# Constraint Handling — Representation, Initialisation and Neighbourhood Operators

Leandro L. Minku

# How to Deal with Constraints in Optimisation Problems?

- Most real world problems have constraints.
- Optimisation algorithms themselves usually do not contain strategies to deal with constraints.
- Instead, strategies need to be designed for each problem.
- Examples of strategies:
    - Representation, initialisation and neighbourhood operators.
    - Objective function.

# How to Deal with Constraints in Optimisation Problems?

- Most real world problems have constraints.
- Optimisation algorithms themselves usually do not contain strategies to deal with constraints.
- Instead, strategies need to be designed for each problem.
- Examples of strategies:
  - Representation, initialisation and neighbourhood operators.
  - Objective function.

# Traveling Salesman Problem Formulation

- Design variables represent a candidate solution.
  - The design variable is a sequence $\mathbf{x}$ of $N$ cities, where $x_i \in \{1, \cdots, N\}$, $\forall i \in \{1, \cdots, N\}$.
  - The $N$ cities to be visited are represented by values $\{1, \ldots, N\}$.
  - The search space is all possible sequences of $N$ cities, where cities are in $\{1, \ldots, N\}$.

- Objective function defines the cost of a solution.

$$\text{minimise totalDistance}(\mathbf{x}) = \left( \sum_{i=1}^{N-1} D_{x_i, x_{i+1}} \right) + D_{x_N, x_1}$$

where $D_{j,k}$ is the distance of the path between cities $j$ and $k$.

- [Optional] Solutions must satisfy certain constraints.

$$\forall i \in \{1, \cdots, N\}, \quad h_i(\mathbf{x}) = \left( \sum_{j=1}^{N} 1(x_j = i) \right) - 1 = 0 \qquad 1(x_j = i) = \begin{cases} 1, \text{ if } x_j = i \\ 0, \text{ if } x_j \neq i \end{cases}$$

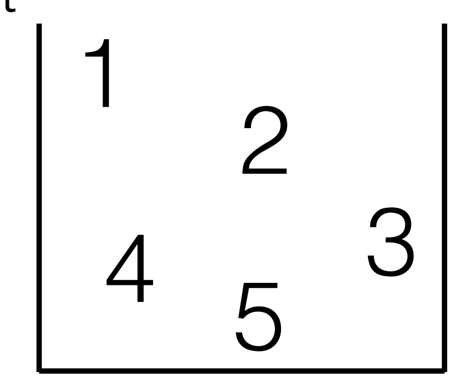
# Designing Representation, Initialisation and Neighbourhood Operators to Deal with Constraints

- Representation:
  - 1-dimensional array of size *N*, where *N* is the number of cities to visit.
  - The fact that the return to the initial city is not in the representation helps to deal with the implicit constraint that we must return to the city of origin.
  - E.g.: for *N* = 5

1 3 2 4 5 1          3 1 2 4 5 3

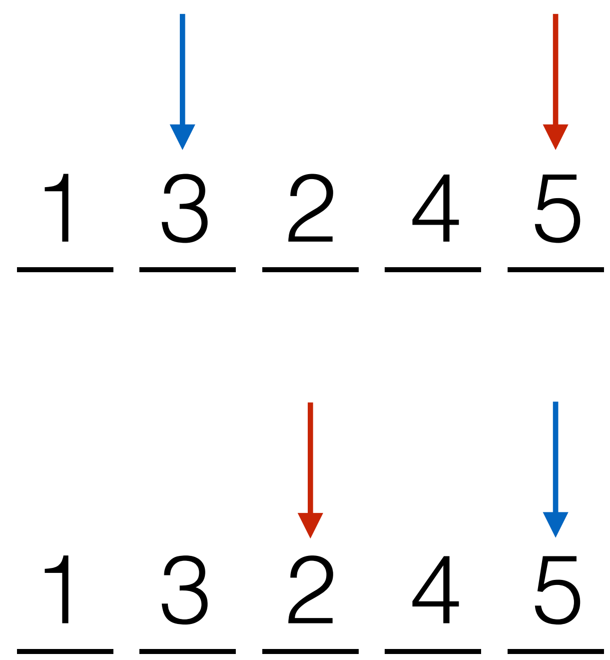
- Initialisation:
  - Draw cities uniformly at random from {1,…,*N*} without replacement,
  - This ensures that there will be no missing or duplicated cities (explicit constraint) and that only cities in {1,…,*N*} are used (implicit constraint).

___ ___ ___ ___ ___

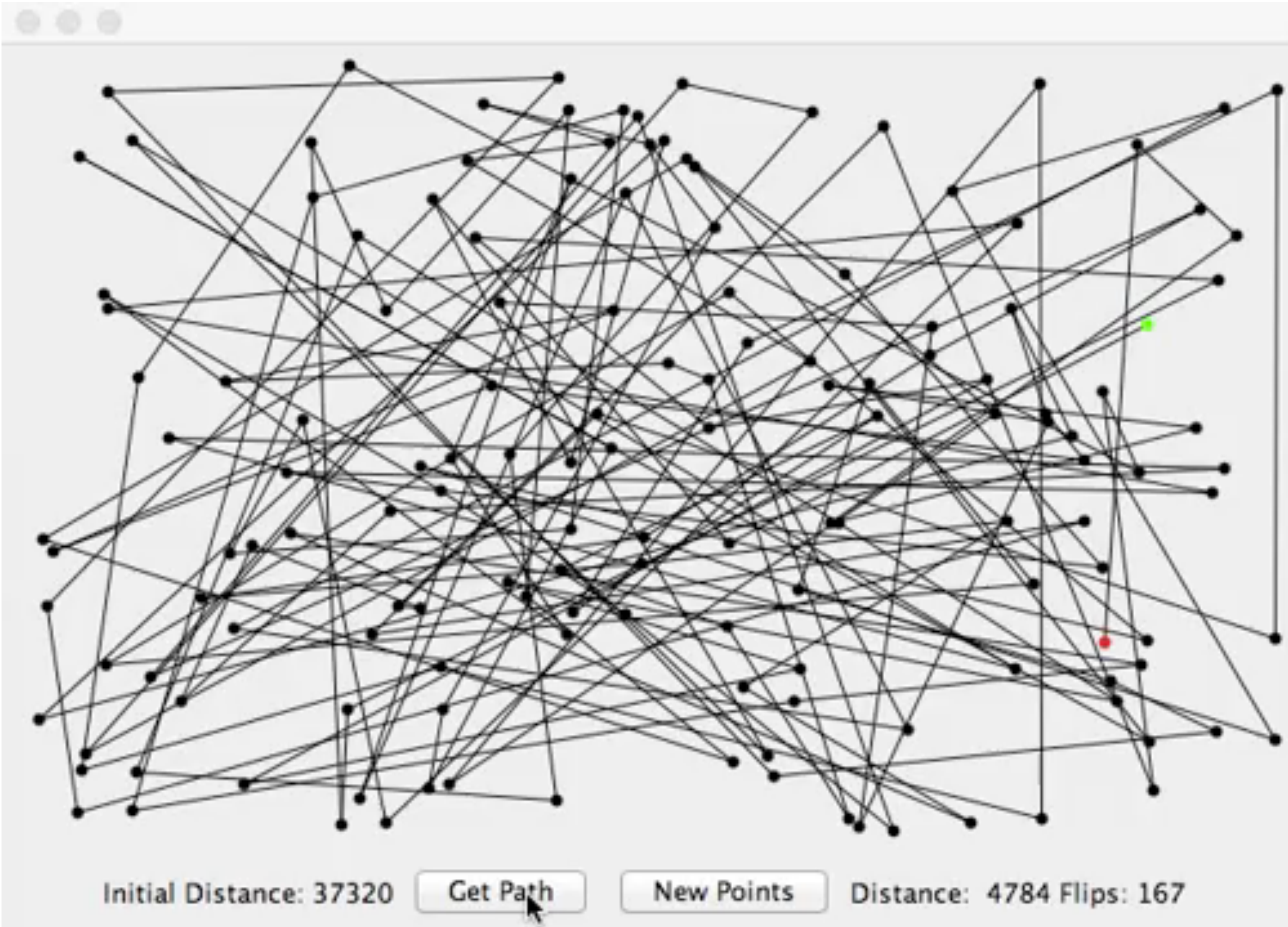


# Designing Representation, Initialisation and Neighbourhood Operators to Deal with Constraints

- Neighbourhood operator:
  - Reverse the path between two randomly picked cities.
  - This ensures that there will be no missing or duplicated cities (explicit constraint) and that only cities in {1,…,$N$} are used (implicit constraint).




  - This design ensures that the constraints are satisfied.




[Video posted by sarahbau: https://youtu.be/3TrnjUKeFg8 ]

# Dealing with Constraints Based on Representation, Initialisation and Neighbourhood Operators

- Advantage:
  - Ensure that no infeasible candidate solutions will be generated, facilitating the search for optimal solutions.
- Disadvantage:
  - May be difficult to design, and the design is problem-dependent.
  - Sometimes, it could restrict the search space too much, making it difficult to find the optimal solution.

# Summary

- We need to design strategies to deal with the constraints.
- Examples of strategies:
  - Representation, initialisation and neighbourhood operators.
  - Objective function.

# Next

- Examples of strategies:
  - Representation, initialisation and neighbourhood operators.
  - Objective function.