Integer vs. constraint programming

Practical Problem Solving

- Model building: Language
- Model solving: Algorithms

IP vs. CP: Language

	IP	CP
Variables	(mostly) 0-1	Finite domain
Constraints	Linear equations and inequalities	Arithmetic constraints **Symbolic/global constraints**

Example

- Variables: $x_1, \dots, x_n \in \{0, \dots, m-1\}$
- Constraint: Pairwise different values

Example (2)

- Integer programming: Only linear equations and inequalities

$$\begin{aligned} x_i \neq x_j \quad & \iff \quad x_i < x_j \quad \vee \quad x_i > x_j \\ & \iff \quad x_i \leq x_j - 1 \quad \vee \quad x_i \geq x_j + 1 \end{aligned}$$

- 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* $x_i = k$, $i = 1, \dots, n$, $k = 0, \dots, m-1$

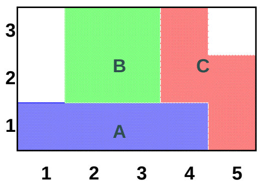
$$z_{i0} + \dots + z_{im-1} = 1, \quad z_{1k} + \dots + z_{nk} \leq 1,$$

- Constraint programming $\rightsquigarrow$ **symbolic constraint**

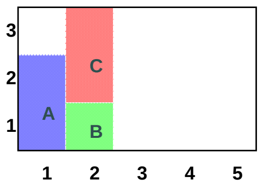
$$\texttt{alldifferent}(x_1, \dots, x_n)$$

Symbolic/global constraints

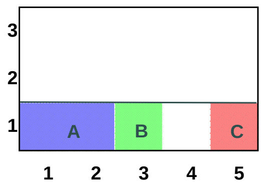
- `alldifferent`$([x_1, \dots, x_n])$
- `cumulative`$([s_1, \dots, s_n], [d_1, \dots, d_n], [r_1, \dots, r_n], c, e)$.
  - n tasks: starting time s_i, duration d_i, resource demand r_i
  - resource capacity c, completion time e




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




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

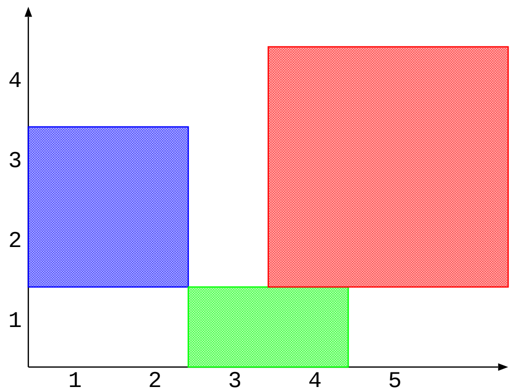



```
cumulative([1,3,5],
           [2,1,1],
           [1,1,1], 1)
```

Diffn Constraint

Beldiceanu/Contejean'94

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



`diffn([[1,2,2,2],[3,1,2,1],[4,2,3,3]])`

- General form: `diffn(Rectangles,Min_Vol,Max_Vol,End,Distances,Regions)`

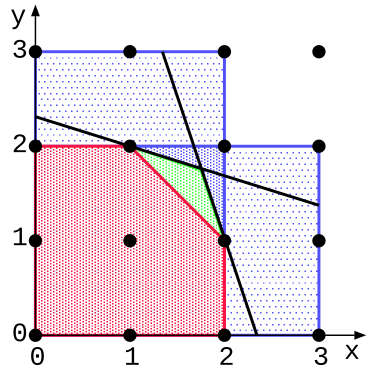
IP vs. CP: Algorithms

	IP	CP
Inference	Linear programming Cutting planes	Domain filtering Constraint propagation
Search	Branch-and-relax Branch-and-cut	Branch-and-bound
Bounds on the objective function	Two-sided	One-sided

Local vs. global reasoning

Linear arithmetic constraints

$$\begin{aligned} 3x+y &\leq 7, \\ 3y+x &\leq 7, \\ x+y &= z, \\ x,y &\in \{0,\dots,3\} \end{aligned}$$



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 ? ⇝ global constraints!

Global reasoning in CP

Example

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

Global reasoning in CP: inside global constraints

Summary

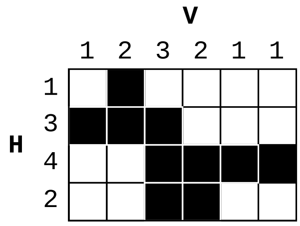
	ILP	CP(FD)
Language	Linear arithmetic —	Arithmetic constraints **Symbolic constraints**
Algorithms	Global consistency (LP) Cutting planes	Local consistency Domain reduction
	Branch-and-bound Branch-and-cut	User-defined enumeration

- Symbolic constraints ⇝ 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)



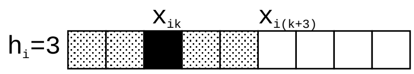
- *Complexity* (Woeginger'01)
  - polynomial: ( ), (p,v,h)
  - NP-complete: (p,v), (p,h), (v,h), (v), (h), (p)

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

$$\sum_{j=1}^{n} x_{ij} = h_i, \qquad \sum_{i=1}^{m} x_{ij} = v_j$$

- *Constraints II:* Convexity

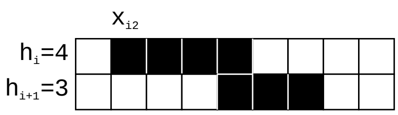


$$h_i x_{ik} + \sum_{l=k+h_i}^{n} x_{il} \leq h_i, \qquad v_j x_{kj} + \sum_{l=k+v_j}^{m} x_{lj} \leq v_j,$$

IP Model (contd)

- *Constraints III:* Connectivity

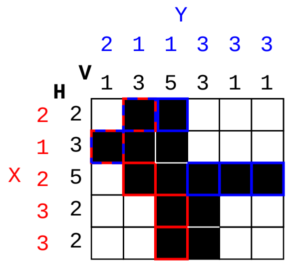
$$\sum_{k=j}^{j+h_i-1} x_{ik} - \sum_{k=j}^{j+h_i-1} x_{i+1,k} \leq h_i - 1$$



- 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 $1 \leq j \leq n$



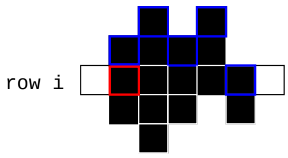
- *Domain*
  - $x_i \in [1, \ldots, n - h_i + 1]$, for $1 \leq i \leq m$
  - $y_j \in \left[1, \ldots, m - v_j + 1\right]$, for $1 \leq j \leq n$

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 $1 \leq j \leq n$

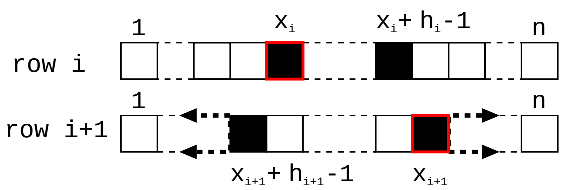


- *Conditional propagation*

$$\texttt{if } x_i \leq j \texttt{ then } (\texttt{if } j < x_i + h_i \texttt{ then } (y_j \leq i,\ i < y_j + v_j))$$

Finite Domain Model (contd)

- *Connectivity*



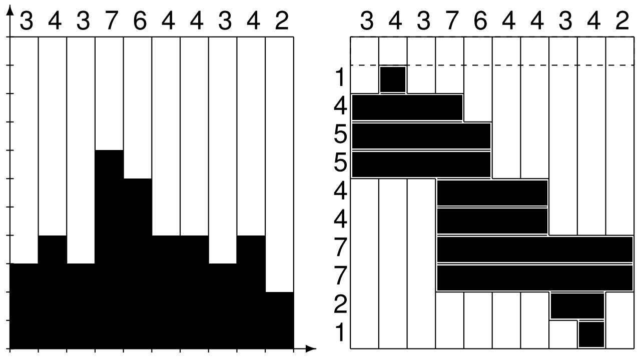
- Block i must start before the end of block $i+1$

$$x_i \leq x_{i+1} + h_{i+1} - 1, \text{ for } 1 \leq i \leq m - 1$$

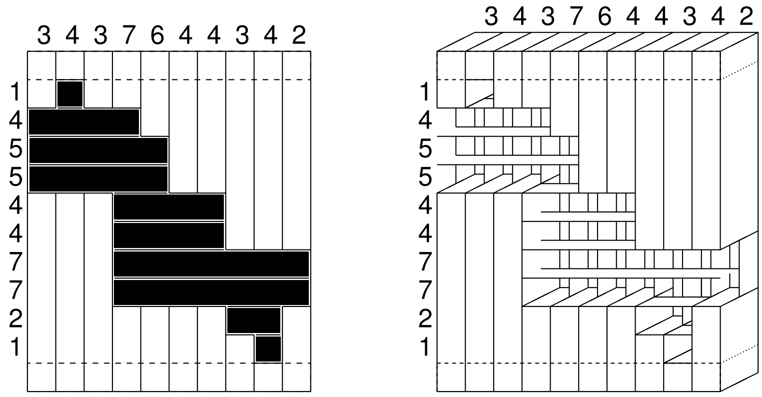
- Block $i+1$ must start before the end of block i

$$x_{i+1} \leq x_i + h_i - 1, \text{ for } 1 \leq i \leq m - 1$$

Cumulative



2d and 3d Diffn Model



Propositional satisfiability

- $x_1, \dots, x_n \in \{0,1\}$ 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 set of literals $C = \{L_1, \dots, L_k\}$ corresponding to the logical disjunction $L_1 \vee \cdots \vee L_k$ or the *clausal inequality* $L_1 + \cdots + L_k \geq 1$.
- A *clause set* is a set of clauses $S = \{C_1, \dots, C_m\}$ corresponding to the logical conjunction $C_1 \wedge \cdots \wedge C_m$.
- A clause set S is *satisfiable* if there exists an assignment $I : \{x_1, \dots, x_n\} \to \{0,1\}$ 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: $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$
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/` $\rightsquigarrow$ SAT competitions
- SAT is a third general approach for solving constraint satisfaction/optimization problems (in addition to IP and CP)

Davis-Putnam procedure

```
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 L̄ in every clause of S containing L̄
      od
      if S = ∅ then true
      else if S contains the empty clause C = ∅ then false
      fi
  until no further changes
  /* branching */
  choose a literal L occuring in S
  if Satisfiable(S ∪ {L}) then true
  else if Satisfiable(S ∪ {L̄}) then true
  else false
  fi
end function
```

Example

Let S be the clause set

$$
\begin{array}{llllllll}
\{ & x_1, & x_2, & x_3, & x_4, & x_5 & \}, \\
\{ & x_1, & x_2, & x_3, & \overline{x_4}, & & \}, \\
\{ & \overline{x_1}, & \overline{x_2}, & \overline{x_3}, & & x_5 & \}, \\
\{ & & \overline{x_2}, & x_3, & & & \}, \\
\{ & \overline{x_1}, & x_2, & & & & \}, \\
\{ & x_1, & \overline{x_2}, & & & \overline{x_5} & \}, \\
\{ & & & & & \overline{x_5} & \}.
\end{array}
$$

Satisfying assignment: $I(x_1) = 0, I(x_2) = I(x_3) = 1, I(x_5) = 0$.