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
Search Tree

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 = value

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

Search tree for 4-queens

\[
\begin{align*}
X_1 & \quad 1 \quad 2 \quad 3 \quad 4 \\
X_2 & \quad \quad \quad \quad \\
X_3 & \quad \quad \quad \quad \\
X_4 & \quad (1,1,1,1) \quad (2,1,1,1) \quad (3,1,1,1) \quad (4,1,1,1) \quad (4,4,4,4)
\end{align*}
\]
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

25 nodes

solution

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td></td>
<td>Q</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>Q</td>
<td>Q</td>
</tr>
</tbody>
</table>
Problems of Backtracking

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

Solutions:
- check not completely assigned constraints: propagation
- jumping to the source of failure: non-chronological backtracking

Non-chronological Backtracking

Changing $X_5$ does NOT remove the dead-end

Backtrack on a culprit variable: $X_4$
Conflict Set

\( \text{CS}(x_k): \text{assign. variables in conflict with some value of } x_k \)

- Backtrack: jumps to the last variable in \( \text{CS}(x_k) \)
- \( \text{CS}(x_k) \) is backed-up:
  \( x_k \nsubseteq x_i \)
  \( \text{CS}(x_i) = \text{CS}(x_k) \setminus \{x_i\} \)
  - conflicts of \( x_k \) 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
  • original constraints
  • during search, new nogoods are discovered

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

\[
(x_1 = a \neq x_3 = a)
\]

or equivalently

\[
\text{lhs} \quad \begin{array}{c}
\text{lhs} \\
\text{rhs}
\end{array}
\]

Nogood resolution: new nogood

\[
\begin{align*}
x_1 = a \neq & x_3 \\
x_2 = b \neq & x_3
\end{align*}
\]

\[
\text{every value for } x_3
\]

Dynamic Backtracking

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

\[
\begin{array}{cccc}
\{x_1 = 1\} & \{x_3 = 2\} & \{x_2 = 4\} & \{\text{other values}\}
\end{array}
\]

\[
\begin{array}{cccc}
1 & 2 & 3 & 4
\end{array}
\]

\[
\begin{array}{cccc}
& Q_1 & \\
x_1 & & \\
x_2 & & Q_3
\end{array}
\]

\[
\begin{array}{cccc}
& Q_2 & \\
x_3 & & \\
x_4 & x_1 = 1 & x_2 = 2 & x_3 = 2 & x_4 = 1
\end{array}
\]

\[
\text{ng: } x_1 = 1 \neq x_3
\]
Dynamic Backtracking (II)

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

Variable Ordering

Static variable ordering: variable associated with level

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

Not needed for an exhaustive tree.
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 search tree size

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

Combination:
- First the variable with min domain/degree
Local Search

- Optimization of objective function: \( \min F(s) \)

- Iterative process: \( s \rightarrow s' \rightarrow s'' \rightarrow s''' \)

- Greedy approach: \( F(s) \geq F(s') \geq F(s'') \geq F(s''') \)
  until solution or resources exhausted

- Problem: \textit{local minima}
  - permitting \( F(s) < F(s') \)
  - random changes, restarts, etc.

- Incomplete algorithms: may fail finding a solution

Local Search (II)

Elements:
- Objective function \( F(s) \): it maps a state \( s \) with a cost \( F(s) \)
- Neighborhood \( N(s) \): states where you can go from \( s \), in the next iteration
- Selection criterion: given \( F(s) \) and \( N(s) \), choose the next state \( s' \)

Complexity: bounded by resources
Local Search and CSP

State $s$: assignment involving every variable

Neighborhood: $N(s)$:
- $s'$ that differ from $s$ in values of $i$ variables
- usually $1 \leq i \leq 2$

Objective function:
- $F(s) = 0$, if $s$ is a solution
- $F(s) > 0$, otherwise

Breakout Algorithm: [Morris 93]
- each constraint has a weight
- $F(s) =$ sum of weights of unsatisfied constraints
- if a constraint is not satisfied, its weight increases