Constraint Programming

Constraint Programming

- *Basic idea:* Programming with constraints, i.e. constraint solving embedded in a programming language
- *Constraints:* linear, non-linear, finite domain, Boolean, . . .
- *Programming:* logic, functional, object-oriented, imperative, concurrent, . . .
- *Systems:* Prolog III/IV, CHIP, ECLIPSE, ILOG, OCRE, NCL, . . .

Finite Domain Constraints

Constraint satisfaction problem (CSP)

- n variables $x_1, \dots, x_n$
- For each variable x_j a *finite domain* D_j of possible values, often $D_j \subset \mathbb{N}$.
- m constraints $C_1, \dots, C_m$, where $C_i \subseteq D_{i_1} \times \dots \times D_{i_{k_i}}$ is a relation between k_i variables $x_{i_1}, \dots, x_{i_{k_i}}$. Write also $C_{i_1,\dots,i_{k_i}}$.
- A *solution* is an assignment of a value D_j to x_j, for each $j = 1, \dots, n$, such that all relations C_i are satisfied.

Coloring Problem

- Decide whether a map can be colored by 3 colors such that neighboring regions get different colors.
- For each region a variable $\mathbf{x_j}$ with domain $D_j = \{\text{red, green, blue}\}$.
- For each pair of variables x_i, x_j corresponding to two neighboring regions, a constraint $\mathbf{x_i} \neq \mathbf{x_j}$.
- NP-complete problem.

Resolution by Backtracking

- Instantiate the variables in some order.
- As soon as all variables in a constraint are instantiated, determine its truth value.
- If the constraint is not satisfied, backtrack to the last variable whose domain contains unassigned values, otherwise continue instantiation.

Efficiency Problems

Mackworth 77

1. If the domain D_j of a variable x_j contains a value v that does not satisfy C_j, this will be the cause of repeated instantiation followed by immediate failure.

2. If we instantiate the variables in the order $x_1, x_2, \ldots, x_n$, and for $x_i = v$ there is no value $w \in D_j$, for $j > i$, such that $C_{ij}(v,w)$ is satisfied, then backtracking will try all values for x_j, fail and try all values for x_{j-1} (and for each value of x_{j-1} again all values for x_j), and so on until it tries all combinations of values for $x_{i+1}, \ldots, x_j$ before finally discovering that v is not a possible value for x_j.

   The identical failure process may be repeated for all other sets of values for $x_1, \ldots, x_{i-1}$ with $x_i = v$.

Local Consistency

- Consider CSP with unary and binary constraints only.
- *Constraint graph G*
  - For each variable x_i a node i.
  - For each pair of variables x_i, x_j occurring in the same binary constraint, two arcs (i,j) and (j,i).
- The node i is *consistent* if $C_i(v)$, for all $v \in D_i$.
- The arc (i,j) is *consistent*, if for all $v \in D_i$ with $C_i(v)$ there exists $w \in D_j$ with $C_j(w)$ such that $C_{ij}(v,w)$.
- The graph is *node consistent* resp. *arc consistent* if all its nodes (resp. arcs) are consistent.

Arc Consistency

Algorithm AC-3 (Mackworth 77):

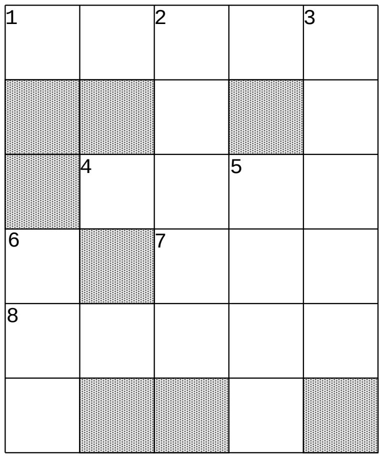
```
begin
   for i ← 1 until n do D_i ← {v ∈ D_i | C_i(v)};
   Q ← {(i,j) | (i,j) ∈ arcs(G), i ≠ j}
   while Q not empty do
      begin
         select and delete an arc (i,j) from Q;
         if REVISE(i,j) then
               Q ← Q ∪ {(k,i) | (k,i) ∈ arcs(G), k ≠ i, k ≠ j}
      end
end
```

Arc Consistency (2)

```
procedure REVISE(i,j):
begin
   DELETE ← false
   for each v ∈ D_i do
      if there is no w ∈ D_j such that C_ij(v,w) then
         begin
            delete v from D_i;
            DELETE ← true
         end;
   return DELETE
end
```

Crossword Puzzle

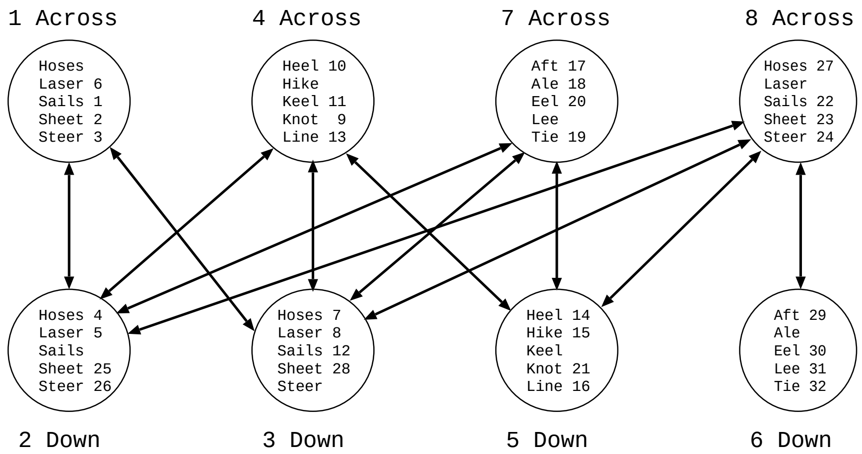
Dechter 92



Word List

Aft	Laser
Ale	Lee
Eel	Line
Heel	Sails
Hike	Sheet
Hoses	Steer
Keel	Tie
Knot	

Solution



Lookahead

Apply local consistency dynamically during search

- *Forward Checking:* After assigning to x the value v, eliminate for all uninstantiated variables y the values from D_y that are incompatible with v.
- *Partial Lookahead:* Establish arc consistency for all (y, y'), where y, y' have not been instantiated yet and y will be instantiated before y'.
- *Full Lookahead:* Establish arc consistency for all uninstantiated variables.

n-Queens Problem

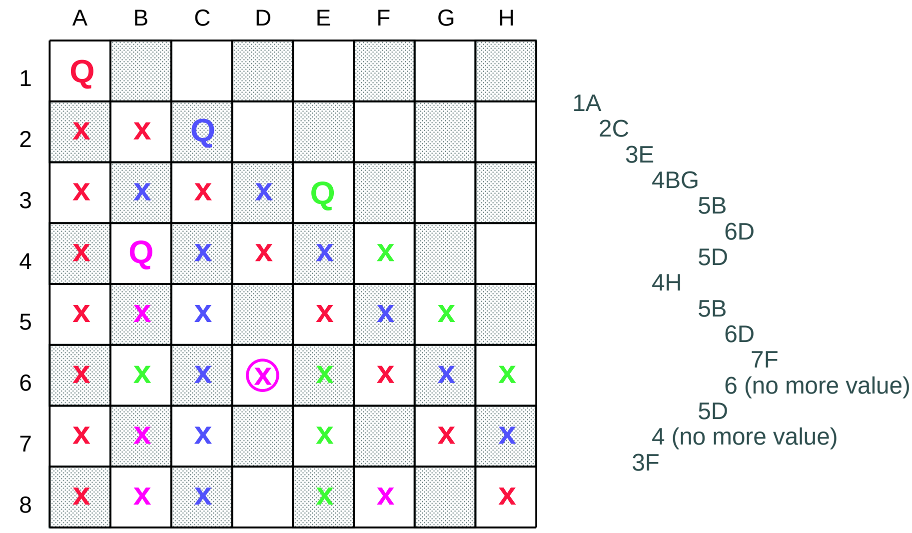
Place n queens in an $n \times n$ chessboard such that no two queens threaten each other.

- *Variables* $x_i, i = 1, \ldots, n$ with domain $D_i = \{1, \ldots, n\}$ indicating the column of the queen in line i.
- *Constraints*
  - $x_i \neq x_j$, for $1 \leq i < j \leq n$ (vertical)

- $x_i \neq x_j + (j-i)$, for $1 \leq i < j \leq n$ (diagonal 1)
- $x_i \neq x_j - (j-i)$, for $1 \leq i < j \leq n$ (diagonal 2)

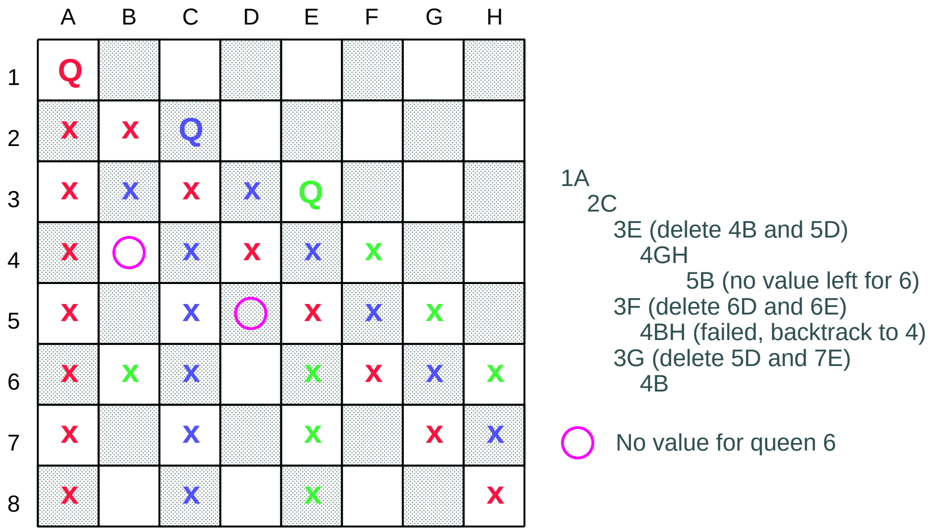
Forward Checking (2)

Forward Checking



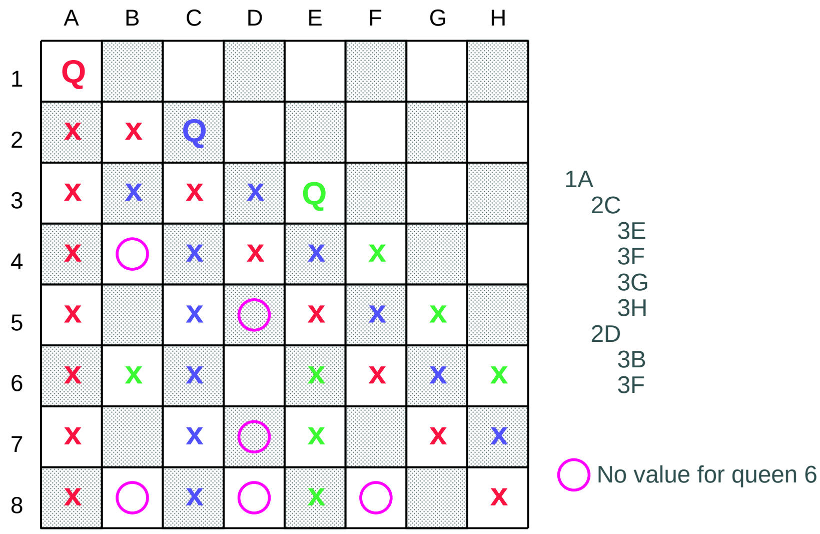
Partial Lookahead (3)

Partial Lookahead



Full Lookahead (4)

Full Lookahead



Typical structure of a constraint program

- Declare the variables and their domains
- State the constraints
- Enumeration (`labeling`)

The constraint solver achieves only local consistency.

In order to get global consistency, the domains have to be enumerated.

Labeling

- Assigning to the variables their possible values and constructing the corresponding search tree.
- *Important questions*
  1. In which order should the variables be instantiated (variable selection) ?
  2. In which order should the values be assigned to a selected variable (value selection) ?
- Static vs. dynamic orderings
- *Heuristics*

Dynamic variable/value orderings

- Variable orderings
  - Choose the variable with the smallest domain *“first fail”*
  - Choose the variable with the smallest domain that occurs in most of the constraints *“most constrained”*
  - Choose the variable which has the smallest/largest lower/upper bound on its domain.
- Value orderings
  - Try first the minimal value in the current domain.

- Try first the maximal value in the current domain.
- Try first some value in the middle of the current domain.

Constraint programming systems

System	Avail.	Constraints	Language	Web site
B-prolog	comm.	FinDom	Prolog	www.probp.com
CHIP	comm.	FinDom, Boolean, Linear $\mathbb{Q}$ Hybrid	Prolog, C, C++	www.cosytec.com
Choco	free	FinDom	Claire	choco-constraints.net
Eclipse	free non-profit	FinDom, Hybrid	Prolog	www.icparc.ic.ac.uk/eclipse/
GNU Prolog	free	FinDom	Prolog	gnu-prolog.inria.fr
IF/Prolog	comm.	FinDom Boolean, Linear $\mathbb{R}$	Prolog	www.ifcomputer.co.jp
ILOG	comm.	FinDom, Hybrid	C++, Java	www.ilog.com
NCL	comm.	FinDom		www.enginest.com
Mozart	free	FinDom	Oz	www.mozart-oz.org
Prolog IV	comm.	FinDom, nonlinear intervals	Prolog	www.prologia.fr
Sicstus	comm.	FinDom, Boolean, linear $\mathbb{R}/\mathbb{Q}$	Prolog	www.sics.se/sicstus/

Integer vs. constraint programming

Practical Problem Solving

- Model building : Language
- Model solving : Algorithms

IP vs. CP : Language

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

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

$$\begin{array}{lcllllll} x_i \neq x_j & \Longleftrightarrow & x_i & < & x_j & \vee & x_i & > & x_j \\ & \Longleftrightarrow & x_i & \leq & x_j - 1 & \vee & x_i & \geq & x_j + 1 \end{array}$$

- Eliminating disjunction

$$x_i - x_j + 1 \leq m y_i, \quad x_j - x_i + 1 \leq m y_j, \quad y_i + y_j = 1,$$
$$y_i, y_j \in \{0,1\}, \quad 0 \leq x_i, x_j \leq m-1,$$

- New variables: $z_{ik} = 1$ *iff* $x_i = k$, $i = 1, \ldots, n$, $k = 0, \ldots, m-1$

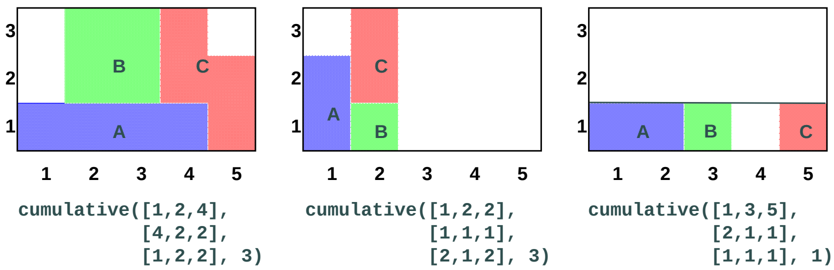
$$z_{i0} + \cdots + z_{im-1} = 1, \quad z_{1k} + \cdots + z_{nk} \leq 1,$$

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

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

Symbolic/global constraints

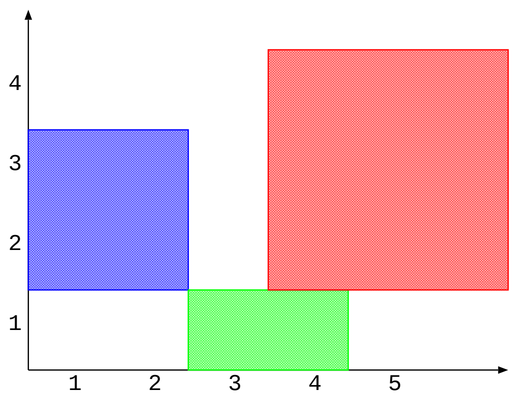
- `alldifferent`$([x_1, \ldots, x_n])$
- `cumulative`$([s_1, \ldots, s_n], [d_1, \ldots, d_n], [r_1, \ldots, r_n], c, e)$.
  - n tasks: starting time s_i, duration d_i, resource demand r_i
  - resource capacity c, completion time e.



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
- `diffn`$([[O_{11}, \ldots, O_{1n}, L_{11}, \ldots, L_{1n}], \ldots, [O_{m1}, \ldots, O_{mn}, L_{m1}, \ldots, L_{mn}]])$



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

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

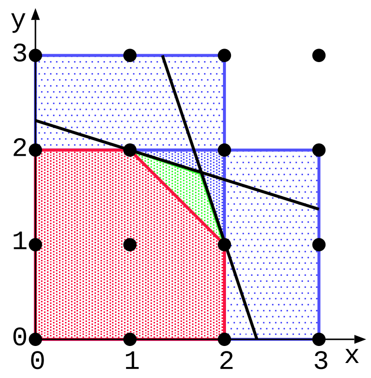
IP vs. CP : Algorithms

	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

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



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 ? $\rightsquigarrow$ global constraints !

Global reasoning in CP

Example

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

Global reasoning in CP : inside global constraints

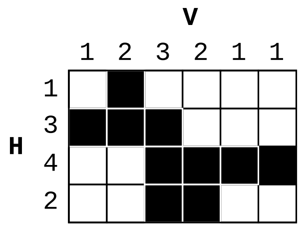
Summary

	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 $\rightsquigarrow$ more expressivity + more efficiency
- Unifying framework for CP and IP: *Branch-and-infer* (Bockmayr/Kasper 98)

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)



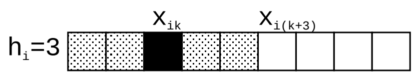
- *Complexity* (Woeginger'01)
  - polynomial: ( ), (p,v,h)
  - NP-complete: (p,v), (p,h), (v,h), (v), (h), (p)

IP Model

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

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

- *Constraints II:* Convexity

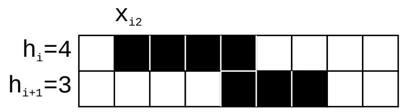


$$h_i x_{ik} + \sum_{l=k+h_i}^{n} x_{il} \leq h_i, \qquad v_j x_{kj} + \sum_{l=k+v_j}^{m} x_{lj} \leq v_j,$$

IP Model (contd)

- *Constraints III:* Connectivity

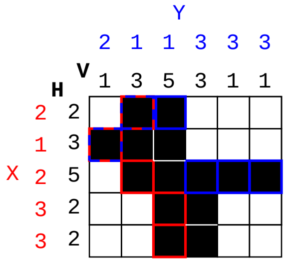
$$\sum_{k=j}^{j+h_i-1} x_{ik} - \sum_{k=j}^{j+h_i-1} x_{(i+1)k} \leq h_i - 1$$



- 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 $1 \leq j \leq n$



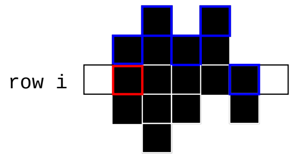
- *Domain*
  - $x_i \in [1, \ldots, n - h_i + 1]$, for $1 \leq i \leq m$
  - $y_j \in [1, \ldots, m - v_j + 1]$, for $1 \leq j \leq n$

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 $1 \leq j \leq n$

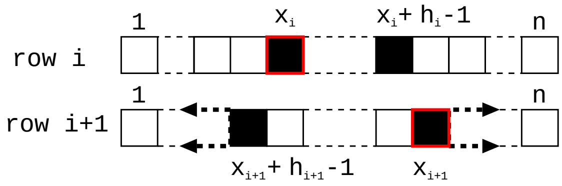


- *Conditional propagation*

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

Finite Domain Model (contd)

- *Connectivity*



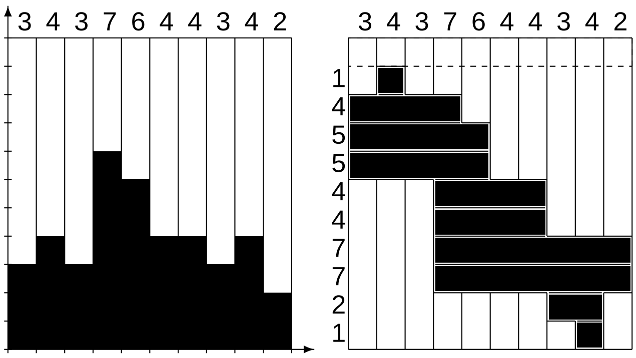
- Block i must start before the end of block $i+1$

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

- 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