# CS517: Theory of Computation

# Final Project

Xuejing Cao

June 12, 2020

## 1 Introduction

The eight queens puzzle is a chess-based problem: how can eight queens be placed on an eight by eight chessboard so that no queen can threaten any other queen directly? To achieve this purpose, no two queens can be on the same horizontal, vertical or diagonal line. It is an old and famous problem which is a typical case of backtracking algorithm. The eight queen puzzle can be generalized to the more general n queens problem. Then the size of the chessboard becomes n by n, and the number of queens becomes n.

In this project, we will provide a software tool for solving the n queens problem. This tool can directly derive solutions to problems from their descriptions. By describing the form of input of the n queen problem, the tool can create all the possible strategy for it.

## 2 Theory

In this section, I will demonstrate how the n-queen problem is decomposed to SAT and then solved by a SAT solver.

### 2.1 N queens problem

Obviously N queens problem is a NP-hard problem whose algorithm complexity is exponential. For the classic eight queen puzzle, it has 64!/(56!*8!) to 4.4*$10^9$ possible arrangements. Two solutions are the same if one of them can be derived from the other one by rotation or symmetry. Then we only have 12 different solutions left. So, for N queens problem on n by n chessboard, it will be computationally more expensive.

## 2.2 Convert N queens to SAT

To solve the n queens problem, we need to express it in conjunctive normal form. The n queens problem can be expressed by the formula containing $n^2$ hypothesis variable $s_{ij}$, where $s_{ij}$ represents the cell (i, j) of the n by n chessboard. If and only if the queen is set to (i, j), then $s_{ij}$ will be defined as true. The different constraints of the problem can be expressed as follows:

The formula for at most one queen in each row and column respectively:

$\psi_1 = \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n} \bigwedge_{k=j+1}^{n} (\neg s_{ij} \vee \neg s_{ik})$

$\psi_2 = \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n} \bigwedge_{k=j+1}^{n} (\neg s_{ji} \vee \neg s_{ki})$

Then we will consider the restriction of the diagonal lines. Define $D_1$ as the set of diagonals parallel to the lower left diagonal to the upper right diagonal and $D_2$ as the set of diagonals parallel to the lower right diagonal to the upper left diagonal. So for the $D_1$, we split it into two subset. The first one is the lines associated with d from 0 to n-2, and d = i - j. Then the formula for at most one queen in these lines:

$\psi_3 = \bigwedge_{d=0}^{n-2} \bigwedge_{j=1}^{n-d} \bigwedge_{k=j+1}^{n+d} (\neg s_{d+j\ j} \vee \neg s_{d+k\ k})$

For the remaining diagonal lines, the formula is:

$\psi_4 = \bigwedge_{d=-(n-2)}^{-1} \bigwedge_{j=1}^{n+d} \bigwedge_{k=j+1}^{n+d} (\neg s_{j\ j-d} \vee \neg s_{k\ k-d})$

For $D_2$, we also split it into two subsets. The first one is lines whose d is from 3 to n+1. Then the formula for at most one queen in these lines:

$\psi_5 = \bigwedge_{d=3}^{n+1} \bigwedge_{j=1}^{d-1} \bigwedge_{k=j+1}^{d-1} (\neg s_{j\ d-j} \vee \neg s_{k\ d-k})$

For the remaining in $D_2$, the formula is:

$\psi_6 = \bigwedge_{d=n+2}^{2n-1} \bigwedge_{j=d-n}^{n} \bigwedge_{k=j+1}^{d-1} (\neg s_{j\ d-j} \vee \neg s_{k\ d-k})$

Now for n queens problem, there are n queens that need to be placed. For $\psi_1$, we need to add the restriction, there is at least one queen in each column, so that the number of queen in each column is exactly one. Then the formula for at least one queen in each column is:

$\psi_7 = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{n} s_{ij}$

Therefore, the CNF is the conjunction of these seven formulas and every result of true means a solution to the n queens problem.

## 2.3 SAT solver

Given Boolean variable and expression in the CNF file, the SAT solver will determine whether a solution that makes the expression true can be found. Therefore, the SAT solver can be applied in many problems, provided that these problems can be decomposed into SAT problems.

```
5  ################ using minisat to solve n-queens #############
6  formula = CNF(from_file = 'queen2sat.cnf')
7  with Minisat22(bootstrap_with = formula.clauses )as m:
8      m.solve()
9      s=(m.get_model())
```

Figure 1: SAT solver

Since we have already converted n-queens problem to CNF file, we use SAT solver to fetch each line content of CNF. According to the figure 1, after SAT solver solves every lines expression successfully, it would provide a set of solution. If we reverse the first solution and add it back into the end of CNF file, then we will get another solution until we get the one we want.

## 3 Implementation

The tools include:

- N queens program in python
- A CNF file as an input for the SAT solver
- A SAT solver program to get solution

First we need to get the value of 'n' from the user and take it as an input for the program. Then the required constraints for sat solver will be generated as a CNF file. This file will be entered as input into the sat solver. Then it will give the solution for the problem according to the given condition which is the coordinate of queens. The final output will be written in a text file.

## 4 Evaluation

With the increasing value of input N, the size of formula and time the solver takes to run will go up. From the table below, we can see the running time of getting the first solution is quite small so that if not giving the position of queen, we can have the result very quick. When it needs to output the specific solution, we will compare each possible solution with the given partial position of queens. However, it will take much more time.

For instance, if we input N=5 and do not set any existed queen, then we will output only one solution. When we input N=5 and set existed queens, it will output the corresponding solution if the existed queens could generate a solution; or it will show "No solution" if the existed queens could not lead to a solution.

| N | time for 1-solution | time for all |
|---|---|---|
| 4 | 0.00012 | 0.00015 |
| 5 | 0.00006 | 0.00048 |
| 6 | 0.00012 | 0.00037 |
| 7 | 0.00006 | 0.00211 |
| 8 | 0.00012 | 0.00874 |
| 9 | 0.00012 | 0.04393 |
| 10 | 0.00012 | 0.25834 |
| 11 | 0.00012 | 3.7292 |
| 12 | 0.00012 | 78.11792 |
| 13 | 0.00012 | 879.82467 |
| 14 | 0.00014 | >25200 |

Figure 2: the running time(second) for different N

```
CAYLAdeMacBook-Pro:Archive 2 copy cayla$ python3 n_queens.py 5
Do you want to insert some existed queens ? (1:yes, 0: no)
 0
CAYLAdeMacBook-Pro:Archive 2 copy cayla$ python3 solve_cnf.py
0.00007s
No given queen's position
```

Figure 3: input for N=5 without setting queen's position

```
[0, 0, 0, 1, 0]
[0, 1, 0, 0, 0]
[0, 0, 0, 0, 1]
[0, 0, 1, 0, 0]
[1, 0, 0, 0, 0]
```

Figure 4: the solution for N=5 without setting queen's position

```
CAYLAdeMacBook-Pro:Archive 2 copy cayla$ python3 n_queens.py 5
Do you want to insert some existed queens ? (1:yes, 0: no)
 1
input the number of Queen's position, ex: if N=4, available position will be 1~16:
2 9
CAYLAdeMacBook-Pro:Archive 2 copy cayla$ python3 solve_cnf.py
```

Figure 5: input for N=5 with setting existed queens

Obviously, the input N should be greater than seven. Since the running time will be at least seven hours for N=14, we recommend the input is less than 14.

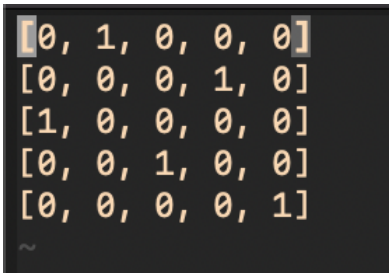


Figure 6: output for N=5 with setting existed queens

## 5 Conclusion

This project demonstrates the n queens problem can be reduced to SAT, and solved by a sat solver. It not only can provide a random solution based on the value of N, but also can output the specific solution according to the determinate position of queens. In the future, we could improve the input of formulas into sat solver and make it more efficient.

## References

Gutiérrez Naranjo, Miguel Ángel, Martínez del Amor, Miguel Ángel, Pérez Hurtado de Mendoza, Ignacio, Pérez-Jiménez, Mario de Jesús. (2009). Solving the N-Queens Puzzle with P Systems. Seventh Brainstorming Week on Membrane Computing, Vol. 1, 2009-01-01, ISBN 9788461328376, Pags. 199-210, 1, 199-210.

Solving NQueens Puzzle using MiniSAT: https://github.com/vivekkvvermaindia/NQueens-Puzzle-using-MiniSAT/blob/master/nqueens.py