Optimal Efficiency of A* Revisited

Research Lecture on work by
Mike Barley & Jorn Christensen

39 years ago, in 1968, Peter Hart, Nils Nilsson, & Bertram Raphael published

"A Formal Basis for the Heuristic Determination of Minimum Cost Paths"

It described a new search algorithm: A*

A* is the most widely used search algorithm today!

A* Algorithm

Mark *s* open and calculate *f(s)*.

While there are open nodes Do

    Select open node *n* with smallest *f* value

        resolve ties in favor of goal nodes,

    If *goal(n)* then terminate with success

    Mark *n* closed

    For all successors, *j*, of *n* Do

        calc *f(j)*

        if *j* not in closed or *f(j)* is lower

        mark *j* open.

A* Optimal Efficiency Result #1

Let A be any optimal algorithm dominated[1] by
A* where *f(n)* = *f(m)* implies *n* = *m*[2],

Then A* is at least as efficient[3] as A.

1. A^*_{h1} *dominates* A^*_{h2} if and only if
   for all non-goal nodes, n, $h1(n) >= h2(n)$.
2. The *no ties* clause.
3. X is as *efficient* as Y means every node expanded by X is also expanded by Y.

No Ties Clause

- OE Result #1 is not very useful, because the “no ties” limitations is too restrictive.

- We need to allow ties.

Handling Ties

The specification of A* does not state exactly how to handle ties. This means that there are many different refinements of A*, each differing in the way they order ties. ***A**** is the set of A* refinements that have different tie handling strategies.

A* Optimal Efficiency Result #2

Let A be any optimal algorithm that is "dominated" by every algorithm in ***A****

Then there exists an A* in ***A**** such that A* is at least as efficient as A.

Is this good enough?

- This only tells us that some A* is optimally efficient but not which one!!!!!
- Also it doesn't tell us what algorithms any given A* is as efficient as.

A* Optimal Efficiency Result #3

If h_1 is *less informed*[1] *than* h_2
then A^*_{h2} is at least as efficient as A^*_{h1}

1. Heuristic h_1 *is less informed than* h_2 iff
for all non-goal nodes, n, $h_1(n) < h_2(n)$.

Where that leaves us

- *Non-Optimal Efficiency:* Don’t know whether an A^*_h will be “as or more efficient” than any other optimal search algorithms that *h* dominates.
- *Heuristic Non-Equivalence:* Given $A^{*\prime}_h$ & A^*_h, they may not expand the same number of nodes.
- *Non-Monotonic Improvement:* Given an “improved” heuristic h_1, A^*_{h1} may be less efficient than A^*_h.
- *Inconsistent Performance:* If we run A^*_h twice in a row, we don’t know whether we will get the same number of nodes both times.

What happened?

- If we don‘t allow ties then we get the result we want.

- If we allow ties then we have two very weak (and unsatisfactory) results.

A* Algorithm - Handling Ties

Mark *s* open and calculate *f(s)*.

While there are open nodes Do

  Select open node *n* with smallest *f* value

    resolve ties in favor of goal nodes,

  If *goal(n)* then terminate with success

  Mark *n* closed

  For all successors, *j*, of *n* Do

    calc *f(j)*

    if *j* not in closed or *f(j)* is lower

    mark *j* open.

Handling Ties

The specification of A* does state exactly that in case of ties in the open list, to always choose goal nodes over non-goal nodes with the same *f*-value.

So why are ties a problem & are all ties a problem?

Critical Ties:The Problem?

HNR defined *critical ties* as those nodes with the same *f*-value as optimal goal node.

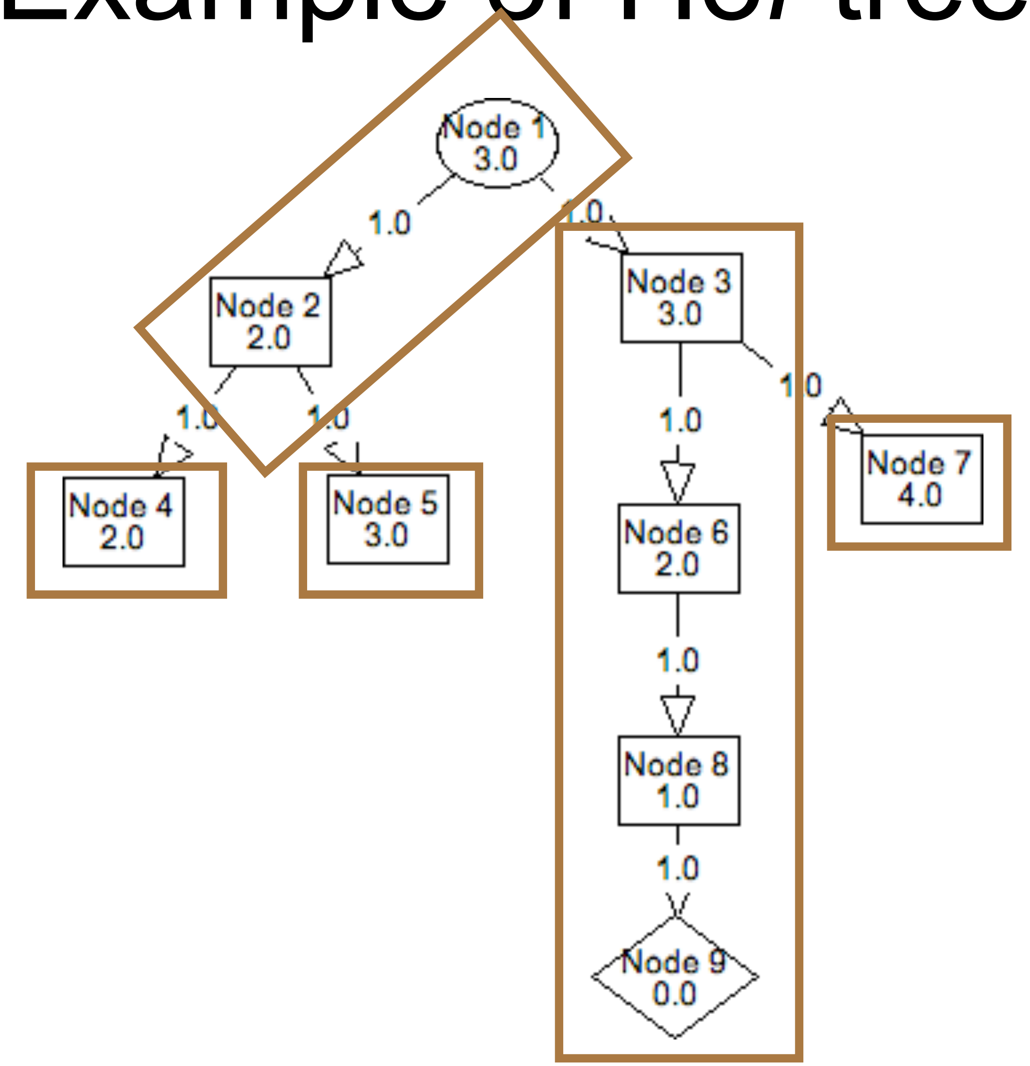
For them, the existence of critical ties was the reason why result #1 had to have the no ties clause.

Are all critical ties a problem?

Tracking down the culprit: Ho*f* trees

- A homogeneous *f*-value (Ho*f*) tree is a sub-tree of a search tree where all nodes have the same *f*-value, say **f**, where the parent (if there is one) of the root has a different *f*-value, and where all the children (if there are any) of the leaves have non-**f** *f*-values.

Example of Ho*f* trees



Why are Ho*f* trees problematic?

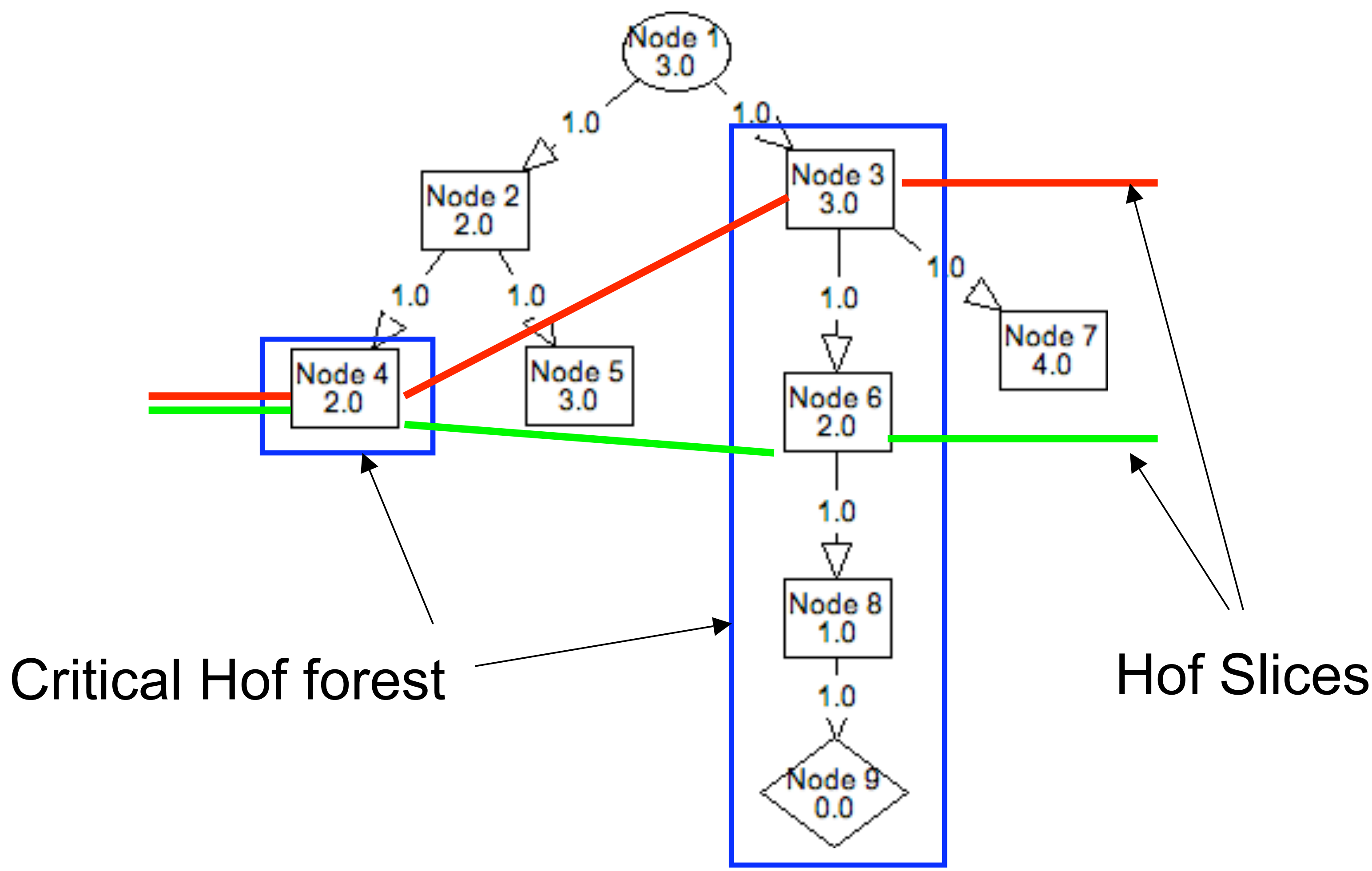
- To understand this we need to see what effect Ho*f* trees have on the open list.

- To understand we need to extend our vocabulary a little.

More Terminology

- A <u>Ho*f* forest</u> is the set of all the Ho*f* trees in a search tree with the same *f*-value.
- A <u>Ho*f* slice</u> is a “cut” through a Ho*f* forest.
- An <u>*f*-open list</u> is the open list sublist that contains all the nodes with the same *f*-value.
- A <u>critical *f*-open list</u> is a *f*-open list with same *f*-value as the optimal goal.

Example of a Hof forest & slices



Tying Things Together

- An f-open list is a slice through a Hof forest.
- A critical f-open list is the set of critical ties that A* can see at a point in time.
- If the goal nodes are not in the current critical f-open list then A* doesn't know where any of the optimal goals are.

Tying Things Together cont'd

- The different A*'s in ***A**** represent the different ways to access the nodes in the critical f-open list.
- Some A*'s will be luckier than others and pick the right node that leads to a goal node in the critical Hof forest.

An Aside

- While A* can’t know which node in the current critical f-open list leads to a goal, it can increase its odds by choosing the node with the lowest *h* value.

Where We are Now

- We should understand now why allowing ties forces us to have such weak results.
- We now show how to get stronger results.
- First, we will look at what we want to get (but can't currently).