Local Search – Genetic Algorithm

Lab 4

Genetic Algorithm Summary

- Start with randomly generated population of valid candidate individuals
- For generation in (0 . . . N)
  - Select 2 individuals (parents) randomly from population for reproduction based on fitness of each individual. The fitter an individual, the more likely it is selected to reproduce.
  - Produce a new individual (child) by combining random parts of the 2 selected individuals (crossover).
  - Mutate a random number of new individuals to allow exploration of other possible individuals.
  - Include the new individuals into the current population
- Best solution is the fittest of all population individuals

Exercise

3-bit binary number

- A trivial problem is to determine the greatest 3-bit binary number.

- **Representation:** An individual is represented by an integer. The 3 bits could be represented by a set such as: (0,0,1) etc.
- **Initial population:** Some random set of individuals, for example: [(1,0,0),(0,1,0),(0,1,0),(0,0,0)]
- **Fitness function:** Determines fitness of each individual; returns the integer value of the 3-bit number, 0 to 7.
- **Selection:** Individuals of each generation are randomly selected using a fitness ratio → their percentage contribution to the total fitness of the population

3-bit binary number

- For the first generation:

Genes	Fitness	Fitness ratio
(1,0,0)	4	50%
(0,1,0)	2	25%
(0,1,0)	2	25%
(0,0,0)	0	0%

- Total fitness is 4+2+2+0 = 8. (1,0,0) selected with 50% probability, (0,1,0) with 25%, (0,1,0) also with 25% and (0,0,0) with 0%.
- Randomly selected pairs using fitness ratio: [(1,0,0),(0,1,0)],[(1,0,0),(1,0,0)],[(0,1,0),(0,1,0)],[(0,1,0),(1,0,0)]

3-bit binary number

- **Reproduce:** combine selected individual pairs at some random point using crossover. The first individual is copied up to the crossover point, then the second individual is copied from there on.

- [(1,0,0),(0,1,0)] combined at bit 1 producing (1,1,0)
- [(1,0,0),(1,0,0)] combined at bit 0 producing (1,0,0)
- [(0,1,0),(0,1,0)] combined at bit 2 producing (0,1,0)
- [(0,1,0),(1,0,0)] combined at bit 1 producing (0,0,0)

- Note that the higher order bits contribute more to our definition of fitness; combining a high and low fitness individual could produce a lower fitness results

3-bit binary number

- **Stagnation:** it is important that less fit individuals occasionally are selected, though all individuals are selected at rate proportional to their fitness. This helps ensure populations do not stagnate by constantly selecting from the same parent individuals. For example, if only the most fit individual, (1,0,0), were selected the optimal individual of (1,1,1) would never be found.

- **Mutation:** Because the least significant bit is 0 throughout the population, no individual with least significant bit 1 can ever be produced using crossover alone. Change a random element in an individual representation at some specified probability. For example:
  (0,1,0) mutated to (0,1,1)

3-bit binary number

- **New population:** [(1,1,0),(1,0,0),(0,1,1),(0,0,0)] has total fitness of 6 + 4 + 3 + 0 = 13

Genes	Fitness	Fitness ratio
(1,1,0)	6	46%
(1,0,0)	4	31%
(0,1,1)	3	23%
(0,0,0)	0	0%

- **Terminate generation or fit enough:** Terminate if individual fit enough or number of generations reached.

Exercise - 3-bit binary number

Must be submitted

- Consider ga_template.py

- Complete the following functions
  - fintess_function
  - random_selection
  - reproduce
  - mutate

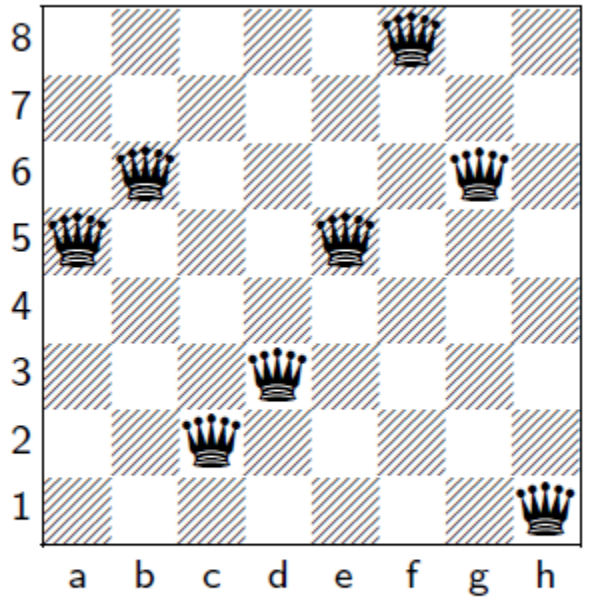
Homework

Homework

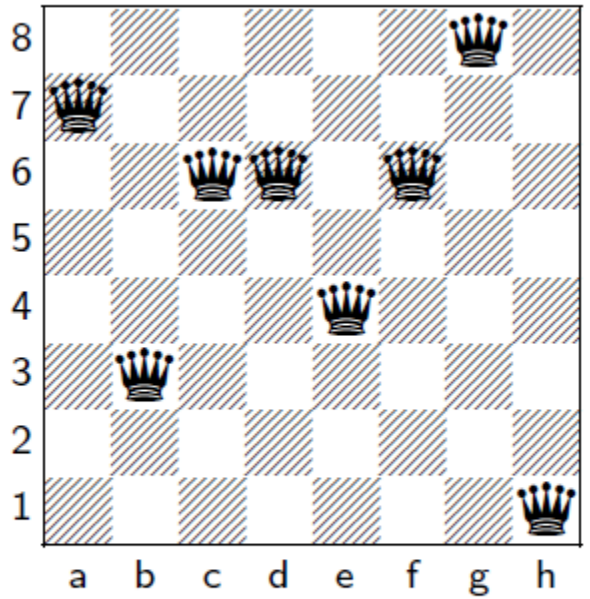
Must be submitted

Modify your GA program given the following problem and code (queens_fitness.py):
Place *n*-queens on a chessboard in non-conflicting positions.
Representation: A solution is represented by a list of integers, the rows of each queen. Each queen has its own column. (3,4,2,6,1,7,8,5) represents a queen in a3, one in b4, etc.
Remember in Python list indices start from 0.
Fitness function: returns the integer value of the number of non-conflicting queen pairs; the maximum for n queens is $\frac{n(n-1)}{2}$.
Fitness function (alternative): returns the integer value of the number of conflicting queen pairs. Minimize instead of maximizing.
Selection: Roulette selection.
Reproduce: Randomly select a crossover point to combine the two parents. Two new children are produced from the crossover. The effect is to maintain the fitness of each parent in the new population; keeping only one child occasionally loses fitness.

Homework – Representation Example

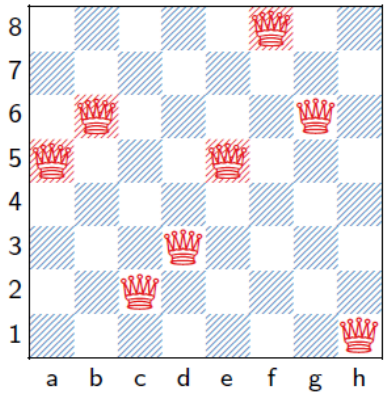


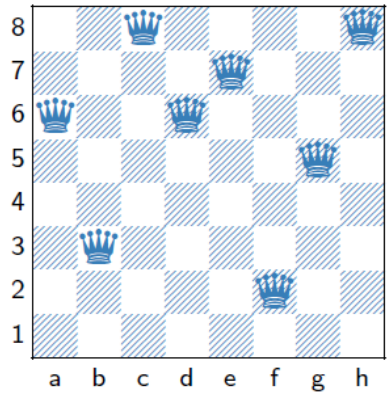
$(5, 6, 2, 3, 5, 8, 6, 1)$

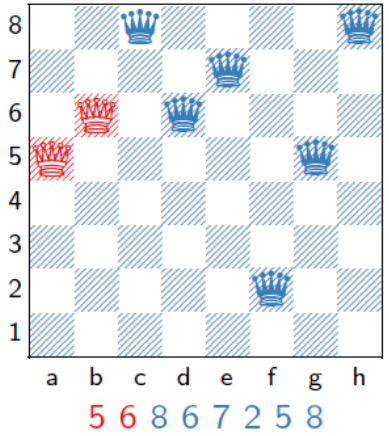


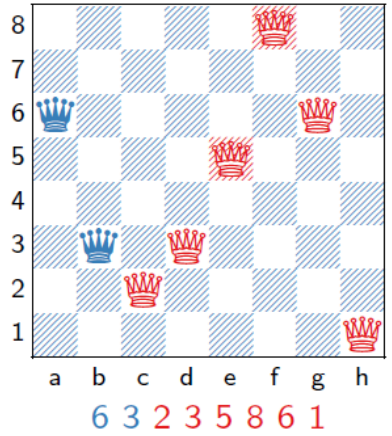
$(7, 3, 6, 6, 4, 6, 8, 1)$

Homework – Crossover Example

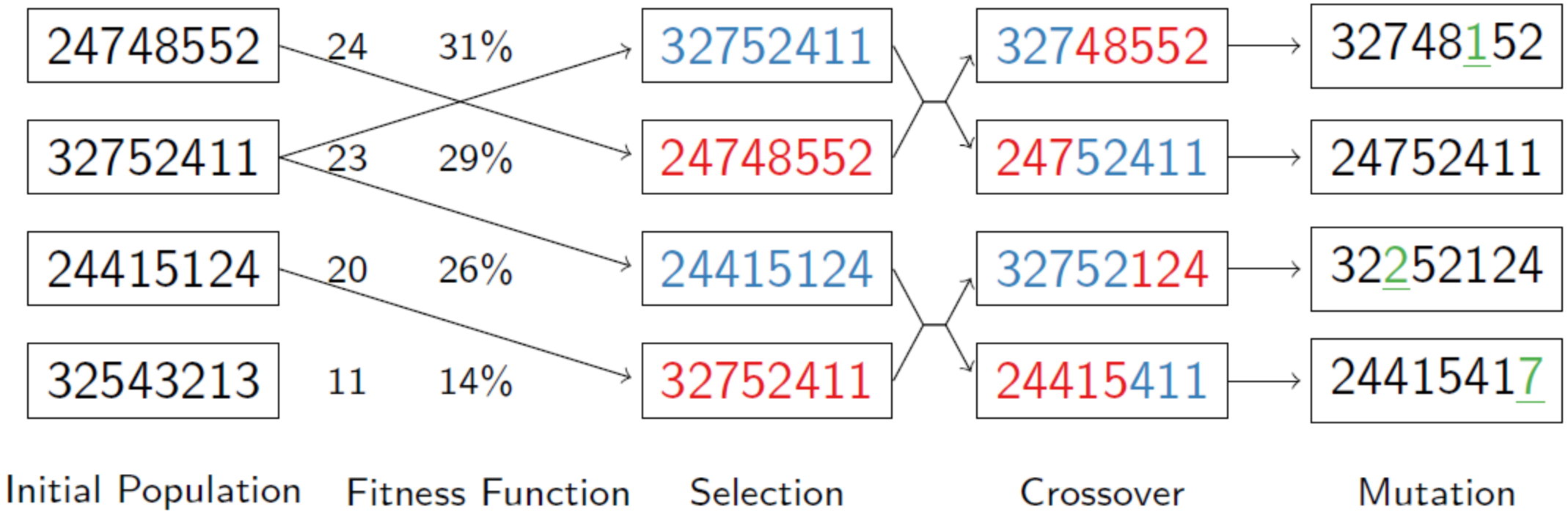








Homework – Full Example



Homework - Hint

Use the queens_fitness module for the homework to calculate the fitness levels.

from queens_fitness import *