

Solving the N-Queens Problem

Interactive Configuration using Binary Decision Diagrams

Thorbjørn Nielsen (thse@itu.dk) and Martin Faartoft (mlfa@itu.dk)

IT University of Copenhagen

1 Introduction

Using the provided library and Java GUI components, we are to create an interactive configurator that helps a user to solve the N-Queens problem. This means doing the following:

- Compile a BDD that represents the rules of N-Queens
- Restrict the BDD every time the user adds a queen
- Relax the BDD restrictions every time the user removes a queen
- Complete the solution, if there are no choices left (the remaining queens can only be placed in one way)

2 Representing the Rules of N-Queens

We have a BDD representing the rules of the board, with one variable for each square on the board. The top left corner is variable $\#0$, and then taking rows before columns, the lower right corner is variable $\#n * n - 1$, where n is the number of squares per row (and column).

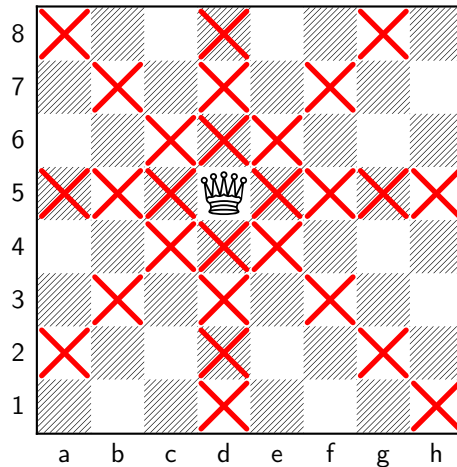


Fig. 1: Example of queen placement, and restricted spaces

Our rules are split in 2, the first part is done for every space on the board and is of the form: $X_i \Rightarrow \neg X_n \wedge \neg X_{n-1} \dots \wedge \neg X_1$, where X_i is the variable we're currently adding rules for, and $X_1 \dots X_n$ are the variables that are mutually exclusive with X_i (same row, same column, or same diagonal)

The second part of the rules is the "one queen per row" part, with rules of the form: $X_1 \vee X_2 \vee X_3 \dots \vee X_n$. Finally we take the conjunction of the n space-restriction rules, and the \sqrt{n} one-queen-per-row rules, and that is our BDD representing the rules of the N-Queens problem.

3 Restricting the BDD