Lab Practice for Local Search

Complete one of the following tasks (Soduku, N-Queens or TSP) in `Simulated_Annealing.ipynb` and `Genetic_Algorithm.ipynb`. Report your implementation and results (solutions and figures).

Detaild walk-throughs and starter codes are provided in the notebooks. Some example outputs are presented in the notebook to give an idea of what should be expected.

The point of using notebooks is to be interactive so that you can play with the code and get immediate feedback. You can of course convert them to python scripts if you like.

1. Simulated Annealing

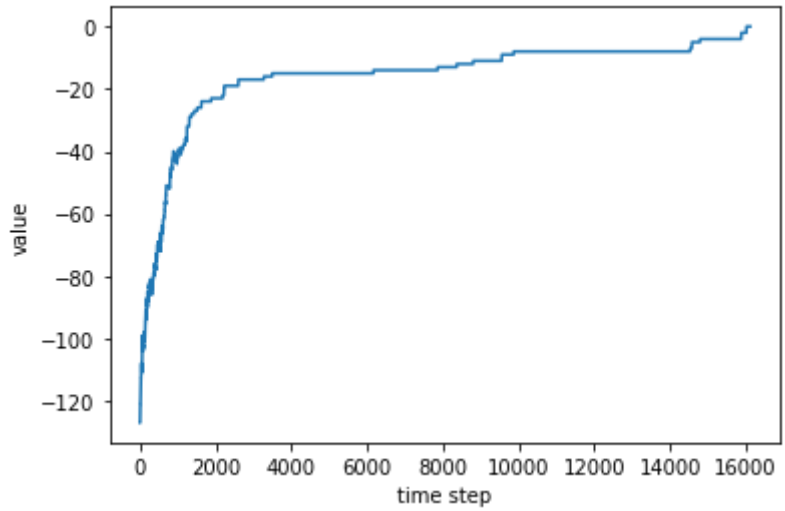
Soduku (easy)

Solve the Soduku problem with Simulated Annealing. Design your own algorithm or refer to Metaheuristics_can_solve_Sudoku_puzzles for the search operator.

- running the algorithm, you are supposed to see something like

```
step 0: T=1.0, current_value=-127
step 200: T=0.8186488294786356, current_value=-88
step 400: T=0.6701859060067401, current_value=-77
step 600: T=0.5486469074854967, current_value=-65
step 800: T=0.4491491486100751, current_value=-47
...
step 15800: T=1.3636847081769104e-07, current_value=-4
step 16000: T=1.1163788901269425e-07, current_value=-2
step 16111: T=9.990352422722691e-08, current_value=0
```

- and a curve



2. Genetic Algorithm

0. Example Problem: Phrase Generation (does not count as practice)

Generate a target phrase (e.g., "genetic algorithm") from an initial polupation of random strings. Assume the length of the target is known.

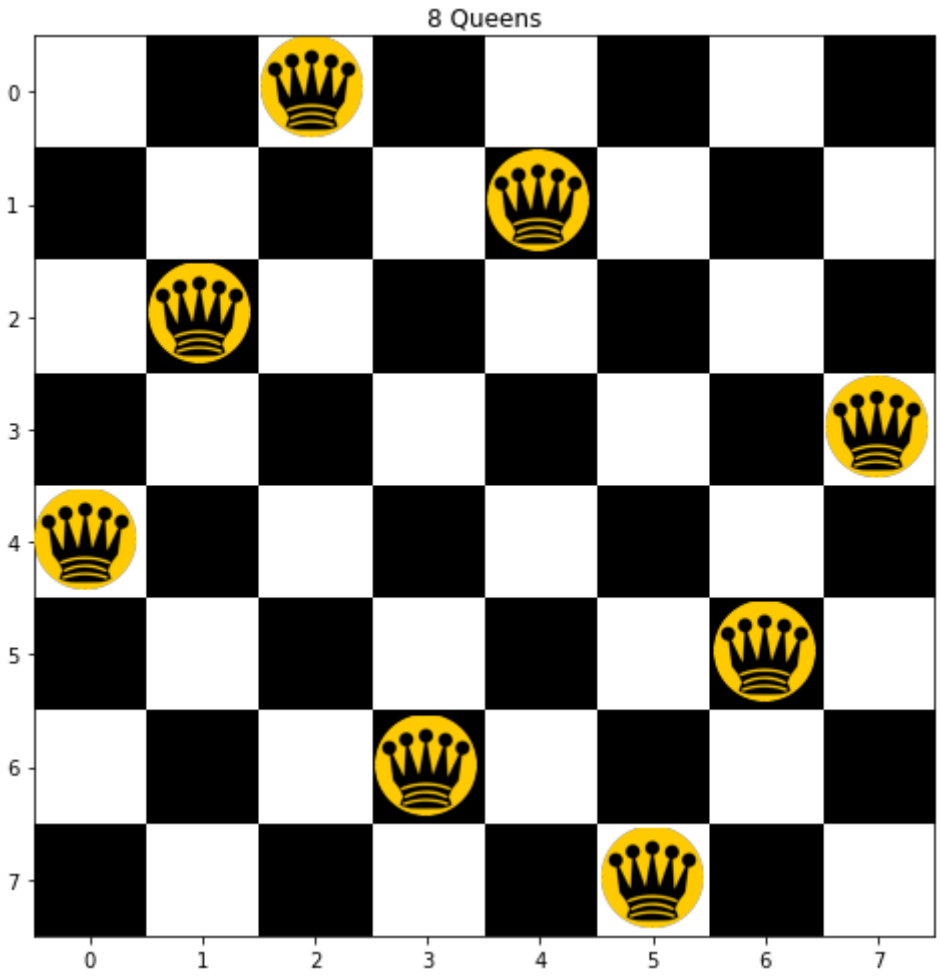
- result to expect:

```
Generation: 0/1000, Best: 2ihkt57fwlmo7dgIubg2uMDWlPQx4,    Fitness=4
Generation: 100/1000,   Best: GenetDc Algo9ithmjlQ 1f8HU924,    Fitness=20
Generation: 200/1000,   Best: Genetic Algo9ithmjlQ 1f8f0924,    Fitness=22
...
Generation: 800/1000,   Best: Genetic Algorithm by 11810424,    Fitness=29
Generation: 900/1000,   Best: Genetic Algorithm by 11810424,    Fitness=29
'Genetic Algorithm by 11810424'
```

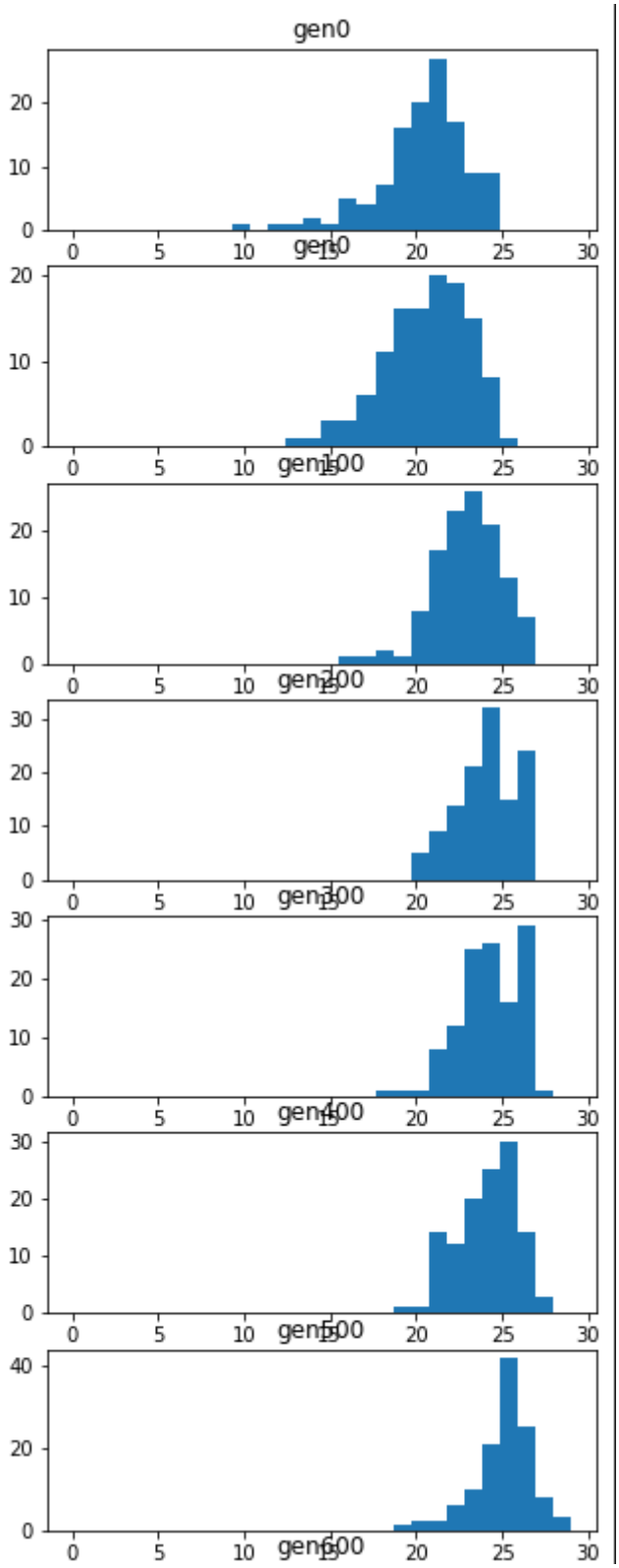
1. N-Queens Problem (easy)

Solve the N-Queens problem with some slight modification to the example problem.

- result to expect:



- you can also see the evolution of the whole population (I didn't know why the text spacing is like this):



2. TSP Problem (hard)

Use GA to solve the traveling salesman probem (TSP). Optional, not required.

For Your Interest

GA is a specific instance of Evolutionary Algorithms (EA). One of the main advantages of these "population-based" methods is that having a population rather than a single solution naturally encourages **diversity** and **robustness**.

They can be applied to

- Multi-Objective Optimisation. When the task has multiple objectives, instead of specifying a weighted combination of them, evolutionary approaches can be used to get a population of solutions each featuring in some aspects.
- Neural Network Architecture Search. A rising trend to automate the designation of deep neural network models.

Also on engineering problems:

- Topology Optimisation. Example of optimizing a building (maybe it is not "evolutionary" in our sense but the algorithm is named so anyway).