Constraint Programming in a Nutshell

Pierre Flener

ASTRA Research Group
on Combinatorial Optimisation
Department of Information Technology
Uppsala University
Sweden
www.it.uu.se/research/group/astra

NordConsNet 2017
2017-05-22
Optimisation is a science of **service**:
to scientists, to engineers, to artists, and to society.
Applications in Air Traffic Management

Demand vs capacity

Airspace sectorisation

Contingency planning

<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>

Workload balancing
Applications in Biology and Medicine

Phylogenetic supertree

Haplotype inference

Medical image analysis

Doctor rostering

- 10 -
Applications in Programming and Testing

Robotic task sequencing

Sensor net configuration

Compiler design

Base station testing
## 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>9:00</td>
<td>MFT332</td>
<td>LMAC33072</td>
<td>MFT336</td>
<td>LMAC33072</td>
</tr>
<tr>
<td>Constraint Problems</td>
<td>Differential Equations</td>
<td>Computer Algebra II</td>
<td>Numerical Analysis</td>
<td>Computer Algebra II</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:00</td>
<td>MFT332</td>
<td>LMAC33072</td>
<td>MFT336</td>
<td>LMAC33072</td>
</tr>
<tr>
<td>Constraint Problems</td>
<td>Differential Equations</td>
<td>Computer Algebra II</td>
<td>Numerical Analysis</td>
<td>Computer Algebra II</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:00</td>
<td>CSY328</td>
<td>MFT336</td>
<td>CSY328</td>
<td>MFT336</td>
</tr>
<tr>
<td>Algorithms and Data</td>
<td>Further Linear Algebra</td>
<td>Numerical Analysis I</td>
<td>Further Linear Algebra</td>
<td>Numerical Analysis I</td>
</tr>
<tr>
<td>Structures</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:00</td>
<td>FAS2</td>
<td>COMP311</td>
<td>FAS2</td>
<td>COMP311</td>
</tr>
<tr>
<td>Mathematics</td>
<td></td>
<td></td>
<td>Mathematics</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1:00</td>
<td>ESY472</td>
<td>MFT336</td>
<td>ESY472</td>
<td>MFT336</td>
</tr>
<tr>
<td>Computer Graphics</td>
<td></td>
<td></td>
<td>Computer Graphics</td>
<td></td>
</tr>
<tr>
<td>2:00</td>
<td>CSPM9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tutorial</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3:00</td>
<td>ESY314</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithms and Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structures</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Sports tournament design

#### School timetabling

#### Sports tournament design

### Security: SQL injection?

### Container packing
Constraint programming (CP) offers methods & tools for:

**what:** Modelling constraint problems in a *high-level* language.
Constraint Programming Technology

Constraint programming (CP) offers methods & tools for:

what: Modelling constraint problems in a high-level language.

and

how: Solving constraint problems intelligently by:

- either default search upon pushing a button
- or systematic search guided by user-given heuristics plus inference, called propagation, but little relaxation.

A solver is a software that takes a model as input and tries to solve the modelled problem.

Slogan of CP: Constraint Program = Model + Search
Constraint Programming Technology

Constraint programming (CP) offers methods & tools for:

what: Modelling constraint problems in a high-level language.

and

describe: Solving constraint problems intelligently by:

- either default search upon pushing a button
Constraint Programming Technology

Constraint programming (CP) offers methods & tools for:

what: Modelling constraint problems in a high-level language.

and

how: Solving constraint problems **intelligently** by:
  - either default **search** upon pushing a button
  - or **systematic search** guided by user-given heuristics
Constraint Programming Technology

Constraint programming (CP) offers methods & tools for:

what: Modelling constraint problems in a high-level language.

and

how: Solving constraint problems intelligently by:

• either default search upon pushing a button
• or systematic search guided by user-given heuristics plus inference, called propagation, but little relaxation.
Constraint Programming Technology

Constraint programming (CP) offers methods & tools for:

what: Modelling constraint problems in a high-level language.

and

how: Solving constraint problems intelligently by:
  • either default search upon pushing a button
  • or systematic search guided by user-given heuristics
    plus inference, called propagation, but little relaxation.

A solver is a software that takes a model as input
and tries to solve the modelled problem.
Constraint Programming Technology

Constraint programming (CP) offers methods & tools for:

what:  Modelling constraint problems in a high-level language.

and

how:  Solving constraint problems intelligently by:
  
  • either default search upon pushing a button
  
  • or systematic search guided by user-given heuristics plus inference, called propagation, but little relaxation.

A solver is a software that takes a model as input and tries to solve the modelled problem.

Slogan of CP:

Constraint Program = Model [ + Search ]
Example (Doctor rostering)

<table>
<thead>
<tr>
<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>
</tr>
<tr>
<td>Doctor B</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>
</tr>
<tr>
<td>Doctor D</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>
</tr>
</tbody>
</table>

**Constraints to be satisfied:**

1. \#doctors-on-call / day = 1
2. \#operations / workday \(\leq\) 2
3. \#operations / week \(\geq\) 7
4. \#appointments / week \(\geq\) 4
5. day off after operation day
6. ...

**Objective function to be minimised:**

- Cost: ...
### Example (Doctor rostering)

<table>
<thead>
<tr>
<th>Doctor</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>A</td>
<td>call</td>
<td>–</td>
<td>oper</td>
<td>–</td>
<td>oper</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>B</td>
<td>app</td>
<td>call</td>
<td>–</td>
<td>oper</td>
<td>–</td>
<td>–</td>
<td>call</td>
</tr>
<tr>
<td>C</td>
<td>oper</td>
<td>–</td>
<td>call</td>
<td>app</td>
<td>app</td>
<td>call</td>
<td>–</td>
</tr>
<tr>
<td>D</td>
<td>app</td>
<td>oper</td>
<td>–</td>
<td>call</td>
<td>oper</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>E</td>
<td>oper</td>
<td>–</td>
<td>oper</td>
<td>–</td>
<td>call</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

**Constraints to be satisfied:**

1. #doctors-on-call / day = 1
2. #operations / workday ≤ 2
3. #operations / week ≥ 7
4. #appointments / week ≥ 4
5. Day off after operation day
6. …

**Objective function to be minimised:**

- Cost: …
Example (Doctor rostering)

```
set of int: Days = 1..7;
set of int: Mon2Fri = 1..5;
set of int: Doctors = 1..5;
enum: ShiftTypes = {app, call, oper, none};

array[Doctors,Days] of var ShiftTypes: Roster;

solve minimize ...;  % objective function

constraint forall (d in Days)
  (count(Roster[..,d],call) = 1);
constraint forall (w in Mon2Fri)
  (count(Roster[..,w],oper) <= 2);
constraint count(Roster,oper) >= 7;
constraint count(Roster,app) >= 4;
constraint forall (d in Doctors)
  (regular(Roster[d,..], (oper none|app|call|none)*));
...
```

... % other constraints
Stand-alone languages:

- **ALICE** by Jean-Louis Laurièrè, France, 1976
- **CHIP** at ECRC, Germany, 1987 – 1990, then marketed by Cosytec, France
- **OPL**, by P. Van Hentenryck, USA, and ILOG, France: front-end to both ILOG CP Optimizer and ILOG CPLEX
- **Comet**, by P. Van Hentenryck and L. Michel, USA
- **MiniZinc**, at U. of Melbourne and Monash U., Australia
- . . .
Stand-alone languages:

- **ALICE** by Jean-Louis Laurière, France, 1976
- **CHIP** at ECRC, Germany, 1987 – 1990, then marketed by Cosytec, France
- **OPL**, by P. Van Hentenryck, USA, and ILOG, France: front-end to both **ILOG CP Optimizer** and **ILOG CPLEX**
- **Comet**, by P. Van Hentenryck and L. Michel, USA
- **MiniZinc**, at U. of Melbourne and Monash U., Australia
  - ...

**Libraries** (the ones listed before “;” are open-source):

- Prolog: **ECLiPSe**, ...; **SICStus Prolog**, ...
- C++: **Gecode**, **OR-Tools**; **IBM CP Optimizer**, **CHIP**, ...
- Java: **Choco**, **Google OR-Tools**, **JaCoP**, ...; ...
- Scala: **OscaR**; ...
  - ...
Success stories: CP = technology of choice in scheduling, configuration, personnel rostering, timetabling, . . .
Scope of Constraint Programming

CP has a wide scope, as it addresses:

- satisfaction problems and optimisation problems
- discrete variables and continuous variables
- linear constraints and non-linear constraints

in principle in any combinations thereof, by:

- systematic search, if optimality more crucial than speed
- local search, if speed is more crucial than optimality
Opportunities for CP

Rapid prototyping, with high solving performance, when:
- Constraints are, still or again, subject to experiments
- Partition into hard & soft constraints yet undetermined

Combinatorial structure is impure, due to side constraints.

It is time to consider all or more problem constraints.

Domain knowledge exploitable for problem-specific search.

It is a configuration problem.

It is a personnel rostering problem.

It is a scheduling problem.

It is a time-tabling problem.


Ph. Baptiste, C. Le Pape, and W. Nuijten.  

P. Flener, M. Carlsson, and Ch. Schulte.  


