Constraint Handling — Objective Function

Leandro L. Minku

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 Objective Functions to Deal With Constraints

- The original objective function of a problem can be modified to deal with constraints.
- A penalty can be added for infeasible solutions, increasing their cost.

Designing Objective Functions to Deal With Constraints

- E.g.: assume that the representation is a list of any size, and that our initialisation procedure is uniformly at random with replacement.

1	2	2	3	4
1	2	3	4	

N=5

- Objetive function:

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

How to modify the objective function to deal with the constraint that each city must appear once and only once?

Designing Objective Functions to Deal With Constraints

- E.g.: assume that the representation is a list of any size, and that our initialisation procedure is uniformly at random with replacement.

1	2	2	3	4
1	2	3	4	

N=5

- Objetive function:

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

where n_m is the number of cities missing, n_d is the number of duplications of cities and P is a large positive constant.

Generalising The Strategy

Minimise $f(\mathbf{x})$

Subject to $g_i(\mathbf{x}) \leq 0, \quad i = 1,\cdots,m$

$h_j(\mathbf{x}) = 0, \quad j = 1,\cdots,n$

Penalty

Minimise $f(\mathbf{x}) + Q(\mathbf{x})$

Only sum here the violated constraints

$$Q(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \text{ is feasible} \\ P \times [g_a(\mathbf{x})^2 + g_b(\mathbf{x})^2 + \cdots + h_{a'}(\mathbf{x})^2 + h_{b'}(\mathbf{x})^2 + \cdots] & \text{otherwise} \end{cases}$$

where P is a large positive constant.

Generalising The Strategy

Minimise $f(\mathbf{x})$

Subject to $g_i(\mathbf{x}) \leq 0, \quad i = 1, \cdots, m$

$h_j(\mathbf{x}) = 0, \quad j = 1, \cdots, n$

Penalty

Minimise $f(\mathbf{x}) + Q(\mathbf{x})$

$$Q(\mathbf{x}) = \quad P \times [v_{g1} g_1(\mathbf{x})^2 + v_{g2} g_2(\mathbf{x})^2 + \cdots + v_{gm} g_m(\mathbf{x})^2 + \\ + v_{h1} h_1(\mathbf{x})^2 + v_{h2} h_2(\mathbf{x})^2 + \cdots + v_{hn} h_n(\mathbf{x})^2]$$

where P is a large positive constant, and v_{gi} and v_{hi} are 1 if the corresponding constraint is violated and 0 otherwise.

Dealing with Constraints Based on Objective Functions

- Advantage:
  - Easier to design.
- Disadvantage:
  - Algorithm has to search for feasible solutions.

Completeness

- If we use a strategy to deal with constraints that never enables any infeasible solution to be generated, algorithms such as Hill Climbing and Simulated Annealing are complete.

- Otherwise:
    - Hill Climbing: not complete if the objective function has local optima.
    - Simulated Annealing: not guaranteed to find a feasible solution within a reasonable amount of time.

Summary

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

Next

- Example applications.