Artificial Intelligence II – Homework 3
8 queens puzzle

Jonáš Petrovský
Faculty of Business and Economics,
Mendel University in Brno,
Czech Republic
`jond@post.cz`

December 5, 2015

1 Introduction

The goal of the work is to implement a solution to the ”8 queens puzzle” using the ”MIN-conflict” heuristic. There are 8 queens on a chess board. The goal is to rearrange the queens so that no two queens threaten each other.

2 Solution design

There are 4,426,165,368 possible arrangements of eight queens on an 8×8 board, but only 92 solutions (Wikipedia, 2015). We could use brute-force, but there exists a smarter way. We apply a rule that in one column can only be one queen. Then we use the MIN-conflict heuristic to select the best following state.

3 Implementation

The program was implemented in Python. A state is saved as a 2D array representing a chess board, where 1 means there is a queen on the square and 0 that there is no queen. Example for lecture state 1:

```
state = [
            [0,0,0,0,1,0,0,0],
            [1,0,0,0,0,0,0,0],
            [0,0,0,0,0,1,0,0],
            [0,0,0,1,0,0,0,0],
            [0,1,0,0,0,0,0,0],
            [0,0,0,0,0,0,1,0],
            [0,0,1,0,0,0,0,0],
            [0,0,0,0,0,0,0,1],
]
```

All visited states are saved in ”solution_path” list (in order). There is a main class **ChessQueenWorld** with following public methods:

- `solve_given_board(start_state)`

- `solve_random_board()`
- `solve_sample_board(number)`
- `bulk_solve(number_of_boards)`

The main method `solve_given_board()` accepts a start state and tries to find a solution using the heuristic.

1. Reset a solution path.
2. Insert a start state into the solution path.
3. While conflicts count > 0 OR iterations count <= 500:
   (a) Choose a new queen position with min. number of conflicts.
   (b) Check if the new state is already in solution path.
   (c) If yes, re-run the method.
   (d) If not, add it to the solution.

If there are no conflicts, a message of success and number of performed moves are displayed.

Class **QueenHeuristics** implements the calculation of "MIN-conflict" heuristic. For every column, we try to place a queen on different squares and count the number of conflicts on every position. Than the position with the lowest number is chosen. However, if the count is higher than count on the queen's original position, we choose nothing.

Class **StatesCreator** contains sample states and `generate_random_start_state()` method.

There is a simple test class in folder /tests, which can be used to check (for sample states), whether heuristic method returns the correct number of conflicts for given state. The class can be used in `test-3-check.py` file.

The program can be run by typing the following command into the operating system's terminal (Python 2.7 has to be installed on the system):

```
python run-3.py
```

What the program should do must be stated in the run-3.py source code (the script has no parameters).

4 Results

All resulting text files are placed in /output folder. The first matrix is the start state, then the performed steps and the final state are displayed.

4.1 Lecture start states

Board 1 (1 conflict): Solved in 2 steps. More in lecture_1.txt.
Board 2 (28 conflicts): Solved in 10 steps. More in lecture_2.txt.

4.2 Generated start states

The method `generate_random_start_state()` can be used to produce some of the 16,777,216 possible queens combinations (one queen in every column). ”Efficiency” means average number of steps (queen moves) for solution. ”Success rate” shows how many percent of the generated start states has the algorithm been able to solve.

Table 1: Total results generated start states

N. of boards	100	1,000	10,000
Success rate [%]	96	95.2	94.37
Efficiency [steps]	18.34	17.6	17.49

From Table 1 can be seen that the algorithm managed to solve the overwhelming majority of generated boards (95 %) in 17 steps on average.

Subsequently, the number of conflicts in the start states and its impact on success rate and efficiency is examined. We take values from simulation of 10,000 boards to present the most accurate conclusions.

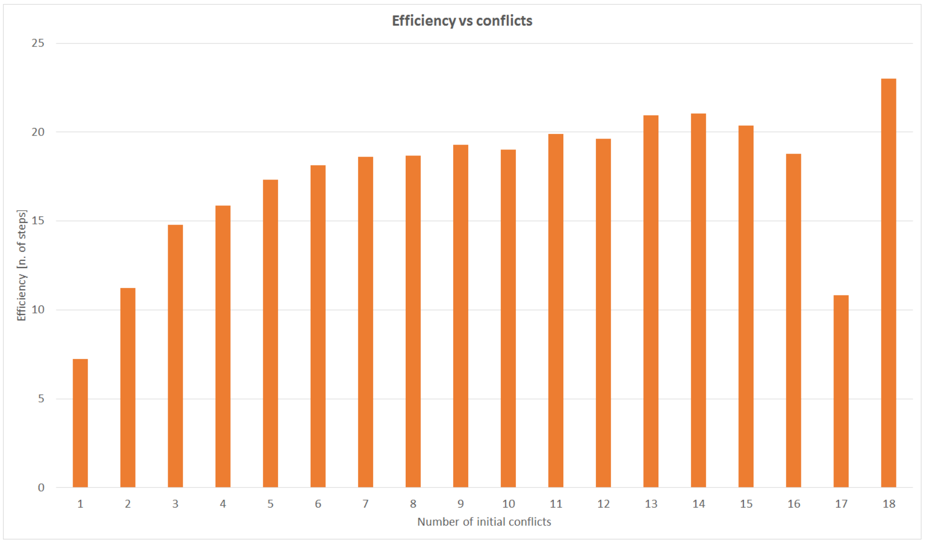

Figure 1: Relation between number of initial conflicts and efficiency

As can be seen in Figure 1, there is an obvious correlation between number of initial conflicts and efficiency – the more initial conflicts, the more steps need to be performed (on average). Of course, there are some irregularities, but on the whole, the conclusion is correct.

The success rate was 100 %, except for 1, 17 and 18 initial conflicts, where the rate was 10.81, 5.83 and 50 %, respectively. The overwhelming majority (94 %) of generated states contained 3–14 conflicts (categories with more than 100 states). The aforementioned success rate might be the result of too low number of tested examples, but also a high complexity of the given start states. For a detailed examination a larger number of generated states would be needed.

5 References

Eight queens puzzle. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, last updated 3. 12. 2015 [accessed 2015-12-05]. Available at: https://en.wikipedia.org/wiki/Eight_queens_puzzle