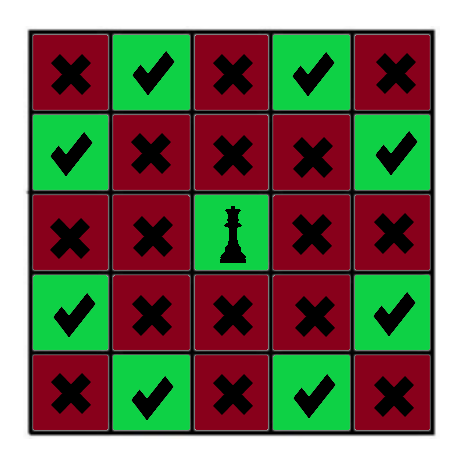
**Technical Document**

**Solving the N-Queen Problem Using a Min-Conflicts Algorithm**

**February 28, 2018**

**Problem**

****The N-Queen problem, inspired by the game of chess, is a constraint satisfaction problem in which the constraints are defined by “n” queens, where {n ∈ ℕ | n = 1 or n > 3}. For n queens, the size of the chess board for which the queens will be placed is n × n squares. Each n queen is a classic chess queen which can attack any other chess piece positioned vertically, horizontally, or diagonally in relation to its location on the board. For the problem to be solved, no queen should be placed on the board such that it can attack another queen without an intermediate move and there should be n queens located on the n × n chess board.

The illustration to the left depicts a queen and its “safe spots” highlighted in green, and “conflicting spots” highlighted in red. For the center queen to not violate problem constraints, no other queen may be placed in a red square. Successive queens placed on the board may only be placed within the green squares. Each time a queen is placed, the problem becomes more constrained.  **Considerations**

In a naïve approach, one might try and randomly place queens on the chess board until a constraint satisfying configuration is found. This is obviously a bad solution because the probability of generating a constraint satisfying configuration is blah.

One method for generating a constraint satisfying configuration is a backtracking algorithm which generates

Backtracking

**Solving with Min-Conflicts Approach**

**Attempted Solution**