CS344: Introduction to Artificial Intelligence (associated lab: CS386)

Pushpak Bhattacharyya
CSE Dept.,
IIT Bombay

Lecture 5: Monotonicity

13th Jan, 2011

Steps of GGS
(*principles of AI, Nilsson,)*

- 1. Create a search graph *G*, consisting solely of the start node *S*; put *S* on a list called *OPEN.*
- *2.* Create a list called *CLOSED* that is initially empty.
- 3. Loop: if *OPEN* is empty, exit with failure.
- 4. Select the first node on *OPEN*, remove from *OPEN* and put on *CLOSED*, call this node *n*.
- 5. if *n* is the goal node, exit with the solution obtained by tracing a path along the pointers from *n* to *s* in *G*. (ointers are established in step 7).
- 6. Expand node *n*, generating the set *M* of its successors that are not ancestors of *n*. Install these memes of *M* as successors of *n* in *G*.

GGS steps (contd.)

- 7. Establish a pointer to *n* from those members of *M* that were not already in *G* (*i.e.*, not already on either *OPEN* or *CLOSED*). Add these members of *M* to *OPEN*. For each member of *M* that was already on *OPEN* or *CLOSED*, decide whether or not to redirect its pointer to *n*. For each member of M already on *CLOSED*, decide for each of its descendents in *G* whether or not to redirect its pointer.
- 8. Reorder the list *OPEN* using some strategy.
- 9. Go *LOOP.*

Illustration for CL parent pointer redirection recursively

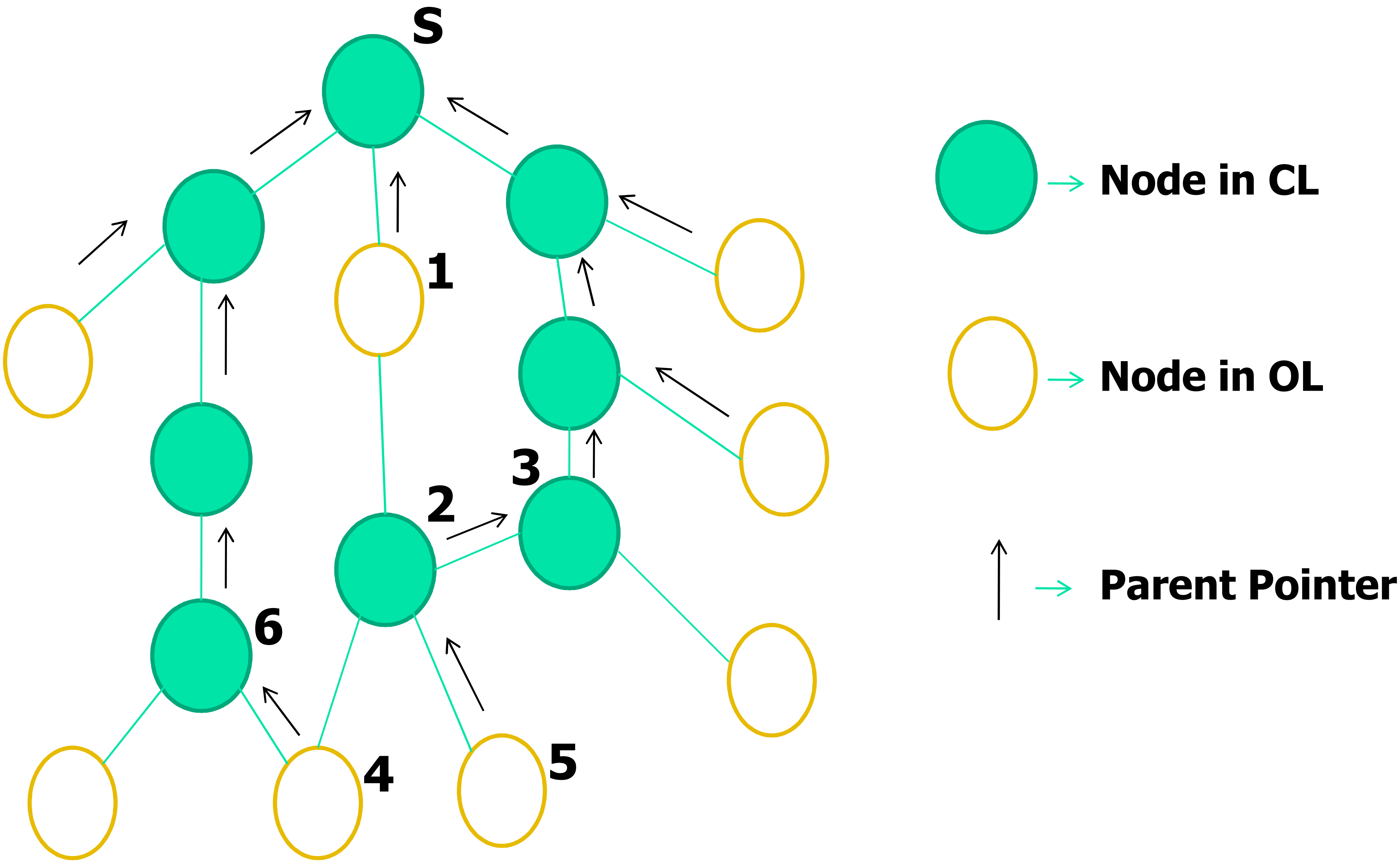
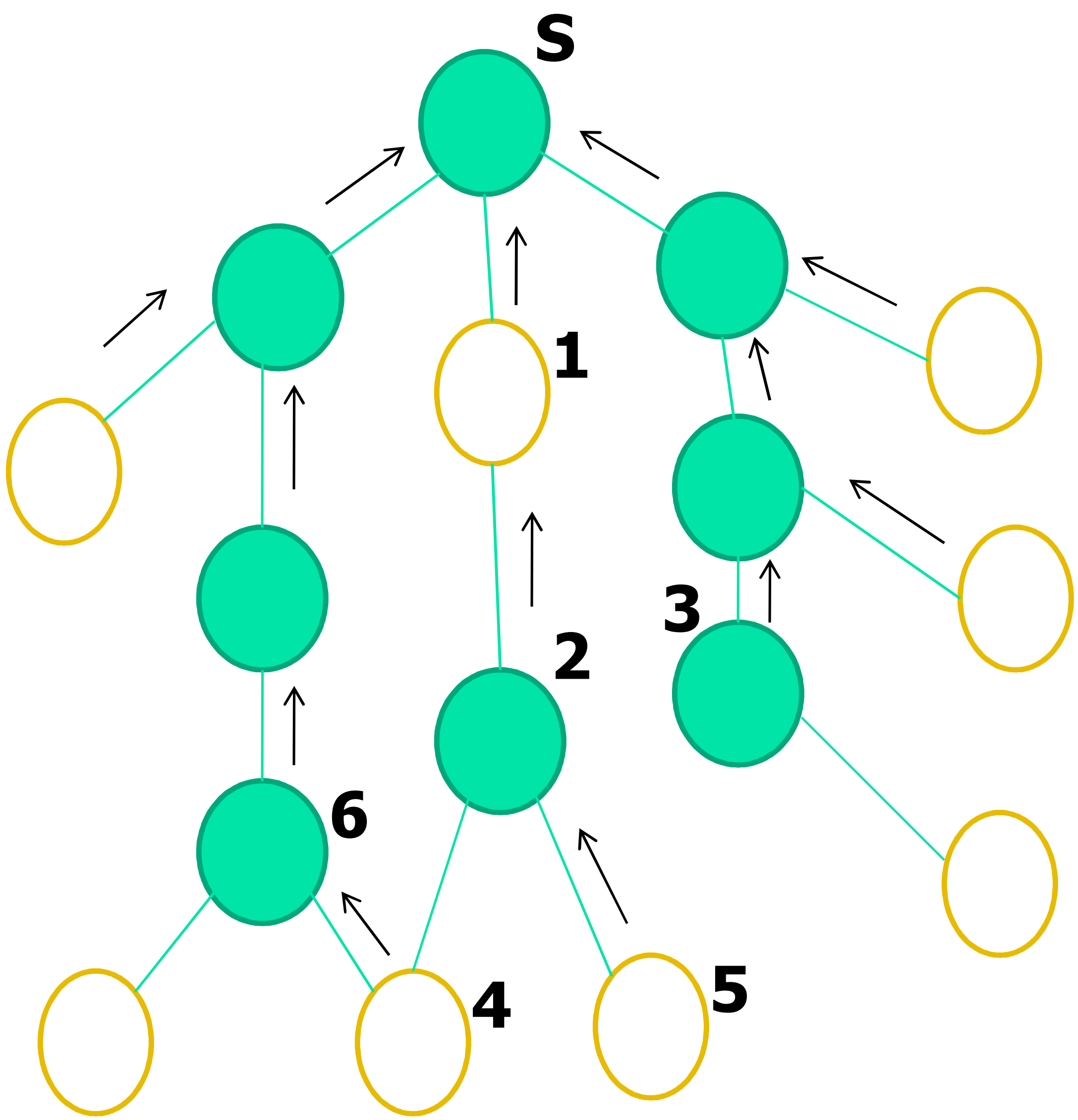


Illustration for CL parent pointer redirection recursively

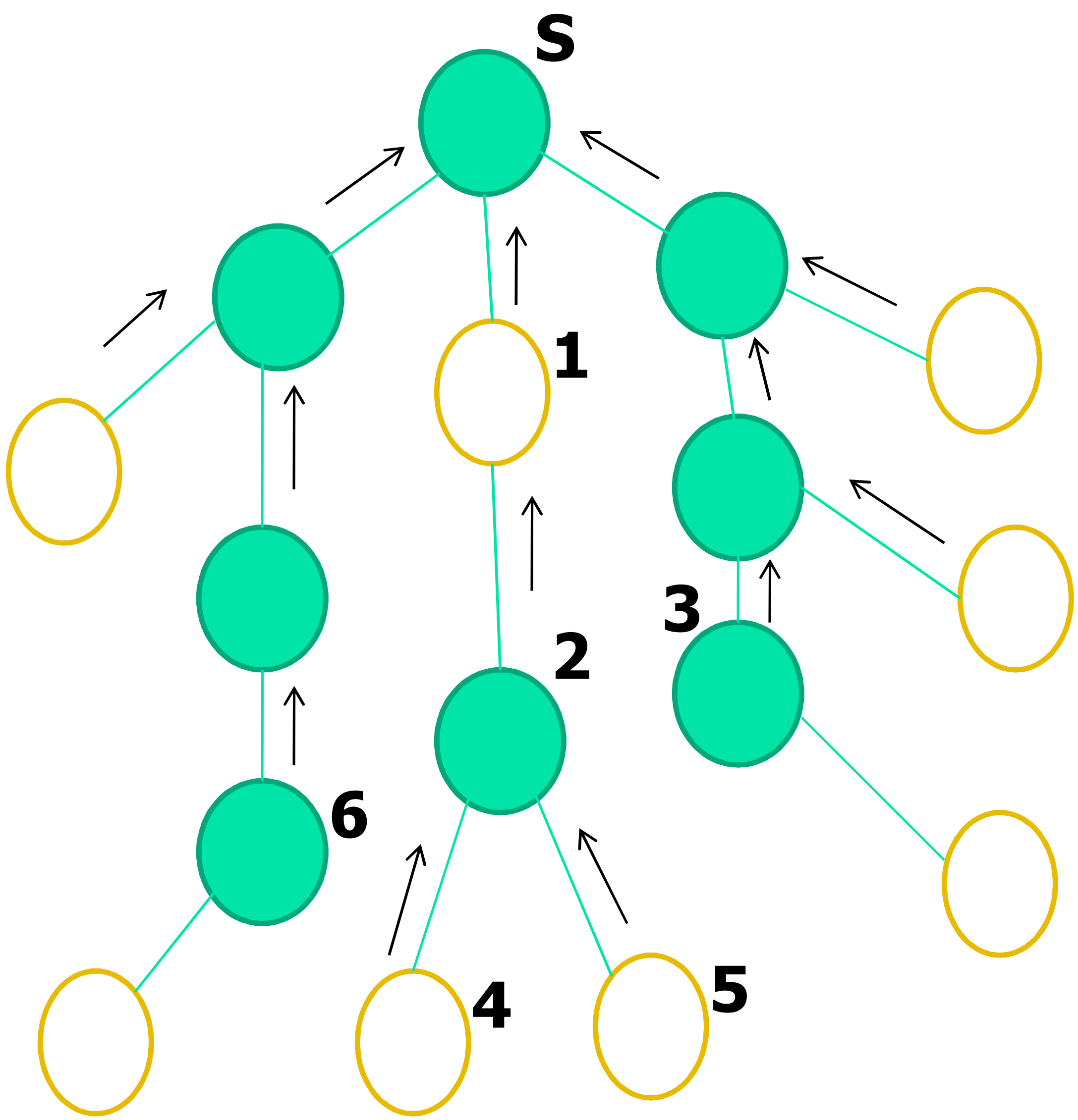


Stage 1 :

Parent Pointer change from

2 - > 3 (Cost = 4)
to
2 - > 1 (Cost = 2)

Illustration for CL parent pointer redirection recursively



Stage 2 :

Parent Pointer change from

4 - > 6 (Cost = 4)
to
4 - > 2 (Cost = 3)

Better Heuristic Performs Better

Theorem

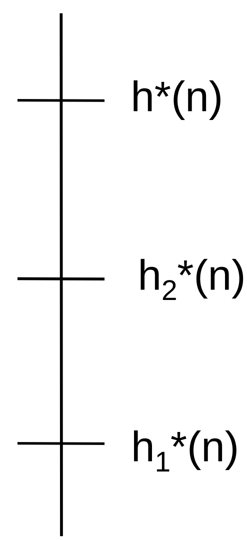
A version A_2^* of A* that has a “better” heuristic than another version A_1^* of A* performs at least “as well as” A_1^*

Meaning of “better”

$h_2(n) > h_1(n)$ for all n

Meaning of “as well as”

A_1^* expands at least all the nodes of A_2^*



For all nodes n, except the goal node

Proof by induction on the search tree of A_2*.

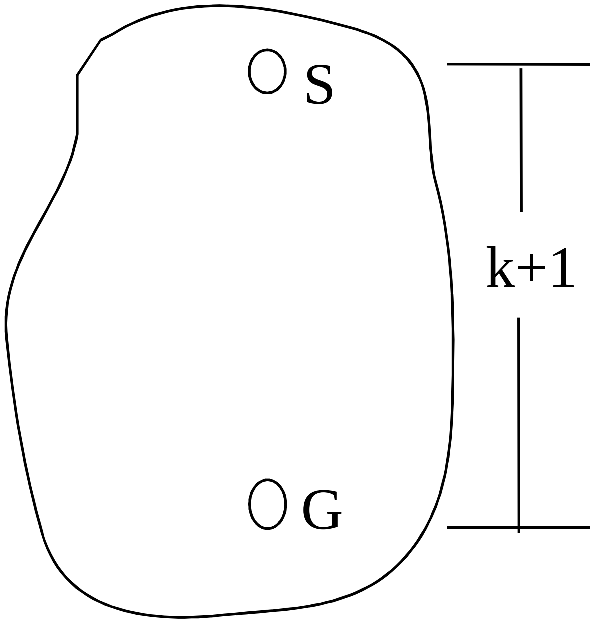
A* on termination carves out a tree out of *G*

Induction
on the depth *k* of the search tree of A_2*. A_1* before termination expands all the nodes of depth *k* in the search tree of A_2*.

k=0. True since start node *S* is expanded by both

Suppose A_1* terminates without expanding a node *n* at depth *(k+1)* of A_2* search tree.

Since A_1* has seen all the parents of *n* seen by A_2*
$g_1(n) <= g_2(n)$ (1)



Since A_1* has terminated without expanding *n*,

$f_1(n) >= f^*(S)$ (2)

Any node whose *f* value is strictly less than $f^*(S)$ has to be expanded.

Since A_2* has expanded *n*

$f_2(n) <= f^*(S)$ (3)

From (1), (2), and (3)

$h_1(n) >= h_2(n)$ which is a contradiction. Therefore, A_1* has to expand all nodes that A_2* has expanded.

<u>Exercise</u>

If better means $h_2(n) > h_1(n)$ for some *n* and $h_2(n) = h_1(n)$ for others, then Can you prove the result ?

Monotonicity

Definition

- A heuristic *h(p)* is said to satisfy the monotone restriction, if for all '*p*', $h(p) <= h(p_c) + cost(p, p_c)$, where '$p_c$' is the child of '*p*'.

Theorem

- If monotone restriction (also called triangular inequality) is satisfied, then for nodes in the closed list, redirection of parent pointer is not necessary. In other words, if any node '*n*' is chosen for expansion from the open list, then $g(n)=g^*(n)$, where $g(n)$ is the cost of the path from the start node '*s*' to '*n*' at that point of the search when '*n*' is chosen, and $g^*(n)$ is the cost of the optimal path from '*s*' to '*n*'

Grounding the Monotone Restriction

7	3	
1	2	4
8	5	6

n

1	2	3
4	5	6
7	8	

g_l

7	3	4
1	2	
8	5	6

n′

h(n) -: number of displaced tiles

Is h(n) monotone ?
h(n) = 8
h(n′) = 8
C(n,n′) = 1

Hence monotone

Monotonicity of # of Displaced Tile Heuristic

- $h(n) <= h(n') + c(n, n')$
- Any move reduces h(n) by at most 1
- $c = 1$
- Hence, $h(parent) <= h(child) + 1$
- If the empty cell is also included in the cost, then *h* need not be monotone (try!)

Monotonicity of Manhattan Distance Heuristic (1/2)

- *Manhattan distance= X-dist+Y-dist* from the target position
- Refer to the diagram in the first slide:
- $h_{mn}(n) = 1 + 1 + 1 + 2 + 1 + 1 + 2 + 1 = 10$
- $h_{mn}(n') = 1 + 1 + 1 + 3 + 1 + 1 + 2 + 1 = 11$
- *Cost = 1*
- Again, $h(n) <= h(n') + c(n, n')$

Monotonicity of Manhattan Distance Heuristic (2/2)

- Any move can either increase the h value or decrease it by **at most 1.**
- Cost again is 1.
- Hence, this heuristic also satisfies Monotone Restriction
- If empty cell is also included in the cost then manhattan distance does not satisfy monotone restriction (try!)
- Apply this heuristic for Missionaries and Cannibals problem

Relationship between Monotonicity and Admissibility

- Observation:

  Monotone Restriction $\rightarrow$ Admissibility but not vice-versa

- Statement: *If* $h(n_i) <= h(n_j) + c(n_i, n_j)$ *for all i, j*

  then $h(n_i) < = h^*(n_i)$ *for all i*

Proof of Monotonicity→admissibility

Let us consider the following as the optimal path starting with a node $n = n_1 - n_2 - n_3 \ldots n_i - \ldots n_m = g_l$

Observe that

$$h^*(n) = c(n_1, n_2) + c(n_2, n_3) + \ldots + c(n_{m-1}, g_l)$$

Since the path given above is the optimal path from *n to* g_l

Now,

$h(n_1) <= h(n_2) + c(n_1, n_2)$ ------ Eq 1

$h(n_2) <= h(n_3) + c(n_2, n_3)$ ------ Eq 2

: : : : : :

$h(n_{m-1}) = h(g_i) + c(n_{m-1}, g_i)$------ Eq (m-1)

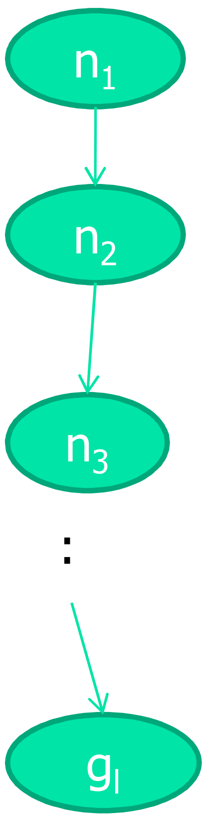
Adding Eq 1 to Eq (m-1) we get

$$h(n) <= h(g_l) + h^*(n) = h^*(n)$$

Hence proved that MR → (h <= h*)

Proof (continued…)

Counter example for vice-versa



$h^*(n_1) = 3$	$h(n_1) = 2.5$
$h^*(n_2) = 2$	$h(n_2) = 1.2$
$h^*(n_3) = 1$	$h(n_3) = 0.5$
: :	: :
$h^*(g_l) = 0$	$h(g_l) = 0$

$h < h^*$ everywhere but MR is not satisfied

Proof of MR leading to optimal path for every expanded node (1/2)

Let S-N_1- N_2- N_3- N_4... N_m ...N_k be an optimal path from S to N_k (all of which might or might not have been explored). Let N_m be the **last** node on this path which is on the open list, i.e., *all* the ancestors from S up to N_{m-1} are in the closed list.

For every node N_p on the optimal path,

$g^*(N_p)+h(N_p) <= g^*(N_p)+C(N_p,N_{p+1})+h(N_{p+1})$, by monotone restriction
$g^*(N_p)+h(N_p) <= g^*(N_{p+1})+h(N_{p+1})$ on the optimal path
$g^*(N_m)+ h(N_m) <= g^*(N_k)+ h(N_k)$ by transitivity

Since all ancestors of N_m in the optimal path are in the closed list,

$g(N_m)= g^*(N_m)$.
$\Rightarrow f(N_m)= g(N_m)+ h(N_m)= g^*(N_m)+ h(N_m) <= g^*(N_k)+ h(N_k)$

Proof of MR leading to optimal path for every expanded node (2/2)

Now if N_k is chosen in preference to N_m,

$$f(N_k) <= f(N_m)$$

$$g(N_k)+ h(N_k) <= g(N_m)+ h(N_m)$$

$$= g^*(N_m)+ h(N_m)$$

$$<= g^*((N_k)+ h(N_k)$$

$$g(N_k)<=g^*(N_k)$$

But $g(N_k)>=g^*(N_k)$, by definition

Hence $g(N_k)=g^*(N_k)$

This means that if N_k is chosen for expansion, the optimal path to this from S has already been found

TRY proving by induction on the length of optimal path

Monotonicity of *f()* values

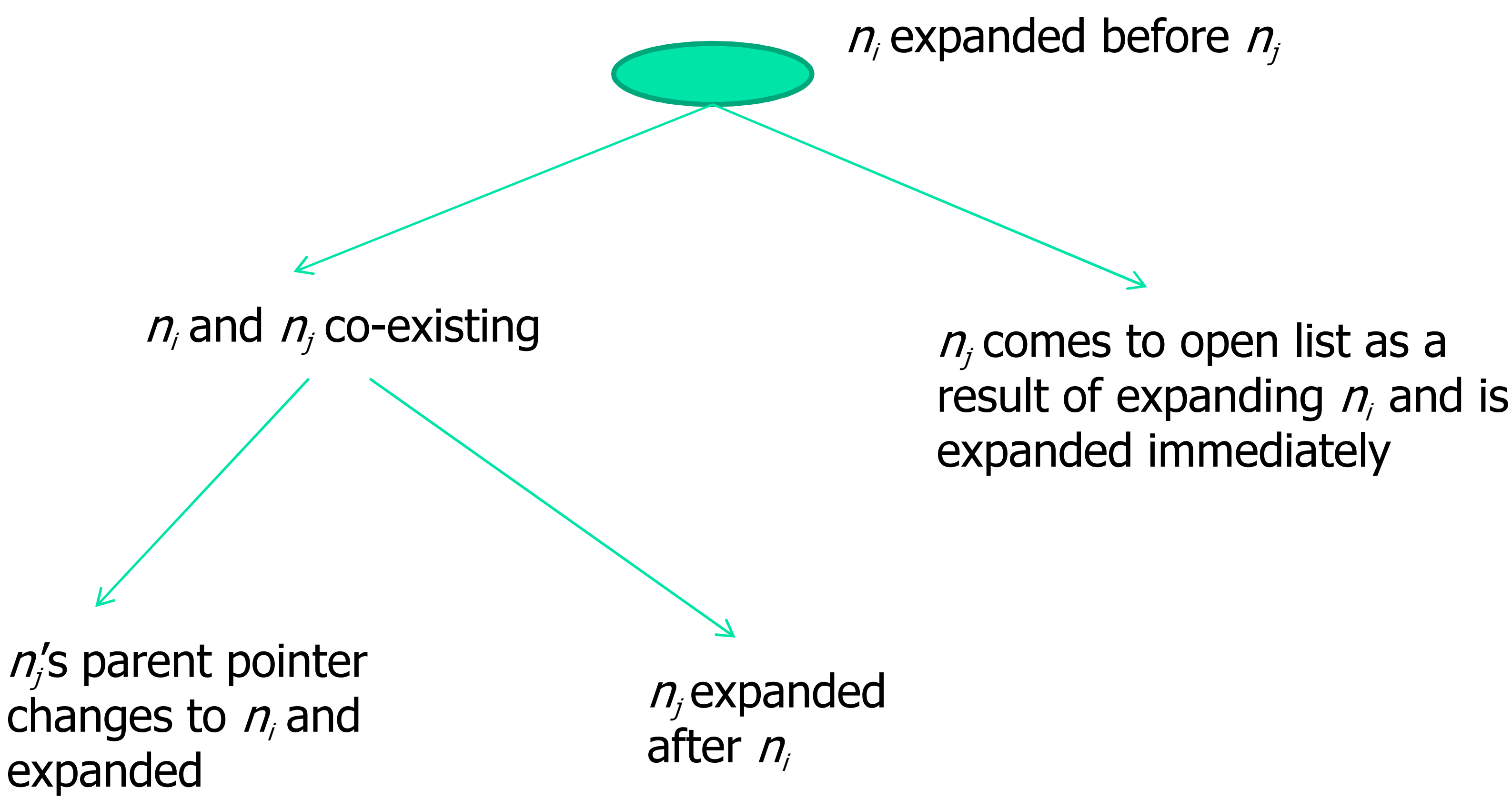
Statement:

f values of nodes expanded by A* increase monotonically, if *h* is monotone.

Proof:

Suppose n_i and n_j are expanded with temporal sequentiality, *i.e.*, n_j is expanded after n_i

Proof (1/3)…


n_i expanded before n_j
n_i and n_j co-existing
n_j comes to open list as a result of expanding n_i and is expanded immediately
n_j's parent pointer changes to n_i and expanded
n_j expanded after n_i

Proof (2/3)…

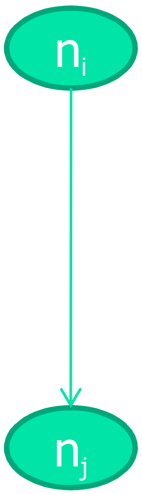
- All the previous cases are forms of the following two cases (think!)
- CASE 1:

  n_j was on open list when n_i was expanded

  Hence, $f(n_i) <= f(n_j)$ by property of A*
- CASE 2:

  n_j comes to open list due to expansion of n_i

Proof (3/3)…



Case 2:

$f(n_i) = g(n_i) + h(n_i)$ (Defn of *f*)

$f(n_j) = g(n_j) + h(n_j)$ (Defn of *f*)

$f(n_i) = g(n_i) + h(n_i) = g^*(n_i) + h(n_i)$ ---*Eq 1*

(since n_i is picked for expansion n_i is on optimal path)

With the similar argument for n_j we can write the following:

$f(n_j) = g(n_j) + h(n_j) = g^*(n_j) + h(n_j)$ ---*Eq 2*

Also,

$h(n_i) <= h(n_j) + c(n_i, n_j)$ ---*Eq 3* (Parent- child relation)

$g^*(n_j) = g^*(n_i) + c(n_i, n_j)$ ---Eq 4 (both nodes on optimal path)

From *Eq 1, 2, 3 and 4*

$f(n_i) <= f(n_j)$

Hence proved.

Better way to understand monotonicity of *f()*

- Let $f(n_1), f(n_2), f(n_3), f(n_4)\ldots f(n_{k-1}), f(n_k)$ be the f values of k expanded nodes.
- The relationship between two consecutive expansions $f(n_i)$ and $f(n_{i+1})$ nodes always remains the same, *i.e.,* $f(n_i) <= f(n_{i+1})$
- This is because
  - $f(n_i)= g(n_i) +h(n_i)$ and
  - $g(n_i)=g^*(n_i)$ since n_i is an expanded node (proved theorem) and this value cannot change
  - $h(n_i)$ value also cannot change Hence nothing after n_{i+1}'s expansion can change the above relationship.

Monotonicity of f()

$f(n_1), f(n_2), f(n_3), \ldots\ldots, f(n_i), f(n_{i+1}), \ldots, f(n_k)$

Sequence of expansion of $n_1, n_2, n_3 \ldots n_i \ldots n_k$

f values increase monotonically

$$f(n) = g(n) + h(n)$$

Consider two successive expansions - > n_i, n_{i+1}

Case 1:

n_i & n_{i+1} Co-existing in OL

n_i precedes n_{i+1}

By definition of A*

$$f(n_i) <= f(n_{i+1})$$

Monotonicity of f()

Case 2:

ni+1 came to OL because of expanding ni and ni+1 is expanded

f(ni) = g(ni) + h(ni)
<= g(ni) + c(ni, hi+1)+ h(ni+1)
= g(ni) + h(ni+1)
= f(ni+1)

Case 3:

ni+1 becomes child of ni after expanding ni and ni+1 is expanded. Same as case 2.

A list of AI Search Algorithms

- A*
  - AO*
  - IDA* (Iterative Deepening)
- Minimax Search on Game Trees
- Viterbi Search on Probabilistic FSA
- Hill Climbing
- Simulated Annealing
- Gradient Descent
- Stack Based Search
- Genetic Algorithms
- Memetic Algorithms