CSP: Solving by Search

Pedro Meseguer
IIIA-CSIC
Bellaterra, Spain

Overview

Systematic Search
- Search Tree
- Backtracking (BT)
- Conflict-Directed Backjumping (CBJ)
- Dynamic Backtracking (DB)
- Heuristics

Local search
- Breakout
State space: explored as a tree
- root: empty
- one variable per level
- successors of a node:
  - one successor per value of the next level variable
  - meaning: variable \[\cdot\] value

Tree:
- each branch defines an assignment
- depth \( n \) (number of variables)
- branching factor \( d \) (domain size)

Search tree for 4-queens:

\[
x_1 \leadsto 1 \leadsto 2 \leadsto 3 \leadsto 4
\]
\[
x_2 \leadsto \ldots \leadsto \ldots \leadsto \ldots \leadsto \ldots
\]
\[
x_3 \leadsto \ldots \leadsto \ldots \leadsto \ldots \leadsto \ldots
\]
\[
x_4 \leadsto (1,1,1) \leadsto (2,1,1) \leadsto (3,1,1) \leadsto (4,1,1) \leadsto (4,4,4)
\]
**Backtracking Algorithm**

Depth-first tree traversal (DFS)

At each node:
- check every completely assigned constraint
- if consistent, continue DFS
- otherwise, prune current branch
- continue DFS

Complexity: $O(d^n)$

---

**Backtracking on 4-queens**

Diagram showing the backtracking process for the 4-queens problem, with 25 nodes and a solution for queens placed in the 1st, 2nd, 3rd, and 4th columns. The diagram illustrates the depth-first search approach with nodes and branches representing the state space exploration.
Problems of Backtracking

Thrashing:
- the same failure can be rediscovered an exponential number of times

Solutions:
- check not completely assigned constraints: propagation
- non-chronological backtracking: backjumping

Non-chronological Backtracking

Backtrack on a culprit variable: $X_4$

Changing $X_5$ does NOT remove the dead-end
**Conflict Set**

CS($x_k$): assign. variables in conflict with some value of $x_k$

- Backtrack: jumps to the last variable in CS($x_k$)
- CS($x_k$) is backed-up:
  - $x_k \notin x_i$
  - CS($x_i$) = CS($x_i$) U CS($x_k$) - $x_i$
  - $x_k$ conflicts with var. before $x_i$ are passed to $x_i$

**Conflict-Directed Backjumping**

Non-chronological backtracking:
- jumps to the last variable in the conflict-set
- the conflict set is backed-up
  - intermediate decisions are removed
Nogoods

Nogood: subset of incompatible assignments

Example: map colouring, $x_1$, $x_2$, $x_3$ adjacent, $D = \{a,b\}$

$(x_1 = a \square x_3 = a)$ or equivalently

Nogood resolution:

$$
\begin{align*}
\text{lhs} & : x_1 = a \quad x_3 \neq a \\
\text{rhs} & : x_2 = b \quad x_3 \neq b
\end{align*}
$$

Dynamic Backtracking

Non-chronological backtracking:

- one nogood per each incompatible value
- empty domain: new $ng$ by nogood resolution
- backtrack to the variable in $rhs(ng)$

$ng: x_1 = 1 \square x_3 \neq 2$
Dynamic Backtracking (II)

Non-chronological backtracking:
- jumps to the last decision responsible for the dead-end
- intermediate decisions are NOT removed

\[
\{x_1 = 1\} \quad \{x_3 = 2\} \quad \{x_2 = 4\} \quad x_4
\]

Variable Ordering

Static variable ordering: variable associated with level

Dynamic variable ordering:
- each branch considers all vars / different ordering per branch

\[
\begin{align*}
Q_1 & \quad x_1 = 1 \\
Q_2 & \quad x_2 = 2 \\
Q_3 & \quad x_3 = 3 \\
Q_4 & \quad x_4 = 4
\end{align*}
\]
**Variable Selection**

Node $q$, what is the next variable to assign?

1. There is a solution in $\text{succ}(q)$: any var is fine
2. There is no solution in $\text{succ}(q)$: assign var that sooner detects that there is no solution

What is more often?
- Except trivial problems, case 2
- *Biggest* effort: escaping from problems without sol.

**Fail-first**: first the variable that sooner detects failure

---

**Heuristics**

**Min-domain:**
- First the variable with less compatible values with the current partial solution
- Minimize the search tree

**Max-degree:**
- First the variable involved in more constraints
- Maximize constraint propagation

**Combination:**
- First the variable with min domain/degree