Contents

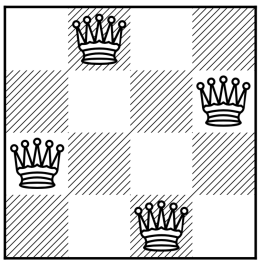
1 Learning Goals

By the end of the exercise, you should be able to

- Formulate a real-world problem as a local search problem.
- Design multiple neighbour relations for a given problem.
- Trace the execution of the hill climbing algorithm with a given neighbour relation.
- Trace the execution of the simulated annealing algorithm with a given neighbour relation.

2 The 4-Queens Problem

The 4-queens problem consists of a 4 x 4 chessboard with 4 queens. The goal is to place the 4 queens on the chessboard such that no two queens can attack each other. Each queen attacks anything in the same row, in the same column, or in the same diagonal. The image below gives one solution to this problem.



2.1 Local Search Formulation

Recall that we've formulated the 4-queens problem as a local search problem.

- State: 4 queens on the board. One queen per column.
  - Variables: x_0, x_1, x_2, x_3 where x_i is the row position of the queen in column i. Assume that there is one queen per column.
  - Domain for each variable: $x_i \in \{0, 1, 2, 3\}, \forall i$.
- Initial state: a random state.
- Goal state: 4 queens on the board. No pair of queens are attacking each other.
- Neighbour relation:
  - Version A: Move a single queen to another square in the same column.
  - Version B: Swap the row positions of two queens.
- Cost function: The number of pairs of queens attacking each other, directly or indirectly.

2.2 Hill Climbing

Algorithm 1 Hill Climbing

```
current ← a random state
while true do
    next ← get-best-neighbour(current)
    if cost(current) ≤ cost(next) then
        break
    end if
    current ← next
end while
return current
```

2.3 Simulated Annealing

Algorithm 2 Simulated Annealing

```
current ← initial state
T ← a large positive value
while T > 0 do
    next ← a random neighbour of current
    ΔE ← current.cost - next.cost
    if ΔE > 0 then
        current ← next
    else
        current ← next with probablity p = e^(ΔE/T)
    end if
    decrease T
end while
return current
```

3 Practice Questions

3.1 Hill Climbing

For questions 1 to 4, consider version B of the neighbour relation: swap the row positions of two queens.

Question 1:

How many neighbours are there for a state?

Question 2:

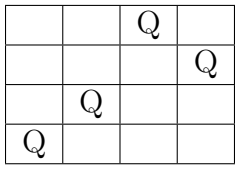
Start with the initial state $x_0 = 3, x_1 = 1, x_2 = 2, x_3 = 0$. Show the steps of executing the hill climbing algorithm until it terminates.

			Q
	Q		
		Q	
Q			

If multiple neighbours have the same cost, choose the neighbour where the pair of queens swapped has the smallest subscript/column number. For example, when we can swap either (x_0, x_4) or (x_2, x_3), we will swap (x_0, x_4). When we can swap either (x_2, x_3) or (x_2, x_4) we will swap (x_2, x_3).

Question 3:

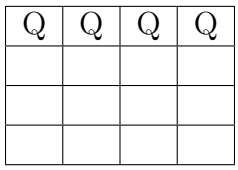
Suppose that we are executing the hill climbing algorithm. Let the current state be $x_0 = 3, x_1 = 2, x_2 = 0, x_3 = 1$.



What is the cost of the current state? Is this state a local optimum? If not, give an example of a neighbour with a lower cost. If yes, is this state a global optimum?

Question 4:

Suppose that we are executing the hill climbing algorithm. Let the current state be $x_0 = 0, x_1 = 0, x_2 = 0, x_3 = 0$.



What is the cost of the current state? Is this state a local optimum? If no, give an example of a neighbour with a lower cost. If yes, is this state a global optimum?

For questions 5 and 6, consider version A of the neighbour relation: move any queen to another square in the same column.

Question 5:

How many neighbours are there for a state?

Question 6:

Suppose that we are executing the hill climbing algorithm. Let the current state be $x_0 = 3, x_1 = 2, x_2 = 0, x_3 = 1$.

		Q	
			Q
	Q		
Q			

What is the cost of the current state? Is this state a local optimum? If not, give an example of a neighbour with a lower cost. If yes, is this state a global optimum?

3.2 Simulated Annealing

Suppose that we are running the simulated annealing algorithm. In the following, fill in the values of ΔE and the probabilities of moving to the neighbour under different temperatures. For probabilities, keep three significant digits.

current.cost	next.cost	ΔE	p(T=100)	p(T=50)	p(T=10)
50	40				
50	100				
50	200				

Based on your calculations, summarize your observations below.

- What is the probability of moving to a neighbour with a lower cost (i.e. the neighbour is better than the current state)?

- Consider a neighbour with a higher cost (i.e. the neighbour is worse than the current state). As the temperature decreases, how does such a worse neighbour change?

- Consider a neighbour with a higher cost (i.e. the neighbour is worse than the current state). As the difference between the neighbour's cost and the current state's cost increases (i.e. as the neighbour becomes worse), how does the probability of moving to such a worse neighbour change?

3.3 An example of the genetic algorithm

- Consider the 8-queens problem. Represent each state as a string of 8 numbers ranging from 1 to 8. Each number denotes the row position of the queen in the respective column.
- For our example, let's start with a population of four individuals:

  24748552, 32752411, 24415124, 32543213
- Calculate the fitness of each individual to be the number of pairs of queens not attacking each other.

  The fitness of the individuals are: 24, 23, 20, and 11.
- If the fitness of an individual is large enough, then the algorithm terminates and returns that individual.

  Otherwise, we continue to produce a new population.
- Calculate the fitness of each individual relative to the total fitness of the population.

  The relative fitness of each individual is: 0.31, 0.29, 0.26, and 0.14.

  $0.31 = 24/(24 + 23 + 20 + 11)$
- For each individual in the new population, choose two parents from the current population to reproduce. The probability of choosing an individual as a parent is proportional to the relative fitness of the individual.

  For example, we have chosen four pairs of parents below.

  24748552 and 32752411

  32752411 and 24415124

  24748552 and 32543213

  24415124 and 32752411
- Each pair of parents produces a child by crossing over at a random point inside the string of number.

  For example, suppose that the following cross-over points are chosen and the child is produced by taking the first part of the first parent and the second part of the second parent.

  24748|552 and 32752|411 produces the child 24748411.

  32|752411 and 24|415124 produces the child 32415124.

  2474|8552 and 3254|3213 produces the child 24743213.

  24|415124 and 32|752411 produces the child 24752411.
- Each child mutates with a small independent probability. For our example, let's assume that one random digit inside the child will mutate with some probability. After possible mutation, the resulting children are as follows.

For the first child, the last digit mutates to 5, resulting in the individual 24748415.

For the second child, no mutation occurs and the resulting individual is 32415124.

For the third child, the first digit mutates to 5, resulting in the inndividual 54743213.

For the fourth child, the fourth digit mutates to 1, resulting in the individual 24712411.

- The new population consists of the four new children 24748415, 32415124, 54743213, and 24712411. Go back to the first step and repeat.