CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT

Decision Analytics

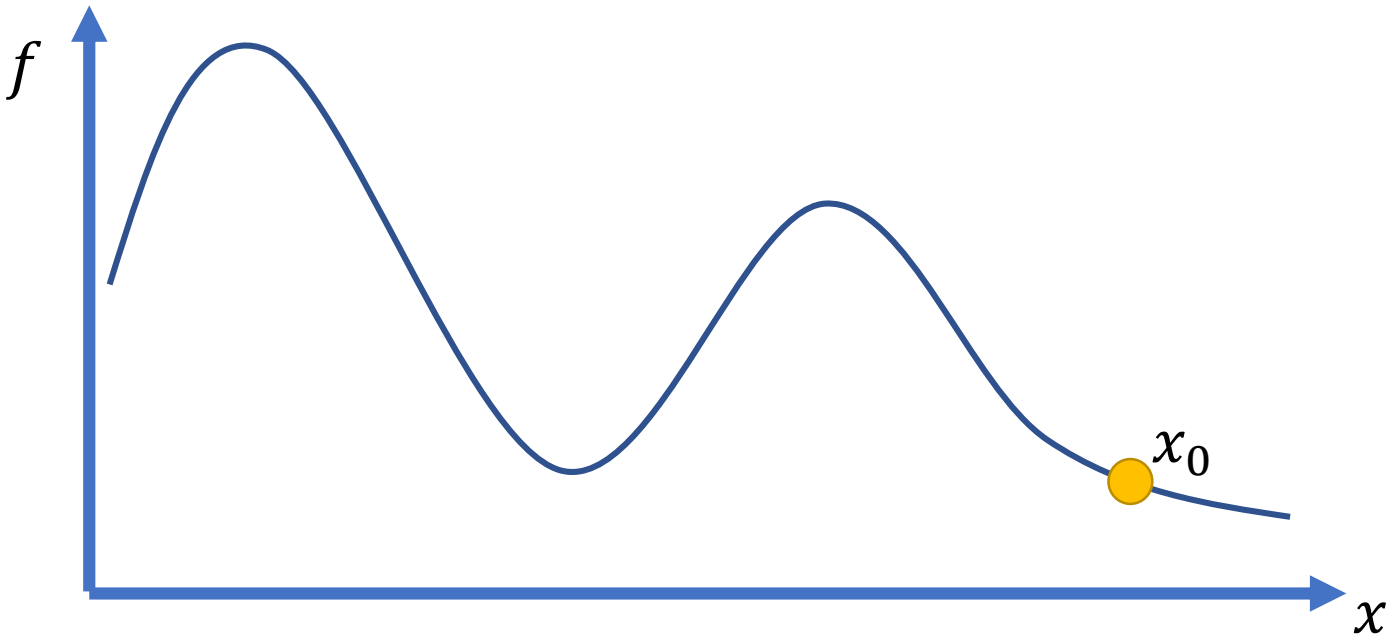
Lecture 6: Solution strategies

Solution strategies

- There are three solution strategies for mathematical optimisation problems that we will consider in this module
  - Neighbourhood based search (briefly)
  - Linear programming
  - Constraint programming

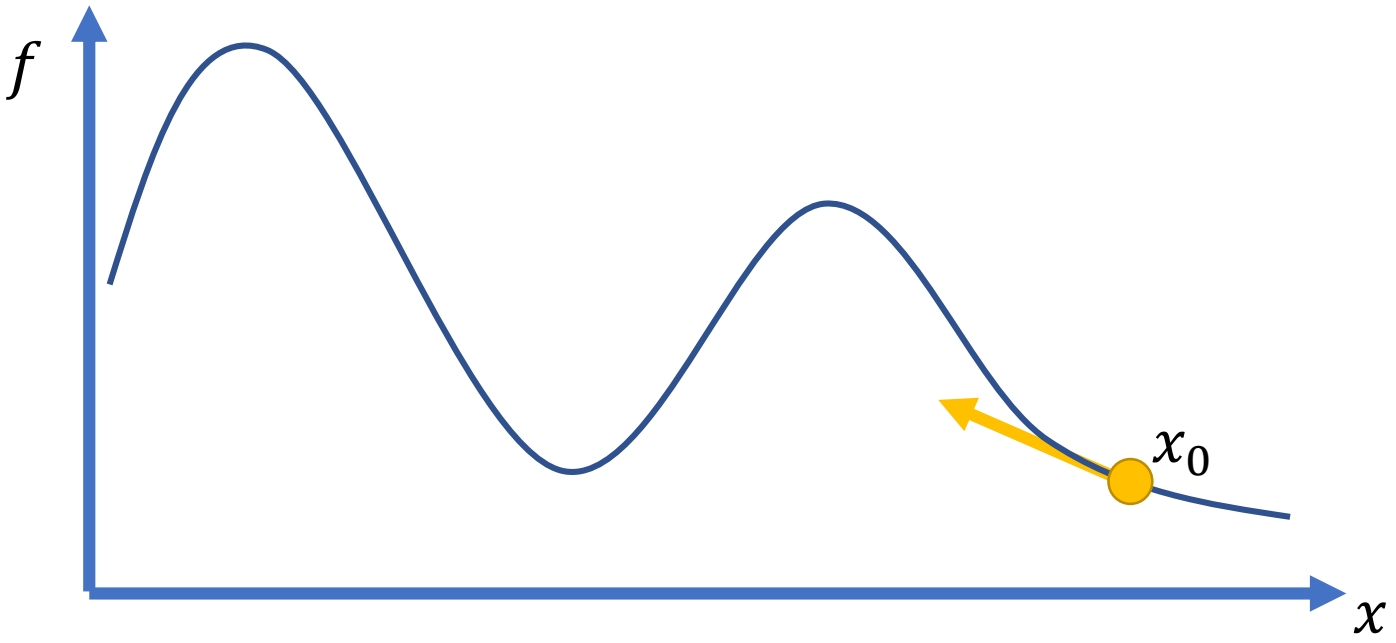
Neighbourhood search algorithms

- Basic idea:
  - Start from an initial feasible solution $\mathrm{x}_0 \in D$



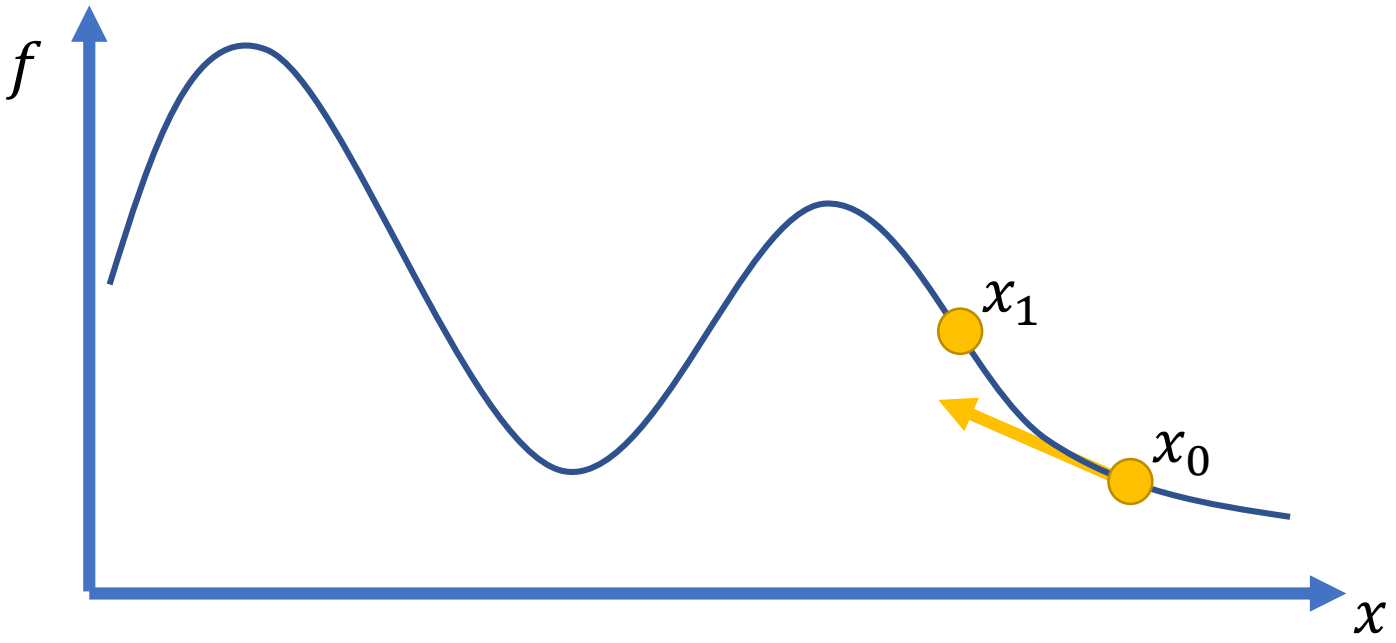
Neighbourhood search algorithms

- Basic idea:
  - Start from an initial feasible solution $\mathrm{x}_0 \in D$
  - Look at "neighbourhood" of x_0 and determine the local "slope" of f



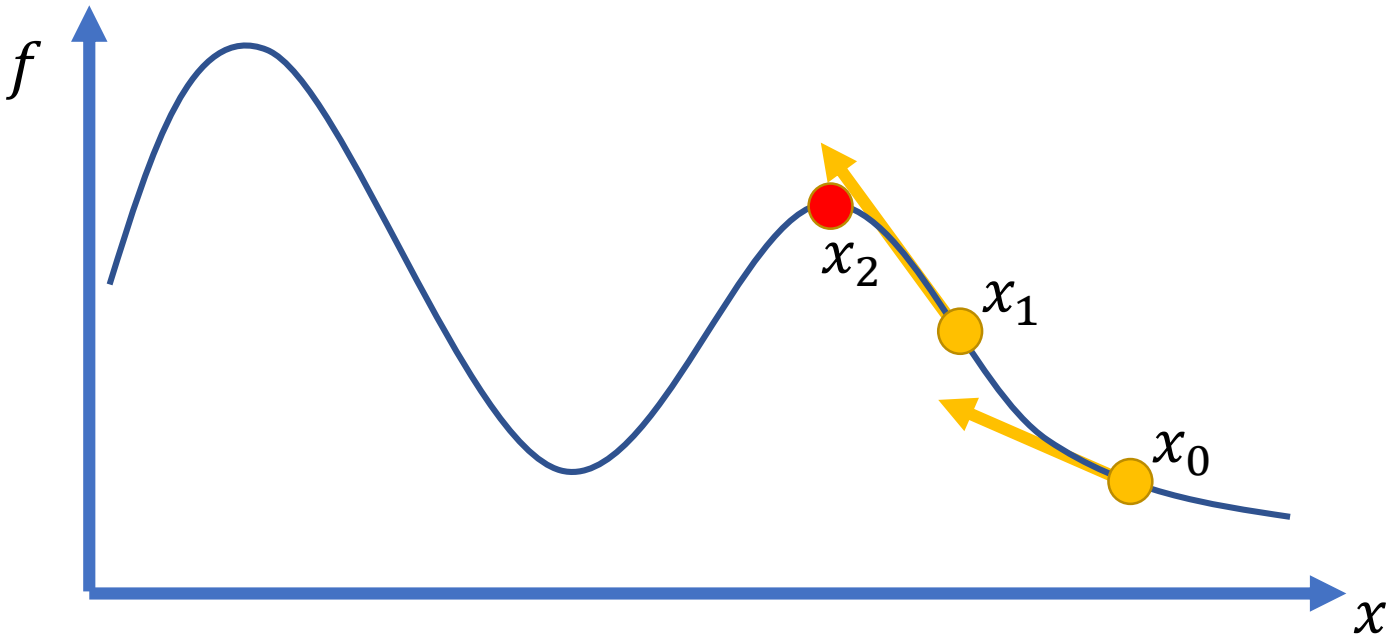
Neighbourhood search algorithms

- Basic idea:
  - Start from an initial feasible solution $\mathrm{x}_0 \in D$
  - Look at "neighbourhood" of x_0 and determine the local "slope" of f
  - Move in the direction where the objective function is ascending



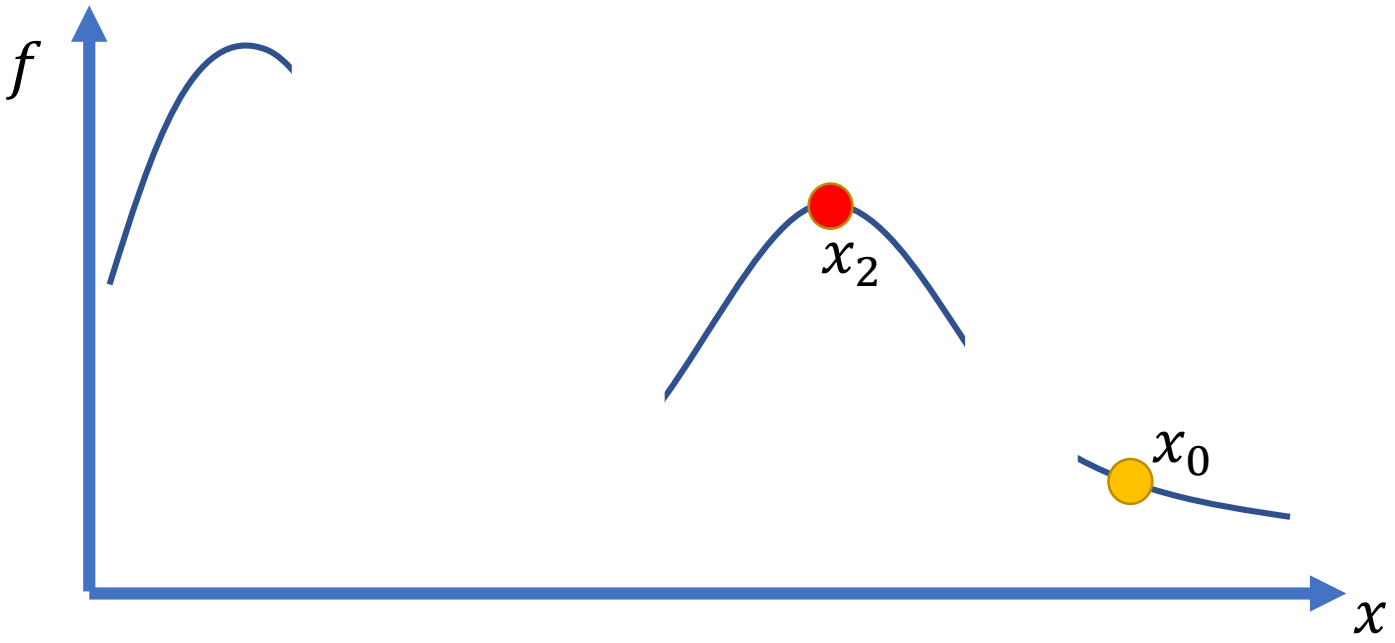
Neighbourhood search algorithms

- Basic idea:
  - Start from an initial feasible solution $\mathrm{x}_0 \in D$
  - Look at "neighbourhood" of x_0 and determine the local "slope" of f
  - Move in the direction where the objective function is ascending
  - Repeat until all local neighbours do not improve the cost function



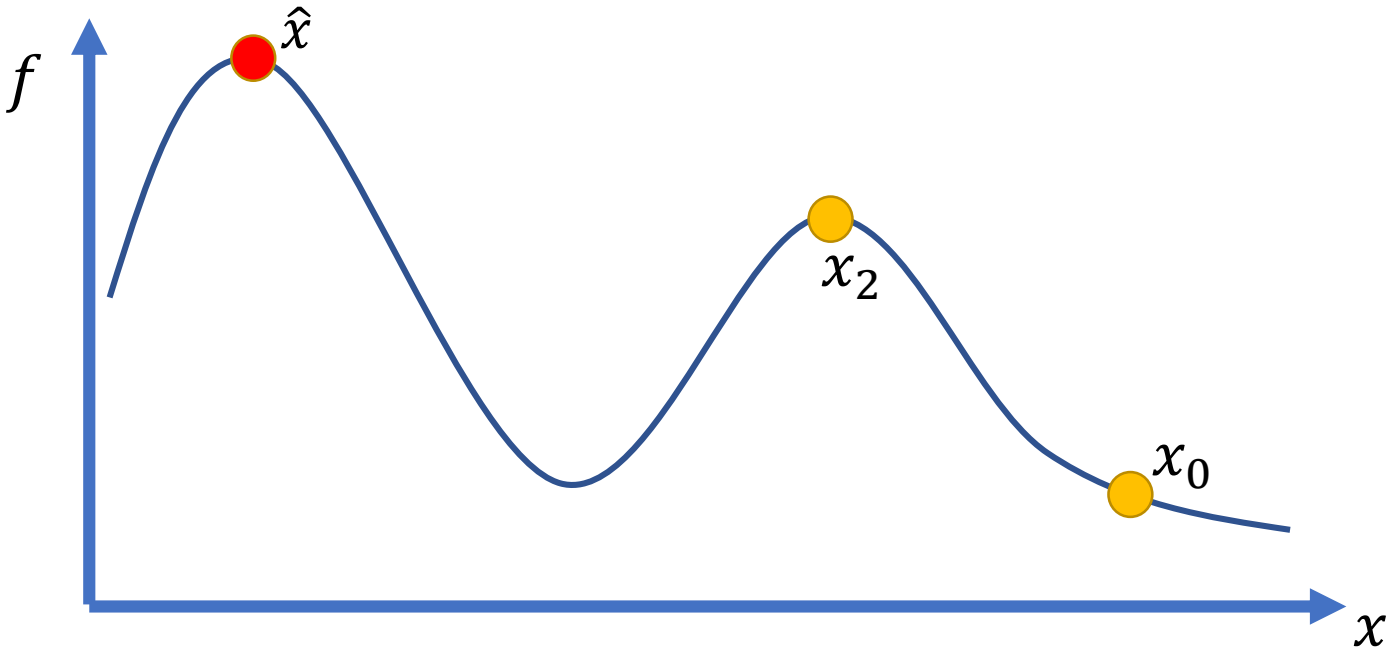
Neighbourhood search algorithms

- Potential issues:
  - Only applicable if the topology of the domain D is connected and allows for a meaningful definition of a neighbourhood relation



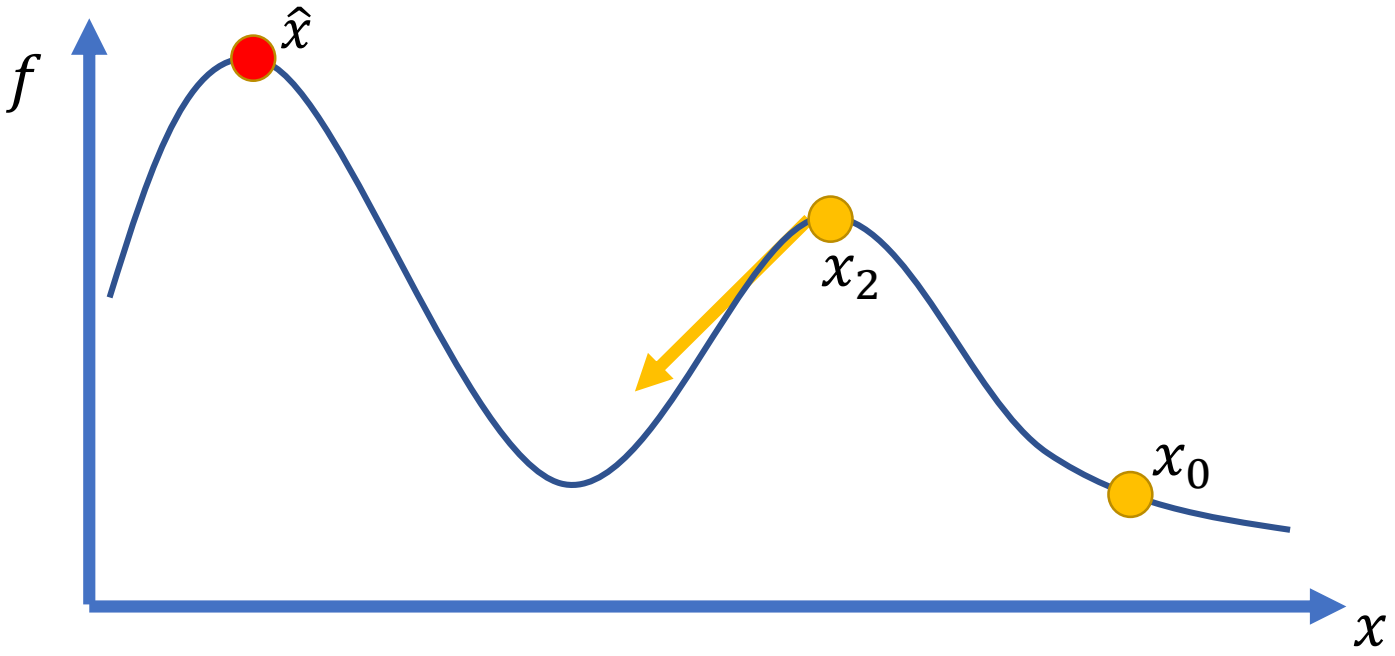
Neighbourhood search algorithms

- Potential issues:
  - Only applicable if the topology of the domain D is connected and allows for a meaningful definition of a neighbourhood relation
  - For non-convex objective functions f the local maximum x_2 that is found does not have to be the global maximum $\hat{x}$



Neighbourhood search algorithms

- The issue with local maxima can be overcome by methods such as simulated annealing or Metropolis-Hastings optimisation
- Both essentially allow for occasionally moving into the direction of decreasing objective function as well

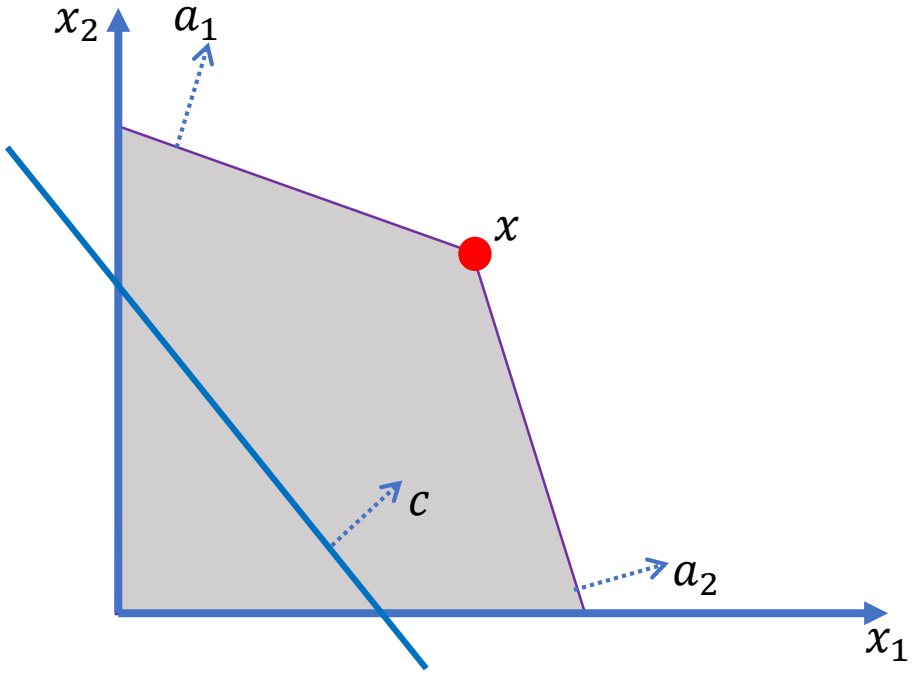


Linear programming

- In case the domain is restricted by linear inequalities

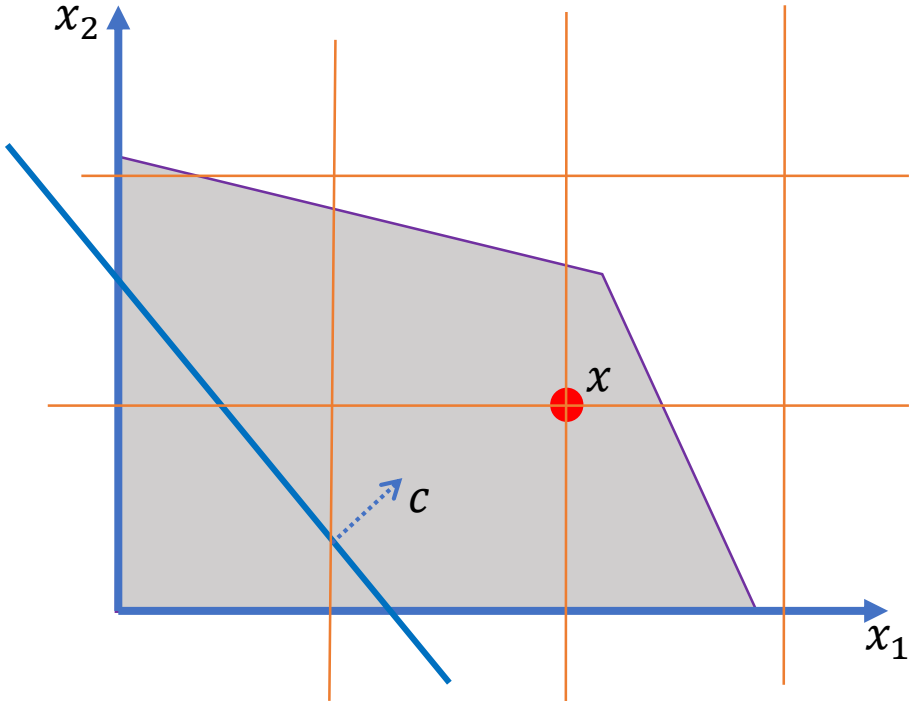
$$a_1 x \leq b_1$$
$$a_2 x \leq b_2$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$

- The feasible solution space becomes a simplex
- If the objective function is linear, the solution is on a corner of the simplex and can be efficiently calculated



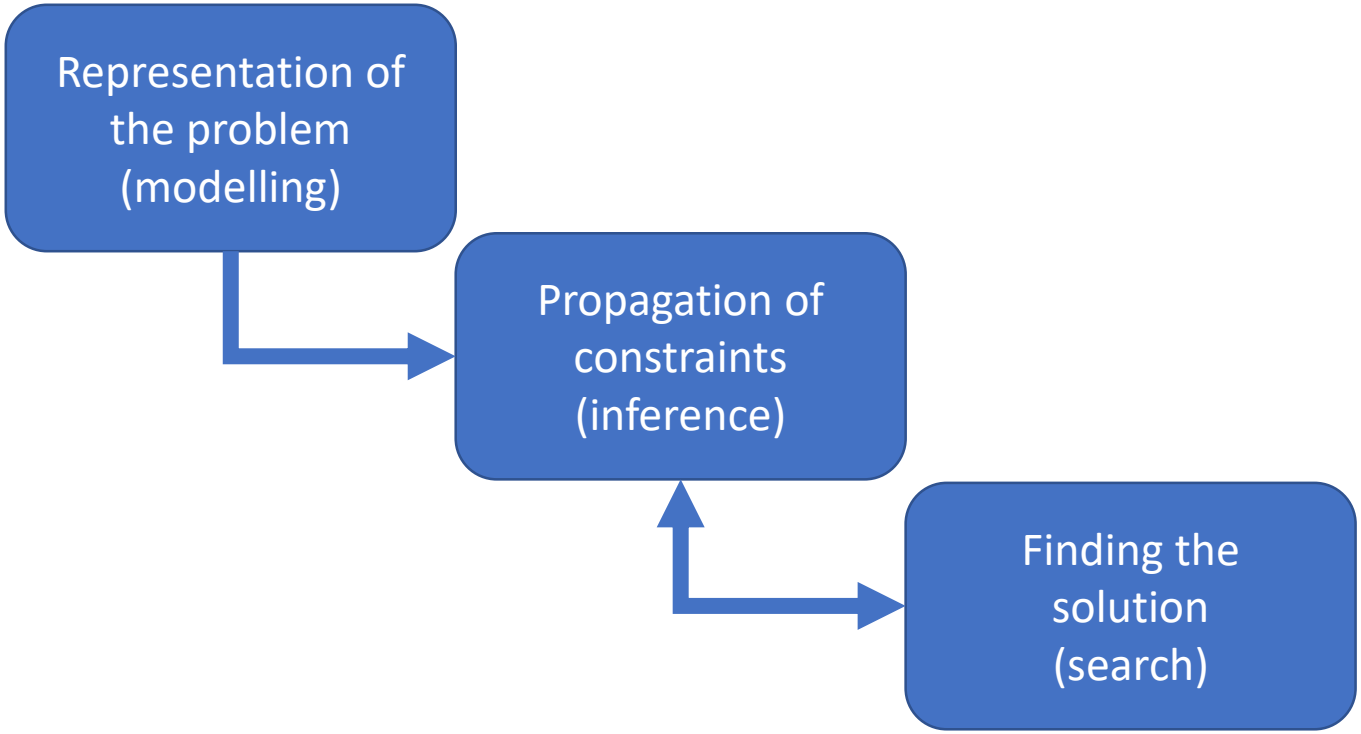
Linear programming

- This is only possible if the domain is continuous, for discrete problems the feasible solution is no longer guaranteed to be on the border of the simplex



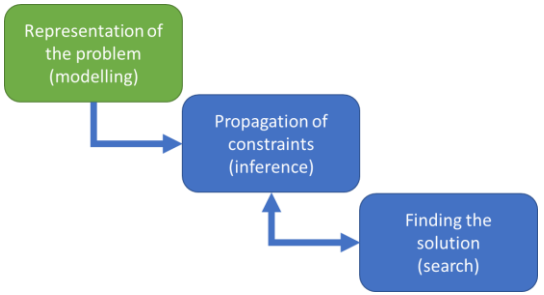
Constraint Programming

- Constraint Programming (CP) is a paradigm for solving combinatorial constraint satisfaction and constrained optimisation problems



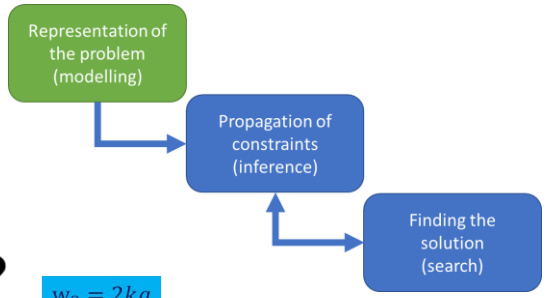
- All of these steps need to be considered to be able to find good solutions for the difficult problems tackled by CP

Constraint Programming

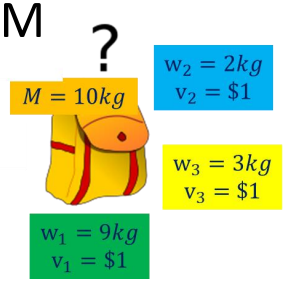


- A **Constraint Satisfaction Problem** (CSP) is defined by
  - A tuple of n variables
    $$X =< x_1, \dots, x_n >$$
  - a corresponding tuple of domains
    $$D =< D_1, \dots, D_n >$$
    defining the potential values each variable can assume
    $$x_i \in D_i$$
  - and a tuple of t constraints
    $$C =< C_1, \dots, C_t >$$
    each being defined itself by a tuple
    $$\mathrm{C}_i =< R_{S_i}, S_i >$$
    - comprising the scope of the constraint $S_i \subset X$, being the subset of variables the constraint operates on
    - and the relation $R_{S_i} \subset D_{s_{i_1}} \times \cdots \times D_{s_{i_{|S_i|}}}$, being the set of valid variable assignments in the scope of the constraint

Constraint Programming

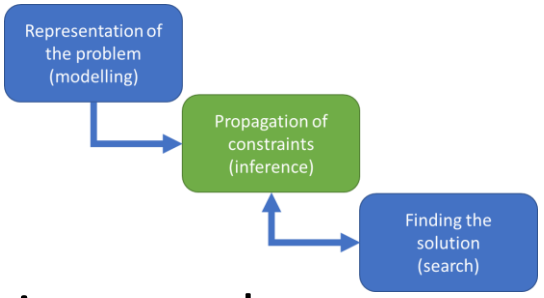


- **Example:** Simple 3-item knapsack of capacity M with item weights w_1, w_2, w_3



- A CSP model for this problem can be defined by
  - A tuple of 3 variables indicating if an item is in the knapsack of not
    $$X =< x_1, x_2, x_3 >$$
  - the corresponding domains are Boolean
    $$D =< \{0,1\}, \{0,1\}, \{0,1\} >$$
  - there is only one constraint stating that the capacity limit should not be exceeded, hence
    $$C =<< R_{S_1}, S_1 >>$$
    with
    - the scope of this single constraint being all variables
      $$S_1 =< 1,2,3 >$$
    - and the set of valid solutions imposed by this constraint being
      $$R_{S_1} = \{< x_1, x_2, x_3 > \in D_1 \times D_2 \times D_3 | x_1 w_1 + x_2 w_2 + x_3 w_3$$

Constraint Programming



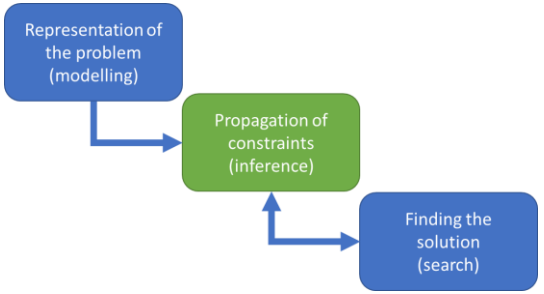
- **Constraint propagation** is used to infer restrictions on the variable domains, thereby reducing the search space
- Intuitively we are enforcing local consistency and propagating this consistency through the hypergraph defined by the constraints
  (a more formal definition will follow later when we cover constraint propagation in depth)
- For instance, each pairwise constraint restricts the domain of direct neighbours

$$D_i \subset \pi_i(R_{ij} \bowtie D_j)$$

- This can be used to iteratively reduce the search space

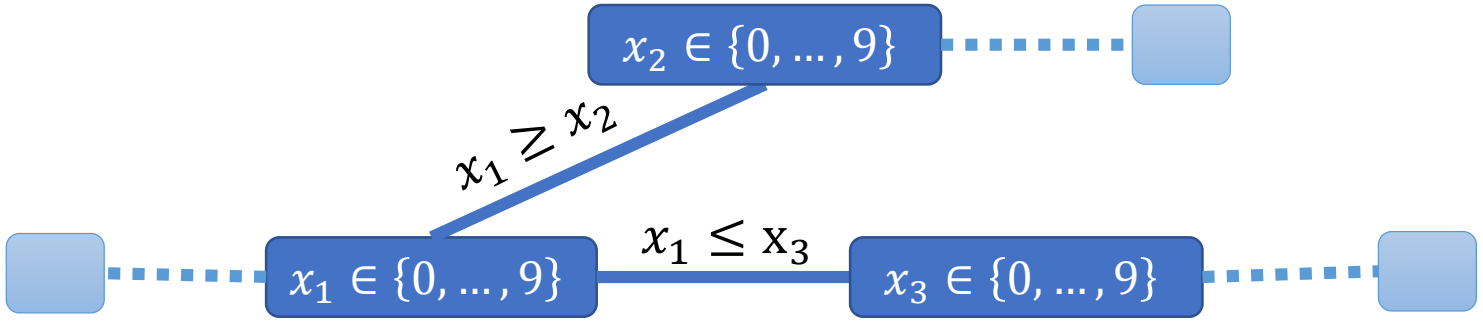
$$D_i := D_i \bigcap \pi_i(R_{ij} \bowtie D_j)$$

Constraint Programming

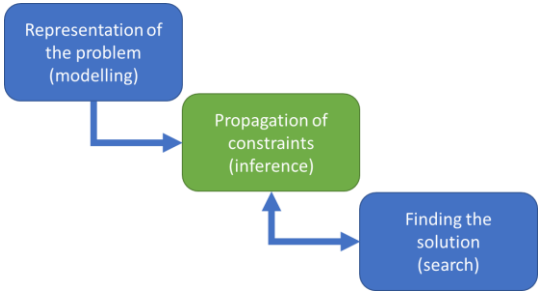


- Example:

$$X =< x_1, x_2, x_3, \dots >$$
$$D =< \{0, \dots, 9\}, \{0, \dots, 9\}, \{0, \dots, 9\}, \dots >$$
$$C =<< R_{S_1}, S_1 >, < R_{S_2}, S_2 >, \dots >$$
$$S_1 =< 1,2 >, R_{S_1} = \{< x_1, x_2 > | x_1 \geq x_2\}$$
$$S_2 =< 1,3 >, R_{S_2} = \{< x_1, x_3 > | x_1 \leq x_3\}$$



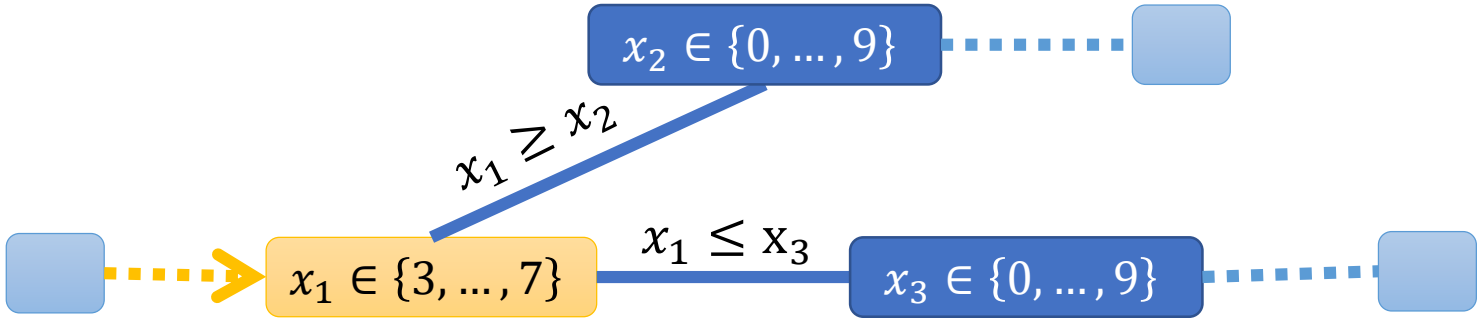
Constraint Programming



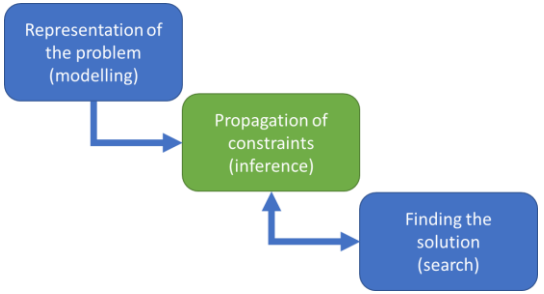
- Example:

$$X = < x_1, x_2, x_3, \dots >$$
$$D = < \{0, \dots, 9\}, \{0, \dots, 9\}, \{0, \dots, 9\}, \dots >$$
$$C = << R_{S_1}, S_1 >, < R_{S_2}, S_2 >, \dots >$$
$$S_1 = < 1,2 >, R_{S_1} = \{< x_1, x_2 > | x_1 \geq x_2\}$$
$$S_2 = < 1,3 >, R_{S_2} = \{< x_1, x_3 > | x_1 \leq x_3\}$$

- Now let's assume that we get to know that $3 \leq x_1 \leq 7$, then



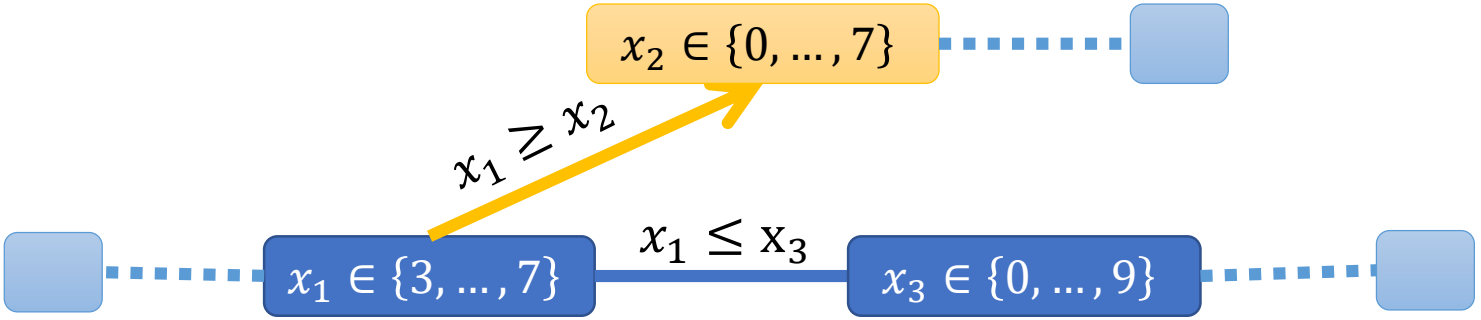
Constraint Programming



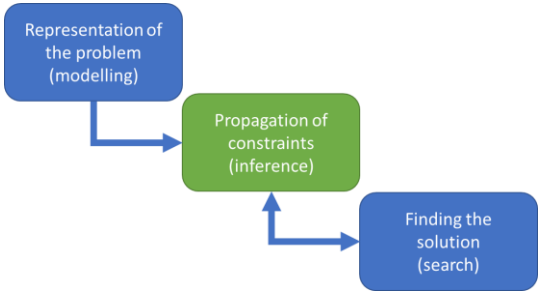
- Example:

$$X =< x_1, x_2, x_3, \dots >$$
$$D =< \{0, \dots, 9\}, \{0, \dots, 9\}, \{0, \dots, 9\}, \dots >$$
$$C =<< R_{S_1}, S_1 >, < R_{S_2}, S_2 >, \dots >$$
$$S_1 =< 1,2 >, R_{S_1} = \{< x_1, x_2 > | x_1 \geq x_2\}$$
$$S_2 =< 1,3 >, R_{S_2} = \{< x_1, x_3 > | x_1 \leq x_3\}$$

- This constraint propagates to its neighbour x_2



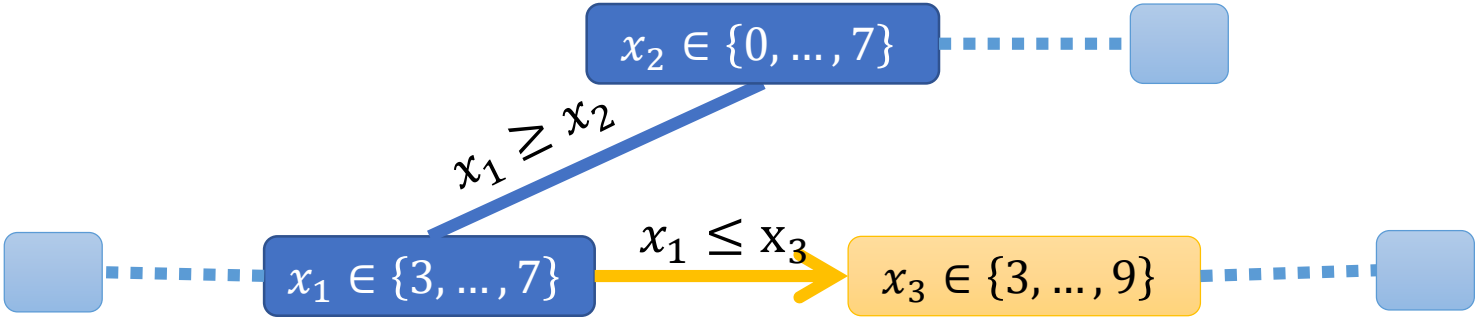
Constraint Programming



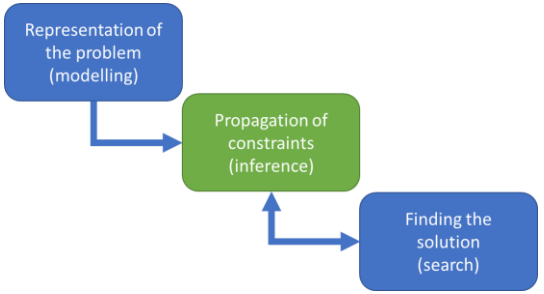
- Example:

$$X =< x_1, x_2, x_3, \dots >$$
$$D =< \{0, \dots, 9\}, \{0, \dots, 9\}, \{0, \dots, 9\}, \dots >$$
$$C =<< R_{S_1}, S_1 >, < R_{S_2}, S_2 >, \dots >$$
$$S_1 =< 1,2 >, R_{S_1} = \{< x_1, x_2 > | x_1 \geq x_2\}$$
$$S_2 =< 1,3 >, R_{S_2} = \{< x_1, x_3 > | x_1 \leq x_3\}$$

- This constraint propagates to its neighbour x_2 and x_3



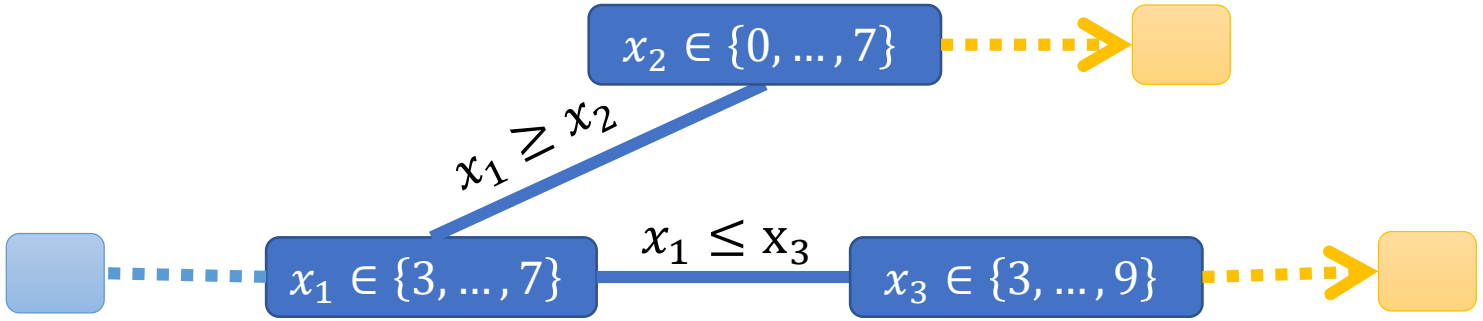
Constraint Programming



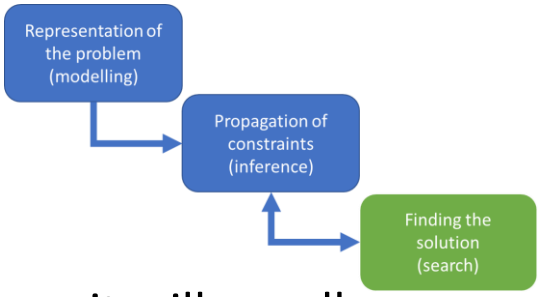
- Example:

$$X =< x_1, x_2, x_3, \dots >$$
$$D =< \{0, \dots, 9\}, \{0, \dots, 9\}, \{0, \dots, 9\}, \dots >$$
$$C =<< R_{S_1}, S_1 >, < R_{S_2}, S_2 >, \dots >$$
$$S_1 =< 1,2 >, R_{S_1} = \{< x_1, x_2 > | x_1 \geq x_2\}$$
$$S_2 =< 1,3 >, R_{S_2} = \{< x_1, x_3 > | x_1 \leq x_3\}$$

- From this locally consistent nodes, constraints can propagate further

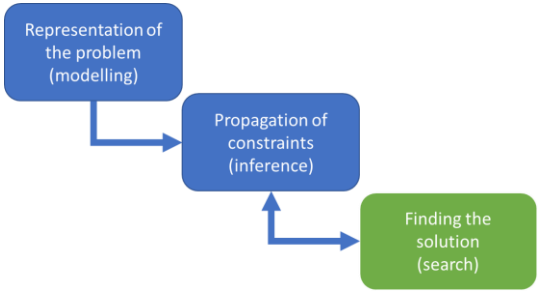


Constraint Programming



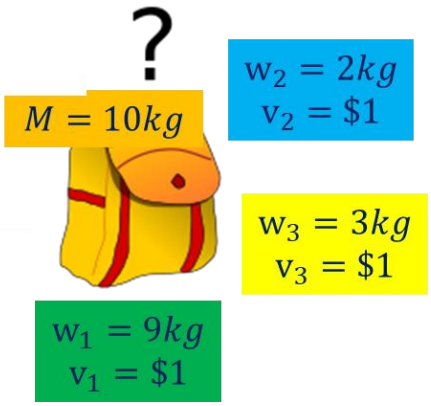
- While constraint propagation will reduce the search space, it will usually not result in a single solution but rather leave us with a (reduced) set of potential solutions
- **Backtracking Search** is used to find feasible solution in the remaining solution space
  - A variable x_i with multiple possible assignment options, i.e. $|D_i| > 1$, is selected for branching (*variable strategy*)
  - For all possible options $x_i \in D_i$ the constraints are propagated to see if all domains remain feasible, i.e. $\forall j: |D_j| \neq \emptyset$
  - For all such feasible options (order according to a *domain strategy*) the search is continued from the top
  - If no such feasible options remain, i.e. after constraint propagation there is an impossible assignment ($\exists j: |D_j| = \emptyset$) for all potential assignment of $x_i \in D_i$, we backtrack to a previous position in the search tree and select a different variable for branching instead

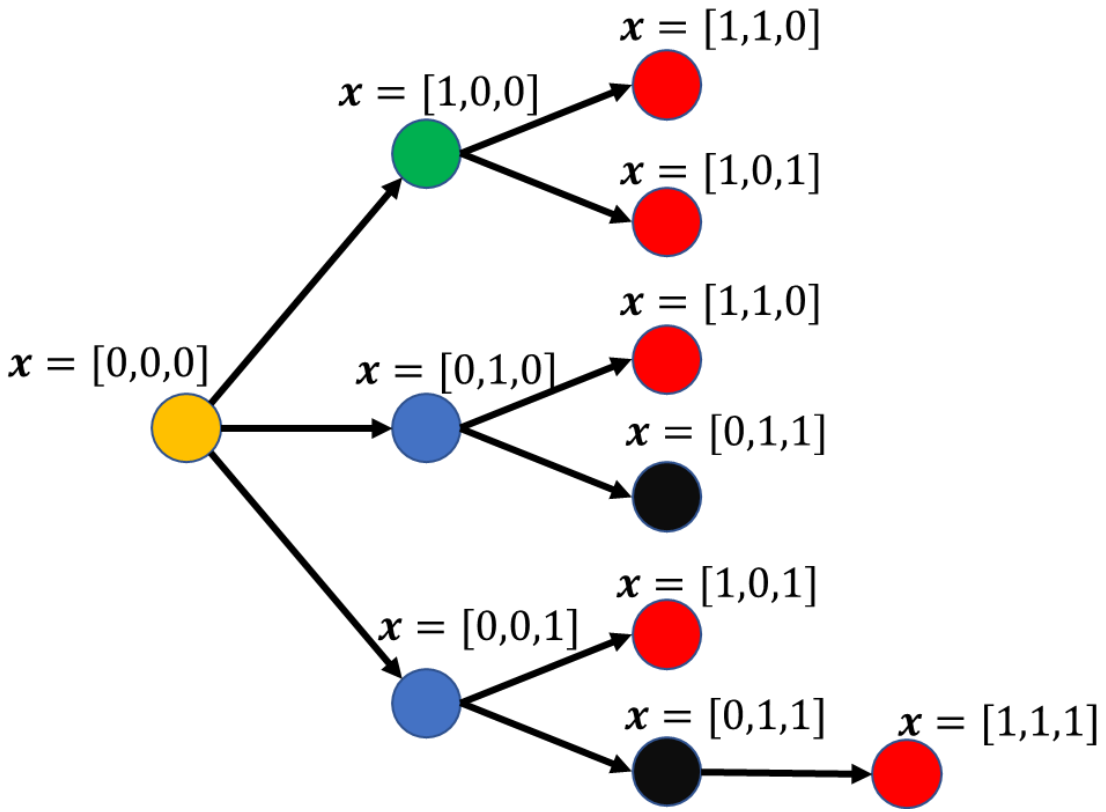
Constraint Programming



- **Example:** Simple 3-item knapsack of capacity M with item weights w_1, w_2, w_3

- Variable strategy:
  CHOOSE_FIRST
- Domain strategy:
  SELECT_MIN_VALUE





Thank you for your attention!