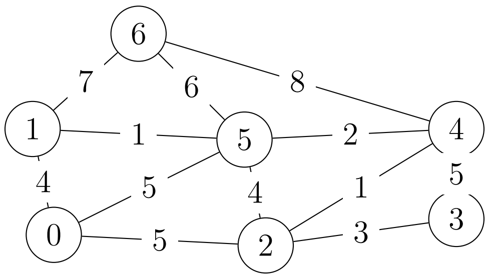
First & last name	Cyril Bousmar
NOMA UCLouvain	63401800

LINFO2266: Advanced Algorithms for Optimization

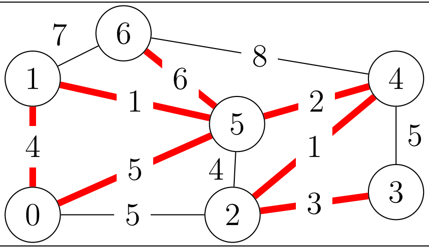
Project 2: Branch and Bound

Exercise 1 Modeling the Traveling Salesman Problem



1. What is the one-tree and its cost on this example.

Lower bound = 22



2. Explain the internal state of your implementation. How does your successor function works?

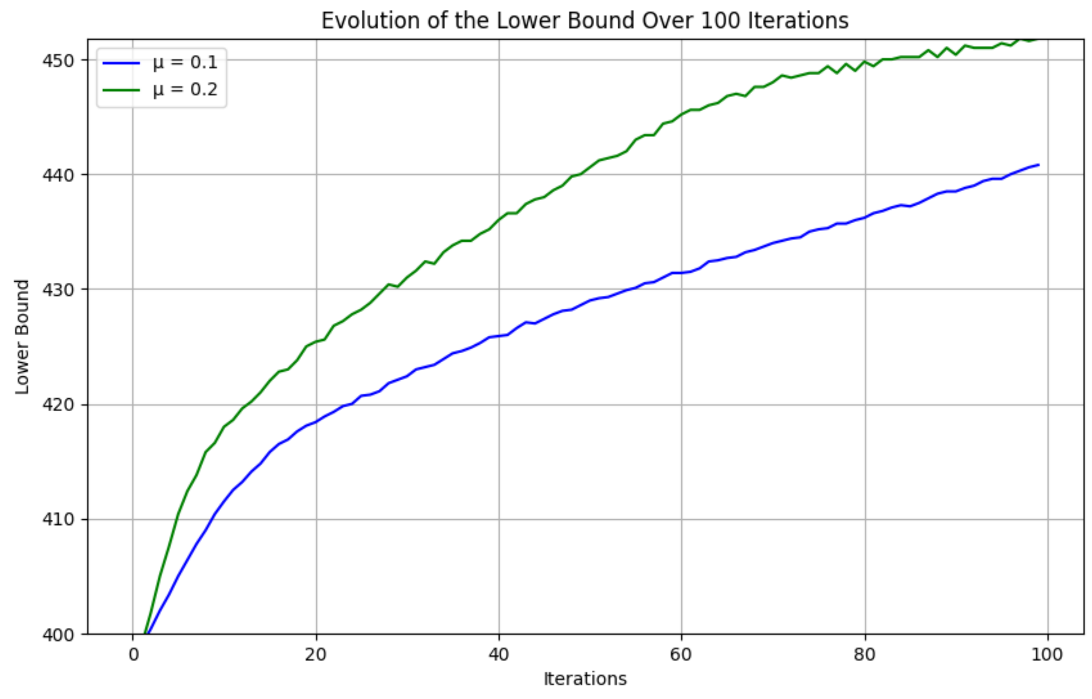
Called a TSPNode, it contains a reference to its parent, its city number, the total cost to reach it, the global lower bound, the distance matrix from the instance, a BitSet to keep track of cities yet to explore, and its depth in the graph search.

The successor method returns a list of TSPNode called children which represents all the cities reachable and yet not visited.
Therefore, it first needs to clone its BitSet for each child and clear the child within, marking it as visited. Secondly, it creates a new TSPNode for each child, each of which receiving the current node as parent, having their depth increased by one compared to their parent one, and having the total cost to reach them equal to the sum of the one from their parent and the cost of the path, recorded in the distance matrix, from parent to them.

Exercise 2 Held and Karp lower-bound

Assume the update-rule of the Langrangian multipliers is $\pi_i \leftarrow \pi_i + \mu(2 - deg(i)), \forall_i$ where μ is a fixed small constant value.

1. For the instance `instance_30_0.xml` make a plot of the evolution of the lower-bound along the 100 first iterations of the gradient descent for $\mu = 0.1, \mu = 0.2$. The x-axis should have the iterations, and the y-axis the value of the lower-bound. The y axis should start at 400.



2. What do you observe, conclude? How can this plot help you to tune the parameters of the Held-Karp lower-bound algorithm.

It appears that having a slightly bigger μ at the beginning is beneficial as it would converge faster to the optimal solution.
Nevertheless, we can see that it becomes more and more unstable, so even though it does not reach a plateau after 100 iterations, it seams better to reduce μ's value reaching that point and going forward.
Ultimately, having a variable μ which value decreases over iterations seams the best approach.

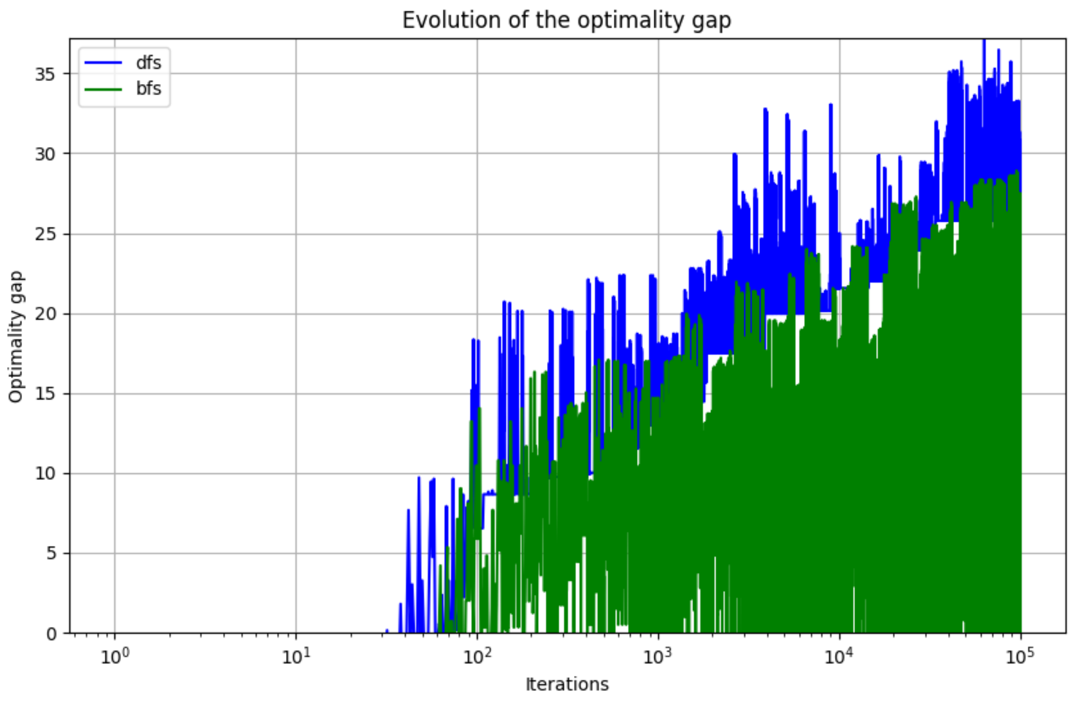
Exercise 3 Optimality Gap

1. Give the formula of the optimality gap (in %). Explain how you can compute it.

Since the problem is a minimization, instead of computing the lower bound with the maximum value of open nodes, it is the minimal open node value that is selected.

$$\textit{optimality gap} = \frac{\textit{upper bound} - \textit{lower bound}}{\textit{upper bound}} \times 100$$

2. For the instance `instance_30_0.xml`, create a plot that shows the evolution of the optimality gap (y-axis) over the iterations (x-axis) during the branch-and-bound process. The plot should include two lines: One line representing the depth-first search strategy and one line representing the best-first search strategy. Make sure to include a legend that distinguishes between the two search strategies.


Evolution of the optimality gap
dfs
bfs
Optimality gap
Iterations
10^0
10^1
10^2
10^3
10^4
10^5