Complete Search Algorithms for Constraint Satisfaction

Lookahead

Gather-information-while-searching

Complete Search Strategies Overview

- Preprocessing
  - Problem reduction; problem transformation
- General Search
  - Chronological backtracking
- Lookahead
  - Forward Checking, DAC- & AC-Lookahead
- Gather-information-while-searching
  - DDBT, Learning Nogoods, Backmarking
- Hybrids, e.g. FC-BM-CBJ

Pre-processing – before search begins

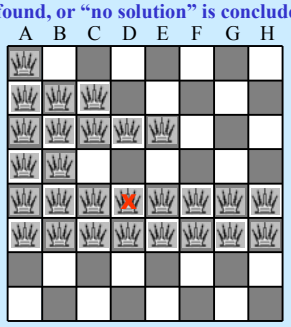
- Problem reduction
  - 1-consistency is always worth achieving
  - How about 2-consistency? 3-consistency?
- Problem transformation
  - Should redundant constraints be added?
    - With A < B, B < C, would adding A < C help?
  - Should the problem be decomposed into sub-problems?
    - Some sub-problems may be tractable
    - E.g. cycle-cutset

Backtracking Search In The 8-Queens Problem

Complete search, till solution found, or "no solution" is concluded

- Place one queen per row
- Place one queen at a time
- Examine each column

Backtrack at dead-ends



Lookahead Algorithms

- Lookahead so as to detect failure asap
- Reduce remaining problem
- More reduction requires more computation
- Does marginal gain justify marginal cost?
- MAC-Lookahead is effective in general
  - Key: to record support in maintaining AC

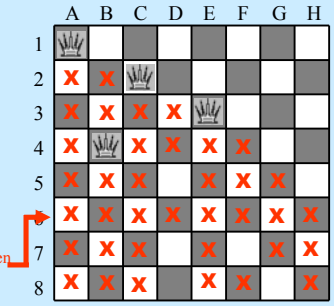
Forward Checking

- Algorithm FC:
  - Label one variable at a time
  - After each label $<x,v_x>$ is committed to:
    - examine every future variable y
    - remove every value v_y from D_y such that $<y,v_y>$ is incompatible with $<x,v_x>$
  - Backtrack if any future domain becomes empty
- Found to be quite effective in general

Forward Checking Search

- Problem reduction
  - a major technique
- Combined with search methods
- Reduce domain of future variables
- Detect dead-ends
  - To backtrack early

Dead-end detected after Queen 4 – no legal space for row 6, backtrack…

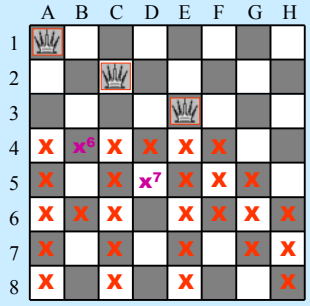


DAC-, AC-Lookahead

- Principle:
  - Do exactly as Forward Checking
  - In addition, after propagating using FC:
    - Maintain DAC or AC in the remaining problem
  - Backtrack if any future domain becomes empty
- DAC, AC can be replaced by *k*-consistency for any value *k*

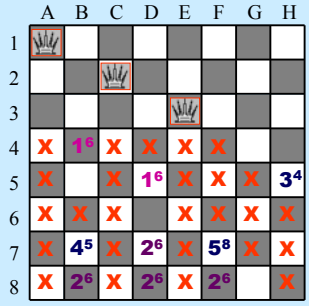
Result of DAC-Lookahead

- 8-Queens Problem
- Three queens have been placed
- Maintaining **DAC**
- Square 4B is not supported by row 6, hence removable
- The same applies to 5D

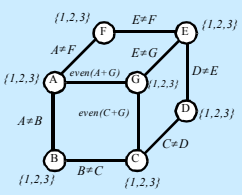


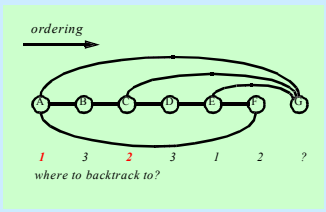
Result of AC-Lookahead

- 8-Queens Problem
- Three queens have been placed
- Maintaining **AC**
- Result: dead-end found in row 7
- Backtrack required – typically remove Queen 3



Dependency-directed Backtracking





- Graph suggests backtracking to F is futile
- E constrains G, but <A,1> & <C,2> are real culprits

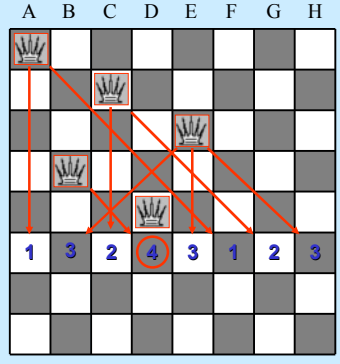
Back Jumping

- Jump to the latest culprit

Recorded earliest conflict

Identify the latest culprit, which is 4

Undo queens 5 and 4, continue



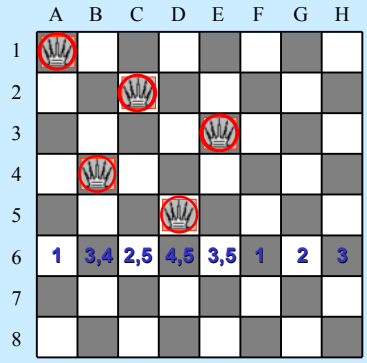
Conflict-directed Backjumping

- Problem with Backjumping (BJ):
  - When all values tried, jumping is possible
  - But if labelled, then later backtrack from below, one can only backtrack chronological
- Conflict-based BJ returns causes of failure when backtracking occurs
  - This allows jumping to take place above
- In general form: ***Truth Maintenance***

Learning Nogoods

- All conflicts recorded
- **Learn Nogoods**:
  - (1A,2C,3E,4B)
  - (1A,2C,3E,5D)

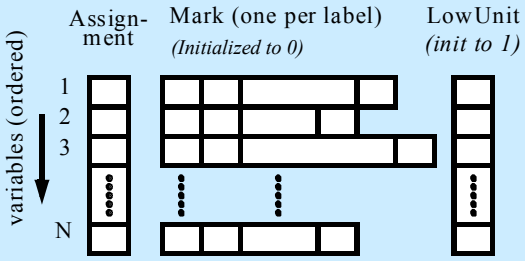
  Set covering prob.
- Next, reject (1A,2C,3E,4B)
- In future, avoid (1A,2C,3E,*4?*,5D)



Backmarking Principles

- Aim: reduce number of constraint checks
  - # of checks often determines cost of search
- Mark(u,l): lowest level at which <u,l> failed
  - No need to re-consider <u,l> as long as the search has not backtracked pass this level
- LowUnit(x): lowest variable which assignment has been changed since the last time x was visited
  - No need to check <x,v> against labels before LowUnit(x), as they were checked to be ok

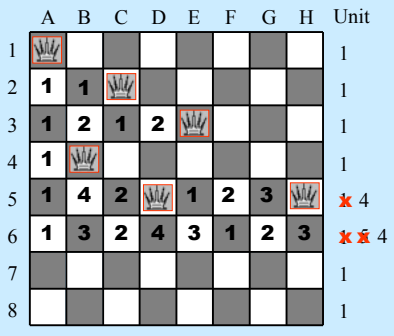
Backmarking Data Structures



BM Example

- After 5H, reject all values in 6, as they all failed before queen 5
- So 5H rejected
- Set LowUnit(5) and (6) to 4
- Next try 4G
- only 5B examined
- Start by checking 5B against 4G (not 1A, 2C, 3E, as they were ok)

- All values for Q6 rejected
- LowUnit(6) set to 5



Program: http://cswww.essex.ac.uk.Research/Download/BackMarking.zip

Search Ordering

Minimal width ordering

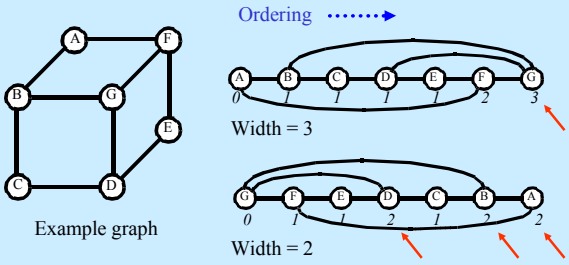
Minimal bandwidth ordering

Smallest-domain-variable-first

Width of a Graph, Definitions

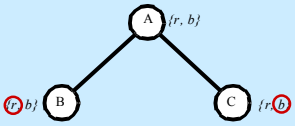
- Given an order O:
  - *width of node x* is the number of nodes before *x* and connected to *x*
  - *width of O* is the maximum width for all nodes
- *Width of a graph* is the minimal width for all ordering

Width, Examples



Motivation: Min. Width Ordering

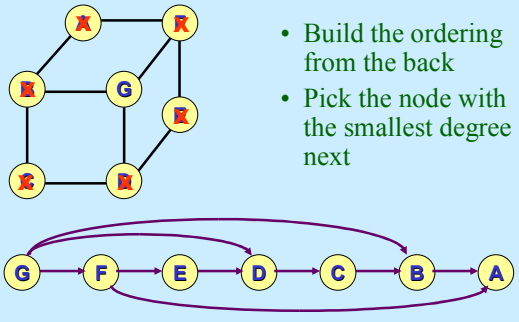
- To reduce dependency
- The hope is to reduce backtracking needs
- This is a *heuristic*
  - Sometimes doesn't work
- Ordering (ABC)
  - No backtrack needed
- Ordering (BCA)
  - May need backtrack



Minimal Width Ordering Algorithm

- **Repeat**
  - **Pick the node with minimum degree**
  - **Put it to the front of the list**
  - **Remove all relevant edges**
- **Until all nodes picked**
- Resulting list has minimal width ordering
  - In reverse order of the picks

Finding Minimal Width, Example



- Build the ordering from the back
- Pick the node with the smallest degree next

Backtrack-free Search

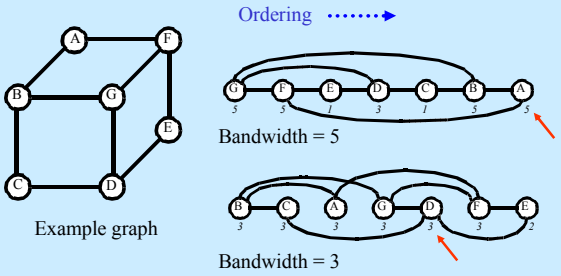
- **A depth first search is *backtrack-free* if**
  - **the level of strong consistency**

  is greater than
  - **the width of the ordering used**
- An important observation!
- E.g. if width of graph is 3, then maintaining *strong 4-consistency* will allow backtrack-free search
- Trees have width 1
  - hence the *Tree-Search Algorithm*

Bandwidth of a Graph, Definition

- Given an ordering O:
  - *bandwidth of node* is the maximum distance to its connected nodes
  - *bandwidth of O* is the maximum distance for all nodes
- *Bandwidth of a graph* is the minimal bandwidth for all ordering

Bandwidth, Example



Minimal bandwidth ordering

- Aim: to reduce the backtracking distance when backtracking is needed
- Algorithm for achieving MBO is complex
  - *(skipped here)*
  - Hill climbing is possible
- MBO is upper-bound for ***induced width***
  - interesting relations, not to be discussed here

Heuristic: Smallest-domain-first

Used to be called "Fail-first principle"

- Variable ordering heuristic:
  - Pick the variable with the smallest domain to label next
  - Break ties randomly
- Bralez variation:
  - Break ties with variable with maximum degree in the remaining graph

Smallest-domain-first

- Dynamic strategy
  - Different branches may use different ordering
- Known as "Fail-first-principle" (FFP)
  - Now we know it does not succeed by failing first
    - Professor Barbara Smith, Huddersfield
- Effective with Lookahead Algorithms
  - As they dynamically change domain sizes
  - "FC+FFP" used to be seen as most practical

Smallest Domain First, Example