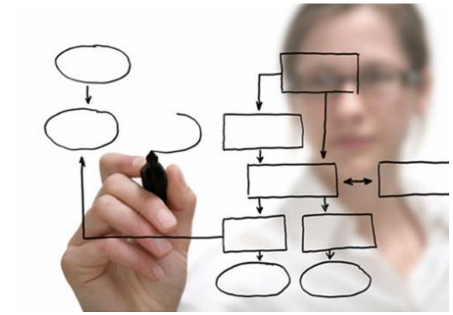
CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT

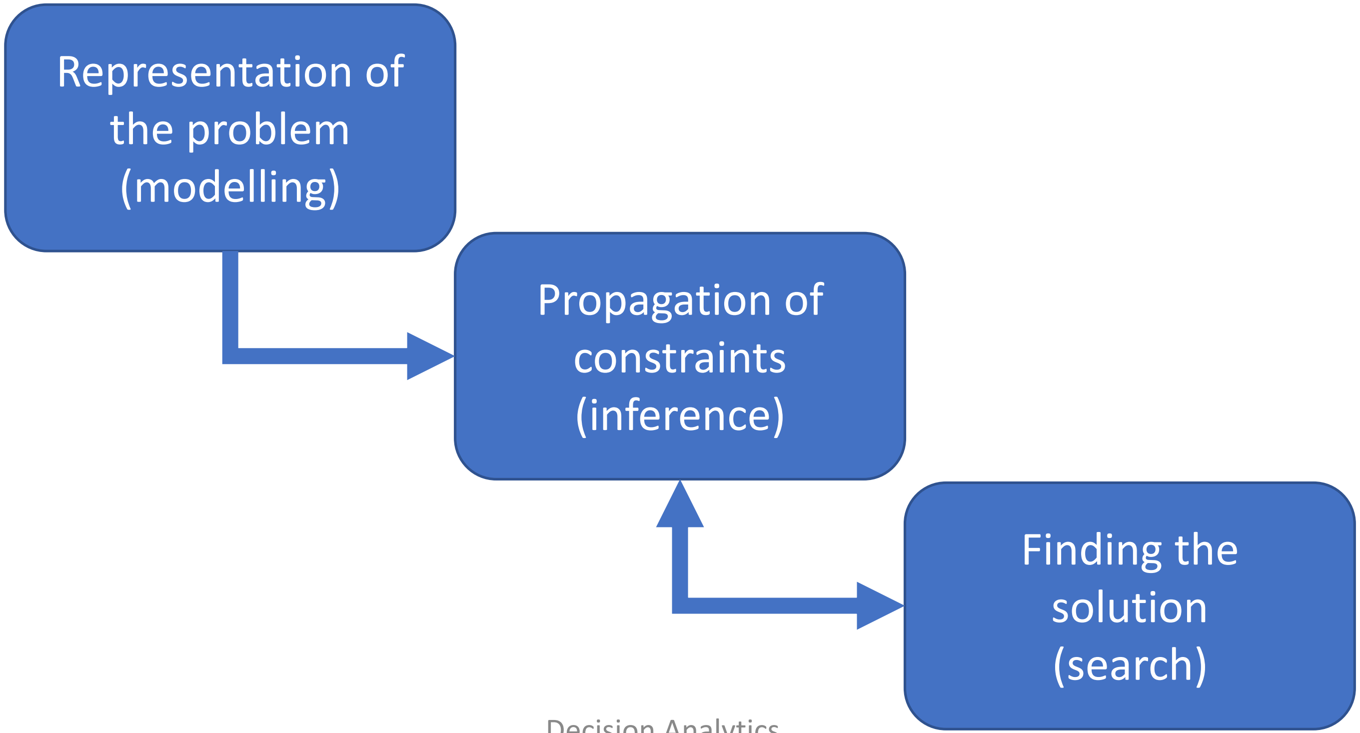


Decision Analytics

Lecture 14: Constraint Propagation beyond Arc Consistency

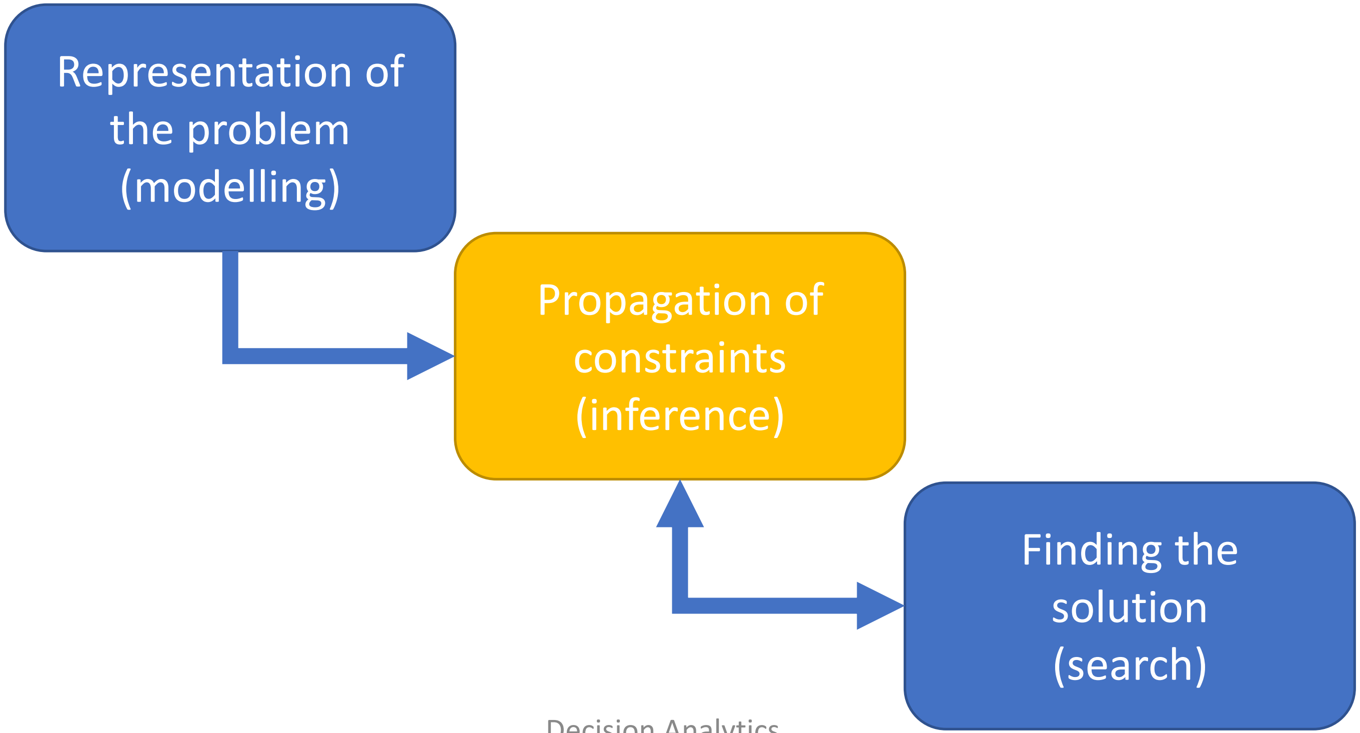
Constraint Programming

- Constraint Programming (CP) is a paradigm for solving combinatorial constraint satisfaction and constrained optimisation problems using a combination of modelling, propagation, and search



Constraint Programming

- Constraint Programming (CP) is a paradigm for solving combinatorial constraint satisfaction and constrained optimisation problems using a combination of modelling, propagation, and search
- This lecture is about **constraint propagation**



Constraint network

- A constraint network (X, D, C) is defined by
  - A sequence of n **variables**
    $$X = (x_1, \dots, x_n)$$
  - A **domain** for X defined by the domains of the individual variables
    $$D = D(x_1) \times \cdots \times D(x_n)$$
  - A set of **constraints**
    $$C = \{c_1, \dots, c_e\}$$
- A network is **normalised** if two different constraints do not contain exactly the same variables, i.e. $c_i \neq c_j \Rightarrow X(c_i) \neq X(c_j)$

AC3 Algorithm

function `Revise3`(**in** x_i: variable; c: constraint)**: Boolean** ;

begin

CHANGE $\leftarrow$ **false**;

foreach $v_i \in D(x_i)$ **do**

if $\nexists \tau \in c \cap \pi_{X(c)}(D)$ *with* $\tau[x_i] = v_i$ **then**

remove v_i from $D(x_i)$;

CHANGE $\leftarrow$ **true**;

return CHANGE ;

end

function `AC3/GAC3`(**in** X: set)**: Boolean** ;

begin

/* initalisation */;

$Q \leftarrow \{(x_i, c) \mid c \in C, x_i \in X(c)\}$;

/* propagation */;

while $Q \neq \emptyset$ **do**

select and remove (x_i, c) from Q;

if `Revise`(x_i, c) **then**

if $D(x_i) = \emptyset$ **then** return **false** ;

else $Q \leftarrow Q \cup \{(x_j, c') \mid c' \in C \land c' \neq c \land x_i, x_j \in X(c') \land j \neq i\}$;

return **true** ;

end

Arc consistency

- A network is $N = (X, D, C)$ is **arc consistent** if all for all variable domains and all constraints

$$D(x_i) \subset \pi_{\{\, x_i\}}(c \cap \pi_{X(c)}(D))$$

- Arc consistency considers each constraint in isolation and makes the domains of the scheme of that constraint locally compatible with the constraint

- Question: can we prune more values if we consider more than one constraint at once?

Path consistency

- Binary normalised networks can be seen as graphs, with the nodes being the variables $X = (x_1, \dots, x_n)$ and the edges being the constraints $C = \{c_{i_1 j_1}, \dots, c_{i_e j_e}\}$

- If a pair of variables x_i and x_j is connected via a path through this graph
$$x_i = x_{k_1} \to x_{k_2} \to x_{k_3} \to \cdots \to x_{k_{p-1}} \to x_{k_p} = x_j$$

- We can look at the constraints along this path
$$c_{k_1 k_2}, c_{k_2 k_3}, \dots, c_{k_{p-1} k_p}$$

- and make sure that the two nodes are consistent with the constraints along the path

- This is more than checking arc consistency, which would only look at the constraint c_{ij}, if it exists, and not consider long range dependencies

Path consistency

- A pair of values $(v_i, \mathrm{v_j}) \in D(x_i) \times D(x_j)$ is **path consistent**, if for every path $Y = (x_{k_1}, \dots, x_{k_p})$ there exists a tuple

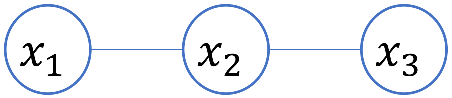
$$\tau = \left(v_i, v_{k_2}, \dots, v_{k_{p-1}}, v_j\right) \in \pi_Y(D)$$

- so that all constraints are satisfied, i.e.

$$\forall 1 \leq q < p: \left(v_{k_q}, v_{k_{q+1}}\right) \in c_{k_q k_{q+1}}$$

- A network N is **path consistent** if every locally consistent pair of values is path consistent

Path consistency



- Example: $N = (X, D, C)$ with

$$X = (x_1, x_2, x_3)$$
$$D(x_1) = D(x_2) = D(x_3) = \{1,2\}$$
$$C = \{x_1 \neq x_2, x_2 \neq x_3\}$$

- N is not path consistent, because for example the pair of values $\big((x_1, 1), (x_3, 2)\big)$ can neither be extended via $(x_2, 1)$ nor $(x_2, 2)$
- We can therefore add a constraint $c_{13} = \{(1,1), (2,2), (2,1)\}$

- The resulting network is still not path consistent, because also $\big((x_1, 2), (x_3, 1)\big)$ cannot be extended via $(x_2, 1)$ nor $(x_2, 2)$
- If can therefore also remove this pair from $c_{13} = \{(1,1), (2,2)\}$
- The resulting network with $C' = C \cup \{x_1 = x_3\}$ is now path consistent

Path consistency

- In contrast to arc consistency, path consistency does not reduce the domains
- Instead pairs of values are excluded, which adds additional (or modifies existing) binary constraints between these pairs
- Typically these constraints have to be maintained explicitly then, potentially adding to the space requirements of the algorithm

Pairwise consistency

- Another avenue to explore is to see if constraints are incompatible, and to add additional constraints to make this explicit

- Two constraints c_1 and c_2 are **pairwise consistent**, if they agree on their overlap, i.e.

$$\pi_{X(c_1)\cap X(x_2)}(c_1) = \pi_{X(c_1)\cap X(x_2)}(c_2)$$

- A network is considered **pairwise consistent**, if every pair of constraints is pairwise consistent

Pairwise consistency

- Example: $N = (X, D, C)$ with

$$X = (x_1, x_2, x_3, x_4)$$
$$D(x_1) = D(x_2) = D(x_3) = D(x_4) = \{1,2\}$$
$$C = \{c_1, c_2\}$$
$$c_1(x_1, x_2, x_3) = \{(1,2,1), (2,1,1), (2,2,2)\}$$
$$c_2(x_2, x_3, x_4) = \{(1,1,1), (2,2,2)\}$$

- This network is not pairwise consistent, because

$$\pi_{\{x_2, x_3\}}(c_1) = \{(2,1), (1,1), (2,2)\}$$
$$\pi_{\{x_2, x_3\}}(c_2) = \{(1,1), (2,2)\}$$

- Therefore the tuple $(1,2,1)$ in c_1 is incompatible and can be removed from the constraint

Directional arc consistency

- Path consistency and pairwise consistency both relied on modifying the constraints not the domains
- We will now return to arc consistency and look at some weaker versions that potentially can be computed more efficiently
- To do that we will revisit the AC3 algorithm and see where it can be modified/improved

Directional arc consistency

- The AC3 algorithm maintains a queue of variables and constraints to check
- Every time a domain is reduced, all variables and constraints linked with this domain need to be re-evaluated
- This can be avoided, if we do not aim for full arc consistency

- A binary network $N = (X, D, C)$ with an ordering of the variables $x_{k_1} <_o \ldots <_o x_{k_n}$ is considered **directional arc consistent**, if for all constraints $c(x_i, x_j)$ where $x_i < x_j$ the first variable x_i is arc consistent on these constraints c

- This is weaker than arc consistency, because consistencies only propagate in one direction from each variable

Directional arc consistency

- The fact that constraints only propagate in one direction can be exploited in the AC3 algorithm

- Directional arc consistency avoids the need for maintaining the processing queue, as long as all revisions are executed from the last variable to the first

procedure $DAC(N, o)$;

for $j \leftarrow n$ ***downto*** 2 **do**

    foreach $c_{ik_j} \in C_N \mid x_i <_o x_{k_j}$ **do**

        if ***not*** $\mathtt{Revise}(x_i, c_{ik_j})$ **then** return false

Thank you for your attention!