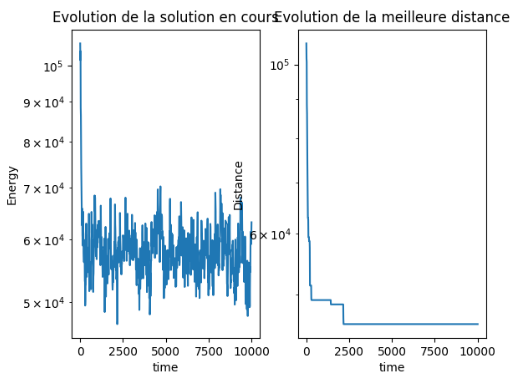
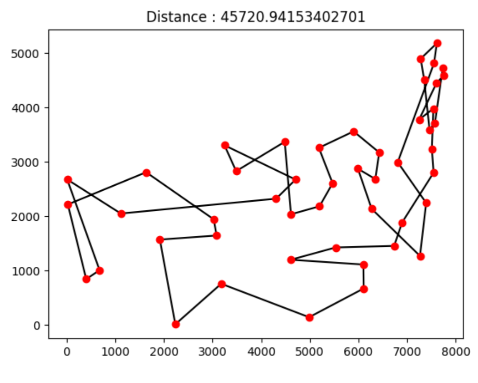
Taboo Search

Course: AI and Optimization, ADEO-M2
Author: Alvise de' Faveri Tron

Exercise 1

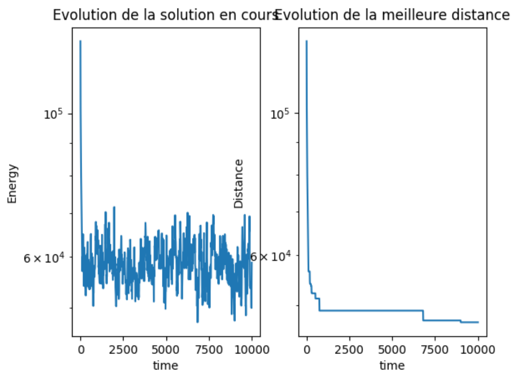
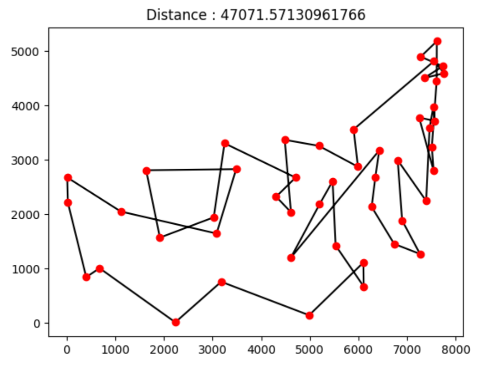
a. Set a size to 1, interpret the behavior.

With `size=1` the algorithm has no *memory*, meaning that it will not remember the solution it already explored and will select them more than once.

Solution	Route
	

b. Set a size to 100, interpret the behavior.

With `size=100` the algorithm will avoid selecting again solutions that have already been selected in the previous 100 iteration. This enables a more efficient exploration of the solution space but can be limiting, since the algorithm can get "stuck" in a certain region if many of the neighbors have already been explored.

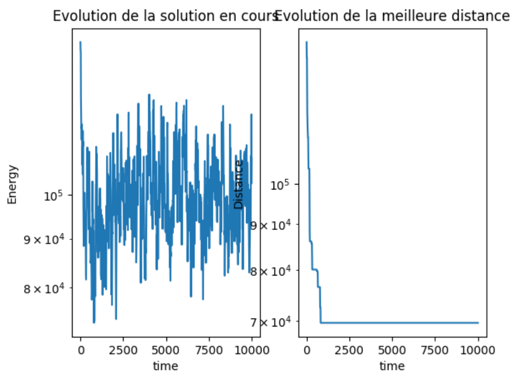
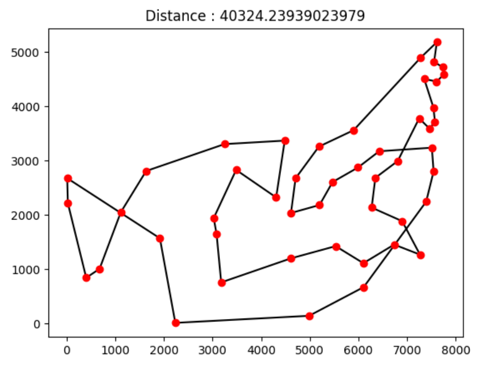
Solution	Route
	

Exercise 2

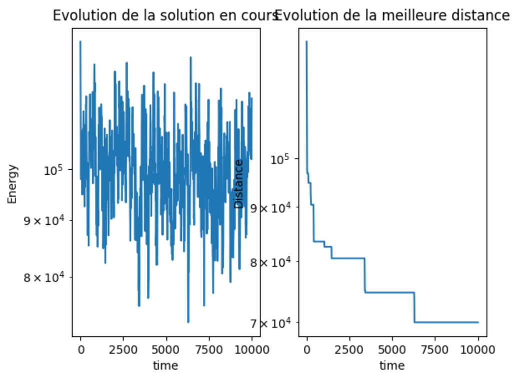
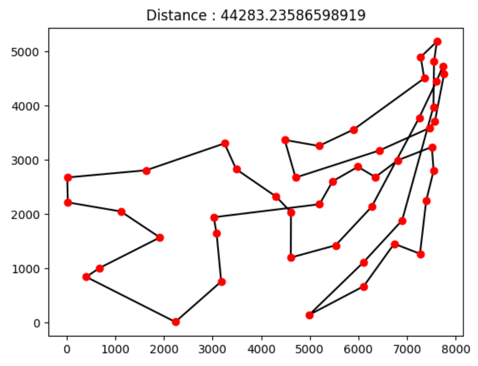
a. Use the permutation 2-opt created in the previous work.

```
def two_opt(route):
      best = route
      improved = True
      while improved:
            improved = False
            for i in range(0, len(route)):
                  for j in range(0, len(route)):
                        new_route = permuteTwo(route, i, j)
                        if evalue(coords,new_route) < evalue(coords,best):
                              best = new_route
                              improved = True
                              draw(best,  evalue(coords,best), x, y)
            route = best
      return best
```

b. Set a size to 1, interpret the behavior.

Solution	Route
	

c. Set a size to 100, interpret the behavior.

Solution	Route
	

d. Modify the algorithm to include the aspiration criterion. We will arbitrarily take the "magic" size of 7.

```
def bestNeighorWithAspiration(path, nbNeigh, ltaboo):
    global bestV, bestDist
    nb = 1
    # case of the first neighbor
    i = scp.random.random_integers(0,N-1)
    j = i
    while (i==j):
        j = scp.random.random_integers(0,N-1)

    bestV = permuteTwo(path,i,j)
    bestDist = evalue(coords,bestV)

    while nb <= nbNeigh:
        i = scp.random.random_integers(0,N-1)
        j = i
        while (i==j):
            j = scp.random.random_integers(0,N-1)

        Neigh = permuteTwo(path,i,j)

        # if it's better, select it
        d = evalue(coords,Neigh)
        if (d < bestDist):
            bestV = Neigh
            bestDist = d
            nb += 1

        # else, check last 7 entries in ltaboo
        else:
            accept = True
            for n in ltaboo[len(ltaboo) - 7:]:
                if(d == evalue(coords, n)):
                    accept = False
            if accept:
                bestV = Neigh
                bestDist = d
                nb += 1

    return (bestV,bestDist)
```

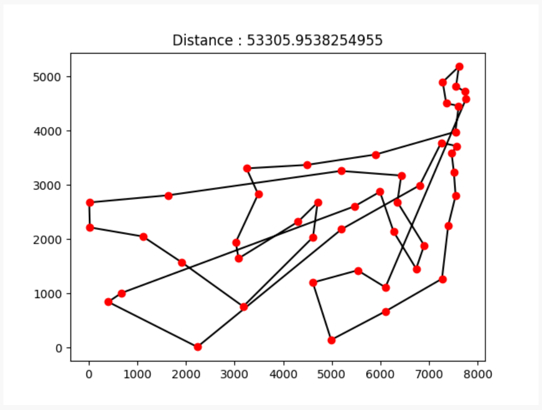
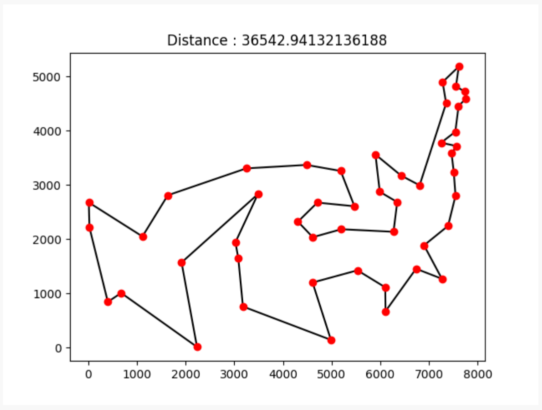
Exercise 3

a. Compare the simulated annealing and the taboo search (the quality of the solution and the number of evaluations).

The number of iterations is set to the same value for both algorithms. The Taboo Search both took less time and got the best solution, but used all the 10000 iterations given. On the other hand, a Simulated Annealing algorithm using exponential decay and very high initial temperature took more time and got a worst result. More fine-tuning of the parameters and many repetitions would be needed to confirm this result.

Annealing	**Taboo**

T0 = 10e4
Tmin = 1e-3
tau = 100
Alpha = 0.99
Step = 7
IterMax = 10000
	ntaboo = 100
	nbNeigh = 10
	iterMax = 10000

	
iterations = 1843	iterations = 10000