Constraint Programming (CP) and IBM CP Optimizer

Speaker: Xinbo Wang
Outline

- A glimpse of Constraint Programming (CP)
- A glimpse of Integer Mathematical Programming (MP)
- Comparison of CP and MP
- A brief introduction of IBM CP optimizer
Outline

- A glimpse of Constraint Programming (CP)
- A glimpse of Integer Mathematical Programming (MP)
- Comparison of CP and MP
- A brief introduction of IBM CP optimizer
What is constraint programming?

- CP is an optimization technology which is complementary to Mathematical programming (e.g. ILP) taking a different approach to optimization, but sharing similarities.

- It is a relatively new technology developed in the computer science and artificial intelligence communities.

- It has found an important role in scheduling and highly combinational problems (for ours).
Applications

- Job shop scheduling
- Assembly line smoothing and balancing
- Cellular frequency assignment
- Airline crew rostering Nurse scheduling
- Shift planning
- Maintenance planning
- and scheduling
- Airport gate allocation and stand planning
- Production scheduling
- Transport scheduling
- Warehouse management
- Course timetabling
How Constraint Programming Works?

- CP is a *constructive* approach
- Values are assigned to variables one at a time to extend a partial solution to a complete solution
- At a point, it may be useless to further extend a partial solution as at least one constraint is already violated by the partial solution
  - The solver *backtracks* and tries a different value for a previously assigned variable
  - All possible assignments of values to variables can be examined in this way
How Constraint Programming Works?

- In CP, the basic search behaviour is tree search
- Including search space reduction via *domain filtering*
- Domain filtering
  - Before each value-variable assignment, *domain filtering* occurs
  - Each value of a variable which cannot be used in a solution (given the current partial assignment) can be removed
  - Each constraint type has a *specialized* algorithm which filters domains
Outline

- A glimpse of Constraint Programming (CP)
- A glimpse of Integer Mathematical Programming (MP)
- Comparison of CP and MP
- A brief introduction of IBM CP optimizer
What is Integer Linear Programming?

- **Integer programming** problem is a mathematical optimization or feasibility program in which some or all of the variables are restricted to be integers.
- The objective function and the constraints (other than the integer constraints) are linear.
- **Mixed integer linear programming** (MILP) involves problems in which only some of the variables are constrained to be integers, while other variables are allowed to be non-integers.
- **Zero-one linear programming** involves problems in which the variables are restricted to be either 0 or 1. Note that any bounded integer variable can be expressed as a combination of binary variables. For example, given an integer variable, \( x \), the variable can be expressed:

\[
x = x_1 + 2x_2 + 4x_3 + \ldots + 2^{\left\lfloor \log_2 U \right\rfloor} x_{\left\lfloor \log_2 U \right\rfloor + 1}.
\]
Outline

- A glimpse of Constraint Programming (CP)
- A glimpse of Integer Linear Programming (ILP)
- Comparison of CP and MP
- A brief introduction of IBM CP optimizer
Comparison

• CP works with the same concepts as mathematical programming: decision variables, objective function, and constraints.

• CP only discrete decision variables (integer or Boolean) vs MP discrete and continuous decision variables.

• CP logical constraints and arithmetic expressions (modulo, integer division, etc.) vs MP models only linear constraints or quadratic convex constraints.

• CP no limitation on the arithmetic constraints that can be set on decision variables vs MP specific to a class of problems whose solution space satisfies certain mathematical properties.

• Each optimization engine uses different techniques and algorithms to find feasible solutions and optimize them.
### A Tabular View

<table>
<thead>
<tr>
<th>Feature</th>
<th>MP</th>
<th>CP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relaxation</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>GAP measure</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Optimality proof</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Modeling limitations</td>
<td>Quadratic problems are limited to PSD (Positive Semi Definite) problems and Second Order Cone Programming (SOCP) problems</td>
<td>Discrete problems</td>
</tr>
<tr>
<td>Specialized constraints</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Logical constraints</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Theoretical grounds</td>
<td>Algebra</td>
<td>Graph theory and algorithmic</td>
</tr>
<tr>
<td>Modeler support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Model and run</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Benefits of constraint programming

- Solve time tabling problems and sequencing problems.

- An alternative to mathematical programming for allocation problems that have a slow convergence.

- Constraint programming has native support for:
  - Nonlinear costs or constraints
  - Logical constraints and statements
  - Constraints on and between interval variables
  - Compatibility or incompatibility constraints
  - More useful features
Expressions and Constraints

- **Arithmetic constraints**
  - $x + y$, $x - y$, $x \times y$, $x / y$, $x \text{ div } y$, $x \% y$
  - $\text{min}$, $\text{max}$, $\text{abs}$, $\text{log}$, $\text{exp}$ etc.
  - Piecewise linear functions

- **Relational constraints**
  - $x == y$, $x != y$, $x <= y$, $x < y$, $lb <= x <= ub$

- **Logical constraints**
  - $!c$, $c||d$, $c && d$
  - $c => d$, $c => d$ else $e$
  - $c$ and $d$ are relational or conditional constraints
Expressions and Constraints

• Reification
  ✓ Relational or logical constraints can be used in a value context, where they evaluate to 0 or 1

• Examples
  ✓ Arithmetic: max(0, abs(load[i] - cap))
  ✓ Relational: wid * hei * depth * density <= maxLoad
  ✓ Logical: end[i] <= start[j] || start[j] <= end[i]
  ✓ Reification: spill == (load[i] > cap)
Expressions and Constraints

- **Count expression**
  - count(dvar int[] x, int c)
  - Evaluates the number of variables in x with value c
  - e.g. Count the number of nurses allocated to ward 5
    - count(wardAllocation, 5) >= 3

- **Element expression**
  - (int[] a)[dvar int x] OR (dvar int[] a)[dvar int x]
  - Evaluates to the xth member of a
  - e.g. travel == 2 * distFromPittsburgh[holidayTown]
  - travel and holidayTown are variables
Expressions and Constraints

• All Different
  ✓ allDifferent(dvar int[] x)
  ✓ All variables in x must take different values
  ✓ e.g. The rank (visit priority) of each city is different
    ▪ allDifferent(rankOfVisit)

• Allowed / Forbidden assignments
  ✓ allowedAssignments({<a,b,c>} A, dvar int[3] x)
  ✓ The assignments to x must fit with a tuple of A
  ✓ forbiddenAssignments is the negation of this
Expressions and Constraints

- **Bin packing constraint**
  - pack(dvar int[m] ld, dvar int[n]x, int[n] sz, dvar int c)
  - ld[i] == sum(j) (x[j] == i) * sz[j]
    - c is the number of containers used

- **Inverse constraint**
  - inverse(dvar int[n] x, dvar int[n] y)
  - x[i] == j <=> y[j] == i --- link primary and dual models

- **Lexicographic ordering constraint**
  - lex(dvar int[n] x, dvar int[n] y) --- break symmetries
Outline

- A glimpse of Constraint Programming (CP)
- A glimpse of Integer Linear Programming (ILP)
- Comparison of CP and MP
- A brief introduction of IBM CP optimizer
What is CP Optimizer

- A Constraint Programming engine with an emphasis on modelling and automatic search
- Available as a toolkit in C++, Java, .NET
  - C++ is the native language and allows more possibilities, like writing incremental custom constraints, and fully controlling the search process
- Available as an engine inside ILOG OPL IDE
  - ILOG: name of the company (acquired by IBM)
  - OPL: optimization programming language
  - Higher level modelling and data manipulation
ILOG OPL IDE
A project =
- a set of model files (.mod)
- a set of data files (.dat)
- a set of settings files (.ops)
- other files (e.g. .xls files ⋮)
- a set of run configurations
ILOG OPL IDE

Model edition:
- data manipulation
- expression
- objective
- constraint
- script for pre- and post-processing
- use Help-> Dynamic Help for information on keywords
Data file:
- explicit data
- connection to:
  * databases
  * Excel worksheets
Overview of an CP Model using OPL

- **Top**
  - ✓ Data manipulation and pre-processing
    - declarative (expressions) and/or imperative (script)
  - ✓ Variable declarations

- **Middle**
  - ✓ Declarative model
    - objective (optional) and constraints

- **Bottom**
  - ✓ Post-processing of solutions
  - ✓ Declarative (expressions) and/or imperative (script)
An CP Model

CP Optimizer model

Slide 26
Data structures and data reading

Here, read input grid
An CP Model

Model:
- Variables
- Expressions
- Objective
- Constraints
An CP Model

Postprocessing

Here: display solution grid
Search in CP Optimizer

• Automatic search is emphasized
  ✓ Simpler, more maintainable, benefit from upgrades

• Search Phases
  ✓ What group of variables to assign first
  ✓ (optionally) define instantiation strategy

• Parameters
  ✓ Inference levels and search control parameters

• Problem still hard?
  ✓ Improve model
  ✓ Simplify or relax specification
  ✓ Decompose: CPLEX often useful here
Typical Use of CP Optimizer

1. Model problem in OPL
2. Use CP Optimizer’s automatic search
3. Use “search phases” to add some domain knowledge to the search process
4. Change parameters or search method
5. Review model (redundant constraints, decision variables etc.)
6. Decompose model
7. Custom goal or constraints (C++)

Satisfied?

YES  NO
Thank You!

xbwang@ucdavis.edu