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, concurrent, imperative/declarative, ...
- Mathematical programming vs. computer programming
- **Systems:** Prolog III/IV, CHIP, ECLIPSE, ILOG, CHOCO, Gecode, JaCoP, MiniZinc, ...

**Recommended reading:** Lustig/Puget'01

Finite Domain Constraints

**Constraint satisfaction problem (CSP)**

- $n$ variables $x_1, \ldots, x_n$
- For each variable $x_j$ a *finite domain* $D_j$ of possible values, often $D_j \subseteq \mathbb{N}$.
- $m$ constraints $C_1, \ldots, C_m$, where $C_j \subseteq D_{i_1} \times \ldots \times D_{i_k}$ is a relation between $k_i$ variables $x_{i_1}, \ldots, x_{i_k}$. Write also $C_{i_1,\ldots,i_k}$.
- A *solution* is an assignment of a value $v_j \in D_j$ to $x_j$, for each $j = 1, \ldots, n$, 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 $x_j$ with domain $D_j = \{ \text{red}, \text{green}, \text{blue} \}$.
- For each pair of variables $x_i, x_j$ corresponding to two neighboring regions, a constraint $x_i \neq x_j$.
- 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 \( i \leftarrow 1 \) until \( n \) do \( D_i \leftarrow \{v \in D_i \mid C_i(v)\} \);
\( Q \leftarrow \{(i, j) \mid (i, j) \in \text{arcs}(G), i \neq j\} \)
while \( Q \) not empty do
begin
select and delete an arc \((i, j)\) from \( Q \);
if \( \text{REVISE}(i, j) \) then
\( Q \leftarrow Q \cup \{(k, i) \mid (k, i) \in \text{arcs}(G), k \neq i, k \neq j\} \)
end
end

procedure \( \text{REVISE}(i, j) \):
begin
\( \text{DELETE} \leftarrow \text{false} \)
for each \( v \in D_i \) do
if there is no \( w \in D_j \) such that \( C_{ij}(v, w) \) then
begin
delete \( v \) from \( D_i \);
\( \text{DELETE} \leftarrow \text{true} \)
end;
return \( \text{DELETE} \)
end

**Complexity:** \( O(d^3e) \), with \( d \) an upper bound on the domain size and \( e \) the number of binary constraints, can be improved to \( O(d^2e) \).
Crossword Puzzle

Dechter 92

Word List

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

Solution

1 Across
- Hoses 1
- Laser 2
- Sails 3
- Steer 4

4 Across
- Hoses 7
- Laser 8
- Sails 9
- Steer 10

7 Across
- Aft 17
- Ale 18
- Eel 19
- Knot 20
- Line 21

8 Across
- Hoses 27
- Laser 28
- Sails 29
- Steer 30

2 Down
- Hoses 4
- Laser 5
- Sails 6
- Sheet 11

3 Down
- Hoses 11
- Laser 12
- Sails 22
- Sheet 23

5 Down
- Heel 10
- Hike 11
- Keel 12
- Knot 21
- Line 13

6 Down
- Aft 29
- Ale 30
- Eel 31
- Lee 32
- Tie 32

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.