the least costly paths in G from node s to the descendants of node n. In addition to the burden of adjusting the search tree, it is often computationally quite expensive to test whether a node has been generated before. We now show that given a rather mild and reasonable restriction on h, when **A*** selects a node for expansion it has already found an optimal path to that node. Thus, with this restriction, there is no need for **A*** to test to see if a newly generated node is already on *CLOSED*, and there is no need to change the parentage in the search tree of any successors of this node in the search graph.

A heuristic function, h, is said to satisfy the *monotone restriction* if 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

$$h(t) = 0.$$

If we write the monotone restriction in the form

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

it is seen to be similar to a triangle inequality. It specifies that the estimate of the optimal cost to a goal from node n_i not be more than the cost of the arc from n_i to n_j plus the estimate of the optimal cost from n_j to a goal. We might say that the monotone restriction imposes the rather reasonable condition that the heuristic function be locally consistent with the arc costs.

In the 8-puzzle, it is easily verified that $h(n) = W(n)$ satisfies the monotone restriction. If the function h is changed in any manner *during* the search process, then the monotone restriction might not be satisfied.

We now show that, given the monotone restriction, when **A*** expands a node, it has found an optimal path to that node. Let n be any node selected for expansion by **A***. If $n = s$, **A*** has trivially found an optimal path to s; so let us suppose that n is not s. Let the sequence $P = (s = n_0, n_1, n_2, \ldots, n_k = n)$ be an optimal path from s to n. Let node n_l be the last node in this sequence that is on *CLOSED* at the time **A*** selects n for expansion. (Node s is on *CLOSED*, but node n_k is not, because it is just now being selected for expansion.) Thus, node n_{l+1} in the sequence P is on *OPEN* at the time **A*** selects node n.

Using the monotone restriction, we have that

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

Since n_i and n_{i+1} are on an optimal path

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

therefore

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

By transitivity, we then have

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

or

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

Therefore, at the time **A*** selected node n, in preference to node n_{l+1}, it must have been the case that $g(n) \leq g^*(n)$; otherwise, $f(n)$ would have been greater than $f(n_{l+1})$. Since $g(m) \geq g^*(m)$ for all nodes m in the search tree, we have

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

The monotone restriction also implies another interesting result, namely, that the f values of the sequence of nodes expanded by **A*** are nondecreasing. Suppose node n_2 is expanded immediately after node n_1. If n_2 was on *OPEN* at the time n_1 was expanded, we have (trivially) that $f(n_1) \leq f(n_2)$. Suppose n_2 is not on *OPEN* at the time n_1 is expanded. (Node n_2 is not on *CLOSED* either, because we are assuming that it has not been expanded yet.) Then, if n_2 is expanded immediately after n_1, it must have been added to *OPEN* by the process of expanding n_1. Therefore, n_2 is a successor of n_1. Under these conditions, when n_2 is selected for expansion we have