Lecture 6 CS6800 Artificial Intelligence:

- Admissibility of A^*
- Additional properties of A^*
- Comparison of A^* algorithms
- "More Informed" algorithms
- The monotone restriction
- Heuristic power of evaluation functions
- Non A^* heuristics
- Related Algorithms
- Measures of Performance

A property of the nodes selected by A^*

The f value of a node selected for expansion is never greater than the cost $f^*(s)$ of an optimal path.

To prove this theorem, let n be any node selected for expansion by A^*. If n is a goal node, we have:

$$f(n) = f^*(s)$$

by RESULT 4.

Suppose n is not a goal node. Now A* selected n before termination, so that at this time we know there existed on *OPEN* some node n' on an optimal path from s to a goal node with:

$$f(n') \leq f^*(s)$$

If $n = n'$, our result is established. Otherwise, we know that A* chose to expand n rather than n'; therefore it must have been the case that:

$$f(n) \leq f(n') \leq f^*(s)$$

which leads to:

RESULT 5:

For any node n selected for expansion by A*,

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

Comparison of A^* algorithms

- the larger the *h* the greater the heuristic knowledge.
- $h(n) = 0$ reflects complete absence of any heuristic information; even though such an estimate leads to an admissible algorithm.

How can we use A^* for this problem?

2	8	3
1	6	4
7		5

$\rightarrow$

1	2	3
8		4
7	6	5

What would be your choice for an evaluation function? What should the arc costs be?

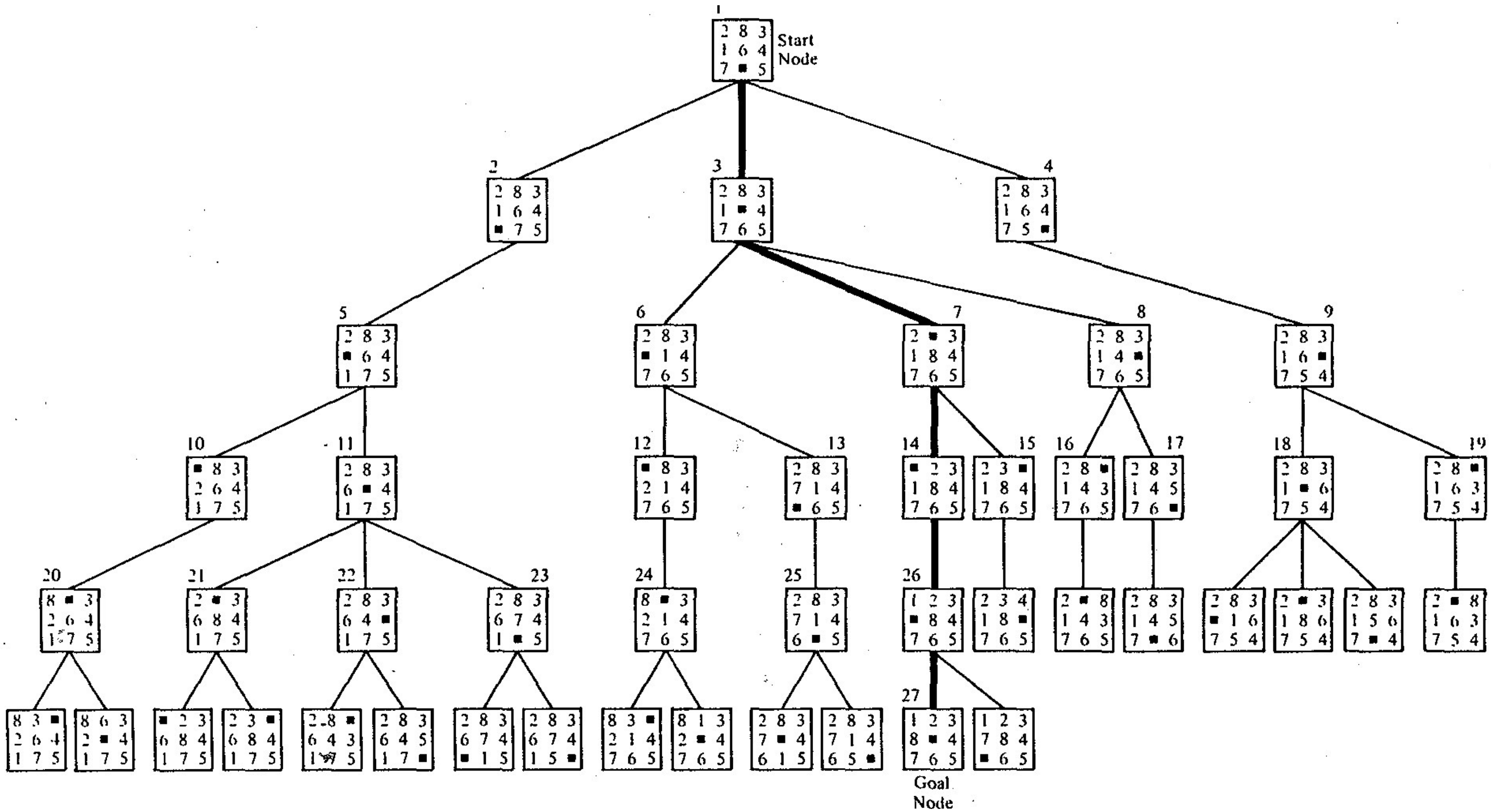
- One possibility: $h(n) = 0$, $g(n) = d(n)$.
- Second possibility: Let's think of a more intelligent choice for the heuristic function.

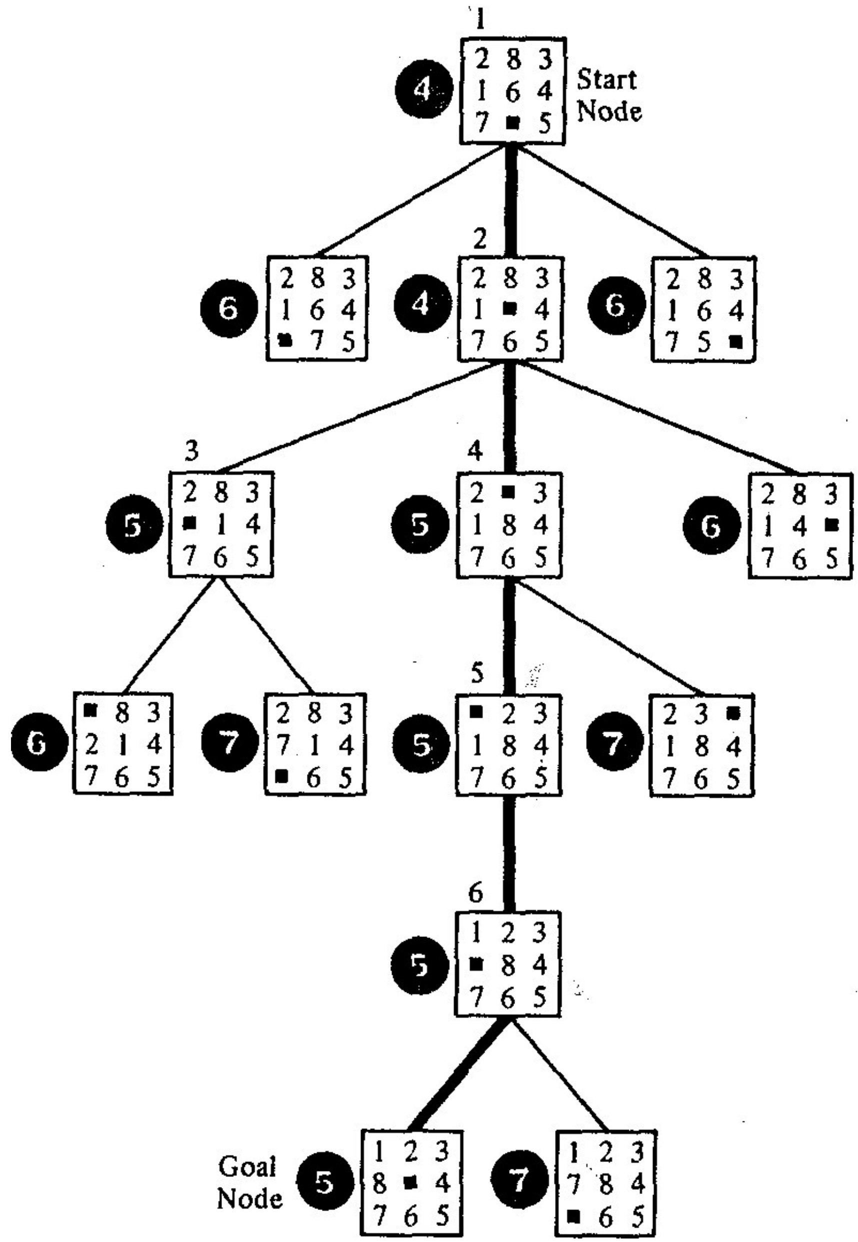
Suppose we try *W(n)*

Where *W(n)* is the number of tiles out of place.

Will this satisfy the A* constraints?

Lets now look at a comparison of these two possibilities.


Start Node
Goal Node


1
Start Node
2
3
4
5
6
Goal Node

What does this say about the efficiency of the two algorithms?

- It appears that the A^* procedure with $h(n) = W(n)$ leads to a smaller expansion of the search graph than with $h(n) = 0$.

Does this mean that this algorithm is more ***efficient***?

Based on the above observations, we could say that the larger the $h(n)$ the more ***informed*** the A^* algorithm.

Formal Definition of "More Informed"

If we have two A^* algorithms A_1 and A_2:

A_1: $f_1(n) = g_1(n) + h_1(n)$

A_2: $f_2(n) = g_2(n) + h_2(n)$

where h_1 and h_2 are both lower bounds on h^*.

If $h_2(n) > h_1(n)$ $\forall$ non-goal nodes n, we say A_2 is ***more informed*** than A_1.

Properties of more informed algorithms

We will now show that if A_2 is more informed than A_1, then A_1 will expand at least as many nodes as A_2. Furthermore, we will show that if a node *n* was expanded by A_2, then it will also be expanded by A_1, but not necessarily vice versa.

We will use induction to prove this result, induction on the depth of a node in the A_2 search tree at termination.

Proof cont.

Base case: if A_2 expands a node n at depth zero, then so will A_1. Why?

Now we will assume that A_1 expands all the nodes expanded by A_2 having depth k, or less, in ***A₂'s search tree***. We must show that any node n that is expanded by A_2 and is of depth $k + 1$ in ***A₂'s search tree*** is also expanded by A_1.

By the induction hypothesis, any ancestor of n in the A_2 search tree, is also expanded by A_1.

Since node n can only have more parents in the A_1 search graph compared to the A_2 search graph, the following must be true:

$$g_1(n) \leq g_2(n)$$

We will now use proof by contradiction to show our result. We will assume that A_1 does not expand node n, while A_2 does.

At termination for A_1, node n must be on *OPEN*. Why? Therefore:

$$f_1(n) \geq f^*(s)$$

Thus,

$$g_1(n) + h_1(n) \geq f^*(s)$$

But we know a relation with $g_1(n)$, thus:

$$h_1(n) \geq f^*(s) - g_2(n)$$

By RESULT 5, since A_2 expanded node n, we have:

$$f_2(n) \leq f^*(s)$$

But $f_2(n) = g_2(n) + h_2(n)$, thus

$$g_2(n) + h_2(n) \leq f^*(s)$$

$$h_2(n) \leq f^*(s) - g_2(n)$$

$$\therefore\ h_2(n) \leq h_1(n)$$

But this violates an assumption! Which one?

Thus node n must also be expanded by A_1.

RESULT 6

If A_1 and A_2 are two versions of A^* such that A_2 is more informed than A_1, then at the termination of their searches on any graph having a path from *s* to a goal node, every node expanded by A_2 is also expanded by A_1.

It follows that A_1 expands at least as many nodes as does A_2.

The Monotone Restriction

One of the inefficiencies of the current method is that if we come to a node that is already on *OPEN*, then we must check if the pointer at this node should be redirected.

If we run into a node that is already on *CLOSED* we have even more work in checking all its descendents.

This leads us to ask the following question:

Are there any heuristic functions that would give us the best (least cost) path to the successor of a node on the very first try?

Monotone Heuristic Functions

With such heuristic methods, the very first time the node *n* is made explicit by expansion, of all the possible paths between *s* and *n* on the implicit search graph, we will already have the best possible path on the search tree - never to be altered as the search continues.

Such heuristic functions must satisfy the ***monotone restriction***.

Monotone Restriction

The monotone restriction says that for all nodes n_i and n_j, such that n_j is a successor of n_i,

$$h(n_i) - h(n_j) \leq c(n_i\,,\, n_j)$$

with $h(t) = 0$.

Similarity to the Triangle Inequality

We can rewrite this restriction in the form:

$$h(n_i) \leq c(n_i\,, n_j) + h(n_j)$$

This makes it look more like the triangle inequality, which says that the distance between any two points must not be less than the distance if measured along a path that passes through a third point.

Examples

In the 8-puzzle, $h(n) = W(n)$, the number of tiles out of place. Does this satisfy the monotone restriction?

Examples cont.

What about $h(n) = 0$?

Note: if the function h is changed in any manner ***during*** the search process, then the monotone restriction might not be satisfied.

A^* using monotone functions

When A^* expands a node, it has already found an optimal path to that node.

To prove this assertion, let n be any node selected for expansion by A^*.

Let the sequence ($s = n_0, n_1, \ldots, n_k = n$) be an optimal path from s to n on the search graph.

Proof cont.

For any pair of nodes, n_i, and n_{i+1} on the optimal path sequence shown on the previous figure, the following is true due to the monotone restriction:

$$g^*(n_i) + h(n_i) \leq g^*(n_i) + c(n_i, n_{i+1}) + h(n_{i+1})$$

But

$$g^*(n_{i+1}) = g^*(n_i) + c(n_i, n_{i+1})$$

Why?

Therefore,

$$g^*(n_i) + h(n_i) \leq g^*(n_{i+1}) + h(n_{i+1})$$

By transitivity,

$$g^*(n_{l+1}) + h(n_{l+1}) \leq g^*(n_k) + h(n_k)$$

But n_{l+1} is on the optimal path to n. Therefore for this node, $g^*(n_{l+1}) = g(n_{l+1})$. Thus,

$$f(n_{l+1}) \leq g^*(n_k) + h(n_k)$$

We also know that for any node n in the search tree $g^*(n) \leq g(n)$. Thus,

$$g^*(n_k) + h(n_k) \leq g(n_k) + h(n_k) = f(n_k)$$

Thus,

$$f(n_{l+1}) \leq f(n_k)$$

But node n_k or n was selected to be expanded before node n_{l+1}! Thus,

$$f(n_{l+1}) = f(n_k)$$

or, finally

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

RESULT 7

If the monotone restriction is satisfied, then A^* has already found an optimal path to any node it selects for expansion. That is, if A^* selects n for expansion, and if the monotone restriction is satisfied, then

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

