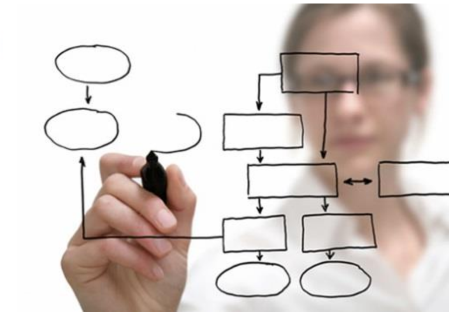
CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT



Decision Analytics

Lecture 23: Wrap-up

Wrap-up: module overview

- Introduction & Tools
- Modelling
- Constraint programming
- Linear programming

Tools

- NumPy
  - Matrix operations
  - Linear algebra
- OR Tools
  - Constraint programming
  - Linear programming

Mathematical Optimisation

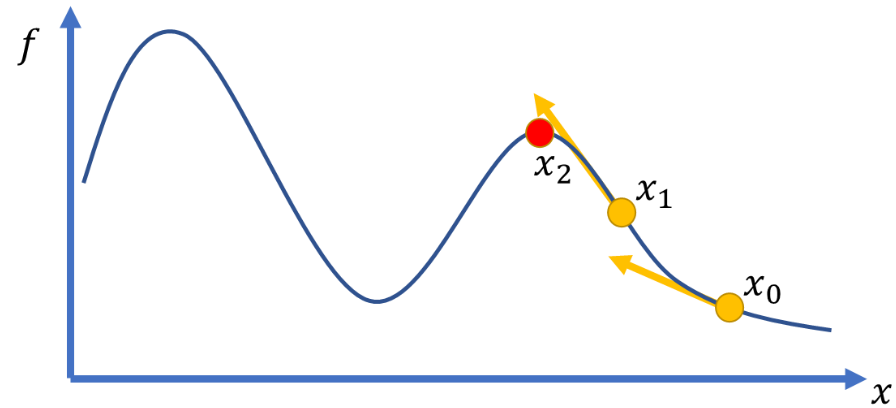
Given a domain D and an objective function

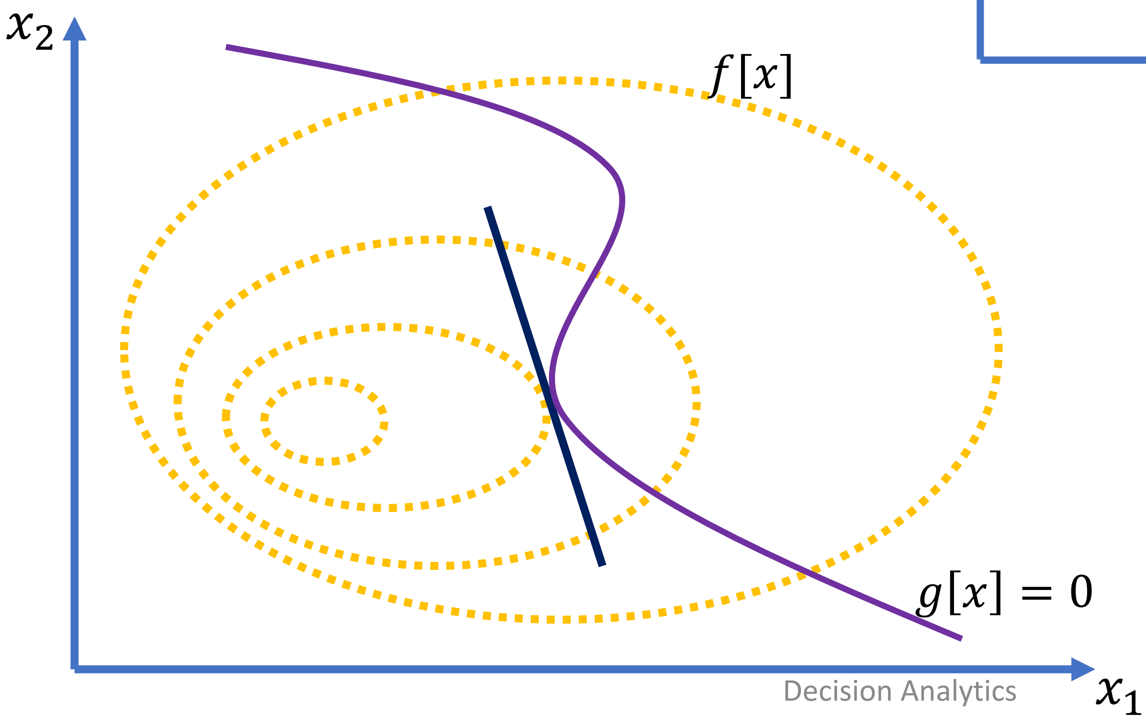
$$f: D \to \mathbb{R}$$

find $\hat{x} \in D$ that minimises f, i.e. $\forall x \in D: f(\hat{x}) \leq f(x)$

(for maximisation problems replace f with $-f$)

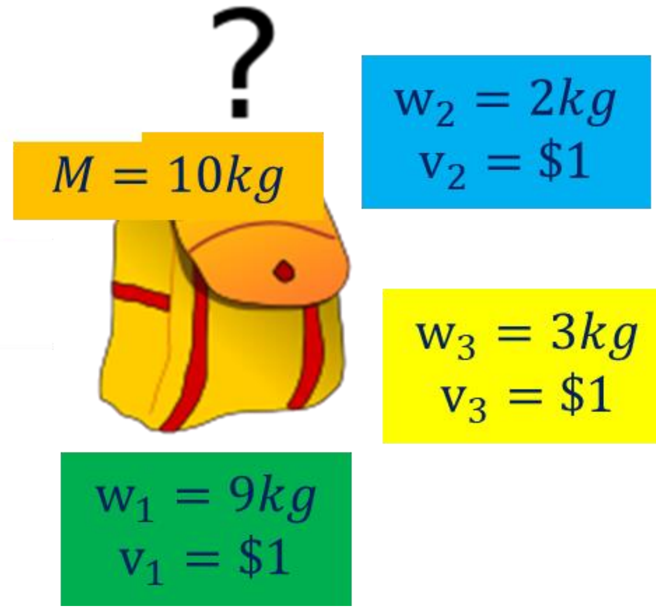
Gradient-based search & Lagrange multipliers



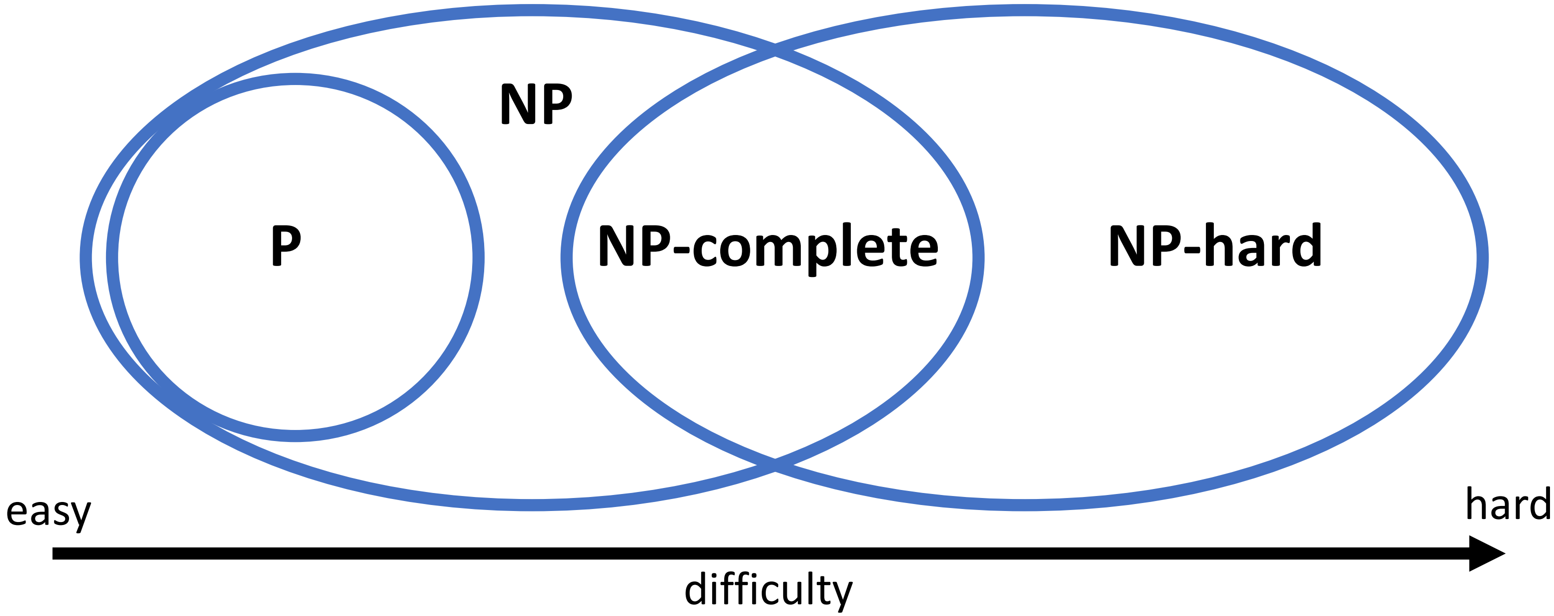


Constraint Satisfaction Problems

$$X = < x_1, \dots, x_n >$$
$$D = < D_1, \dots, D_n >$$
$$x_i \in D_i$$
$$C = < C_1, \dots, C_t >$$
$$C_i = < R_{S_i}, S_i >$$
$$S_i \subset X, R_{S_i} \subset D_{s_{i_1}} \times \cdots \times D_{s_{i_{|s_i|}}}$$



Complexity classes



Boolean algebra and binary domains

$$(x_1 \lor \neg x_2 \lor x_3) \land (\neg y_1 \lor y_2) \land (z_1 \lor z_2 \lor z_3 \lor z_4)$$

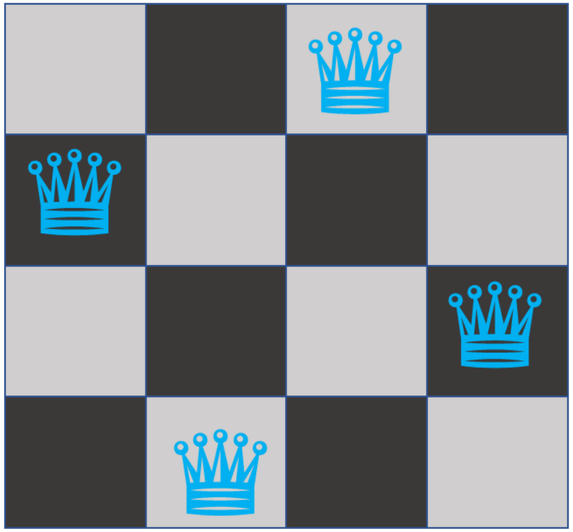
```
model.AddBoolOr([x1,x2.Not(),x3])
model.AddBoolOr([y1.Not(),y2])
model.AddBoolOr([z1,z2,z3,z4])
```

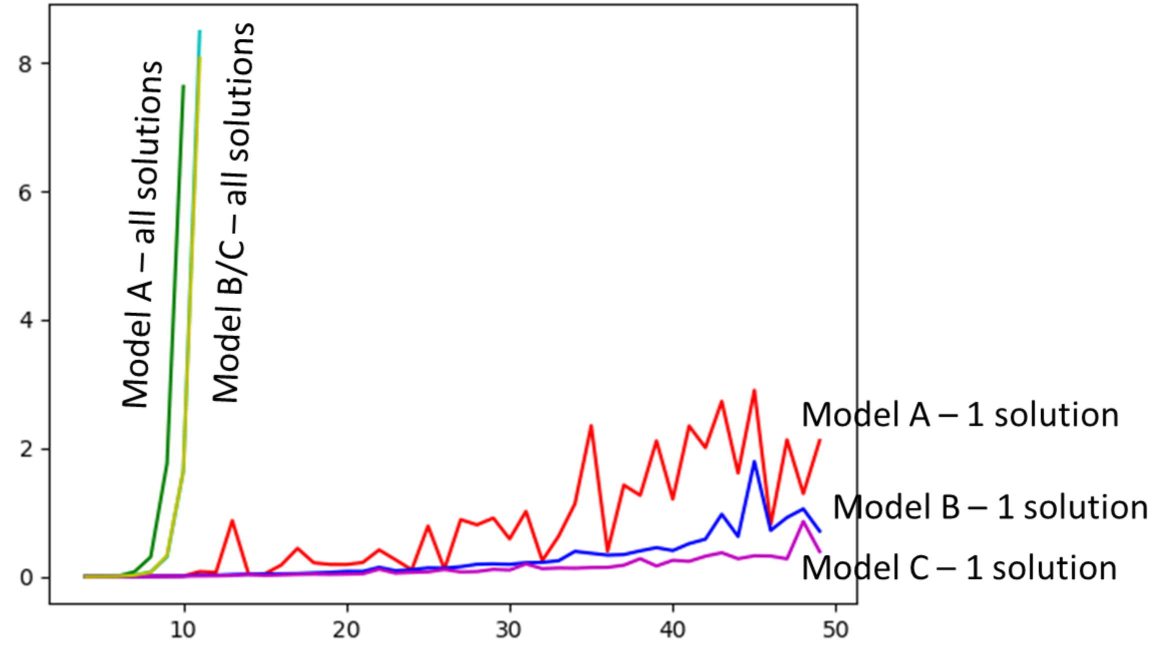
x	y	$x \land y$	$x \lor y$	$\neg x$
False	False	False	False	True
True	False	False	True	False
False	True	False	True	
True	True	True	True	

Planning problems and SAT

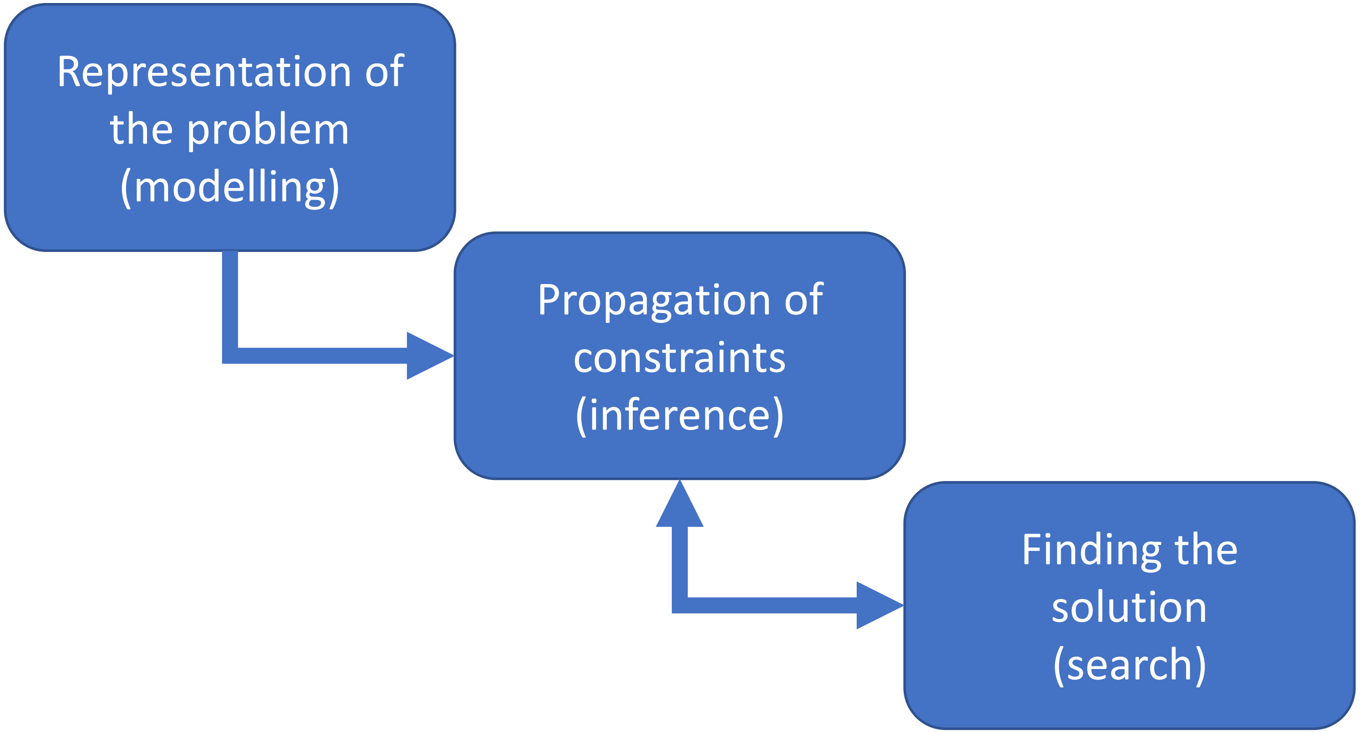
First-order logic and SAT

Modelling beyond SAT



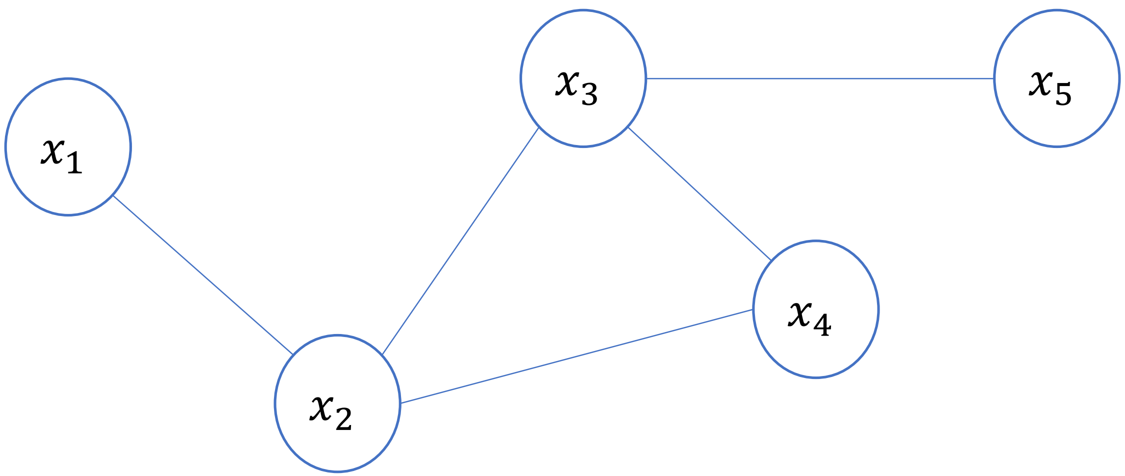


Constraint programming



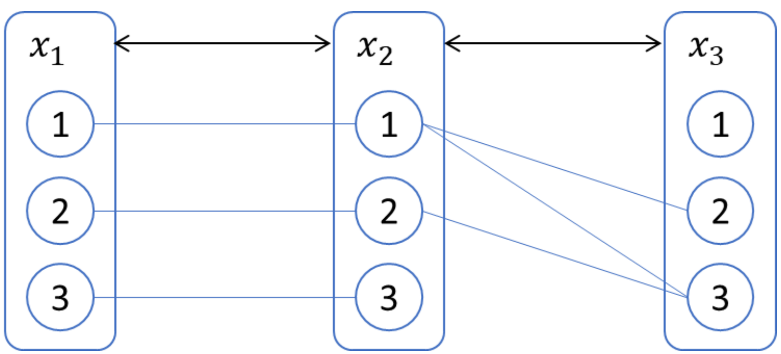
Constraint networks

$$X = (x_1, x_2, x_3, x_4, x_5)$$
$$D(x_i) = \{1,2,3,4,5\}$$
$$C = \{x_1 < x_2, x_2 = x_3, x_3 \geq x_4, x_3 \geq x_5, x_2 \leq x_4\}$$

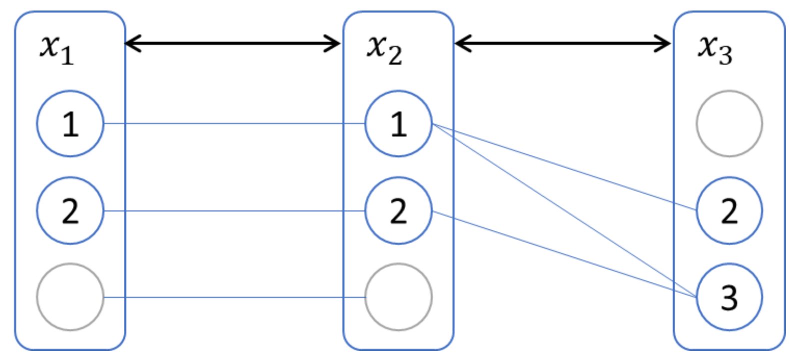


Arc consistency

$$X = (x_1, x_2, x_3)$$
$$D(x_1) = D(x_2) = D(x_3) = \{1,2,3\}$$
$$C = \{x_1 = x_2, x_2 < x_3\}$$



$$D'(x_1) = \{1,2\}$$
$$D'(x_2) = \{1,2\}$$
$$D'(x_3) = \{2,3\}$$



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 $\mathtt{Revise}(x_i, c)$ **then**

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

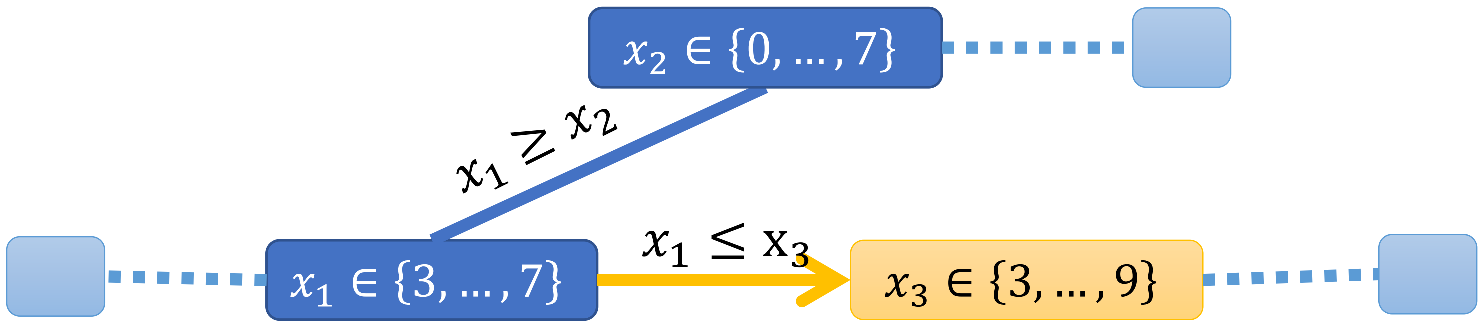
else $Q \leftarrow Q \cup \{(x_j, c') \mid c' \in C \wedge c' \neq c \wedge x_i, x_j \in X(c') \wedge j \neq i\}$;

return **true** ;

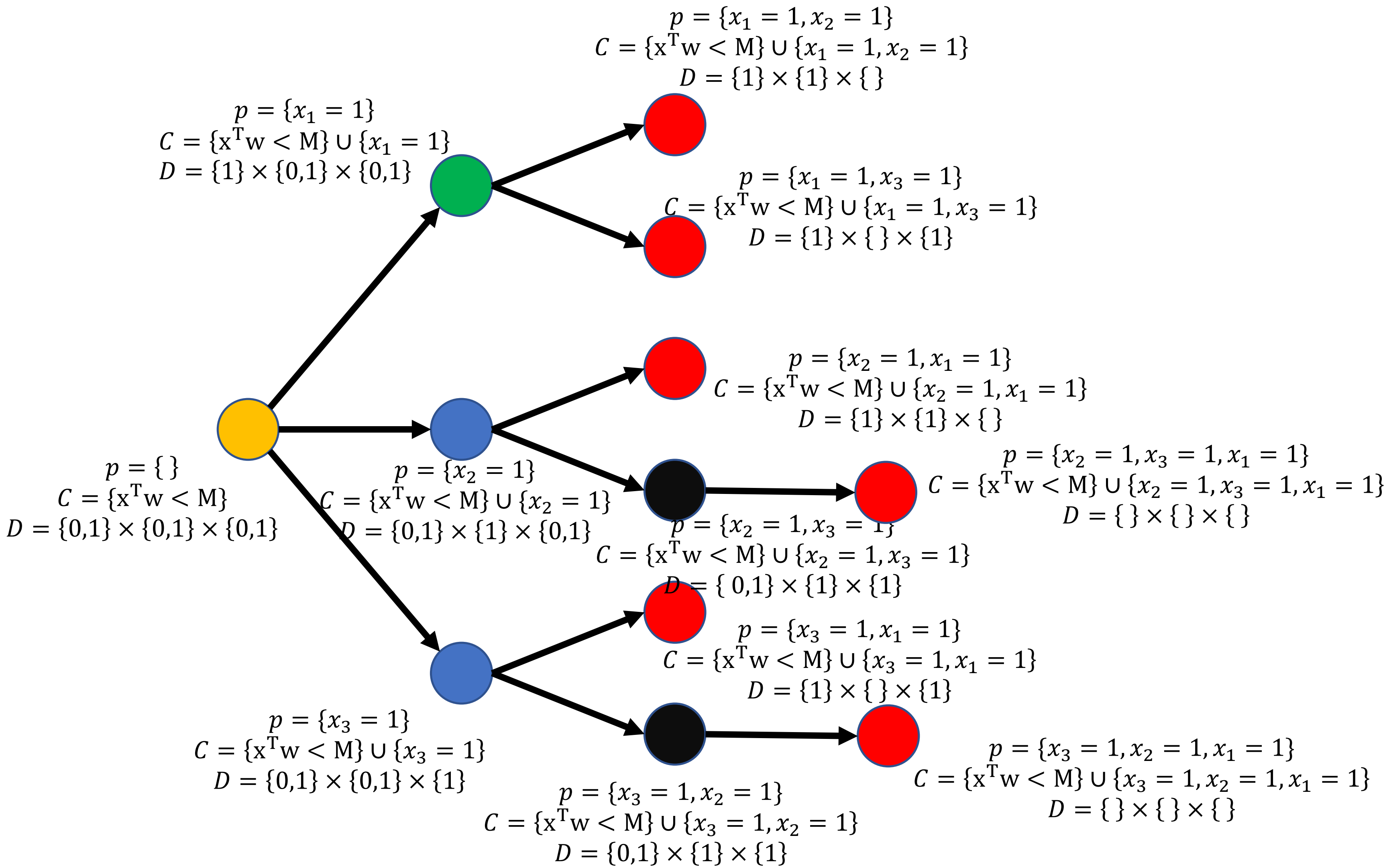
end

Propagator iteration

$$D''(x_i) = \pi_{\{x_i\}}(c_j \cap \pi_{X(c_j)}(D'))$$



Posting constraints



Branching strategies

```
model.AddDecisionStrategy(variables,
                          cp_model.CHOOSE_FIRST,
                          cp_model.SELECT_MIN_VALUE)
```

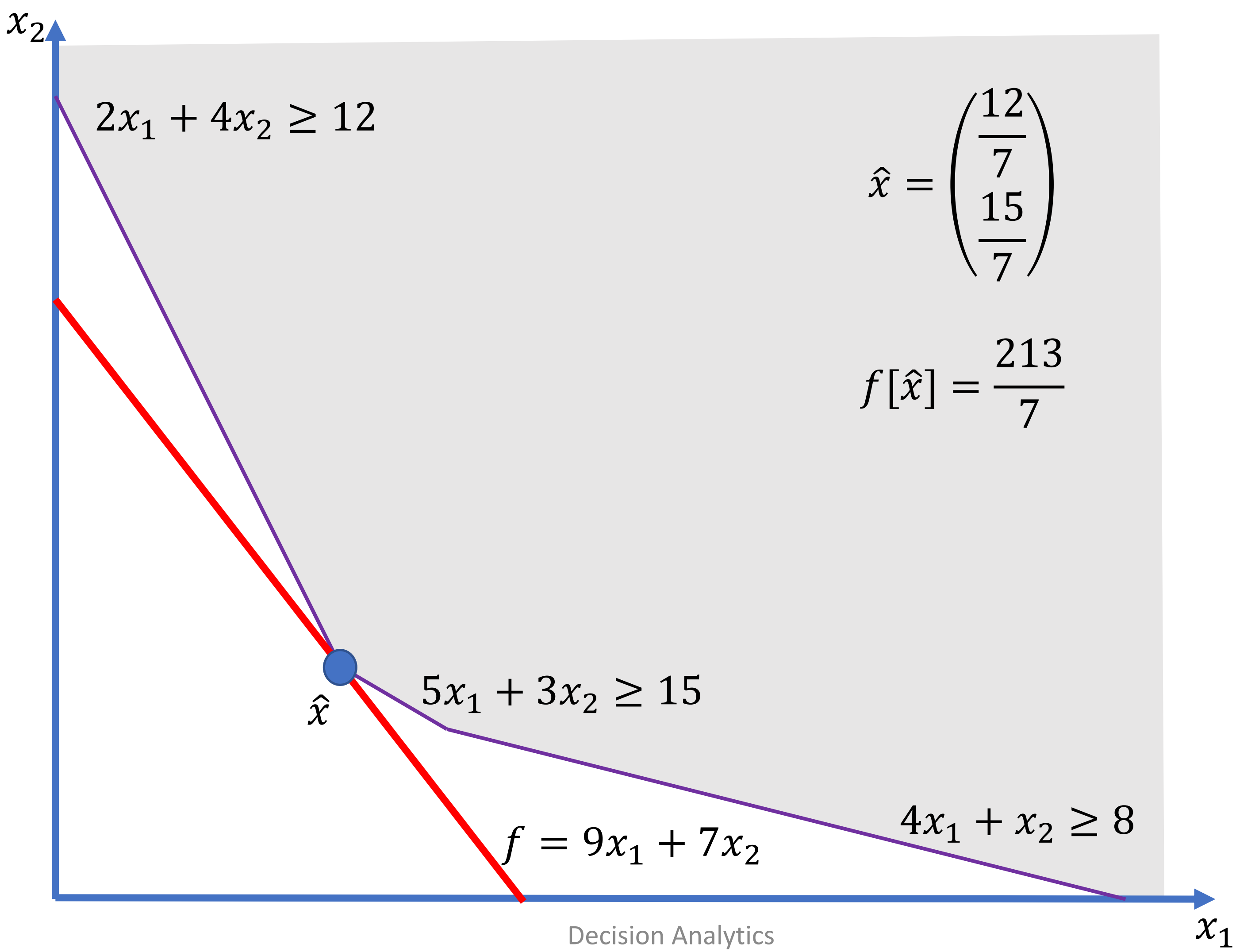
```
CHOOSE_FIRST
CHOOSE_LOWEST_MIN
CHOOSE_HIGHEST_MAX
CHOOSE_MIN_DOMAIN_SIZE
CHOOSE_MAX_DOMAIN_SIZE
```

```
SELECT_MIN_VALUE
SELECT_MAX_VALUE
SELECT_LOWER_HALF
SELECT_UPPER_HALF
```

Maintaining Arc Consistency (MAC)

```
Search(p, (X,D,C)):
    (X,D',C) = make arc consistent (X,D,C)
    if ∃D'(x_i): |D'(x_i)| = 0
        (X,D',C) is a no-good
        return
    else if ∀D'(x_i): |D'(x_i)| = 1
        (X,D',C) is a solution
        return
    else
        choose x_i so that |D'(x_i)| > 1
        choose v for b^1 = {x_i ≤ v}, b^2 = {x_i > v}
        Search(p ∪ {b^1}, (X,D',C ∪ {b^1}))
        Search(p ∪ {b^2}, (X,D',C ∪ {b^2}))
```

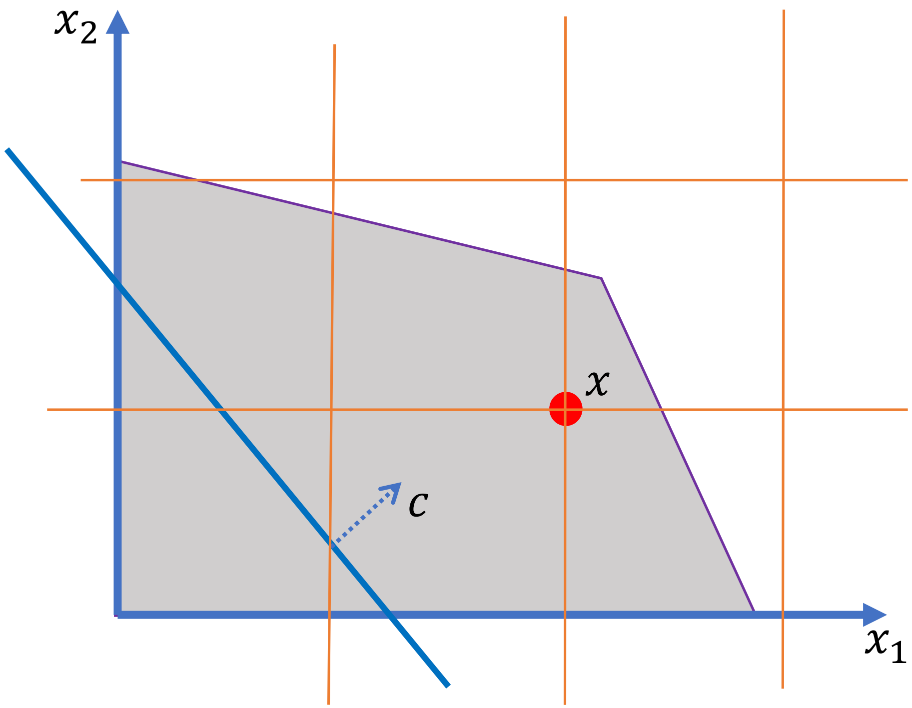
Linear programming



The simplex algorithm

$$
\begin{array}{c}
\begin{array}{cccc} \quad -\boldsymbol{f} \quad & \boldsymbol{x_B} & \qquad \boldsymbol{x_A} \qquad & \qquad \end{array} \\
\left(\begin{array}{c|cc|c}
1 & 0 & c_N^T - y^T A_N & -y^T b \\
\hline
0 & I & A_B^{-1} A_N & A_B^{-1} b
\end{array}\right)
\end{array}
$$

Integer Linear Programming



Thank you for your attention!