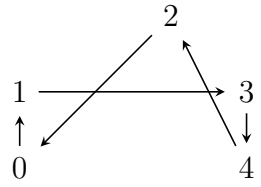
First & last name	Cyril Bousmar
NOMA UCLouvain	63401800

LINFO2266: Advanced Algorithms for Optimization

Project 4: Local Search

Exercise 1 2Opt Move

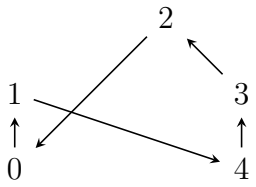
Below is an instance of the Travelling Salesman Problem with 5 cities and a starting solution.

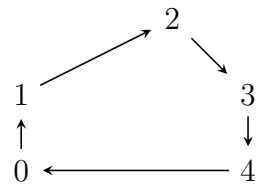


City ID	X position	Y position
0	0	0
1	0	10
2	20	20
3	30	10
4	30	0

1. This is the initial circuit representation [0,1,2,3,4]. Apply an imroving 2Opt move of your choice. Show the sequence of the circuit after and give the delta cost on the objective (it should be negative number).

Using the heuristic for the best first, the result is:





δ	total
-6.6	-6.6
-7.55	-14.15

Exercise 2 Initialization

1. Your PilotInitialization class extends the BeamSearchInitialization class. Explain how you achieved this design (how does it avoid code duplication?).

Since *Pilot Search* can be considered as a special case of *Beam Search* for which the *beam width* has a size of 1, the implementation of `PilotInitialization` can simply be initialized with an instance of `BeamSearchInitialization` where the given parameter for the *beam width* equals one. Thus preventing code duplication.
However, this also mean that the code still uses `PriorityQueues` to contain nodes reached by the beam even though only one will ever be taken into account.

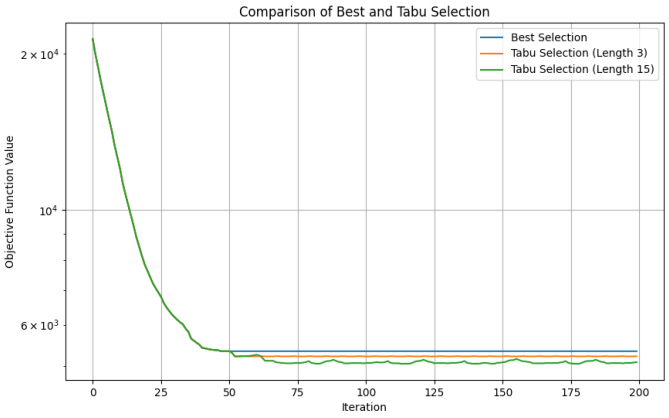
2. Justify the time complexity of your implementation.

The beam search can be seen as a tree search that searches through all cities of the tour, amounting to n. For each city, only a handful possible successors are considered depending on the $beam\_width$.
Since a `PriorityQueue` is used to manage the beam with next cities to take into account, the time complexity to insert element is of $O(log(k))$ where $k = n \times beam\_width$. Pulling out an element is $O(1)$.
Finally, each candidate loops over a maximum of $O(n)$ cities it can reach before considering adding them to the beam (effectively less that n since already visited cities are not considered).

Therefore, the overall complexity is : $O(n^2 \times beam\_width \times log(n \times beam\_width))$.

Exercise 3 Improving the solution

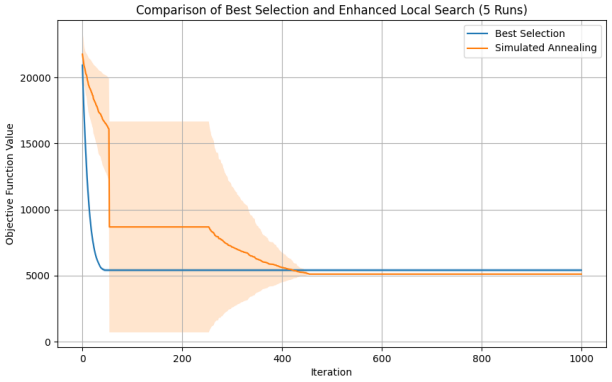
1. Using RandomInitialization, compare Tabu Selection with Best Selection by plotting the objective function of the current candidate at each iteration over 200 iterations on the instance data/TSP/gr48.xml. Overlay the curves for tabu lengths of 3 and 15.



We can definitely see on the plot that *Best Selection* is very efficient but much prone to be stuck in local minima, whereas *Tabu Search* has a mechanism to escape some. As expected, the longer the *Tabu Length*, the more local minima it can escape, thus finding better solutions.

2. Using a local search with random initialization and BestSelection within a 2Opt neighborhood as a baseline, implement an enhanced local search incorporating any components of your choice (e.g., selection, initialization, meta-heuristics). Explain your choices, and compare both approaches by showing the progression of the best solution found over the iterations for each method. Since the process may involve pseudorandom number generators, it is good practice to represent the standard deviation around these curves.

The solution implemented is a *Simulated Annealing (SA)* algorithm based on the *Best Selection (BS)*. It allows to pick worse solution randomly based on a threshold which changes overtime (high initial temperature which decreases over iterations). Moreover, to improve it even further, the algorithm has a heuristic which allows to go back to already visited cities and start over from there.



To prevent it to start completely over, the algorithm has a memory which allows it to revert to the best known solution after too many unsuccessful iterations.

SA has slight improvement over the *BS* but is difficult to fine tuned since it has a lot of parameters that can have a huge impact. It would definitely benefit from having other start than a random initialization.
It is also much slower to converge, even when tuning parameter focusing only on that aspect, and is much less stable in its current implementation compared to the simple *BS*.