The 8-queens problem

Jesus Armando Beltran Verdugo
Spring 2025

1. Introduction

1.1. **Problem Statement:** How can eight queens be placed on an 8x8 chessboard so that no two attacks each other?

The 8-queen problem involves placing eight queens on a chessboard so no queen attacks another. A queen can attack any piece in the same row, column, or diagonal. Figure 1 shows a queen configuration that is nearly an optimal solution, apart from the two queens in the fourth and seventh columns, which attack each other along the diagonal.

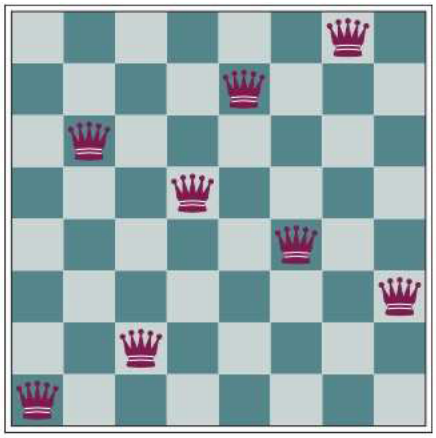

Figure 1. Example of Candidate Solution with One Queen Conflict: 8-Queens Problem [1].

1.2. Existing approaches

Numerous algorithms and heuristics have been devised to address the N-queen problem, operating constructively or incrementally. These algorithms include:

1. Backtracking algorithms
2. Local search methods (e.g., hill climbing, simulated annealing)
3. Genetic algorithms
4. Constraint programming techniques

These methods are designed to effectively search the solution space and identify configurations that meet all problem constraints. By treating the N-Queens Problem as a combinatorial optimization issue, various optimization techniques can be used, helping

advance theoretical computer science and practical applications like scheduling and resource allocation.

2. Evolutionary Algorithm Approach

2.1. Overview

In this section, I summarize what we covered in the lecture over the past two weeks, focusing on the concept of an evolutionary algorithm (EA).

Various evolutionary algorithms share a familiar concept. Within environments with limited resources, competition drives natural selection, boosting population fitness. By aiming to maximize a quality function, we randomly generate candidate solutions. Applying the function as a fitness measure, higher values indicate better candidates. These are selected for the next generation through recombination and/or mutation. Recombination involves two or more parents creating new children, while mutation alters one candidate to produce another. New candidates' fitness is then evaluated, and they compete for survival with existing ones. This cycle repeats until a sufficiently high-quality solution is found or a computational limit is reached.

It is important to note that two main forces form the basis of evolutionary systems:

• **Variation operators** (recombination and mutation) create the necessary diversity within the population, facilitating novelty.

• **Selection acts** as a force increasing the mean quality of solutions in the population.

Applying variation and selection together typically enhances fitness values in successive populations. **One might interpret this process as evolution optimizing (or 'approximating') the fitness function by progressively nearing optimal values over time**. Alternatively, evolution can be viewed as an adaptation process. In this view, fitness is not considered an objective function to optimize but rather an indicator of environmental requirements. Better alignment with these requirements means increased viability, shown by a higher number of offspring. Consequently, the evolutionary process leads to a population that becomes progressively better adapted to its environment [2].

2.2. Key Element of an EA's Functioning

Evolutionary algorithms have some general properties that explain how they work. To demonstrate how an evolutionary algorithm (EA) typically works, let's consider maximizing a one-dimensional objective function. Figure 3.4 displays three stages of the evolutionary

search, depicting the distribution of individuals at the beginning, midway, and end of the evolution.

1. Immediately following initialization, the individuals are dispersed randomly across the entire search space (see Fig. 3.4, left).
2. After a few generations, the distribution shifts: due to selection and variation, the population leaves low-fitness areas and begins ascending the hills (Fig. 3.4, middle).

3. Near the end of the search, if the termination condition is set correctly, the population converges around a few peaks, some possibly suboptimal.

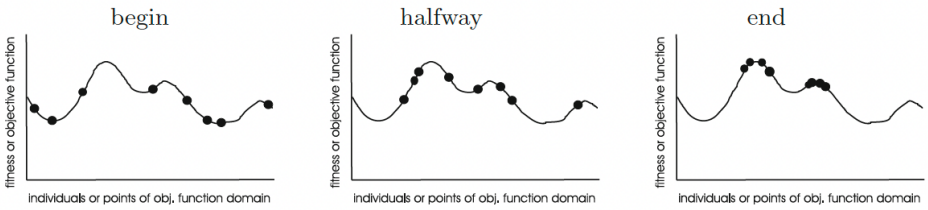

Fig. 3.4. Typical progress of an EA illustrated in terms of population distribution. For each point x in the search space y shows the corresponding fitness value.

Exploration involves generating new candidates in uncharted areas of the search space, whereas **exploitation** focuses on intensively searching near known optimal solutions. Evolutionary search processes are frequently described as balancing exploration and exploitation. Excessive exploration can result in inefficient searching, while excessive exploitation risks premature convergence.

Premature convergence is the well-known effect of losing population diversity too quickly, and getting trapped in a local optimum [2].

N-Queen Problem Statement: Constrained Optimization Approach

Given the number of queens n, find a configuration for the queens on an nxn chessboard such that **every row and column contains exactly one queen** while maximizing the number of non-attacking pairs of queens

Statement: The formal definition of the N-queen Constrained optimization problem is:

$$\max_{Q} f(Q) = \frac{n(n-2)}{2} - \mathrm{A(Q)}$$

$subject\ to$
$i \neq j\ for\ all\ Q_i, Q_j$
$Q_i \neq Q_j\ for\ all\ i \neq j$

Where A(Q) is the number of attacking pairs of the queens. For a state Q. Here $\mathrm{Q} = [\mathrm{Q_1}, \mathrm{Q_2}, \ldots, \mathrm{Q_n}]^{\mathrm{T}}$ defines a specific queen configuration on a nxn chessboard, where each Q_i denotes the column position of the queen in a row i.

3. **TRY IT YOURSELF:** Evolutionary Algorithm to Solve the 8-queen Problem (40 points)

Write a program that solves the 8-queen problem using the evolutionary scheme below. You can start using the code elaborated in the Jupyter notebook we covered in week 2.

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

3.1. Representation

To design an EA to search P, we need to define a representation of phenotype form P, P is the set of all such board configurations, which specify the positions of all eight queens.

A genotype, or chromosome, is a permutation of the numbers $1, \ldots, 8$, and a given $\mathrm{g} = < \mathrm{i1}, \ldots, \mathrm{i8} >$ denotes the (unique) board configuration, where the nth column contains exactly one queen placed on the $\mathrm{i_n}$ th row.

3.2. Fitness Function/ Evaluate:

The quality of a solution is evaluated.

The quality $q(p)$ of any phenotype $p \in P$ can be quantified by the number of checking queen pairs. The lower this measure, the better a phenotype (i.e., board configuration), and a zero value, q(p) = 0, indicates a good solution.

3.3. Parent Selection

The methods used to select individuals for reproduction. We will do this by **choosing five individuals randomly from the population** and taking the best two as parents.

3.4. Crossover:

The crossover operators are used to combine genetic material from two parents.

1. Select a random position, the crossover point, $i \in \{1, \ldots, 7\}$
2. Cut both parents into two segments at this position
3. Copy the first segment of parent 1 into child 1 and the first segment of parent 2 into child 2
4. Scan parent 2 from left to right and fill the second segment of child 1 with values from parent 2, skipping those that it already contains
5. Do the same for parent 1 and child 2

Fig. 3.3. 'Cut-and-crossfill' crossover

3.5. Mutation

The mutation operators are used to introduce diversity into the population. We can use an operator that randomly selects two positions in each chromosome and swaps the value found in those positions.

Example

1. Randomly select two positions (e.g., g_3, g_5)

g=<5,1,**3**,2,**6**,7,8,4>

2. Swap the value found in those positions g_3 =3 and g_5=6.

g=<5,1,**6**,2,**3**,7,8,4>

3.6. Survive Selection: Population Update Mechanism

We will choose a simple scheme for managing the population. In each evolutionary cycle, we will select two parents, producing two children, and the new population of size n will contain the best n of the resulting n + 2 individuals (the old population plus the two new ones).

Survivor selection checks which old individuals should be deleted to make place for the new ones - provided the new ones are better. We define a replacement strategy. The strategy is to merge the population and offspring, then rank them according to fitness and delete the worst two.

3.7. Termination condition

The termination condition is satisfied if we find a solution or when 10,000 fitness evaluations have elapsed, whichever happens sooner.

4. Experimental Setup (40 points)

After constructing your Evolutionary Algorithm, evaluating its performance on the 8-queens problem is important.

4.1 EA parameters for the eight-queen problem

Here, we outline the EA parameters, including population size, number of generations, recombination probability, mutation probability, and other pertinent parameters.

Representation	Permutations
Recombination	'Cut-and-crossfill' crossover
Recombination probability	100%
Mutation	Swap
Mutation probability	80%
Parent selection	Best 2 out of random 5
Survival selection	Replace worst
Population size	100
Number of offspring	2
Initialisation	Random
Termination condition	Solution or 10,000 fitness evaluations

4.2. Number of Runs

Run the EA at least three times with different numbers of mutation probability, such as (60%, 80%, and 100%). Then, save the average and the best individual for each generation.

Example

Run 1 (pm=60%)		
Generation	Avg. fitness	Best fitness
1	5.5	4
2	4.5	2

This is to analyze the performance of EA, in which we measure the time evolution of the average fitness of its population from generation to generation or the time evolution of the fitness of the best individual of each generation.

4.3. Time evolution of Fitness

Display your findings by graphing **the best fitness value** and **the population's average fitness over time**. Figure 3.5 illustrates swift initial progress that levels off later, though note that while Figure 3.5 is a maximization problem, yours is a minimization problem.

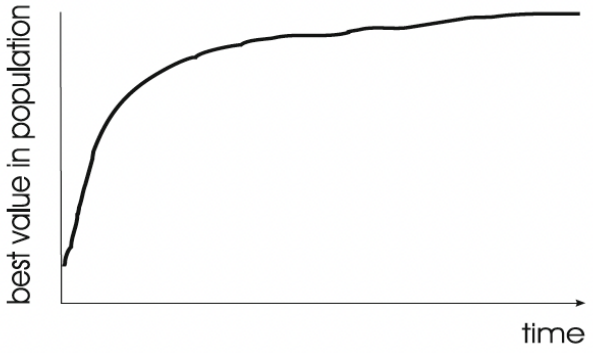

Fig. 3.5. Typical progress of an EA illustrated in terms of development over time of the highest fitness in the population

5. Create a Report Document (20 points)

- **Structure Your Report:**

- **Introduction:**
  - Clearly state the purpose of your report and provide a brief overview of the 8-Queens problem.
  - Define Evolutionary Algorithms (EA) and explain how they can be applied to solve optimization problems.

- **Methodology:**
  - Describe the specific EA algorithm you used
  - Explain how you represented the 8-Queens problem as an optimization problem.
  - Detail the parameters and settings used in your EA implementation.
- **Results and Analysis:**
  - Present your experimental results, including:
    - The number of generations required to find a solution.
    - The quality of the solution (number of conflicts).
    - Any visualization or analysis of the search process.
  - Discuss the effectiveness of EA in solving the 8-Queens problem.
- **Conclusion:**
  - Summarize your findings and conclusions.
  - Discuss the strengths and limitations of using EA for this problem.
  - Suggest potential areas for future research or improvements.

- **Formatting and Style:**
  - Follow a consistent formatting style (e.g., APA, MLA, Chicago).
  - Use clear and concise language.
  - Include appropriate figures, tables, and code snippets to support your findings.
  - Proofread carefully for grammar and spelling errors.

6. References

1. Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: a modern approach*. Pearson.
2. Eiben, A. E., & Smith, J. E. (2015). *Introduction to evolutionary computing*. Springer-Verlag Berlin Heidelberg.