{A}_{gc}$: Subclass of $\underline{A}_{ad}$ that will return optimal solutions whenever A^* does, but may not obey $h > h^*$

Algorithm Classes (cont.)



Problem Instances

Problem Instance: A particular "setup" of a general problem.

Defined by a quadruple: $I = (G, s, \Gamma, h)$

- G is the graph representation of the problem
- s is the start node in the graph
- Γ is the set of goal nodes
- h is a heuristic that any algorithm run on this instance will use

$\underline{I}$ is a *set* of problem instances that share some common trait

Problem Instances (cont.)

There are four important sets of problem instances:

- $\underline{I}_{AD}$: The set $I \in \underline{I}$ that has an admissible heuristic:

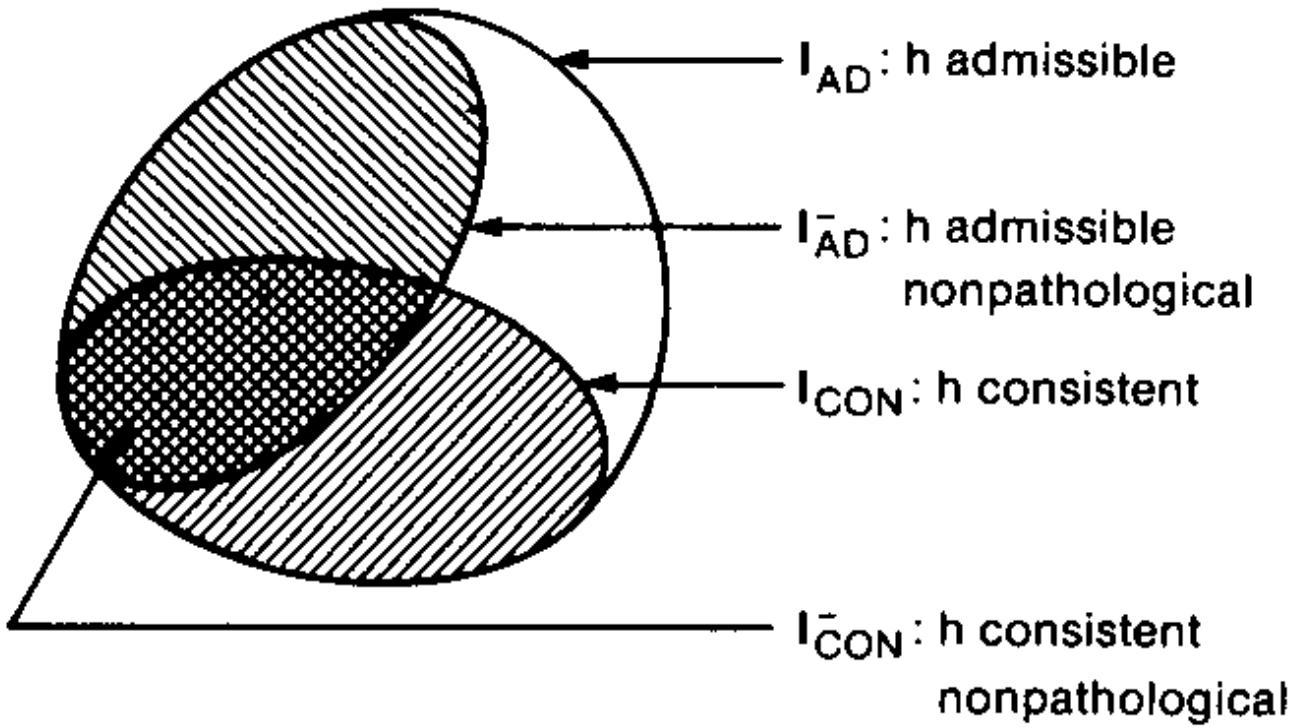
$$\underline{I}_{AD} = \{(G, s, \Gamma, h) | h \leq h^* \text{ on } (G, \Gamma)\}$$

- $\underline{I}_{CON}$: The set $I \in \underline{I}$ that has a consistent heuristic:

$$\underline{I}_{AD} = \{(G, s, \Gamma, h) | h \text{ is consistent on } G\}$$

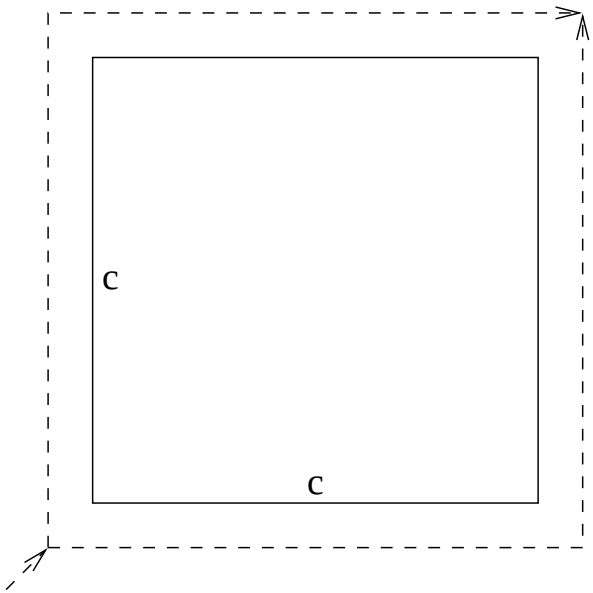
- $\underline{I}_{\overline{AD}}$: The non-pathological case of $\underline{I}_{AD}$
- $\underline{I}_{\overline{CON}}$: The non-pathological case of $\underline{I}_{CON}$

Problem Instances (cont.)



Tie-Breaking Rules

A^* can be considered a class of algorithms, defined by the tie-breaking rule chosen



0-3 Optimality

A^* is optimal relative to $\underline{A}$ in the following senses:

0-Optimal: All A^*'s tie-breaking rules dominate all members of $\underline{A}$

1-Optimal: One of A^*'s tie-breaking rules expands a subset of all $\underline{A}$'s members

2-Optimal: No member of $\underline{A}$ expands a proper subset of any of A^*'s members

3-Optimal: No tie-breaking rule in A^* (or A^{**}) is strictly dominated by some member of $\underline{A}$

Strongest $\rightarrow$ Weakest
0-Optimal $\rightarrow$ 3-Optimal

Outline

Theorem 1

Theorem

Any algorithm that is admissible on $\underline{I}_{AD}$ will expand, in every instance $I \in \underline{I}_{CON}$, all nodes surely expanded by A^*.

Meaning

Problems with a consistent heuristics cannot be solved any better than A^* can solve them by algorithms with the same information.

Theorem 1 Proof

Proof :
Given $I = (G, S, \Gamma, h) \in \underline{I}_{CON}$, and assuming that n is surely expanded by A^*, then there exists a path P_{s-n} s.t.

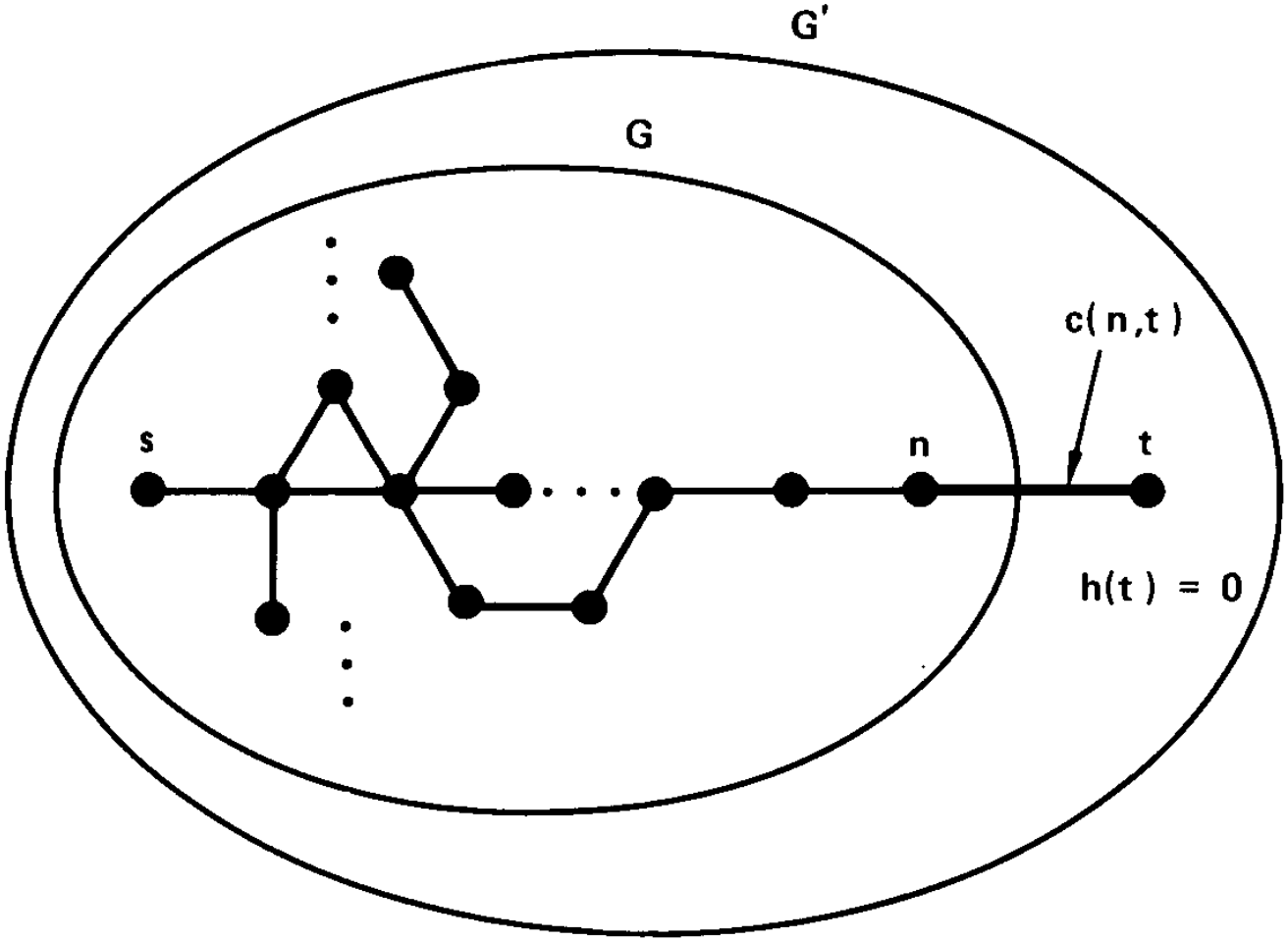
$$\forall n' \in P_{s-n}, g(n') + h(n') < C^*$$

Let B be an algorithm compatible with $A*$ that halts with cost C^* in I. Assume that B does not expand n. We can then create G' as follows:

Translation

What we do here is setup a contradiction: For B to be better than A^*, it must skip some node n that A^* visits. We assume B does this, and setup a new graph G' that will introduce a contradiction.

Constructing G and G'



Theorem 1 Proof

where we have added a goal node t to G. The costs of t is given by $h(t) = 0$ and the edge from n to t is given as $c = h(n) + \triangle$ where:

$$\triangle = \frac{1}{2}(C^* - D) > 0$$

$$D = max\left\{f(n')|n' \in \underline{N}_{g+h}^{G^*}\right\}$$

This creates a new path P^* whose cost is $\leq C^* - \triangle$ yet is still consistent (and admissible) on the new I'.

Translation

Here, we setup a new node in G' and make sure it has the proper costs associated with it so that the new G' obeys all the rules that the old G did.

Theorem 1 (cont.)

It remains to be proven that h is consistent on I' for the new node t in G' (this is trivially true for all the previous nodes since the h values of all the nodes in G remain unchanged).
This is done by establishing $h(n') \leq k(n', t)\ \forall n' \in G$.
At any node n' we should also have

$$h(n') > k(n', n) + c = k(n', n) + h(n) + \triangle$$

but this violates h's consistency on I, so $I' \in \underline{I}_{CON}$.

Translation

We show that by using the above weight for the $k(n', t)$ edge, the new graph G' remains consistent.

Theorem 1 (cont.)

Thus A^* will find the new path P^* (which costs $C^* - \triangle$) because

$$f(t) = g(n) + c = f(n) + \triangle \leq D + \triangle = C^* - \triangle < C^*$$

So t is reachable from s by a $C^* - \triangle$ bounded path, so it will be selected.
But algorithm B must behave the same as if it were running on I, halting with cost $C^* > C^* - \triangle$. This is a contradiction that B is both admissible on I, and avoids the expansion of n.

Translation

We have a contradiction, because after creating G', which has a shortest path goal node attached to n (which A^* finds properly), B is unable to find a shortest path and therefore must not be admissible.

Theorem 2

Theorem

If any admissible algorithm B does not expand a node which is surely expanded by A^* in some problem instance where h is admissible and non-pathological, then in that very problem instance B must expand a node which is avoided by every tie-breaking-rule in A^*.

No algorithm (A^* or otherwise) is 1-optimal over those guaranteed to find an optimal solution when given $h \leq h^*$.

Outline

A^{**}

A^{**} differs from A^* in that it relies not only on the $g + h$ value of node n, but also considers all the $g + h$ values along the path from s to n. The maximum is then used as a criterion for node selection.

A^{**} Evaluation Function

$$f'_{P_{s-n}} = max\{f(n') = g_{P_{s-n}}(n') + h(n') | n' \in P_{s-n}\}$$

A^{**}

Optimality

A^{**} is 3-optimal over all algorithms relative to $\underline{I}_{AD}$.

The type-3 optimality of A^{**} over $\underline{A}_{ad}$ demonstrates that those "smart" algorithms that prevent A^* from achieving optimality are not smart after all, but simply lucky; each takes advantage of the peculiarity of the graph for which it was contrived, and none can maintain this superiority over all problem instances. If it wins on one graph, there must be another where it is beaten, and by the very same opponent, A^{**}. It is in this sense that A^{**} is 3-optimal; it exhibits a universal robustness against all its challengers.

A^{**}

A^{**} strictly dominates A^*

Theorem

(a) For every tie-breaking rule of A^* and for every problem instance $I \in \underline{I}_{AD}$, there exists a tie-breaking rule for A^{**} that expands a subset of the nodes expanded by A^*.
(b) Moreover, there exists a problem instance and a tie-breaking rule for A^{**} that expands a proper subset of the nodes that are expanded by any tie-breaking rule of A^*.

A^{**}

A^{**} is admissible over I_{AD}.

Theorem

Algorithm A^{**} will terminate with an optimal solution in every problem instance in which $h \leq h^*$.

A^{**}

Note

Because of the pathological cases in $\underline{I}_{AD}$, A^* cannot be 3-optimal. A^{**} exists due to this fact, because it can dominate all instances of A^*. A^{**} is 3-optimal in the most general case.

A^{**} is admissible and in non-pathological cases A^{**} expands the same set of nodes as does A^*, namely, the surely expanded nodes in $N^{C^*}_{g+h}$. In pathological cases, however, there exist tie-breaking rules in A^{**} that strictly dominate every tie-breaking rule in A^*. This immediately precludes A^* from being 3-optimal relative to $\underline{I}_{AD}$ and nominates A^{**} for that title.

A^{**}

A** is not a BF^* algorithm since it uses one function f' for ordering nodes for expansion and a different function g for redirecting pointers. Had we allowed A^{**} to use f for both purposes, it would not be admissible relative to $\underline{I}_{AD}$, since f' is not order preserving.

Outline

Conclusion

Domain of Problem Instances	Class of Algorithms: Admissible if $h \leq h^*$ ($\mathbf{A}_{ad}$)	Globally Compatible with A^* ($\mathbf{A}_{gc}$)	Best-First ($\mathbf{A}_{bf}$)
Admissible I_{AD}	A^{**} is 3-optimal No 2-optimal exists	A^* is 1-optimal No 0-optimal exists	A^* is 1-optimal No 0-optimal exists
Admissible and non pathological $\bar{I}_{AD}$	A^* is 2-optimal No 1-optimal exists	A^* is 0-optimal	A^* is 0-optimal
Consistent I_{CON}	A^* is 1-optimal No 0-optimal exists	A^* is 1-optimal No 0-optimal exists	A^* is 1-optimal No 0-optimal exists
Consistent nonpathological $\bar{I}_{CON}$	A^* is 0-optimal	A^* is 0-optimal	A^* is 0-optimal

Conclusion (cont).

- In consistent cases, no $\underline{A}$ can beat A^* (the node-by-node superiority of type 0 and 1 optimality)
- In the most general case, A^* can be beaten outright by other "smarter" algorithms
- By reformulating A^* as a non-best-first algorithm A^{**}, type 3 optimality can be achieved in even the most general case

References


Rina Dechter and Judea Pearl.
Generalized best-first search strategies and the optimality of a*.
Journal of the ACM, 32:505–536, 1985.


Othar Hansson, Andrew Mayer, and Marco Valtorta.
A new result on the complexity of heuristic estimates for the a* algorithm.
Artificial Intelligence, 55:129–143, 1992.


Laszlo Mero.
A heuristic search algorithm with modifiable estimate.
Artif. Intell., 23(1):13–27, 1984.

Web References

http://theory.stanford.edu/~amitp/GameProgramming/
http://www.policyalmanac.org/games/aStarTutorial.htm
http://idm-lab.org/applet.html
http://www.cs.umd.edu/class/fall2003/cmsc421-0101/heuristic-search.pdf
http://en.wikipedia.org/wiki/A*

[1], [2], [3]