The Quadratic Assignment Problem

University of Geneva

Metaheuristics for Optimization
3 October 2022

The Quadratic assignment Problem (QAP)

The QAP belongs to a class of **combinatorial optimization** problems.

Structure:
Starting from a set of n given *locations* and n given *objects*, *flow values* between the objects and *distances* between the locations; find the best way to

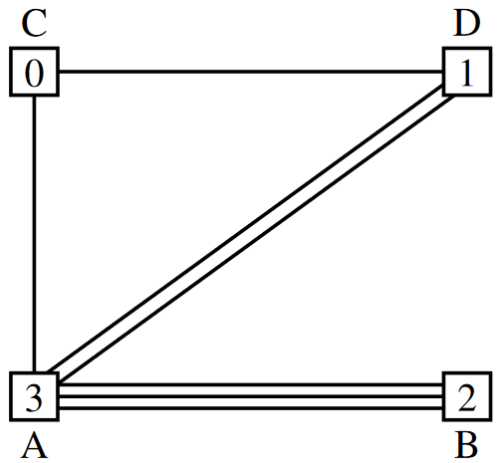
- assign all n objects to the n locations in order to
- **minimize** the sum of the product, for each pair of objects, inter-object flow and related inter-location distance

The search space is the **Permutation Space**

- For a general $n \geq 1$ the search space size is $n! = n(n-1)(n-2)...1$

QAP Example

Imagine we have a set of n cities (Locations) and a set of n factories (Objects). The factories work together so there are some trucks going from one factory to another and the **flow** of trucks between factories changes from pair to pair. The trucks have to travel between two factories so we also have to account for the traveling **distance**.



- Locations: the squares A, B, C, D
- Objects: the numbers 0, 1, 2, 3
- Distances: $d_{AB} = d_{CD} = 4, d_{AC} = d_{BD} = 3, d_{AD} = d_{BC} = 5$
- Flows: $w_{13} = 2, w_{23} = 3, w_{01} = w_{03} = 1$

QAP Example

Objective:

Assign an object to each location (a factory to each city) so that the product between inter-city distance and inter-factory flow is minimized. This means minimize the function:

$$F(x) = \sum_{i,j=0}^{n-1} d_{ij} w_{ij}$$

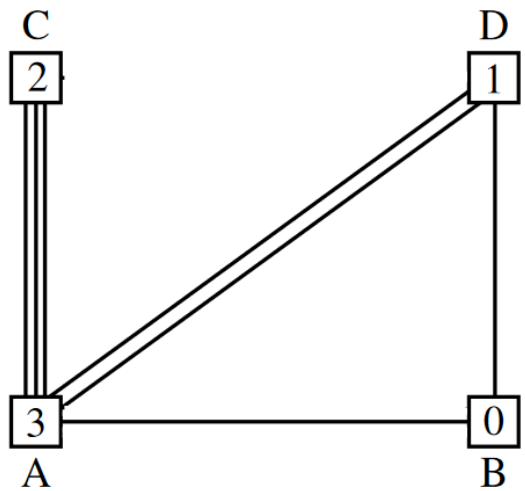
QAP Example

If we choose the solution x = (C, D, B, A) the fitness function is going to be calculated as:

$$F(x) = d_{CD}w_{01} + d_{DA}w_{13} + d_{AC}w_{30} + d_{AB}w_{32} = 29$$

If we consider a permutation of this solution, x' = (B,D,C,A), the fitness function is going to be:

$$F(x') = d_{CA}w_{23} + d_{BA}w_{03} + d_{AD}w_{31} + d_{DB}w_{10} = 26$$

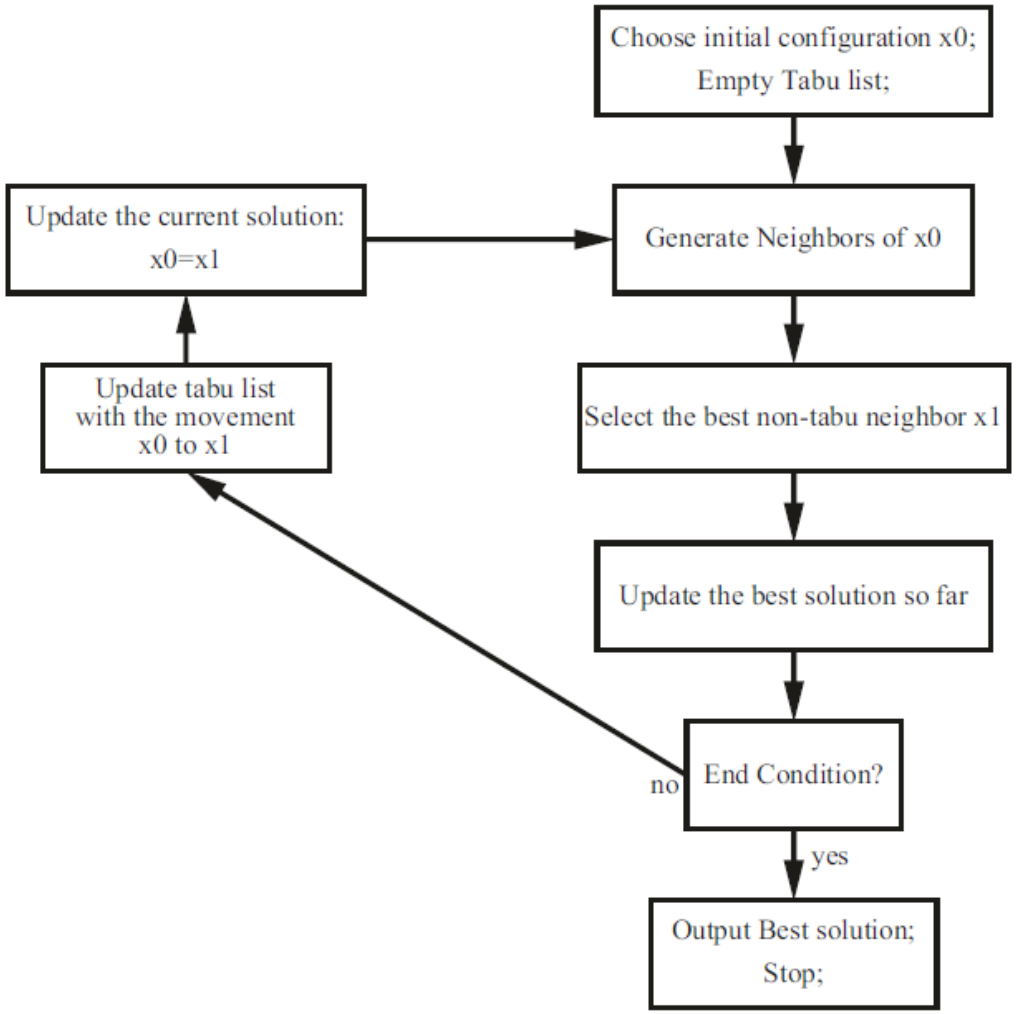


Tabu Search

It gives good results on combinatorial optimization problems, such as the QAP.

- Similarly to the *Deterministic Hill Climbing* compute the fitness of the neighbours and choose as the next possible solution, the one with the **highest fitness value**
- Novelty element = Already explored solutions become part of the *Tabu List* and cannot be visited, they are forbidden solutions for a certain amount of iterations.
- You don't get stuck at a local optimum

Tabu Search - Flow Diagram



The Tabu List

It is created to prevent the exploration to circle back to already visited solutions. It stores the last M moves or points visited.

Characteristics:

- Keeps track of points in the search space that are not allowed or moves that are forbidden
- It has a finite capacity $\rightarrow$ A move or point is not going to be 'tabu' forever $\rightarrow$ This is called **Short Term Memory**
- At every iteration it gets updated: the oldest memorized point becomes available again and the newest visited point enters the end of the list

The Tabu List - Short Term Memory

Consider the current solution and the next solution

$$x = (C, D, B, A)$$

$$x' = (B, D, C, A)$$

We want to remember that factory C cannot go back to address 0 and factory B cannot go back to address 2 for some time. This amount of time is a number l of iterations.

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \longrightarrow \begin{bmatrix} 0 & 0 & l & 0 \\ 0 & 0 & 0 & 0 \\ 0 & l & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The Tabu List - Long Term Memory

A list of frequencies of each move. If one particular move is less frequent it gets imposed.

- Not based on the fitness of a possible solution
- Allows diversification
- Allows exploration of the entire search space