AND/OR Graphs

Decomposible Production Systems

Up until now
each branch was a OR branch.

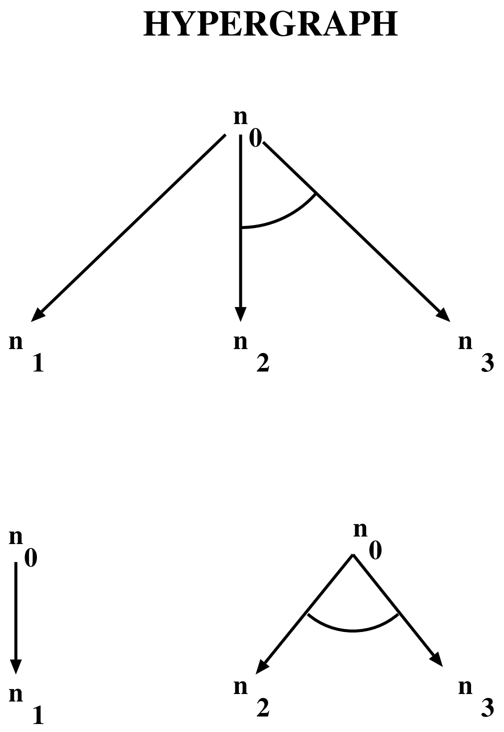
The solution was a single path of solutions

Introduce
AND branching

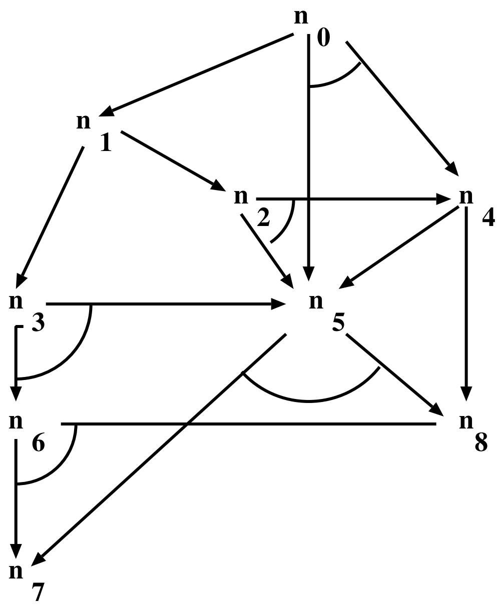
If a parent has an AND branch
all
of the sons must be fulfilled
to lead to a solution

The solution is now a graph of nodes

Connectors and Successors

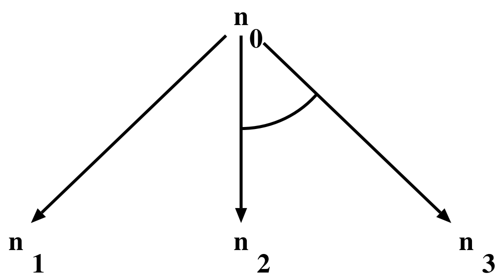


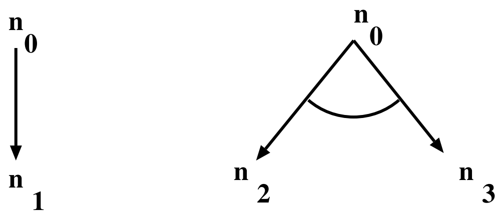
AND/OR Graph



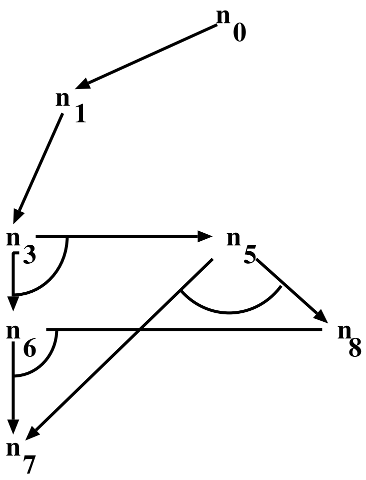
Solutions

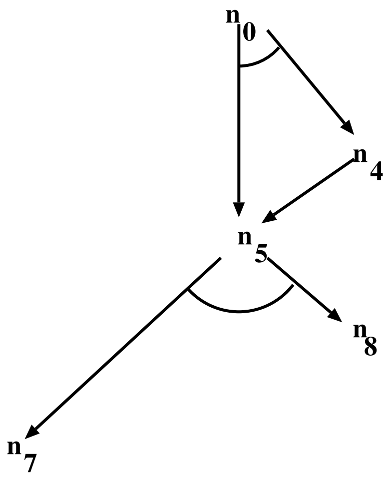
HYPERGRAPH





AND/OR Graph Solutions





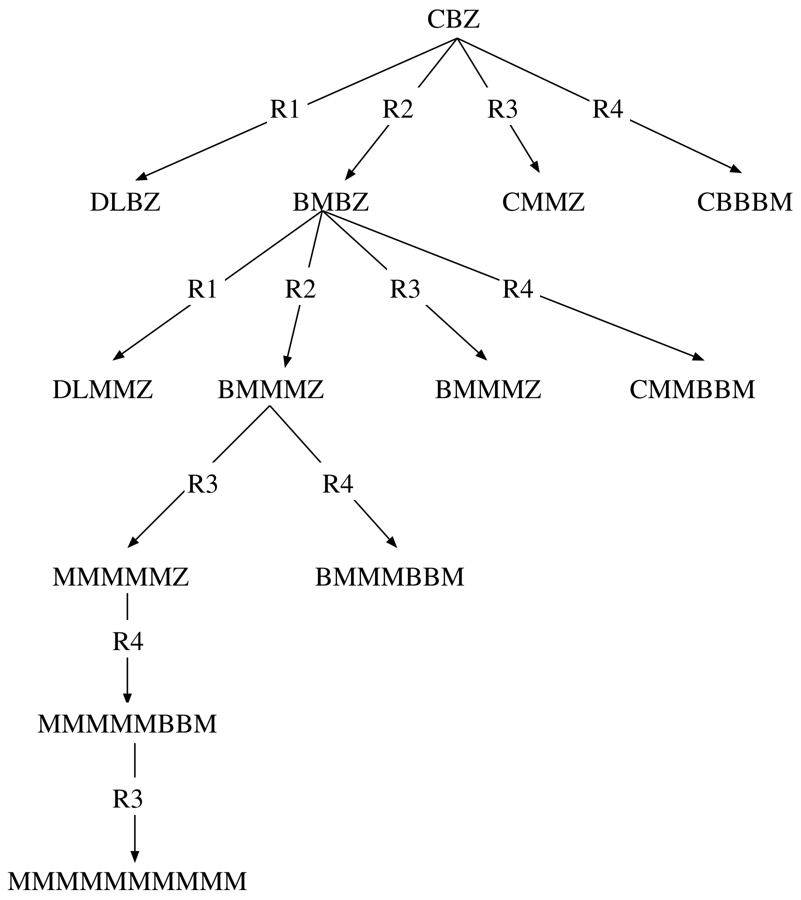
Comparison

Suppose we have a set of rewrite rules

$$
\begin{aligned}
R1 : C &\longrightarrow (D, L)\\
R2 : C &\longrightarrow (B, M)\\
R3 : B &\longrightarrow (M, M)\\
R4 : Z &\longrightarrow (B, B, M)
\end{aligned}
$$

Goal to have a sequence of all M's

Search Space



Redundancy

Such a search space can lead to many redundant paths

In solution space:

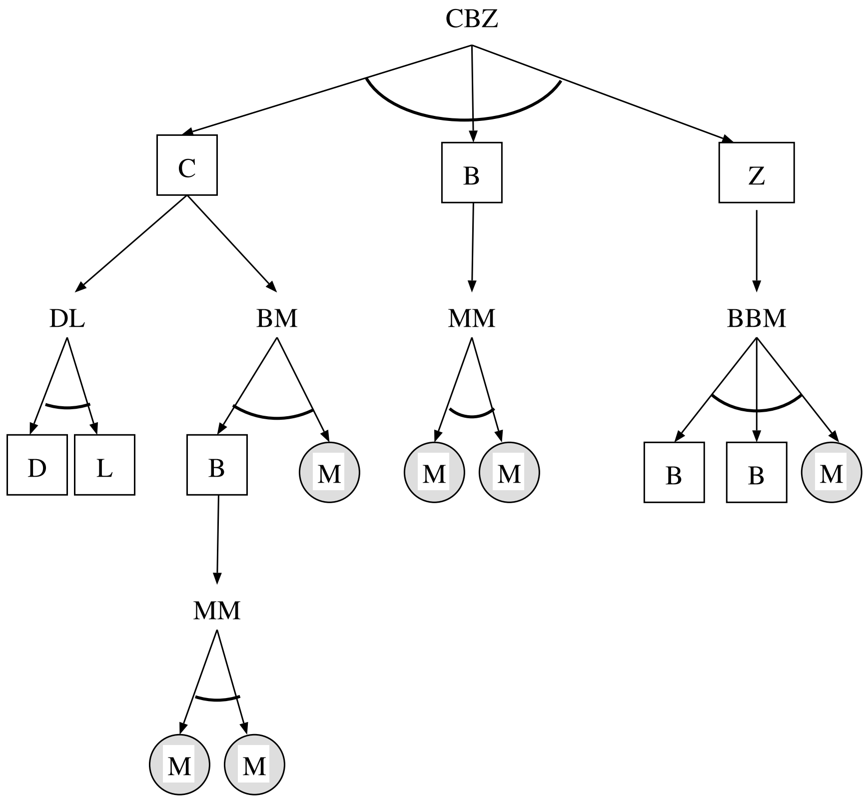
$$CBZ \xrightarrow{R2} \xrightarrow{R3} BMMMZ$$

$$CBZ \xrightarrow{R3} \xrightarrow{R2} BMMMZ$$

(Here we are looking for one solution
in a large solution space)

One way to avoid redundacy
is to recognize that there
independent components which can be processed
individually (in parallel)

Decomposed Problem



Generalized Best First Search

G

The entire search space

G'

The currently explored portion of G

G' is a subgraph of G

G' is connected

G' contains s (the start node)

Frontier Nodes

Nodes to be expanded

(That have not been labeled *unsolvable*)

Solution Base

G”

Contains the start node

If an expanded AND node n is in G”
Then all n’s successors are also in G”

If an expanded OR node n is in G”
then exactly one successor of n is in G”

None of the nodes in G” are labeled unsolvable

Which Node should be Expanded

With just OR graphs

One-to-One correspondence
Candidates for Expansion
and
Candidate Solution Bases

Each path from the root to the frontier
is represented by one frontier node to be expanded

With AND/OR graphs this is not the case

Each solution may contain many candidates for expansion
(The result of the AND connections)

One node can participate
in many solution candidates

Thus, what is the best solution

Best First Principle: AND/OR graphs

f_1
Graph Evalution Function

Estimates properties of the set of solutions
emanating
from a given candidate base

f_2
Node Evalution Function

Estimates the amount of information
that a given node may provide
regarding
alledged superiority of its hosting graph

Most Promising Solution

From the explicite search graph, G'
(initially just s)

Compute the most promising solution base G_0,
using f_1, Graph Evalution Function

Using also the set of *heuristics* provided
(see below)

Expand Best Node

Using f_1
Select node n

n is both on OPEN and in G_0

Remove n from OPEN and put on CLOSED

Expand node n
generating all successors

For each successor
Put a pointer back to n
Provide a set of *heuristic* information
(to be used by f_1)

Terminal Node?

If any successor is a terminal node

Solved
if a goal
bf Unsolved
if not

Labeling Procedure
If the node it induces solved or unsolved
on any of its ancestors, label them so (see below)

If the start node is solved
then label with G_0

If the start node is Unsolved
then exit with failure

Remove any nodes of G'
Which do not contribute to solution
(Space saving property)

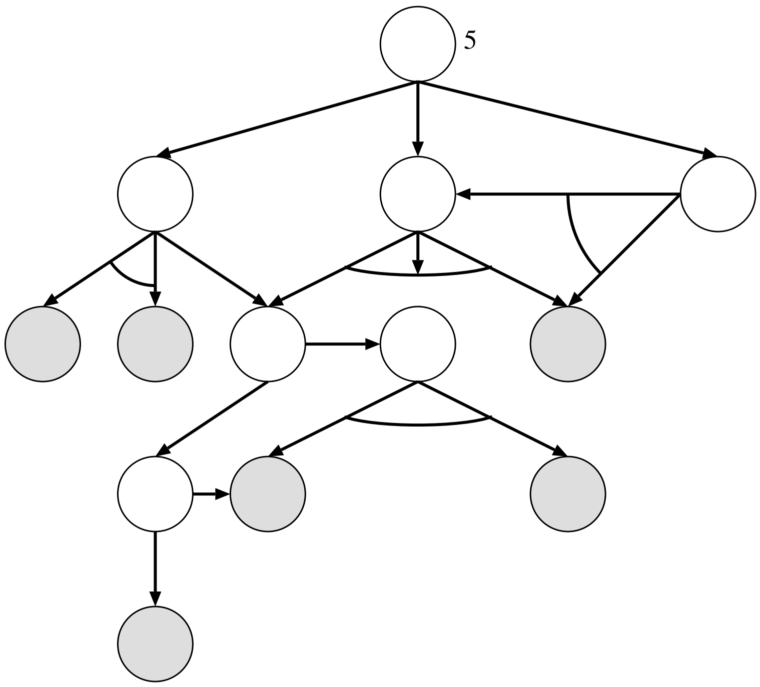
Labeling Procedure

Terminal Node
Labeled **Solved** if it is a goal node
Labeled **Unsolved** if it cannot lead to solution

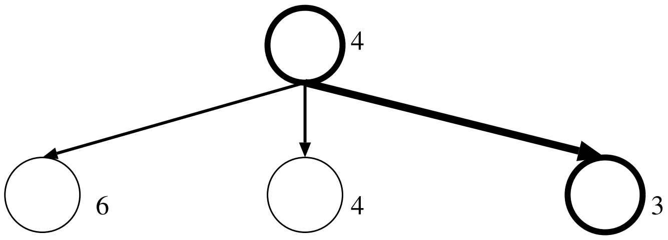
Nonterminal AND
If *all* successors *Solved* then label solved
If *any* successor is *Unsolved* then label *unsolved*

Nonterminal OR If *any* successor *Solved* then solved
If *all* successors *Unsolved* then solved

An Example



Root Node Expanded



Three Solution Bases (3 OR branches)

Estimates will be based on
Small-Is-Quick

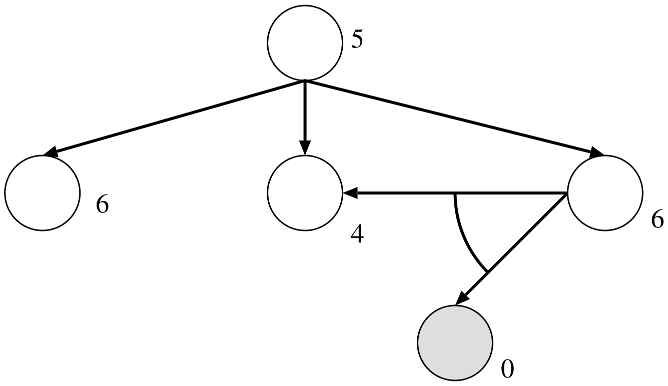
How many nodes until a complete solution

The set of heuristics would be the set of information about the system needed to estimate how many nodes are to come

The score of the root node is
the score of the node
plus
one (the connection)

The rightmost node is expanded next

Rightmost Node Expanded



Rightmost Node Expanded

The center node is common node

The heuristic is updated appropriately

One of the AND branches of the expanded node is a goal node

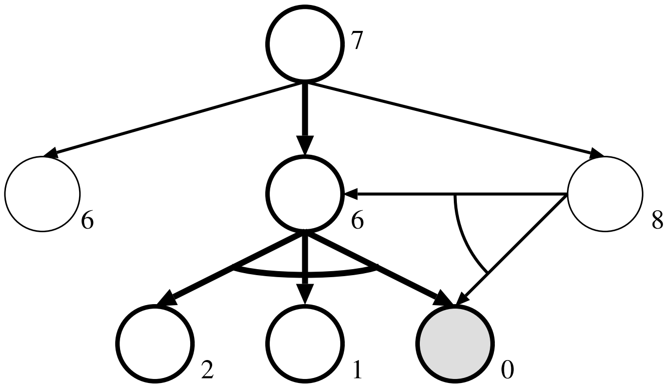
Goal nodes have heuristic 0
(no nodes follow)

Since the middle node is now the best node to a solution,
the root node score reflects going through this node

The middle node is expanded

The rightmost node was underestimated
It cannot lead to a solution, because it involves the middle node

Third Node Expanded



Third Node Expanded

All nodes updated

Note that up to this point
f_1 was used to select the best
Solution Graph

Now, within the solution graph
There are two nodes to expand

The function, f_2 is used
to select which to expand
(It is not the same criteria as selection of solution graph)

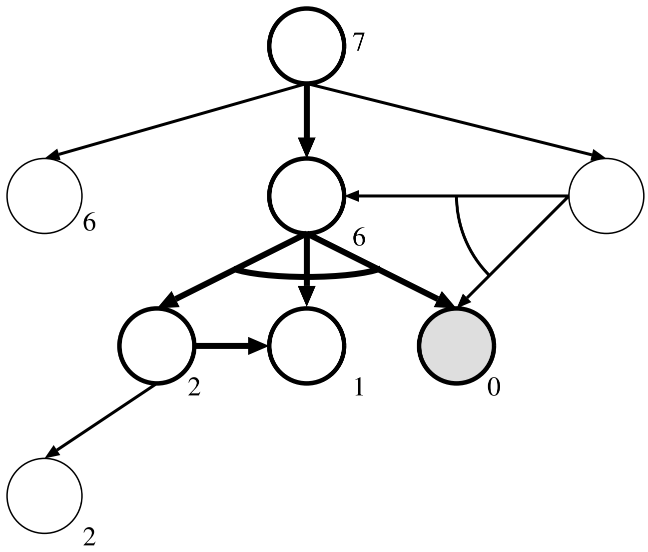
Third Node Expanded

f_2 strategy
Since all nodes will have to be expanded eventually
(if the solution graph stays the same)
Select the node which most likely to show an error in choice

f_2 is of course an estimate
The subtree with the largest number of nodes
would probably be the one with the most
errors in the estimates

Expand the one with the highest estimate of the number of nodes

Fourth Node Expansion



Common Node

There is a common node in the solution

How do we count the number of nodes to solution?

Should it be counted with the second father

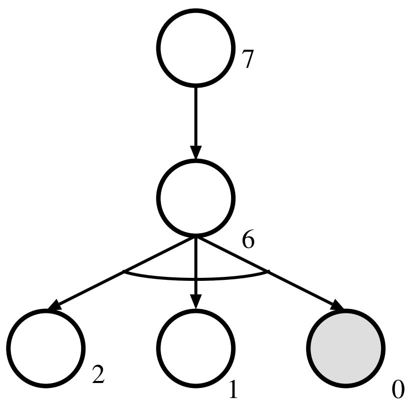
Or ignored because it has already been counted with the first

With Second Father
An over-estimate of the number of nodes

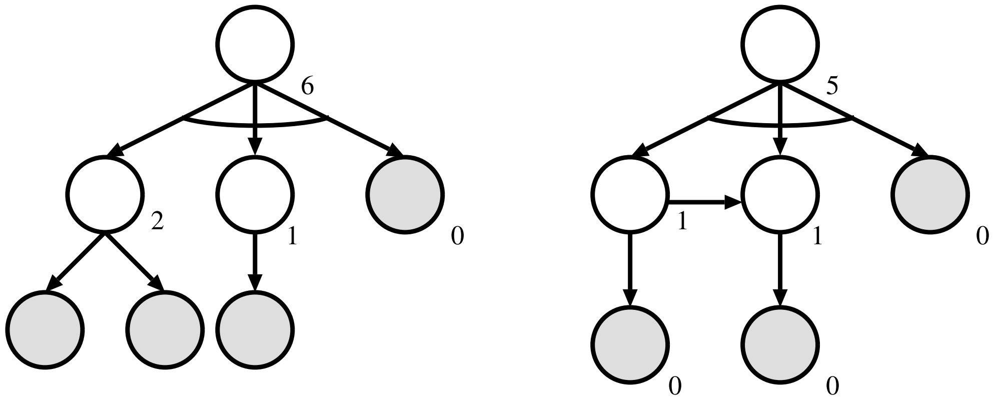
With First Father
complicates the estimate of the calculations
of the number of nodes needed

Depends on the number of common nodes
The nature of the solution space
Has to be recalculated for each solution

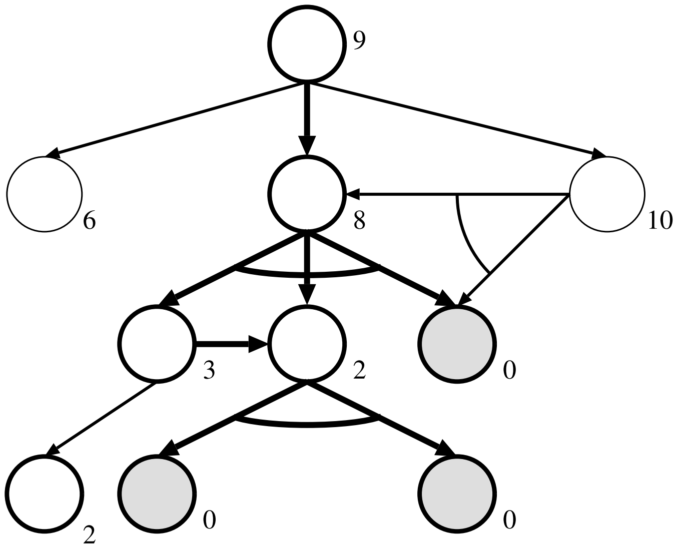
Nature of Solution Space



Varies according to Solution
Would have to recalculate depending on solution space



Fifth Node Expanded



Solution Node Found

Though a solution node was found

It was not optimal

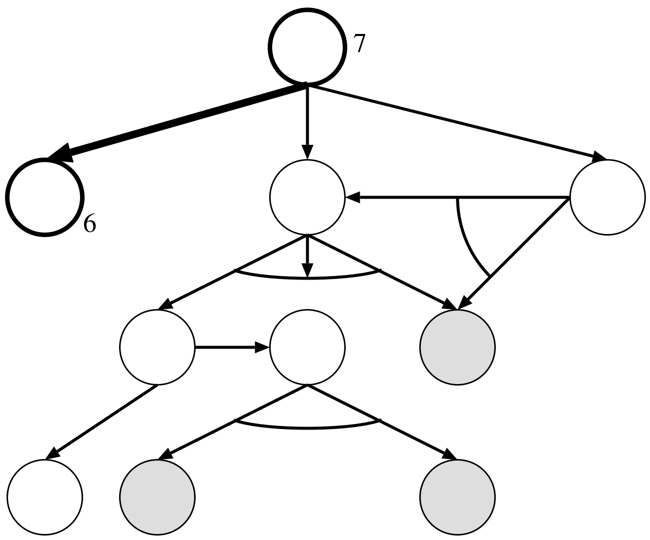
To find the Best Solution:

Optimistic Choice of f_1

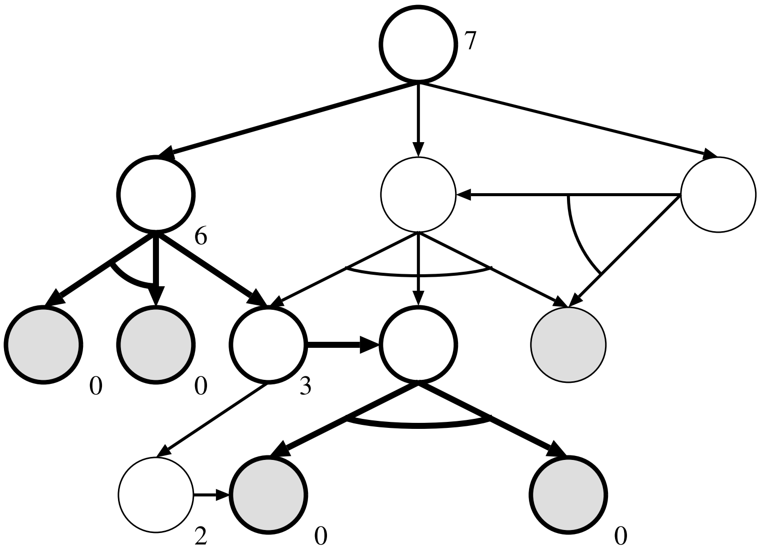
Modify Procedure:
Find Most Promising Solution Based Graph First
Then Check to see if it a solution

In the example the left-most node would be expanded

Toward Best Solution



Toward Best Solution



Optimization Function

Weight Function (W):
Measures Some Property of G

Can Represent
Cost (C)
Merit (Q)

Recursive Weight Function

$$W_G(n) = F(E(n); W_G(n_1), W_G(n_2), ...W_G(n_b))$$

$n_1, n_2...n_b$
The immediate successors of n
$E(n)$
local properties characterising node n
F
Arbitrary combination function

Bottom-Up
Evalution starting with the terminal nodes
and working back to the root
Sometimes called **Rollback Function**

Identifying G_0

Backed-up Evalution Function
$e(n)$

h(n)
If n is on OPEN

$F(E(n); e(n_1), e(n_2)..., e(n_b))$
If n is a CLOSED AND node

$min_i F(E(n); e(n_i))$

Finding G_0 can be done
Bottom up
or
By marking the smallest OR branch

The Selection Function
$f_1(G') = e_{G'}(s)$