DM841
Discrete Optimization

Part II
Lecture 1
Introduction to Constraint Programming

Marco Chiarandini
Department of Mathematics & Computer Science
University of Southern Denmark
Outline

1. Course Introduction
2. Constraint Programming
   Example
3. Modelling
4. Modeling in MP and CP
Outline

1. Course Introduction

2. Constraint Programming
   Example

3. Modelling

4. Modeling in MP and CP
Schedule and Material

▶ Schedule:
  ▶ Wednesday 10:15-12:00
  ▶ Friday 8:15-10:00
  ▶ Tuesday 14:15-16:00 (only from week 48)
  ▶ Officially last lecture in Week 51, Friday, 19th December, 2014

▶ Communication tools
  ▶ Public Course Webpage (Wp)
    http://www.imada.sdu.dk/~marco/DM841/
  ▶ In Blackboard (Bb):
    ▶ Announcements
    ▶ Documents (Photocopies)
  ▶ Personal email
  ▶ You are welcome to visit me in my office in working hours.
Contents

- The technology behind Constraint Programming
- CP in gecode
- Modelling and solving constrained optimization problems
Evaluation

- Two obligatory assignments (pass/fail)

- Final assignment (50% of final grade)
  - Model
  - Implementation
  - Report (Max 10 pages)
References

- Main References:
  
  **RBW** F. Rossi, P. van Beek and T. Walsh (ed.), *Handbook of Constraint Programming*, Elsevier, 2006
  

- Photocopies (Bb)
- Articles from the Webpage
- Lecture slides
- Assignments
- Active participation
Under development:

Here, we will use free and open-source software:

- Gecode (C++) – MIT license
Outline

1. Course Introduction

2. Constraint Programming
   Example

3. Modelling

4. Modeling in MP and CP
Outline

1. Course Introduction

2. Constraint Programming
   Example

3. Modelling

4. Modeling in MP and CP
Put a different number in each circle (1 to 8) such that adjacent circles cannot take consecutive numbers.
Constraint Programming
An Introduction
by example

Patrick Prosser
with the help of Toby Walsh, Chris Beck,
Barbara Smith, Peter van Beek, Edward Tsang, ...
A Puzzle

- Place numbers 1 through 8 on nodes
  - Each number appears exactly once
  - No connected nodes have consecutive numbers

You have 8 minutes!
Heuristic Search

Which nodes are hardest to number?
Heuristic Search
Heuristic Search

Which are the least constraining values to use?
Heuristic Search

Values 1 and 8
Heuristic Search

Values 1 and 8

Symmetry means we don’t need to consider: 8 1
We can now eliminate many values for other nodes
Inference/propagation

\{1,2,3,4,5,6,7,8\}
Inference/propagation

\{2,3,4,5,6,7\}
Inference/propagation

\{3,4,5,6\}
Inference/propagation

\{3,4,5,6\}

By symmetry
Inference/propagation

\{3,4,5,6\} \quad \{1,2,3,4,5,6,7,8\}
Inference/propagation

\{3,4,5,6\} \quad \{2,3,4,5,6,7\}

\[\begin{array}{ccccccc}
\text{?} & 1 & \text{?} & \text{?} & \text{?} & \text{?} & \text{?} \\
\text{?} & \text{?} & \text{?} & \text{?} & \text{?} & \text{?} & \text{?} \\
\text{?} & \text{?} & \text{?} & \text{?} & \text{?} & \text{?} & \text{?} \\
\text{?} & \text{?} & \text{?} & \text{?} & \text{?} & \text{?} & \text{?} \\
\end{array}\]
Inference/propagation

\{3,4,5,6\} \quad \{3,4,5,6\}

\begin{align*}
&\text{?} & & \text{?} \\
&\text{?} & & \text{1} \\
&\text{1} & & \text{8} \\
&\text{?} & & \text{?} \\
\end{align*}

\{3,4,5,6\}
Inference/propagation

By symmetry
Inference/propagation

{3,4,5,6}  {3,4,5,6}

{3,4,5,6}  {3,4,5,6,7}  {3,4,5,6}  {3,4,5,6,7}  {3,4,5,6}  {3,4,5,6,7}  {3,4,5,6,7}  {2,3,4,5,6}
Value 2 and 7 are left in just one variable domain each
And propagate …
And propagate …
And propagate …
Inference/propagation

Guess a value, but be prepared to backtrack …
Inference/propagation

{3,4,5} \quad \{4,5,6\}

Guess a value, but be prepared to backtrack …
And propagate …
Inference/propagation

And propagate …
Inference/propagation

Guess another value …
Inference/propagation

Guess another value …
And propagate …
And propagate …
Inference/propagation

One node has only a single value left …
Inference/propagation
Solution
The Core of Constraint Computation

- Modelling
  - Deciding on variables/domains/constraints
- Heuristic Search
- Inference/Propagation
- Symmetry
- Backtracking
Hardness

• The puzzle is actually a hard problem
  – NP-complete
Constraint programming

• Model problem by specifying constraints on acceptable solutions
  – define variables and domains
  – post constraints on these variables

• Solve model
  – choose algorithm
    • incremental assignment / backtracking search
    • complete assignments / stochastic search
  – design heuristics
Example CSP

- Variable, $v_i$ for each node
- Domain of $\{1, \ldots, 8\}$
- Constraints
  - All values used
    \[ \text{allDifferent}(v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8) \]
  - No consecutive numbers for adjoining nodes
    \[ |v_1 - v_2| > 1 \]
    \[ |v_1 - v_3| > 1 \]
    \[ \ldots \]
more examples?
Do you know any constraint satisfaction problems?

To a man with a hammer, everything looks like a nail.
In the pyramid above, two adjacent bricks added together give the value of the brick above. Find the value for the brick marked ?
Crossword puzzle generation
Monday, 28 Sep 2009

Across

1 Mass-produced legislation? [3,3]
4 Guard ordered to breached line [9]

Down

1 Not subject put on in advance [7]
2 Club sought to give female support [9]
An example

Make a crossword puzzle!
Given the above grid and a dictionary, fill it.
Then go get the clues (not my problem)
1A 1 across
4D 4 down
2D 2 down
4A 4 across
7D 7 down

An example

Variables
An example

Constraints

1A-4D: 4th of 1A equals 1st of 4D
1A-2D: 2nd of 1A equals 1st of 2D
2D-4A: 4th of 2D equals 2nd of 4D
4D-4A: 4th of 4A equals 4th of 4D
4A-7D: 7th of 4A equals 2nd of 7D
An example

1A: any 6 letter word
4A: any 8 letter word
4D: any 5 letter word
2D: any 7 letter word
7D: any 3 letter word

Domains (also unary constraints!)
An example

Find an assignment of values to variables, from their domains, such that the constraints are satisfied (or show that no assignment exists)

A CSP!
Choose a variable
Assign it a value
Check compatibility
If not compatible try a new value
If no values remain re-assign previous variable

Good old fashioned BT!
What variable should I choose?

What value should I choose?

What reasoning can I do when making an assignment?

What reasoning can I do on a dead end?

Questions?

Decisions, decisions!
An example

Is there an alternative representation?
These are some of the problems that have been tackled by CP

- factory scheduling (JSSP)
- vehicle routing (VRP)
- packing problems (NumPart and BinPack)
- timetabling (exams, lectures, trains)
- configuration and design (hardware)
- workforce management (call centres, etc)
- car sequencing (assembly line scheduling)
- super tree construction (bioinformatics)
- network design (telecoms problem)
- gate arrival (at airports)
- logistics (Desert Storm an example)
- aircraft maintenance schedules
- aircraft crew scheduling (commercial airlines)
- air cover for naval fleet
Constraint Programming = model (representation) + 
propagation (reasoning, inference) + 
search (reasoning, inference)
Applications

- Operation research (optimization problems)
- Graphical interactive systems (to express geometrical correctness)
- Molecular biology (DNA sequencing, 3D models of proteins)
- Finance
- Circuit verification
- Elaboration of natural languages (construction of efficient parsers)
- Scheduling of activities
- Configuration problem in form compilation
- Generation of coherent music programs ?.
- Data bases
- ...

http://hsimonis.wordpress.com/
Applications

Distribution of technology used at Google for optimization applications developed by the operations research team

[Slide presented by Laurent Perron on OR-Tools at CP2013]
List of Contents

- **Modeling**
- Overview on global constraints
- Introduction to Gecode
- Notions of local consistency
- Constraint propagation algorithms
- Filtering algorithms for global constraints
- Search
- Set variables
- Symmetries
- Logic-Based Benders Decomposition and/or Large Neighborhood Search
Outline

1. Course Introduction

2. Constraint Programming
   Example

3. Modelling

4. Modeling in MP and CP
The **domain** of a variable $x$, denoted $D(x)$, is a finite set of elements that can be assigned to $x$.

A **constraint** $C$ on $X$ is a subset of the Cartesian product of the domains of the variables in $X$, i.e., $C \subseteq D(x_1) \times \cdots \times D(x_k)$. A tuple $(d_1, \ldots, d_k) \in C$ is called a **solution** to $C$.

Equivalently, we say that a solution $(d_1, \ldots, d_k) \in C$ is an assignment of the value $d_i$ to the variable $x_i$, $\forall 1 \leq i \leq k$, and that this assignment satisfies $C$. If $C = \emptyset$, we say that it is **inconsistent**.

**Extensional**: specifies the good (or bad) tuples (values)

**Intensional**: specifies the characteristic function
Constraint Satisfaction Problem (CSP)

A CSP is a finite set of variables $X$, together with a finite set of constraints $C$, each on a subset of $X$. A solution to a CSP is an assignment of a value $d \in D(x)$ to each $x \in X$, such that all constraints are satisfied simultaneously.

Constraint Optimization Problem (COP)

A COP is a CSP $P$ defined on the variables $x_1, \ldots, x_n$, together with an objective function $f : D(x_1) \times \cdots \times D(x_n) \to Q$ that assigns a value to each assignment of values to the variables. An optimal solution to a minimization (maximization) COP is a solution $d$ to $P$ that minimizes (maximizes) the value of $f(d)$. 
Task:

- determine whether the CSP/COP is consistent (has a solution):
- find one solution
- find all solutions
- find one optimal solution
- find all optimal solutions
Solving CSPs

- **Systematic search:**
  - choose a variable $x_i$ that is not yet assigned
  - create a choice point, i.e. a set of mutually exclusive & exhaustive choices, e.g. $x_i = v$ vs $x_i \neq v$
  - try the first & backtrack to try the other if this fails

- **Constraint propagation:**
  - add $x_i = v$ or $x_i \neq v$ to the set of constraints
  - re-establish local consistency on each constraint
    - remove values from the domains of future variables that can no longer be used because of this choice
  - fail if any future variable has no values left
Representing a Problem

- If a CSP $\mathcal{P} = \langle X, D, \mathcal{C} \rangle$ represents a problem $\mathcal{P}$, then every solution of $\mathcal{P}$ corresponds to a solution of $\mathcal{P}$ and every solution of $\mathcal{P}$ can be derived from at least one solution of $\mathcal{P}$

- More than one solution of $\mathcal{P}$ can be represented by the same solution of $\mathcal{P}$, if modelling removes symmetry

- The variables and values of $\mathcal{P}$ represent entities in $\mathcal{P}$

- The constraints of $\mathcal{P}$ ensure the correspondence between solutions

- The aim is to find a model $\mathcal{P}$ that can be solved as quickly as possible (Note that shortest run-time might not mean least search!)
Interactions with Search Strategy

Whether a model is better than another can depend on the search algorithm and search heuristics

- Let’s assume that the search algorithm is fixed although different level of consistency can also play a role

- Let’s also assume that choice points are always $x_i = \nu$ vs $x_i \neq \nu$

- Variable (and value) order still interact with the model a lot

- Is variable & value ordering part of modelling?

  In practice it is.
  but it depends on the modeling language used
Global Constraint: alldifferent

Global constraint:
set of more elementary constraints that exhibit a special structure when considered together.

alldifferent constraint
Let \( x_1, x_2, \ldots, x_n \) be variables. Then:

\[
\text{alldifferent}(x_1, \ldots, x_n) = \\
\{(d_1, \ldots, d_n) \mid \forall i, d_i \in D(x_i), \forall i \neq j, d_i \neq d_j\}.
\]

Constraint arity: number of variables involved in the constraint

Note: different notation and names used in the literature
Global Constraint Catalog

http://www.emn.fr/z-info/sdemasse/gccat/sec5.html

Global Constraint Catalog

Corresponding author: Nicolas Beldiceanu nicolas.beldiceanu@emn.fr
Online version: Sophie Demassey sophie.demassey@emn.fr

Global Constraint Catalog
html / 2009-12-16

Search by:

<table>
<thead>
<tr>
<th>NAME</th>
<th>Keyword</th>
<th>Meta-keyword</th>
<th>Argument pattern</th>
<th>Graph description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bibliography</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Index</td>
</tr>
</tbody>
</table>

Keywords (ex: Assignment, Bound consistency, Soft constraint,...) can be searched by Meta-keywords (ex: Application area, Filtering, Constraint type,...)

About the catalogue

The catalogue presents a list of 348 global constraints issued from the literature in constraint programming and from popular constraint systems. The semantic of each constraint is given together with a description in terms of graph properties and/or automata.
Outline

1. Course Introduction
2. Constraint Programming
   Example
3. Modelling
4. Modeling in MP and CP
Computational Models

Three main Computational Models to solve (combinatorial) constrained optimization problems:

▶ Mathematical Programming (LP, ILP, QP, SDP, ...)

▶ Constraint Programming (CSP as a model, SAT as a very special case)

▶ Local Search (... and Meta-heuristics)

▶ Others? Dynamic programming, dedicated algorithms, satisfiability modulo theory, etc.
Modeling:

1. identify:
   - variables and domains
   - constraints
   - objective functions

   that formulate the problem

2. express what in point 1) in a way that allows the solution by available software
Variables

In MILP: real and integer (mostly binary) variables

In LS: integer variables

In CP:

- finite domain integer (including Booleans),

- continuous with interval constraints

- structured domains: finite sets, multisets, graphs, ...
Constraint Programming vs LS

► In LS modelling, we define: i) variables and domains, ii) objective function and iii) neighborhood. Constraints entered in the objective function. In CP constraints are declared explicitly.

► In LS, constraints handled indirectly either by solution representation or relaxed in the objective function. They can be violated. More constraints may be bad.

► In CP handled directly, used to remove values from variable domains, they cannot be violated. More constraints may be good.
Constraint Programming vs MILP

- In MILP we formulate problems as a set of linear inequalities.
- In CP we describe substructures (so-called global constraints) and combine them with various combinators.
  - Substructures capture building blocks often (but not always) computationally tractable by special-purpose algorithms.

- CP models can:
  - be solved by the constraint engine
  - be linearized and solved by their MIP solvers;
  - be translated in CNF and solved by SAT solvers;
  - be handled by local search.

- In MILP the solver is often seen as a black-box.
  In CP and LS solvers leave the user the task of programming the search.

- CP = model + propagation + search
  constraint propagation by domain filtering $\rightarrow$ inference
  search = backtracking or branch and bound (local search)
Example: Sudoku

How can you solve the following Sudoku?

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
<td>8</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>9</td>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>
Sudoku: LS model

Model:
- Variables: $X_{ij} \in N$, restrict to be permutation over rows
- Evaluation function: count number of conflicts in the subsquares and on columns
- Neighborhood: exchange two values on a row

Search: eg, iterative improvement, stochastic local search, tabu search, etc.
Sudoku: ILP model

Let $y_{ijt}$ be equal to 1 if digit $t$ appears in cell $(i,j)$. Let $N$ be the set \{1, \ldots, 9\}, and let $J_{kl}$ be the set of cells $(i,j)$ in the $3 \times 3$ square in position $k,l$.

\[
\sum_{j \in N} y_{ijt} = 1, \quad \forall i, t \in N,
\]
\[
\sum_{j \in N} y_{jit} = 1, \quad \forall i, t \in N,
\]
\[
\sum_{i,j \in J_{kl}} y_{ijt} = 1, \quad \forall k,l = \{1, 2, 3\}, t \in N,
\]
\[
\sum_{t \in N} y_{ijt} = 1, \quad \forall i,j \in N,
\]
\[
y_{i,j,a_{ij}} = 1, \quad \forall i,j \in \text{given instance}.
\]
Sudoku: CP model

Model:

\[ X_{ij} \in N, \]
\[ X_{ij} = a_{ij}, \]
\[ \text{alldifferent}([X_{1i}, \ldots, X_{9i}]), \]
\[ \text{alldifferent}([X_{i1}, \ldots, X_{i9}]), \]
\[ \text{alldifferent}\{X_{ij} \mid ij \in J_{kl}\}, \]

\[ \forall i, j \in N, \]
\[ \forall i, j \in \text{given instance}, \]
\[ \forall i \in N, \]
\[ \forall k, l \in \{1, 2, 3\}. \]

Search: backtracking
Sudoku: CP model (revisited)

\[ X_{ij} \in \mathbb{N}, \]
\[ X_{ij} = a_t, \]
\[ \text{alldifferent}\left([X_{1i}, \ldots, X_{9i}]\right), \]
\[ \text{alldifferent}\left([X_{i1}, \ldots, X_{i9}]\right), \]
\[ \text{alldifferent}\left(\{X_{ij} \mid ij \in J_{kl}\}\right), \]

\[ \forall i, j \in \mathbb{N}, \]
\[ \forall i, \text{given instance}, \]
\[ \forall i \in \mathbb{N}, \]
\[ \forall k, l \in \{1, 2, 3\}. \]

Redundant Constraint:

\[ \sum_{j \in \mathbb{N}} X_{ij} = 45, \quad \forall i \in \mathbb{N}, \]
\[ \sum_{j \in \mathbb{N}} X_{ji} = 45, \quad \forall i \in \mathbb{N}, \]
\[ \sum_{ij \in J_{kl}} X_{ij} = 45, \quad k, l \in \{1, 2, 3\}. \]
Hybrid Methods?

Strengths:
- CP is excellent to explore highly constrained combinatorial spaces quickly
- Math programming is particularly good at deriving lower bounds
- LS is particularly good at deriving upper bounds

How to combine them to get better “solvers”?
- Exploiting OR algorithms for filtering
- Exploiting LP (and SDP) relaxation into CP
- Hybrid decompositions:
  1. Logical Benders decomposition
  2. Column generation
  3. Large-scale neighborhood search
Models interact with solution process hence models in CP and IP are different.

To integrate one needs:

▷ to know both sides
▷ to have available a modelling language that allows integration (python, C++, MiniZinc)

There are typically alternative ways to formulate a problem. Some may yield faster solutions.

Typical procedure:

▷ begin with a straightforward model to solve a small problem instance
▷ alter and refine the model while scaling up the instances to maintain tractability
CP and LS as alternative approaches

- Constraint Programming
  1. Declarative modeling language (model&solve paradigm)
  2. Global constraints and filtering algorithm
  3. Search tree

- Constraint-based Local Search
  1. Declarative modeling language (model&solve paradigm)
  2. Global constraints and penalty functions
  3. Local Search and Meta-heuristic (incremental evaluation and commitment of moves)

- Hybrid methods (... once there was Comet)
Resume

- LS models
  - largest flexibility but everything to implement

- MILP models
  - impose modelling rules: linear inequalities and objectives
  - emphasis on tightness and compactness of LP, strength of bounds (remove dominated constraints)

- CP models
  - a large variety of algorithms communicating with each other: global constraints
  - more expressiveness
  - emphasis on exploiting substructures, include redundant constraints