

ECE 566

Constraint Satisfaction Problems and Techniques



Algorithms for Arc-Consistency achievement

An arc (V_i, V_j) can be made consistent by simply deleting values from the domain of $D_i(V_i)$ for which we cannot find consistent values in the domain of the variable V_j . The following algorithm which does that will be used as a part of the algorithm for Arc-consistency.

Algorithm REVISE

```
procedure REVISE( (Vi, Vj), (Z, D, C) )
  DELETE <- false;
  for each X in Di do
    if there is no such Vj in Dj such that ( X, Vj ) is consistent,
      then
        delete X from Di;
        DELETE <- true;
    endif;
  endfor ;
return DELETE;
end REVISE
```



```
procedure AC-3 (Z,D,C)  
   $Q \leftarrow \{ (V_i, V_j) \in \text{arcs}(G), i \neq j \};$   
  while  $Q$  not empty  
    select and delete any arc  $(V_k, V_m)$  from  $Q$ .  
    if  $(\text{REVISE}( (V_k, V_m), (Z, D, C) ))$  then  
       $Q \cup \{ (V_i, V_k) \text{ such that } (V_i, V_k) \in \text{arcs}(G), i \neq k, i \neq m \}$   
    endif;  
  endwhile;  
end AC-3
```



Searching:

Special search strategies that make use of CSP features.

- General Search Strategies:
 - Chronological Backtracking.
 - Iterative broadening.
- Look-ahead Strategies:
 - Forward Checking.
 - Directional Arc Consistency Look-ahead algorithm.
 - AC - Look-ahead algorithm.
- Gather information while searching strategies:
 - Dependency directed backtracking
 - Back-Jumping.
 - Graph based back-Jumping.



Illustrative examples for the Searching strategies.

We use the n queens problem to illustrate the concepts presented.

The n-queens is a standard benchmark in the CSP literature.

In this problem we have to place n queens on a n x n chess board such that no queen is in conflict with any other queen.

Notation:

We will represent a square which cannot be occupied by a X and a square which is occupied by a queen by •

	A	B	C	D
1	•	X	X	X
2	X	X		
3	X		X	
4	X			X

In the four queens problem shown, once we assign 1A, the squares marked X cannot be used.



Illustrative examples for the searching methods.

Backtracking(for comparison) (For a four queens problem)

A dead-end configuration
is represented by
a X under it

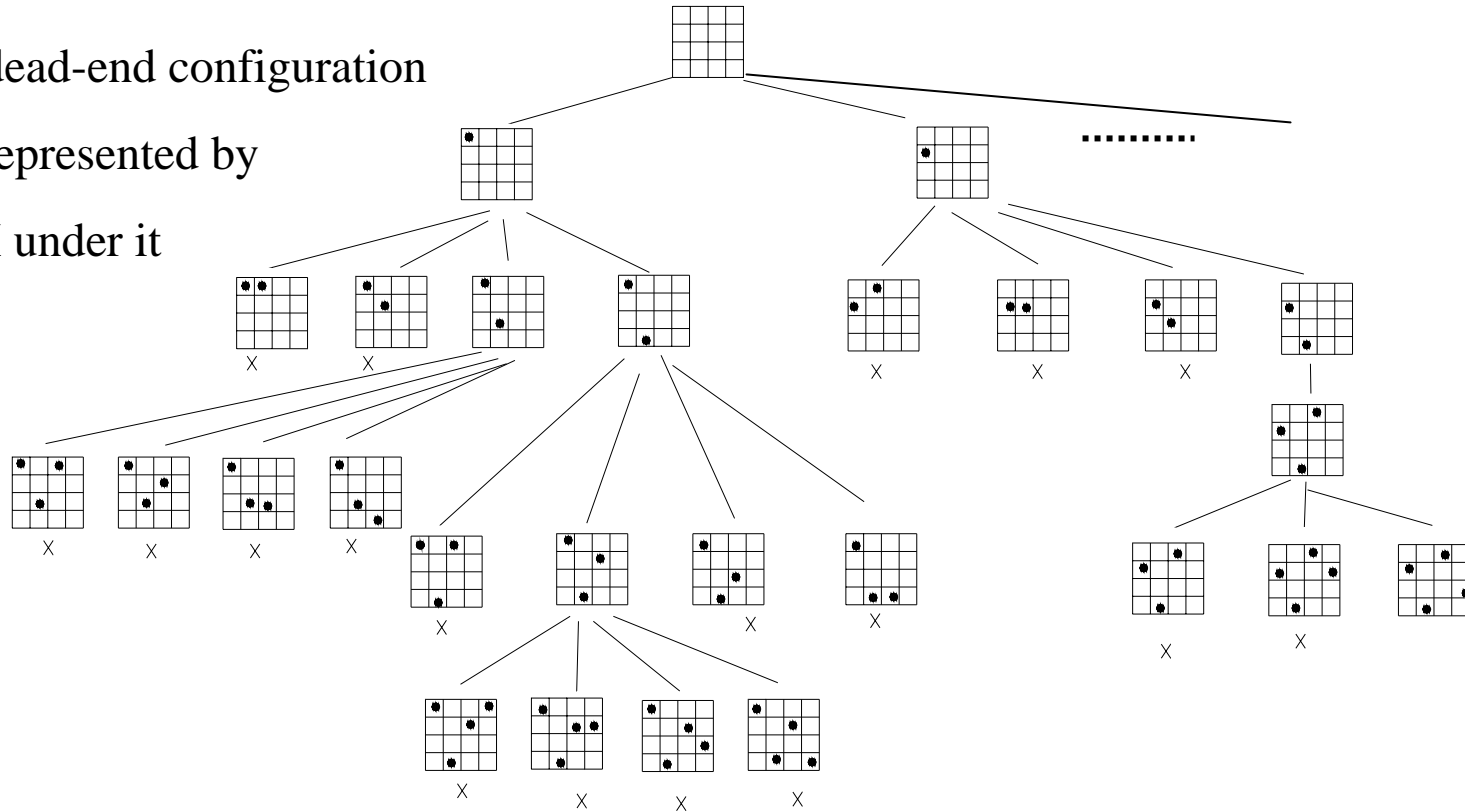


Figure shows assigning one queen to each square and trying to assign queens for the other squares one by one. Clearly a lot of back-tracking has occurred. All the other methods will be compared to this simple backtracking to show efficiency. *(Note that we are assigning queens column-wise, first assign to 1st column, then the second column and so on)*



Searching strategies in CSP:(More details)

Iterative broadening: This is an improvement over Chronological backtracking (BT). In BT a full branch in a tree is exhausted before going to another branch. However since choice points are ordered randomly, We have no reason to believe that earlier choices will lead to solutions. Here computational effort is distributed. A depth first search with a breadth cutoff threshold b is used, If a node is visited b times(initial visit plus backtracking) then the unvisited children are ignored so that we can concentrate on some other branch. If no solutions found then b is increased.

Advantages: More efficient over backtracking when the depth is large, useful when single solutions are quickly needed, efficient if sub-trees are similar.



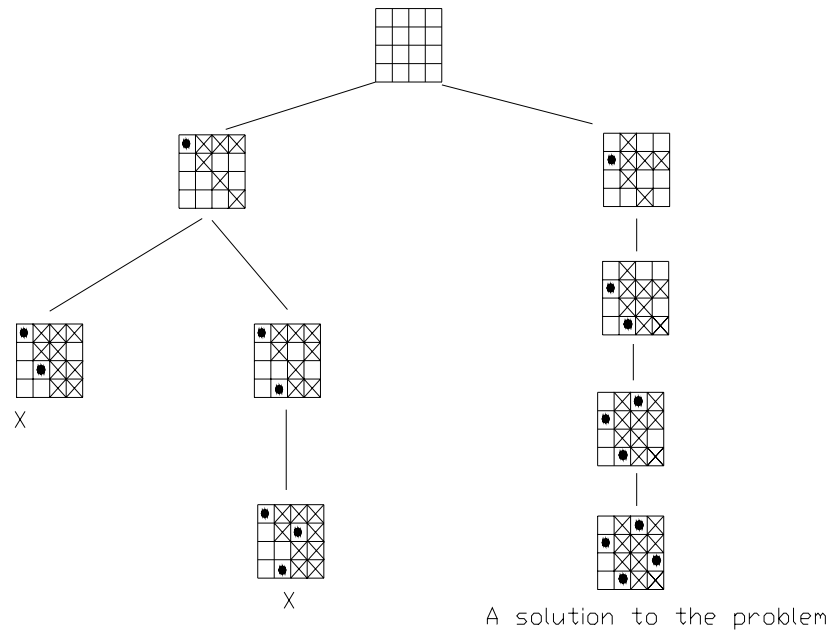
Look-ahead algorithms: Here consistency techniques are used such that backtracking is reduced.

Forward Checking: This uses the concept of Chronological backtracking, but when a value is committed to a variable, values from the domain of the other unlabeled variables are removed which are incompatible with the assigned value. Thus backtracking is reduced. If by choice of any value, any of the other domains are reduced to null, then that first value is immediately rejected.



(2) Four queens problem by Forward checking.

Queens are being assigned column by column.



Here, the X's denote the decisions that lead to failure since there is a conflict, The forward checking algorithm eliminates these values and thus a large portion of the search space is pruned.



Look-ahead algorithms: Here consistency techniques are used such that backtracking is reduced.

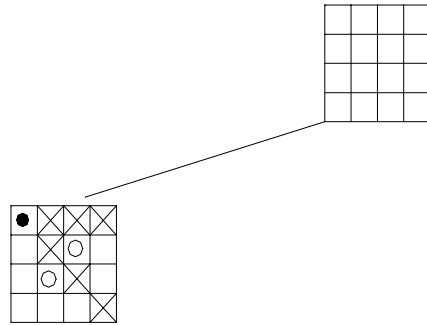
Directional Arc consistency Look-ahead algorithm: This uses problem reduction to reduce more redundant values than forward checking. Here for the unlabeled variables we maintain Directional arc consistency among them.

Larger part of the search space is pruned off by this method.

This method is called “ Partial Look-ahead”.



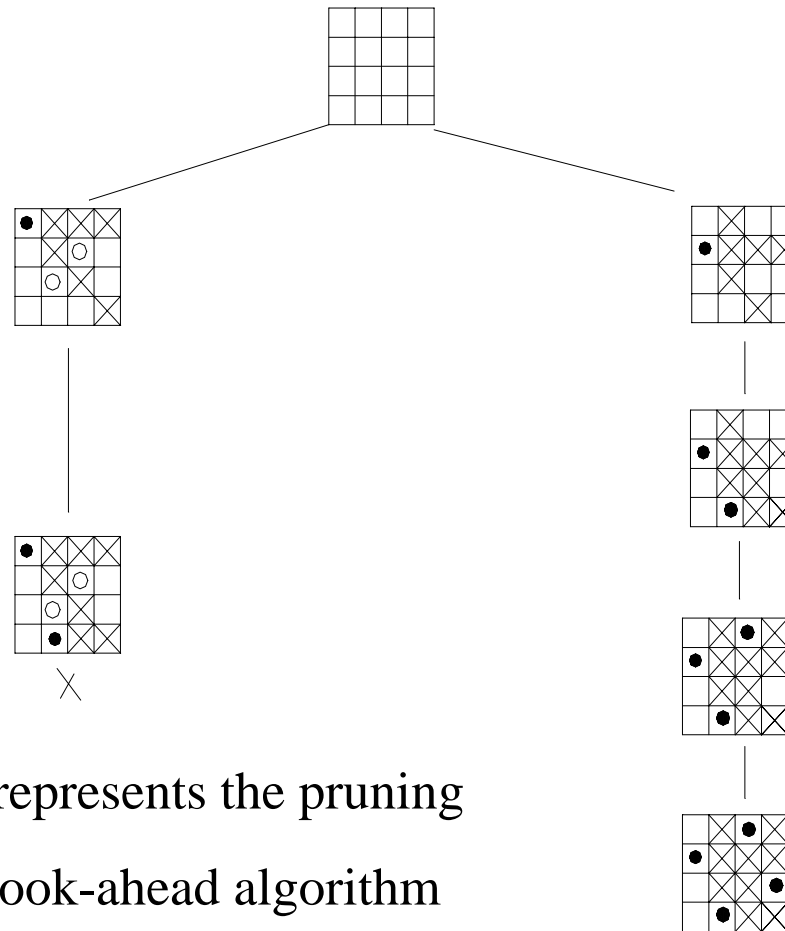
(3) Four queens by Directional Arc consistency Look-ahead.



O in the search tree represents the pruning off caused by the Look-ahead algorithm



(3) Four queens by Directional Arc consistency Look-ahead.



O in the search tree represents the pruning off caused by the Look-ahead algorithm

A solution to the problem



DAC- Lookahead continued.

This algorithm not only removes conflicting values but also maintains Directional arc consistency among variables. We can prune off larger regions of search space by this method.

In the example shown of the four queens problem:

Consider the left branch, Once 1A has been assigned to first queen, We immediately find out that we cannot assign either 3B, or 2C. If we assign 3B, we cannot assign queens anywhere in column C. This is a

Directed Arc consistency maintenance since we maintain consistency between domain values of the variables in the direction $Queen1 < Queen2 < Queen3 < Queen4$

Clearly comparing with forward checking, we have pruned off a part of the left hand side search space.



Full lookahead or the arc consistency lookahead algorithm

By using the AC-Lookahead algorithm, we can prune much larger space than DAC since AC is a stronger condition than the Directional Arc consistency. But we naturally spend more time in pre-processing.

In AC -lookahead, we maintain Arc consistency among the unlabeled variables,

If arc-consistency was used for the previous problem, we wouldn't have assigned 1A at all! And the left hand search space would be even smaller.

Why?

If we assign 1A, Consider arc consistency among the other variables. Now for the unlabeled variables column C and D ,We cannot assign 2C since it makes any assignment to column D impossible .

We cannot assign 4C since it makes any assignment in column B impossible.

Therefore without even assigning anything to the second column, we decide that we cannot assign a queen to 1A. Much larger search space is thus pruned.



Gather information while searching strategies:

Backjumping: In backjumping, unlike Backtracking, when there is a failure in assigning a value to an unlabeled variable, we backtrack to the variable causing the incompatibility (i.e. causing the failure) rather than the immediate previous variable.



An example of Back-jumping.

Here we use an 8 queens problem to illustrate the concept better.

In back-jumping, it is similar to backtracking except that when backtracking takes place, we backtrack to the culprit decision rather the immediate variable in the case of Back-tracking.

	A	B	C	D	E	F	G	H
1	●							
2			●					
3					●			
4		●						
5				●				
6	1	3,4	2,5	4,5	3,5	1	2	3
7								
8								

In the above example, numbers in row 6 indicate the queens in conflict, Clearly backtracking to queen 5 and only changing queen 5 position will be useless.

We need to backtrack to row 4, the most recent culprit.

Note: Backtracking would just go back to row 5 itself.



Graph Based Back-jumping: This is a variation of Back-jumping in which backtracking is done to the most recent variable involved in a constraint with the current variable rather than the most recent variable causing incompatibility as in back jumping. This is easier than the regular back-jumping.

Learning no-good compound labels algorithms:

Here we retain information regarding what combinations of labels caused over constraining.

Backchecking: This reduces the number of compatibility checks, by remembering combinations of values which can cause conflicts. Example: If a value $\langle x, a \rangle$ conflicts a value $\langle y, b \rangle$, then as long as $\langle x, a \rangle$ is used, we do not use $\langle y, b \rangle$.



Solution Synthesis:

This is a method of solving the CSP when the aim is to find all the solutions or we have to find an optimal solution.

Main idea:

- Searching multiple branches simultaneously.
- Collect sets of legal labels for larger and larger sets of variables, until we have labeled all the variables (The final solution).

For further details on this read:

Foundations of Constraint Satisfaction, E. Tsang, 1993.



Search Orders in CSP:

Why Ordering can affect Searching?

In Look-ahead algorithms - Failures can be detected earlier,

Larger search space pruned.

Less Backtracking needed and also less undoing of labels when backtracking occurs.

Heuristics for Ordering:

- (1) Minimal Width Ordering.
- (2) Minimal Bandwidth Ordering.
- (3) Fail first Principle.
- (4) Minimal Cardinality Ordering.



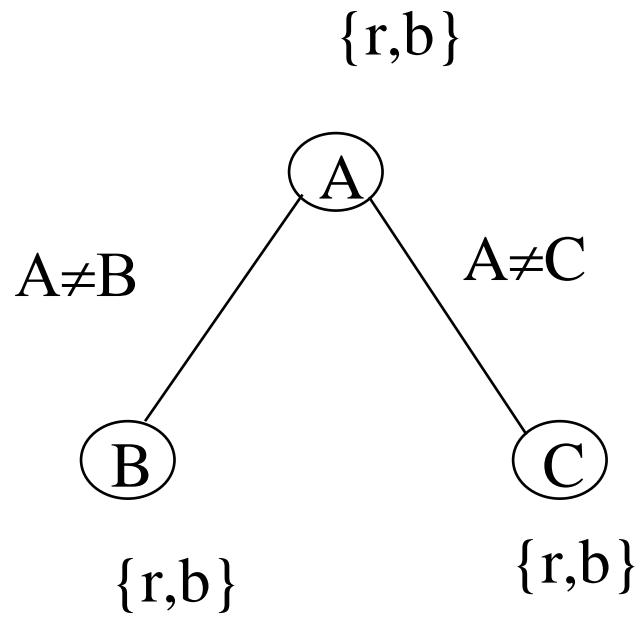
Minimal width ordering:(MWO)

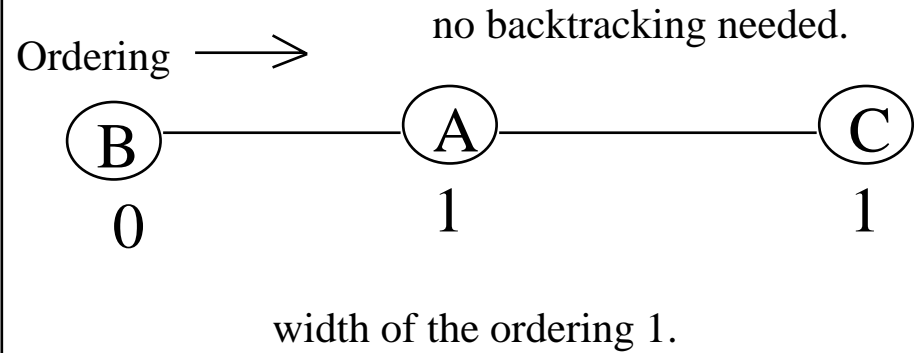
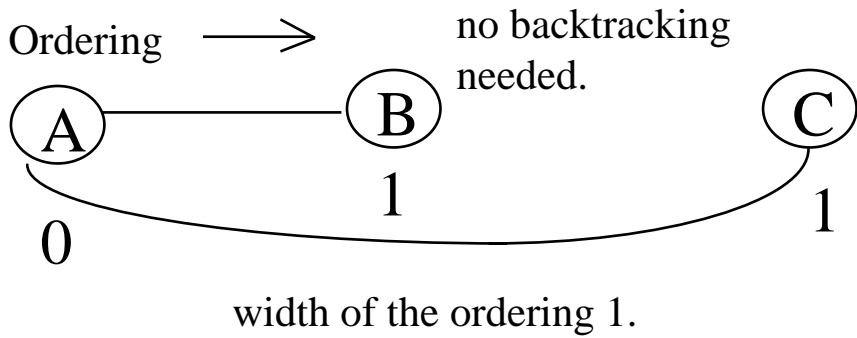
This method Reduces need to backtrack.

Width of a node: Number of nodes before it in the ordering which are adjacent to it in the constraint graph.

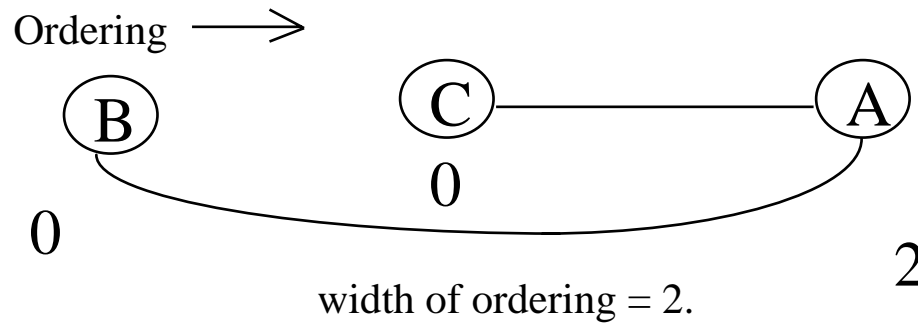
Width of the Ordering:Maximum width among all nodes.

Example:





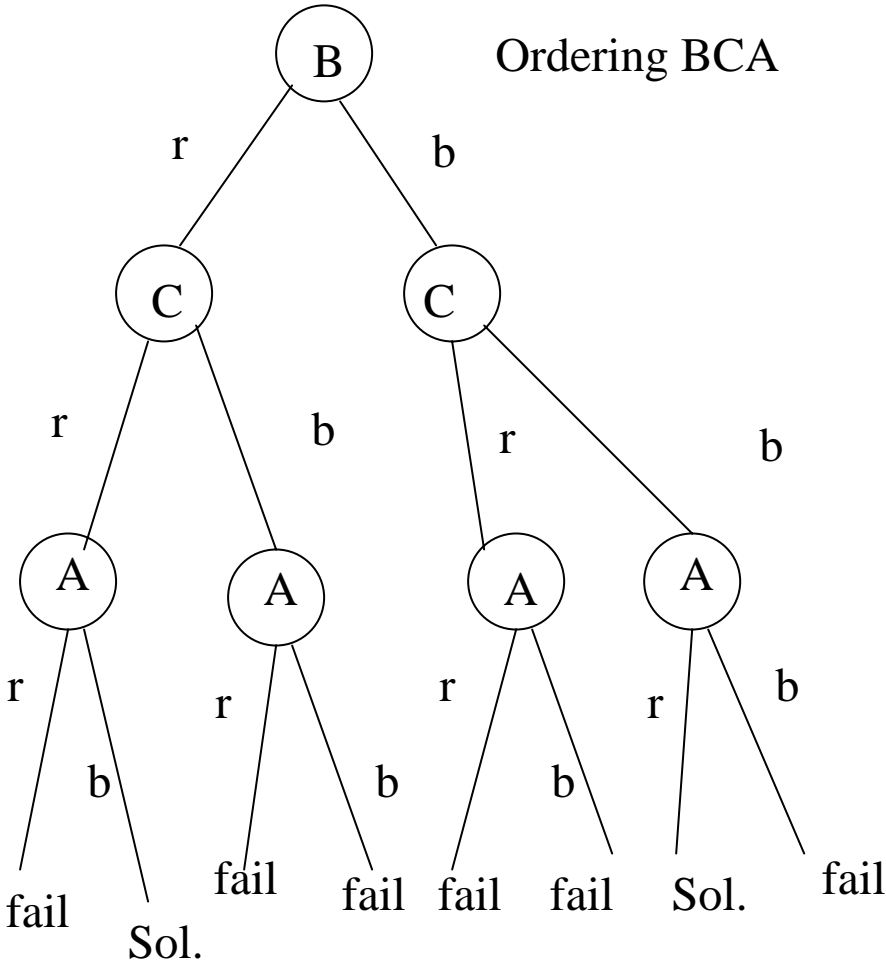
may need backtracking.



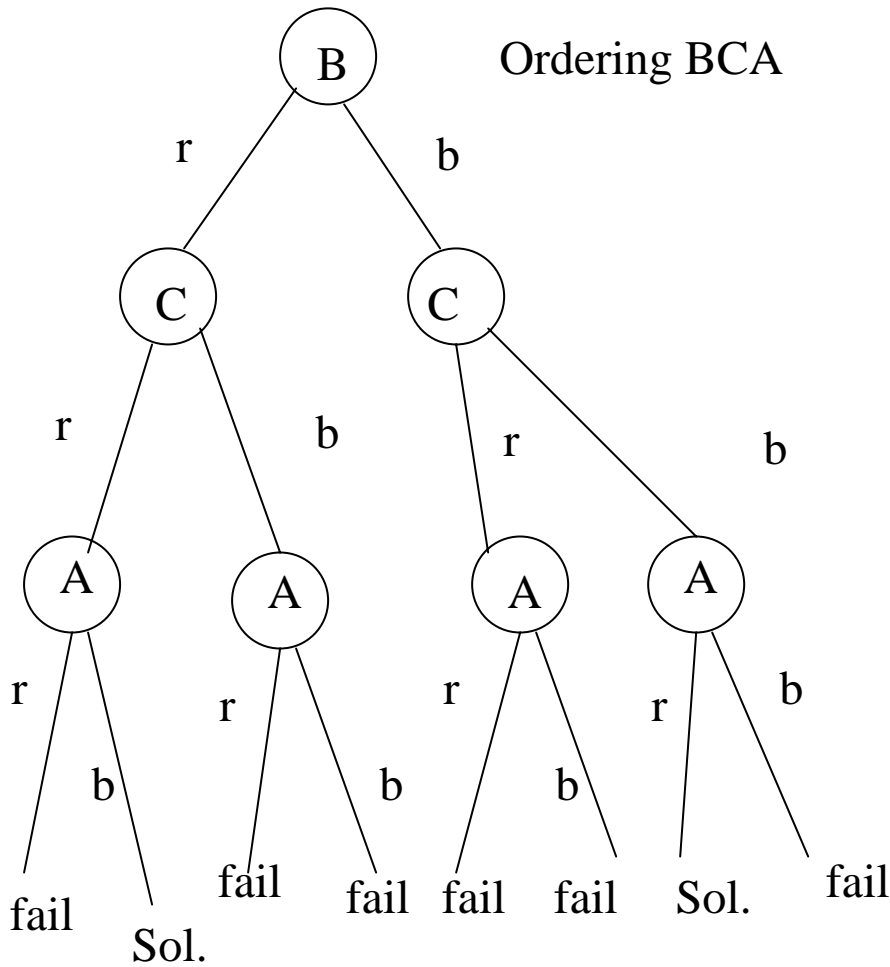
Some orderings of the problem, Clearly shows backtracking can be avoided by the top orderings (with Minimal width).



Efficiency of MWO when we want to find all solutions to previous problem. (using backtracking)

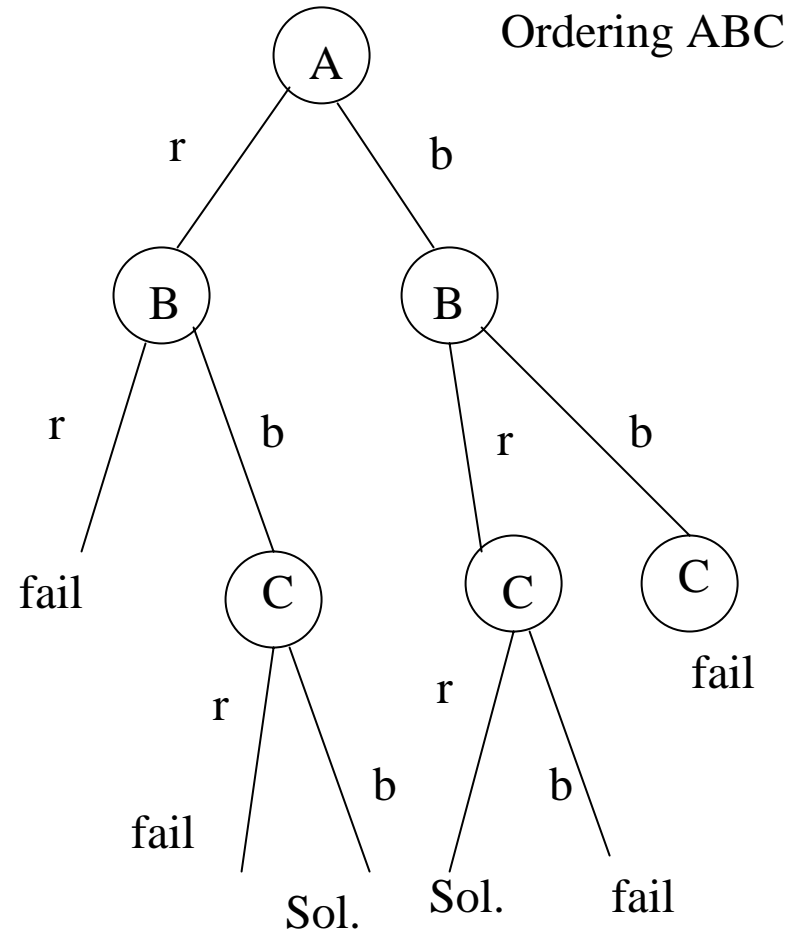


Efficiency of MWO when we want to find all solutions to previous problem. (using backtracking)



Number of branches=14

“Sol.” is a Solution.



Number of branches=10



Heuristic (2): Minimal Bandwidth Ordering (MBO).

This is applicable to CSP's in which the constraint graph is not complete. (A complete graph is a graph in which an edge exists between every two nodes). This heuristic is used for pre-processing before searching. It reduces the number of labels to be undone during backtracking.

Before search starts, the nodes are given a Minimal bandwidth ordering

Idea: The closer together the variables which constrain each other are, the less distance we have to backtrack.

Bandwidth: The bandwidth of a node "v" in an ordered graph is the maximum distance between "v" and any other node adjacent to v in its constraint graph.



Minimal Bandwidth Ordering (Continued)

Bandwidth of the ordering: Maximum bandwidth of all the nodes in the graph with this ordering.

Bandwidth of the graph: Minimal bandwidth of all the orderings in the graph.

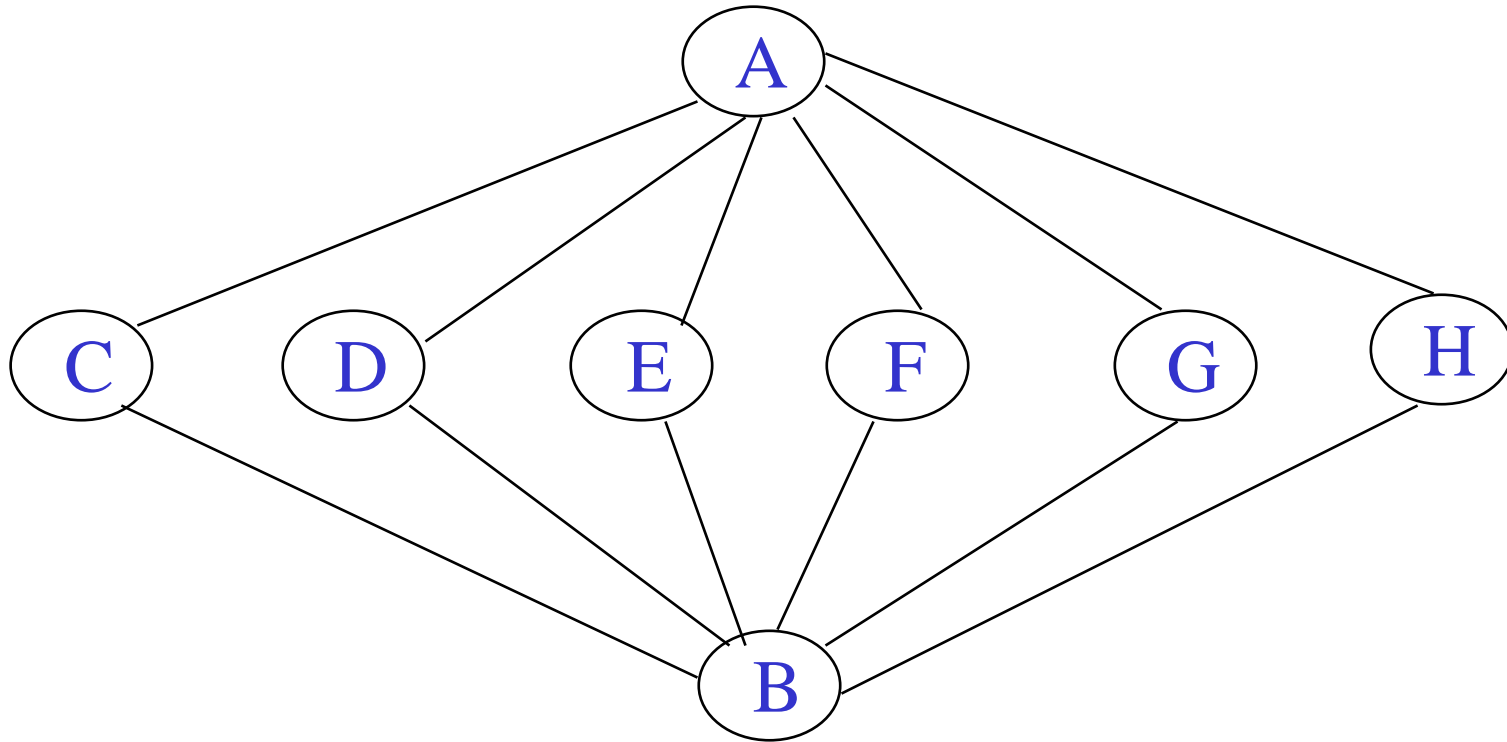
We can show that when the bandwidth of a graph is small, the worst time complexity can be improved over backtracking and look-ahead algorithms which do not use this pre-processing

We may be tempted to assume that MWO and MBO give us the same results, As the next example shows, this is not true in all cases.

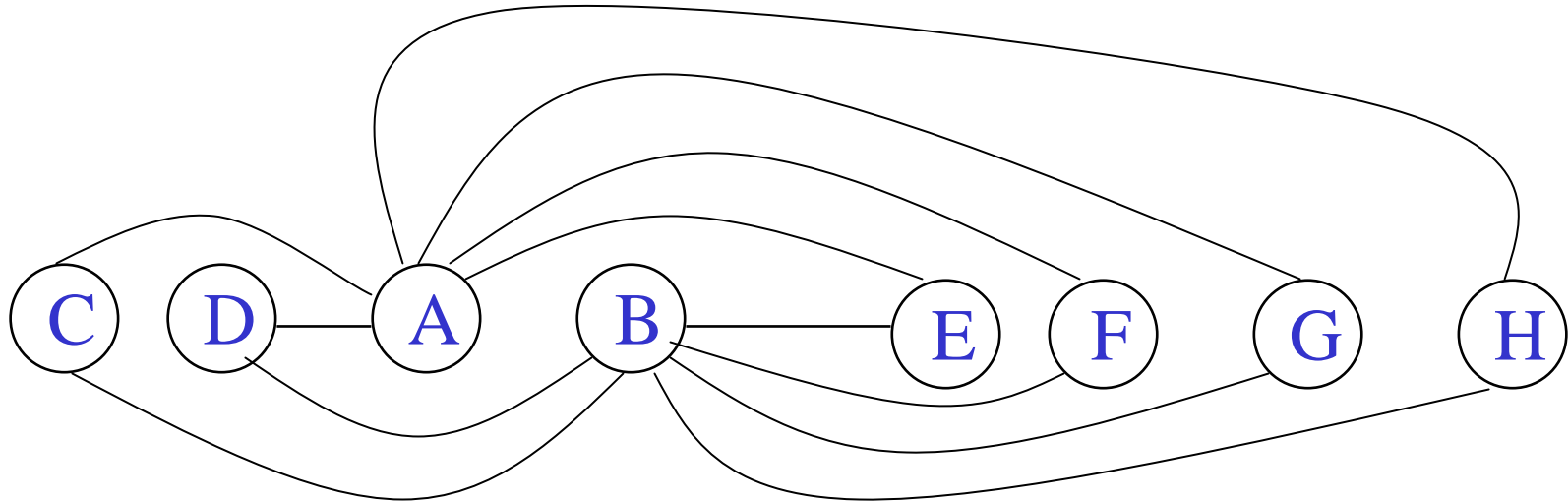
This is because we may not be able to minimize both the bandwidth and width simultaneously.



Example: Consider the example shown below.



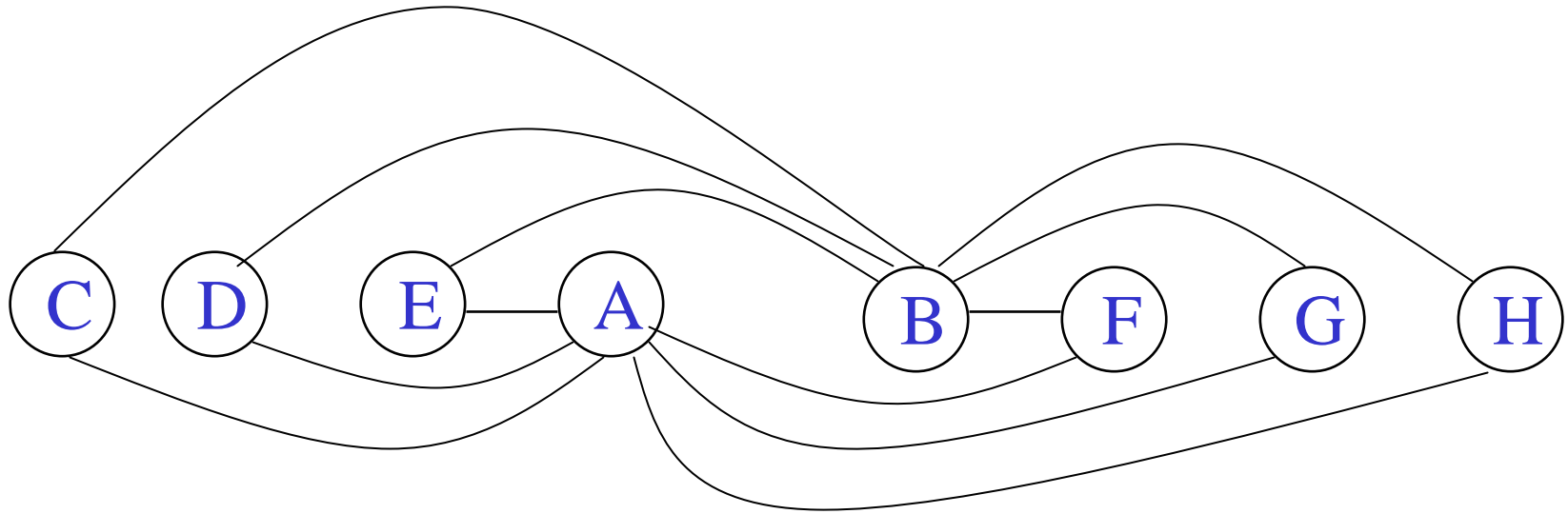
An ordering with minimal width is given below.



The width of this ordering is 2 and its bandwidth is 5



An ordering with minimal band-width is given below.



The bandwidth of this ordering is 4 and its width is 3



We therefore get different orderings with minimal - bandwidth(MBO) and minimal width (MWO)and therefore performance may be different with each algorithm.

The choice of the algorithm to use is highly problem dependent. There may be problems in which MWO performs better and others in which MBO performs better.

Some algorithms use a combination of MBO and MWO to get good results.



Heuristic 3: Maximum Cardinality Ordering.

This is an approximation of the MWO heuristic.

How to get the MCO ordering?

The MCO is obtained by picking nodes in the reverse order using:

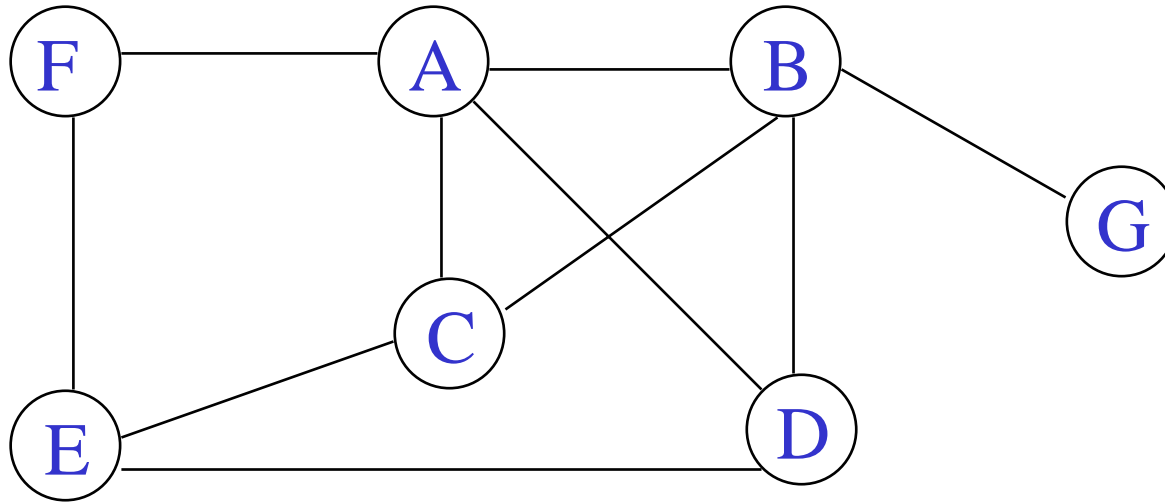
- (1) A node with a high degree is made the last node.
- (2) Among unordered nodes, the one adjacent to maximum number of already ordered nodes will be made last, Ties will be arbitrarily broken.

Advantages:

Simple to use, Has a linear worst case complexity.



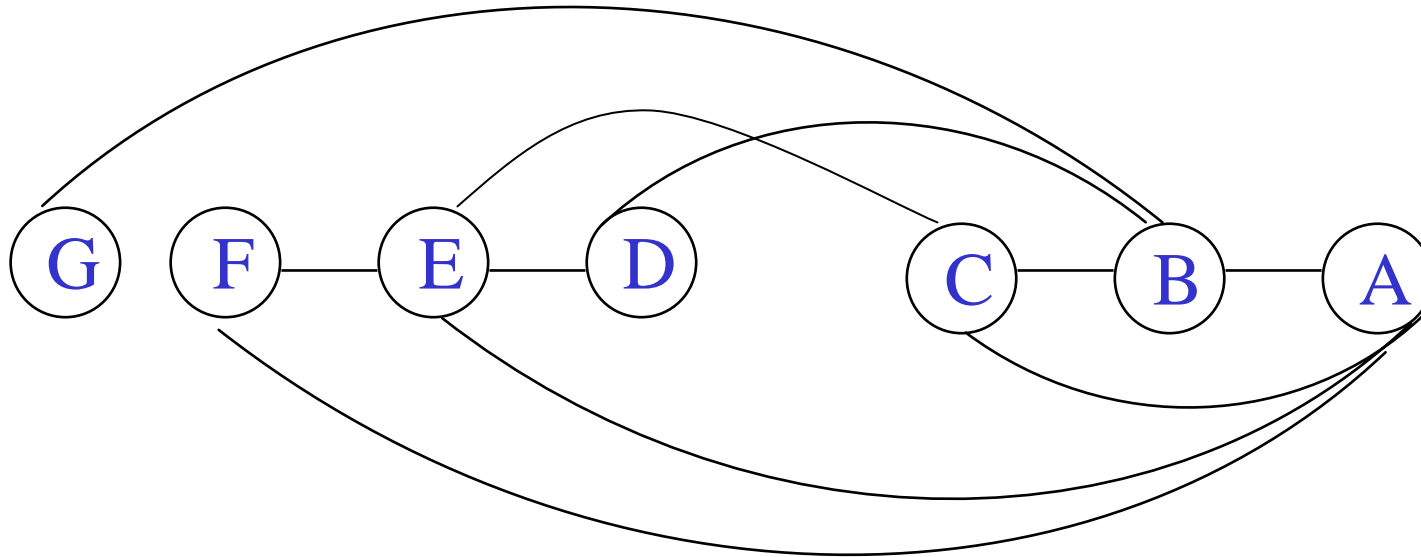
Consider the example shown below.



Let us use the Maximum Cardinality ordering algorithm. Choosing node A arbitrarily and then B arbitrarily, We get..



An ordering with maximum Cardinality is:



The width of this graph is 4, But using the ordering ABCDEFG we can get a width of 2. Thus it is a simple approximation of the MWO, but the advantage here is that this algorithm takes linear time.



Heuristic (4)

Fail First Principle.

This is based on the principle that a task that is most likely to fail should be performed first, We can recognize dead-ends earlier. According to this the next variable chosen must be the most-constrained. A simple rule would be to pick a variable with the smallest domain size

The Fail first principle can be combined with look-ahead algorithms to determine which node to process next dynamically.



When should we use these heuristics?

Minimal width ordering:

Can be used effectively when: (a) Degrees of nodes vary significantly (It will not help solve a problem like the N-queens well.) (Degree of node is the number of nodes adjacent to it.) (b) When consistency is maintained, this can be used to see if a backtrack free search can be done.

Minimal Bandwidth Ordering:

(a) When No node in the CSP has high degree, We note that the higher the degree the more *likely* that the *bandwidth* will be high. (Maximum degree will dictate the lower bound on the MBO).

Fail first principle:

will be useful when (a) Domain size varies considerably and (b) tight constraints exist, (this will ensure that domain size of unlabeled variables will vary significantly).

