Combinatorial Optimisation using Constraint Programming (course 1DL441, 10 credits)

Solving problems that are larger than the universe!

Pierre Flener

ASTRA Research Group on Combinatorial Optimisation
Uppsala University Sweden

Autumn semester, periods 1 and 2:
http://www.it.uu.se/edu/course/homepage/consprog/
Optimisation is a science of service: to scientists, to engineers, and to society.
**Example (Touristic town competition)**

<table>
<thead>
<tr>
<th></th>
<th>Alva</th>
<th>Dan</th>
<th>Eva</th>
<th>Jim</th>
<th>Leo</th>
<th>Mia</th>
<th>Ulla</th>
</tr>
</thead>
<tbody>
<tr>
<td>Birka</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Falun</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lund</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mora</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sigtuna</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uppsala</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ystad</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Constraints to be **satisfied**:

- Fixed jury size: Every town is evaluated by 3 judges.
- Fixed travel load: Every judge evaluates 3 towns.
- Fairness: Every town pair has 1 common judge.
**Example (Touristic town competition)**

<table>
<thead>
<tr>
<th></th>
<th>Alva</th>
<th>Dan</th>
<th>Eva</th>
<th>Jim</th>
<th>Leo</th>
<th>Mia</th>
<th>Ulla</th>
</tr>
</thead>
<tbody>
<tr>
<td>Birka</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Falun</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lund</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Mora</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sigtuna</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Uppsala</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Ystad</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Constraints to be **satisfied**:

- Fixed jury size: Every town is evaluated by 3 judges.
- Fixed travel load: Every judge evaluates 3 towns.
- Fairness: Every town pair has 1 common judge.
### Example (Doctor allocation)

<table>
<thead>
<tr>
<th></th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
<th>Sun</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doctor A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Doctor B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Doctor C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Doctor D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Doctor E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Constraints to be satisfied:

- #doctors-on-call / day = 1
- #operations / workday ≤ 2
- #operations / week ≥ 7
- #appointments / doctor ≤ 1
- #appointments / week ≥ 4
- ...
### Example (Doctor allocation)

<table>
<thead>
<tr>
<th></th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
<th>Sun</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doctor A</td>
<td>call</td>
<td>–</td>
<td>op</td>
<td>–</td>
<td>op</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Doctor B</td>
<td>app</td>
<td>call</td>
<td>–</td>
<td>op</td>
<td>op</td>
<td>–</td>
<td>call</td>
</tr>
<tr>
<td>Doctor C</td>
<td>op</td>
<td>–</td>
<td>call</td>
<td>–</td>
<td>app</td>
<td>call</td>
<td>–</td>
</tr>
<tr>
<td>Doctor D</td>
<td>app</td>
<td>op</td>
<td>–</td>
<td>call</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Doctor E</td>
<td>op</td>
<td>–</td>
<td>op</td>
<td>app</td>
<td>call</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Constraints to be **satisfied**:

- #doctors-on-call / day = 1
- #operations / workday ≤ 2
- #operations / week ≥ 7
- #appointments / doctor ≤ 1
- #appointments / week ≥ 4
- ...
Example (Financial investment instrument design)

<table>
<thead>
<tr>
<th>Basket</th>
<th>Acer</th>
<th>Apple</th>
<th>Dell</th>
<th>HP</th>
<th>IBM</th>
<th>Sony</th>
<th>Sun</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basket 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basket 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basket 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basket 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basket 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basket 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basket 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Constraint to be **satisfied**:  
- Fixed basket size: Every basket contains 3 shares.

Objective value to be **minimised**:  
- Risk: Maximum observed overlap of any basket pair.
Example (Financial investment instrument design)

<table>
<thead>
<tr>
<th></th>
<th>Acer</th>
<th>Apple</th>
<th>Dell</th>
<th>HP</th>
<th>IBM</th>
<th>Sony</th>
<th>Sun</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basket 1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basket 2</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basket 3</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Basket 4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basket 5</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Basket 6</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Basket 7</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

Constraint to be satisfied:
- Fixed basket size: Every basket contains 3 shares.

Objective value to be minimised:
- Risk: Maximum observed overlap of any basket pair.
Example (What supertree is maximally consistent with several given trees that share some species?)

Oceanodroma castro
Hydrobates pelagicus
Macronectes giganteus
Fulmarus glacialoides
Fulmarus glacialis
Bulweria bulwerii
Procellaria cinerea
Calonectris diomedea
Puffinus assimilis
Puffinus puffinus
Puffinus yelkouan
Puffinus mauretanicus
THALASSARCHE BULLERI
Thalassarche chrysostoma
Phoebetria fusca
Phoebetria palpebrata
Phoebastria albatrus
Phoebastria immutabilis
Diomedea amsterdamensis
DIOMEDEA EPOMOPHORA

Pygoscelis adeliae
Eudyptula minor
Megadyptes antipodes
Eudyptes pachyrhynchos
Pelagodroma marina
DIOMEDEA EPOMOPHORA
THALASSARCHE BULLERI
Daption capense
Pelecanoides georgicus
Pachyptila vittata
Pachyptila turtur
Procellaria westlandica
Puffinus griseus
Puffinus huttoni
Pterodroma inexpectata
Pterodroma cookii
Example (Air traffic demand-capacity balancing)

Reroute flights, in height and speed, so as to balance the workload of air traffic controllers in a multi-sector airspace:
Example (Airspace sectorisation)

**Given:** an airspace split into \( c \) cells, and a targeted number \( s \) of sectors.

**Find:** a colouring of the cells into \( s \) connected convex sectors, with minimal imbalance of the workloads of their air traffic controllers.

There are \( s^c \) possible colourings, but very few satisfy the constraints and are optimal: intelligent search is necessary!
Air Traffic Management

Demand-capacity balancing

Airspace sectorisation

Contingency planning

Complexity resolution

<table>
<thead>
<tr>
<th>Flow</th>
<th>Time Span</th>
<th>Hourly Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>From: Arlanda</td>
<td>00:00 – 09:00</td>
<td>3</td>
</tr>
<tr>
<td>To: west, south</td>
<td>09:00 – 18:00</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>18:00 – 24:00</td>
<td>2</td>
</tr>
<tr>
<td>From: Arlanda</td>
<td>00:00 – 12:00</td>
<td>4</td>
</tr>
<tr>
<td>To: east, north</td>
<td>12:00 – 24:00</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Biology and Medicine

Phylogenetic supertrees

Medical image analysis

Haplotyping inference

Doctor allocation

Combinatorial Optimisation
Constraint Programming (CP)
Success Stories
Course Organisation

Periods 1 & 2

Pierre Flener
Programming and Testing

Robotic task sequencing

Macro-programming of wireless sensor networks

Compiler construction

Testing of base stations
Other Application Areas

School timetabling

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.30 AM</td>
<td>ME3302</td>
<td>LAB350702</td>
<td>ME33088</td>
<td>LAB350702</td>
</tr>
<tr>
<td>8.00 AM</td>
<td>CS5425</td>
<td>LAB350702</td>
<td>CS5673</td>
<td>LAB350702</td>
</tr>
<tr>
<td>9.00 AM</td>
<td>CS5425</td>
<td>LAB350702</td>
<td>CS5673</td>
<td>LAB350702</td>
</tr>
<tr>
<td>10.00 AM</td>
<td>CS5425</td>
<td>LAB350702</td>
<td>CS5673</td>
<td>LAB350702</td>
</tr>
<tr>
<td>11.00 AM</td>
<td>CS5425</td>
<td>LAB350702</td>
<td>CS5673</td>
<td>LAB350702</td>
</tr>
<tr>
<td>12.00 PM</td>
<td>CS5425</td>
<td>LAB350702</td>
<td>CS5673</td>
<td>LAB350702</td>
</tr>
<tr>
<td>1.00 PM</td>
<td>CS5425</td>
<td>LAB350702</td>
<td>CS5673</td>
<td>LAB350702</td>
</tr>
<tr>
<td>2.00 PM</td>
<td>CS5425</td>
<td>LAB350702</td>
<td>CS5673</td>
<td>LAB350702</td>
</tr>
<tr>
<td>3.00 PM</td>
<td>CS5425</td>
<td>LAB350702</td>
<td>CS5673</td>
<td>LAB350702</td>
</tr>
<tr>
<td>4.00 PM</td>
<td>CS5425</td>
<td>LAB350702</td>
<td>CS5673</td>
<td>LAB350702</td>
</tr>
<tr>
<td>5.00 PM</td>
<td>CS5425</td>
<td>LAB350702</td>
<td>CS5673</td>
<td>LAB350702</td>
</tr>
<tr>
<td>6.00 PM</td>
<td>CS5425</td>
<td>LAB350702</td>
<td>CS5673</td>
<td>LAB350702</td>
</tr>
</tbody>
</table>

Sports tournament design

Security
Detection of SQL injection vulnerabilities in JavaScript programs

Financial designs
Many important real-life problems are NP-hard and thus must be solved by intelligent search (unless P=NP):

- Personnel allocation, scheduling, time-tabling
- Transportation logistics: vehicle routing, ... 
- Packing: truck loading, ... 
- Configuration, design 
- Alignment of bio-molecules, phylogeny, ... 
- Financial investment portfolio design 
- ... 

Definition

In a constraint problem, decisions have to be made so that:

- The given constraints are satisfied.
- Optionally: A cost is minimal, or a benefit is maximal.

Combinatorial optimisation covers satisfaction problems and optimisation problems, for unknowns over discrete sets.
Combinatorial optimisation offers methods and tools for:

- **Modelling** constraint problems in some language.

- **Solving** constraint problems by *intelligent search*:
  - either by default `search` upon pushing a button
  - or by *systematic search* guided by user-given heuristics
  - or by *local search* guided by user-given heuristics
  - or by *hybrid search*

  plus *inference* or *relaxation* or both.

\[
\text{Solving} = \text{Search} + \text{Inference} + \text{Relaxation}
\]
Combinatorial Optimisation

Constraint Programming (CP)

Success Stories

Course Organisation

 Constraint Modelling

Example (Touristic town competition: \( \checkmark \) \( \nRightarrow 1 \), \( \nRightarrow 0 \))

- set of \textit{int:} \( \text{Towns}, \text{Judges} \)
- \textit{int:} \( r, c, \lambda \)
- \text{array}[\text{Towns, Judges}] \text{ of var } 0..1: \text{TTC}
- satisfy
  - \text{forall}(t \text{ in } \text{Towns}) \quad r = \text{sum}(j \text{ in } \text{Judges}) \quad \text{TTC}[t, j]
  - \text{forall}(j \text{ in } \text{Judges}) \quad c = \text{sum}(t \text{ in } \text{Towns}) \quad \text{TTC}[t, j]
  - \text{forall}(t_1, t_2 \text{ in } \text{Towns} \text{ where } t_1 < t_2)
    \quad \lambda = \text{sum}(j \text{ in } \text{Judges}) \quad \text{TTC}[t_1, j] \cdot \text{TTC}[t_2, j]

Example (Instance data)

- \text{Model-independent} instance data for the Sweden instance:
  - \text{Towns} = \{Birka, Falun, Lund, Mora, Sigtuna, Uppsala, Ystad\}
  - \text{Judges} = \{Ali, Dan, Eva, Jim, Leo, Mia, Ulla\}
  - \langle r, c, \lambda \rangle = \langle 3, 3, 1 \rangle
Constraint Programming

Constraint programming (CP) offers methods and tools for:

- **Modelling** constraint problems in a high-level language.

- **Solving** constraint problems by intelligent search:
  - either by default search upon pushing a button
  - or by systematic search guided by user-given heuristics
  - or by local search guided by user-given heuristics
  - or by hybrid search

  plus *inference*.

**Slogan of CP:**

Constraint Program $=$ Model $+$ Search
CP Solving = Search + Inference

A CP solver conducts search interleaved with inference:

Each constraint has an inference algorithm.
Inference

Example

Consider the constraint \texttt{CONNECTED\([C_1, \ldots, C_n]\)}, which enforces max one stretch per colour among the \(n\) cells.

From

\[
\begin{array}{ccccccc}
\ldots & \ldots & \black & \red & \? & \? & \? & \red & \white & \ldots & \? & \ldots
\end{array}
\]

the \texttt{CONNECTED} constraint \textit{infers}

\[
\begin{array}{ccccccc}
\text{no red white} & \black & \red & \red & \red & \red & \red & \white & \text{no red black}
\end{array}
\]

\textbf{Inference} is the elimination of (some) impossible values, and thereby accelerates otherwise blind search.
Is CP a Silver Bullet for NP-Hard Problems?

No!
CP solvers work in a way orthogonal & complementary to:

- Operations research (OR):
  - linear programming (LP)
  - integer linear programming (IP)
  - mixed integer linear programming (MIP)
  - non-linear programming (NLP)
  - ...

- Local search (LS): tabu, genetic, ...

- ...

- Boolean satisfiability (SAT), modulo theories (SMT)

This leads to hybridised optimisation technologies!
CP principles also apply to continuous optimisation!
Success Stories: Users and Contributors

CP has become the technology of choice in scheduling, configuration, timetabling, personnel allocation, . . .
Course Contents

- Consistency
- Propagation
- Systematic search
- Local search
- Global constraints: DISTINCT, LINEAR, ELEMENT, REGULAR, etc
- Modelling, including symmetry detection and breaking
- Set variables and set constraints
- Scheduling
- Guest lectures: applications, extensions, etc

Check the previous course instance (at http://www.it.uu.se/edu/course/homepage/consprog) for an idea of what the course may look like this time.
Learning Outcomes

In order to pass, the student must be able to:

- describe the basic underlying concepts of CP
- describe how a generic CP solver works
- model a combinatorial problem using CP
- devise suitable search heuristics
- compare CP programs for a combinatorial problem
- evaluate impact of redundant variables or constraints
- detect and break (some of the) symmetries in a CP
- enhance a CP solver with an additional constraint
- outline other combinatorial optimisation technologies
Course Organisation

- Periods 1 and 2 of the autumn semester
- 22 lectures, in English
- No textbook required
- **C++ programming** with the free [Gecode.org](http://www.gecode.org) library:
  - 3 assignments, to be done in pairs (2 credits)
  - 3 project parts, to be done in pairs (3 credits)
    in $6 \cdot 3 = 18$ help sessions and 6 solution sessions

- 1 closed-book **exam**, to be done alone (5 credits)

**Prerequisites**: able to define or learn basic concepts in algebra, combinatorics, logic, graph theory, and set theory; able to implement basic search algorithms