Integer vs. constraint programming

Practical Problem Solving

- Model building: Language
- Model solving: Algorithms

### IP vs. CP: Language

<table>
<thead>
<tr>
<th></th>
<th>IP</th>
<th>CP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables</td>
<td>(mostly) 0-1</td>
<td>Finite domain</td>
</tr>
<tr>
<td>Constraints</td>
<td>Linear equations</td>
<td>Arithmetic constraints</td>
</tr>
<tr>
<td></td>
<td>and inequalities</td>
<td>Symbolic/global constraints</td>
</tr>
</tbody>
</table>

**Example**

- Variables: \(x_1, \ldots, x_n \in \{0, \ldots, m-1\}\)
- Constraint: Pairwise different values

**Example** (2)

- Integer programming: Only linear equations and inequalities
  \[ x_i \neq x_j \iff x_i < x_j \lor x_i > x_j \]
  \[ \iff x_i \leq x_j - 1 \lor x_i \geq x_j + 1 \]

- Eliminating disjunction
  \[ x_i - x_j + 1 \leq my_1, \quad x_j - x_i + 1 \leq my_2, \quad y_1 + y_2 = 1, \]
  \[ y_1, y_2 \in \{0,1\}, \quad 0 \leq x_i, x_j \leq m-1, \]

- New variables: \(z_{ik} = 1\) iff \(x_i = k, i = 1, \ldots, n, k = 0, \ldots, m-1\)
  \[ z_0 + \cdots + z_{m-1} = 1, \quad z_{1k} + \cdots + z_{nk} \leq 1, \]

- Constraint programming \(\leadsto\) **symbolic constraint**
  \[ \text{alldifferent}(x_1, \ldots, x_n) \]
Symbolic/global constraints

- \text{alldifferent}([x_1, \ldots, x_n])
- \text{cumulative}([s_1, \ldots, s_n], [d_1, \ldots, d_n], [r_1, \ldots, r_n], c, e).
  - \(n\) tasks: starting time \(s_i\), duration \(d_i\), resource demand \(r_i\)
  - resource capacity \(c\), completion time \(e\)

\begin{center}
\begin{tabular}{cccc}
1 & 2 & 3 & 4 \\
A & B & C & \\
\end{tabular}
\end{center}

cumulative([1,2,4], [4,4,2], [1,2,2], 3)

cumulative([1,2,2], [1,1,3], [2,1,2], 3)

cumulative([1,3,5], [2,1,1], [1,1,1], 1)

Diffn Constraint

Beldiceanu/Contejean'94

- Nonoverlapping of \(n\)-dimensional rectangles \([O_1, \ldots, O_n, L_1, \ldots, L_n]\), where \(O_i\) (resp. \(L_i\)) denotes the origin (resp. length) in dimension \(i\)
- \text{diffn}([[O_{11}, \ldots, O_{1n}, L_{11}, \ldots, L_{1n}], \ldots, [O_{m1}, \ldots, O_{mn}, L_{m1}, \ldots, L_{mn}]])

\begin{center}
\begin{tabular}{cccc}
1 & 2 & 3 & 4 \\
\end{tabular}
\end{center}
diffn([[1,2,2,2], [1,1,1,1], [4,2,3,3]])

- General form: \text{diffn}(Rectangles, Min\_Vol, Max\_Vol, End, Distances, Regions)

IP vs. CP: Algorithms

\begin{center}
\begin{tabular}{|c|c|c|}
\hline
 & IP & CP \\
\hline
\textit{Inference} & Linear programming & Domain filtering \\
& Cutting planes & Constraint propagation \\
\hline
\textit{Search} & Branch-and-relax & Branch-and-bound \\
& Branch-and-cut & \\
\hline
\textit{Bounds on the objective function} & Two-sided & One-sided \\
\hline
\end{tabular}
\end{center}
Local vs. global reasoning

Linear arithmetic constraints

\[ 3 x + y \leq 7, \]
\[ 3 y + x \leq 7, \]
\[ x + y = z, \]
\[ x, y \in \{0, ..., 3\} \]

- **CP** \( x, y \leq 2, z \leq 4 \)
- **LP** \( x, y \leq 2, z \leq 3.5 \)
- **IP** \( x, y \leq 2, z \leq 3 \)

Global reasoning in CP? \(\leadsto\) global constraints!

**Global reasoning in CP**

**Example**

- \( x_1, x_2, x_3 \in \{0, 1\} \)
- pairwise different values
- **Local** consistency. 3 disequalities: \( x_1 \neq x_2, x_1 \neq x_3, x_2 \neq x_3 \)
  \(\leadsto\) \( x_1, x_2, x_3 \in \{0, 1\} \), i.e., no domain reduction is possible
- **Global** constraint: \( \text{alldifferent}(x_1, x_2, x_3) \)
  \(\leadsto\) detects infeasibility (uses bipartite matching)

Global reasoning in CP: inside global constraints

**Summary**

<table>
<thead>
<tr>
<th>Language</th>
<th>ILP</th>
<th>CP(FD)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithms</strong></td>
<td>Global consistency (LP)</td>
<td>Local consistency</td>
</tr>
<tr>
<td></td>
<td>Cutting planes</td>
<td>Domain reduction</td>
</tr>
<tr>
<td></td>
<td>Branch-and-bound</td>
<td>User-defined enumeration</td>
</tr>
<tr>
<td></td>
<td>Branch-and-cut</td>
<td></td>
</tr>
</tbody>
</table>

- **Symbolic constraints** \(\leadsto\) more expressivity + more efficiency
- Unifying framework for CP and IP: **Branch-and-infer**
  (Bockmayr/Kasper 98), . . . , SCIP
Discrete Tomography

- Binary matrix with $m$ rows and $n$ columns
  - Horizontal projection numbers ($h_1, \ldots, h_m$)
  - Vertical projection numbers ($v_1, \ldots, v_n$)

- **Properties**
  - Horizontal convexity ($h$)
  - Vertical convexity ($v$)
  - Connectivity (polyomino) ($p$)

- **Complexity** (Woeginger’01)
  - polynomial: $(p,v,h)$
  - NP-complete: $(p,v), (p,h), (v,h), (v), (h), (p)$

**IP Model**

- **Variables** $x_{ij} = \begin{cases} 0 & \text{cell}(i,j) \text{ is labeled white} \\ 1 & \text{cell}(i,j) \text{ is labeled black} \end{cases}$

- **Constraints I:** Projections
  $$\sum_{j=1}^{n} x_{ij} = h_i, \quad \sum_{i=1}^{m} x_{ij} = v_j$$

- **Constraints II:** Convexity
  $$h_j x_{ik} + \sum_{k=1}^{n} x_{ij} \leq h_i, \quad v_j x_{kj} + \sum_{l=1}^{m} x_{lj} \leq v_j$$

**IP Model (contd)**

- **Constraints III:** Connectivity
  $$\sum_{k=j}^{j+h-1} x_{ik} - \sum_{k=j}^{j+h-1} x_{i+1,k} \leq h_j - 1$$

- Various linear arithmetic models possible, e.g. convexity
- Enormous differences in size and running time, e.g. 1 day vs. < 1 sec
- Large number of constraints ($\sim 3mn$ in the above model)
Finite Domain Model

• Variables
  – $x_i$ start of horizontal convex block in row $i$, for $1 \leq i \leq m$
  – $y_j$ start of vertical convex block in column $j$, for $1 \leq j \leq n$

\[
\begin{array}{cccccc}
 & 1 & 3 & 5 & 3 & 3 \\
\hline 
2 & 1 & 1 & 3 & 3 & 3 \\
1 & 3 & 2 & 3 & 2 & 3 \\
2 & 5 & 3 & 5 & 2 & 1 \\
3 & 2 & 2 & 1 & 3 & 1 \\
3 & 2 & 2 & 1 & 3 & 1 \\
\end{array}
\]

• Domain
  – $x_i \in [1, \ldots, n - h_i + 1]$, for $1 \leq i \leq m$
  – $y_j \in [1, \ldots, m - v_j + 1]$, for $1 \leq j \leq n$

Conditional Propagation

• Projection/Convexity modelled by FD variables

• Compatibility of $x_i$ and $y_j$
  \[x_i \leq j < x_i + h_i \iff y_j \leq i < y_j + v_j\]
  for $1 \leq i \leq m$ and $1 \leq j \leq n$

\[
\begin{array}{cccccccc}
\hline 
& & & & & & & & \\
& & & & & & & & \\
\hline 
& & & & & & & & \\
\hline 
& & & & & & & & \\
\hline 
& & & & & & & & \\
\hline 
& & & & & & & & \\
\hline 
& & & & & & & & \\
\hline 
& & & & & & & & \\
\hline 
& & & & & & & & \\
\hline 
& & & & & & & & \\
\hline 
& & & & & & & & \\
\hline 
& & & & & & & & \\
\hline
\end{array}
\]

• Conditional propagation
  \[
\text{if } x_i \leq j \text{ then (if } j < x_i + h_i \text{ then } (y_j \leq i, i < y_j + v_j))
\]

Finite Domain Model (contd)

• Connectivity

• Block $i$ must start before the end of block $i + 1$
  \[x_i \leq x_{i+1} + h_{i+1} - 1, \text{ for } 1 \leq i \leq m - 1\]

• Block $i + 1$ must start before the end of block $i$
  \[x_{i+1} \leq x_i + h_i - 1, \text{ for } 1 \leq i \leq m - 1\]
Propositional satisfiability

- \( x_1, \ldots, x_n \in \{0, 1\} \) 0-1 variables (or atomic formulae in propositional logic)
- A literal \( L \) is a 0-1 variable \( x \) or its negation \( \overline{x} = 1 - x \).
- A clause is a set of literals \( C = \{L_1, \ldots, L_k\} \) corresponding to the logical disjunction \( L_1 \lor \cdots \lor L_k \) or the clausal inequality \( L_1 + \cdots + L_k \geq 1 \).
- A clause set is a set of clauses \( S = \{C_1, \ldots, C_m\} \) corresponding to the logical conjunction \( C_1 \land \cdots \land C_m \).
- A clause set \( S \) is satisfiable if there exists an assignment \( I : \{x_1, \ldots, x_n\} \rightarrow \{0, 1\} \) making the logical formula true (equivalently, if the system of clausal inequalities has a 0-1 solution).

Examples

1. Clause set

   \[ S = \{ \{x_1, x_2\}, \{\overline{x_1}, \overline{x_2}\} \} \]

   Corresponding logical formula: \((x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2})\)
   Corresponding system of clausal inequalities: \(x_1 + x_2 \geq 1, -x_1 - x_2 \geq -1\)
   Satisfying assignments: \(I(x_1) = 1, I(x_2) = 0\) and \(I'(x_1) = 0, I'(x_2) = 1\).

2. The clause set

   \[ S = \{ \{x_1, x_2\}, \{\overline{x_1}, \overline{x_2}\}, \{x_1, \overline{x_2}\}, \{\overline{x_1}, x_2\} \} \]

   is unsatisfiable.
SAT problem

- **SAT problem**: Given a set of clauses \( S \), is \( S \) satisfiable?
- **Theorem (Cook’71)**: SAT is NP-complete.
- There exist highly efficient SAT solvers.
- Enormous progress has been made during the last 10-15 years, see e.g. [http://www.satlive.org/](http://www.satlive.org/) SAT competitions
- SAT is a third general approach for solving constraint satisfaction/optimization problems (in addition to IP and CP)

**Davis-Putnam procedure**

```plaintext
function Satisfiable(S) return boolean
  /* unit resolution */
  repeat
    for each clause \( \{L\} \) in \( S \) of length 1 do
      delete from \( S \) every clause containing \( L \)
      delete \( L \) in every clause of \( S \) containing \( \overline{L} \)
    od
    if \( S = \emptyset \) then true
    else if \( S \) contains the empty clause \( \{0\} \) then false
    fi
  until no further changes
  /* branching */
  choose a literal \( L \) occurring in \( S \)
  if Satisfiable(\( S \cup \{L\} \)) then true
  else if Satisfiable(\( S \cup \{\overline{L}\} \)) then true
  else false
  fi
end function
```

**Example**

Let \( S \) be the clause set

\[
\{ x_1, x_2, x_3, x_4, x_5 \}, \quad \{ x_1, x_2, x_3, \overline{x_4} \},
\{ x_1, x_2, x_3, x_5 \}, \quad \{ \overline{x_2}, x_3 \},
\{ \overline{x_1}, x_2 \}, \quad \{ x_1, \overline{x_2}, \overline{x_5} \},
\{ \overline{x_5} \}.
\]

Satisfying assignment: \( l(x_1) = 0, l(x_2) = l(x_3) = 1, l(x_5) = 0. \)