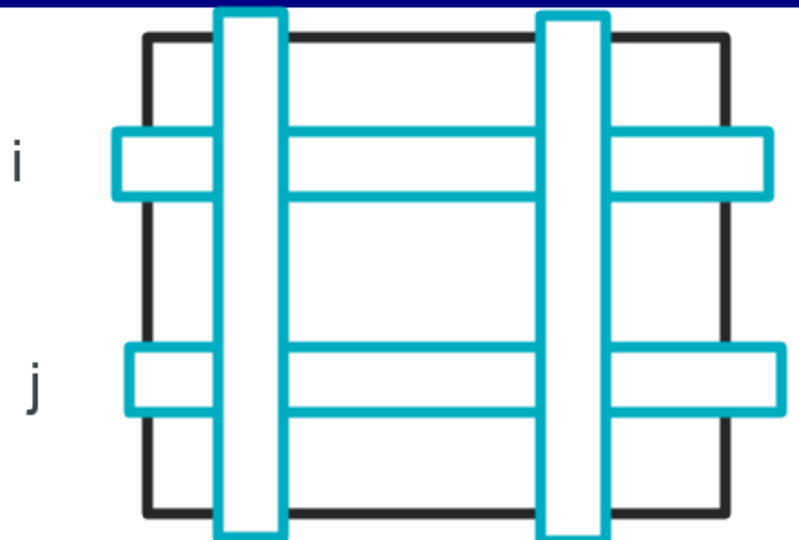
```
def score(self, chromosome):
    return sum(np.sum(self.wMtx * self.dMtx[chromosome[:, None], chromosome], 1))
```

```
def mutationScoreOpt(self, currentChromosome, currentScore, i, j):
    idx = list(range(self.dim)) ; del(idx[i]) ; del(idx[j-1])

    mutationScore = currentScore - sum(np.sum(self.wMtx[[i,j], :] * self.dMtx[currentChromosome[[i,j], None], currentChromosome], 1))
    mutationScore -= sum(sum(self.wMtx[idx][:,[i,j]] * self.dMtx[currentChromosome[idx, None], currentChromosome[[i,j]]]))

    currentChromosome[i], currentChromosome[j] = currentChromosome[j], currentChromosome[i]

    mutationScore += sum(np.sum(self.wMtx[[i,j], :] * self.dMtx[currentChromosome[[i,j], None], currentChromosome], 1))
    mutationScore += sum(sum(self.wMtx[idx][:,[i,j]] * self.dMtx[currentChromosome[idx, None], currentChromosome[[i,j]]]))

    #revert changes
    currentChromosome[i], currentChromosome[j] = currentChromosome[j], currentChromosome[i]

    return mutationScore
```



```cpp
int Model::CalcularCoste(const std::vector<int> & solution) const{
    if(solution.size() == tam){
        int coste = 0;

        for(int i = 0; i < tam; ++i){
            for(int j = i + 1; j < tam; ++j){
                coste += distance[i][j] * flow[solution[i]][solution[j]];
            }
        }

        return coste;
    }

    return -1;
}
```

```cpp
int Model::ReCalcularCoste(const std::vector<int> & solution, int cost, int pos_a, int pos_b) const{
    int new_cost = cost;

    for(int i = 0; i < tam; ++i){
        if(i != pos_a && i != pos_b){
            int change_flow = flow[solution[i]][solution[pos_a]] - flow[solution[i]][solution[pos_b]];

            new_cost += distance[i][pos_b] * change_flow - distance[i][pos_a] * change_flow;
        }
    }

    return new_cost;
}
```