Overview

Constraint Programming

- Modelling
- Search space size
- Primal / Dual models
- Global constraints
- Solving
- Guidelines
- CP Styles
Constraint Programming

CP:
- provides a platform for solving CSPs
- proven useful in many real applications

Platform:
- set of common structures to reuse
- best known algorithms for propagation & solving

Two stages:
- modelling
- solving

CP: Modelling

Modelling decisions: select among alternatives
- the choice of the variables
- the choice of the domains
- how we state the constraints

Example: Map Colouring
- variables: are regions or colours?

Any CSP can be modelled in different ways
- Efficiency of algorithms can vary dramatically
- No strong results are known
- Formulating an effective model is not easy, requires considerable skills in modelling
**N-queens: Model 1**

**Variables:** \( n^2 \), one per cell, matrix \( B \) \( n \times n \)

**Domains:** \( \{0,1\} \), \( B(a,b) = 0 \), no queen  
\( B(a,b) = 1 \), queen

**Constraints:** If \( B(a,b) = 1 \) then  
- ***same row*** \( B[_,b]=0 \)  
- ***same column*** \( B[a,_]=0 \)  
- ***same diagonal*** \( B[a+d,b+d]=0, B[a-d,b-d]=0 \)  
- ***same diagonal*** \( B[a-d,b+d]=0, B[a+d,b-d]=0 \)

---

**N-queens: Model 2**

**Variables:** \( n \), one per row

**Domains:** \( \{0,1,\ldots,n-1\} \), queen column

**Constraints:**  
- ***different columns*** \( x_i \neq x_j \)  
- ***different diagonals*** \( |x_i - x_j| \neq |i - j| \)

---

Different row constraint is included in the formulation!!
N-queens: Model 3

Variables: \( n \), one per row

Domains: \( \{0,1,\ldots,n-1\} \), queen column

Constraints:
- Different columns: \( \text{all-different}(x_1, x_2, \ldots, x_n) \)
- Different diagonals: \( |x_i - x_j| = |i - j| \)

Different row constraint is included in the formulation!!

N-queens Models

<table>
<thead>
<tr>
<th>Constraints number pruning</th>
<th>Model 1</th>
<th>Model 2</th>
<th>Model 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search space size d#vars</td>
<td>( 2^{n^2} )</td>
<td>( n^n )</td>
<td>( n^n )</td>
</tr>
<tr>
<td>4</td>
<td>65,536</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>10</td>
<td>1.27 E30</td>
<td>1.00 E10</td>
<td>1.00 E10</td>
</tr>
<tr>
<td>20</td>
<td>ERROR!!</td>
<td>1.05 E26</td>
<td>1.05 E26</td>
</tr>
<tr>
<td>n rows</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n columns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2(n-1) diagonals</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constraints number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n columns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2(n-1) diagonals</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Equal model 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 all-diff</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2(n-1) diagonals</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>More than model 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Constraint Formulations

Binary (arity \(\leq 2\)):
- conceptually simple, easy to implement
- may generate weak formulations

Non-binary (arity > 2):
- more complex constraints
- GAC: stronger (filter more) than AC on equivalent binary decomposition

Equivalence: any non-binary CSP can be reformulated as a binary one

---

Primal / Dual Formulations

Primal CSP: \((X, D, C)\)
\[X = \{x_1, x_2, \ldots, x_n\}, \quad D = \{d_1, d_2, \ldots, d_n\}, \quad C = \{c_1, c_2, \ldots, c_r\}\]

Dual CSP: \((X', D', C')\)
\[X' = \{x'_1, x'_2, \ldots, x'_r\}, \quad D' = \{d'_1, d'_2, \ldots, d'_r\}, \quad C' = \{c'_{ij}\}\]

- \(d'_i = \text{rel}(c_i)\)
- \(\text{var}(c'_{ij}) = \{x'_i, x'_j\}\)
- \(\text{var}(c) \cap \text{var}(c) = \emptyset\)
- \(\text{rel}(c'_{ij}) = \text{same values for shared primal vars}\)

Always binary!!
Example: Crossword puzzles

Primal model (Non-binary)

variables: cells
domains: ‘a’, …, ‘z’
constraints: contiguous letters must form words in dictionary
Dual model (binary)

- Variables: words across and down
- Domains: words from dictionary
- Constraints: intersecting words must agree on common letter

Hidden Variable Formulation

Primal CSP: \((X, D, C)\)
- \(X = \{x_1, \ldots, x_n\}\)
- \(D = \{d_1, \ldots, d_n\}\)
- \(C = B \setminus \{c_1, \ldots, c_q\}\)

Hidden formulation: \((X', D', C')\)
- \(X' = X \setminus \{x_a, \ldots, x_b\}\)
- \(D' = D \setminus \{d_a, \ldots, d_b\}\)
- \(C' = B \setminus \{c_q\}\)

- A new variable per non-binary constraint
- \(d_p = \text{rel}(c_p)\) values = permitted primal tuples
- \(\text{var}(c_p) = \{x_p, x_p\}\)
- \(x_i \in \text{var}(c_p)\)
- \(\text{rel}(c_p) = \) same values for \(x_i\)
Global Constraints

Real-life constraints: often complex, non-binary

c is global iff:
• arity(c) > 2
• c is logically equivalent to \( \{c_1, c_2, \ldots, c_k\} \) binary
• AC\( (c) \) prunes more than AC\( (c_1, c_2, \ldots, c_k) \)

Propagation:
• specialized algorithms
• exploit constraint semantics decrease AC complexity

Example: all-different

Var: F, N, S; Val: \{ \( \bullet \), \( \circ \) \}; Ctrs: \( N \neq S \neq F \neq N \)

3 binary constraints, they are AC, no pruning

1 ternary constraint, not AC, AC pruning \[ \] empty domain no solution!!
Example: *all-different*

Enforcing arc-consistency:
- $n$ variables, $d$ values
- $n(n-1)/2$ binary constraints: $O(n^2 d^2)$
- 1 $n$-ary constraint:
  - general purpose algorithm $O(d^n)$
  - specialized algorithm $O(n^2 d^2)$

---

**CP: Solving**

Solving decisions: select among alternatives
- search algorithm
- local consistency: level / how often
- heuristics: variable / value

Example: Map Colouring
- static or dynamic variable ordering?

Efficient solving:
- reasonable initial size of the search space
- drastic *reductions* of space during search
**CP Solving: Some Guidelines**

Easy/hard problems:
- hybrid search
- dynamic variable ordering: min domain / degree
- easy: FC / hard: MAC

One solution/All solutions:
- one solution: hybrid search
- all solutions: hybrid search or complete inference

For specific problems (scheduling, routing...) check:
- formulation, global constraints
- heuristics, experiences

---

**CP: Declarative Programming**

Declarative Programming: you declare
- Variables
- Domains
- Constraints
and ask the SOLVER to find a solution!!

SOLVER offers:
- Implementation for variables / domains / constraints
- Hybrid algorithm: backtracking + incomplete inference
- Global constraints + optimized AC propagation
- Empty domain detection
- Embedded heuristics
**Constraint Logic Programming**

Logic Programming:
- Depth-first search
- Unification: substitute equals by equals clauses/database

Existing solvers:
- Chip, Eclipse, Mozart, Sictus Prolog (and many others)

**Constraint Programming Libraries**

Library to be included in your program:
- Imperative programming

Provides:
- Special objects:
  - Variables / Domains / Constraints (global)
- Special functions to find:
  - One solution / the next solution

Existing Solvers:
- Ilog Solver, Choco