# Eight queen problem

**Chris Godnič**
UP FAMNIT

***Abstract*—In this report we present the problem, how to find a placement of 8 queens on a chess board so that no queen can capture another queen. We will make three different programs for same problem, so the problem will be solved sequentialy, parallel with multiple threads and at the end serially with Message Passing Interface (MPI).**
**At the end, we will also generalize the problem to N-queens so we will see how time behaves in a connection to bigger chessboard.**

## I. Introduction

In 1848, chess composer Max Bezzel published the eight queens puzzle problem. In 1850, was published the first solution by Franz Nauck. Nauck also extended the puzzle to n queens problem with n queens on a chesboard of nxn squares.

Since then, many mathematicians, including Carl Friderich Gauss, have worked on both the eight and its generalized n-queens version.

The problem is classic example of the use of backtracking.

## II. Eight problem solution

The 8-queens problem asks us to place 8 queens in a 8x8 chessboard so that no queen can capture another queen according to the rules of standard chess. The problem is an example of CSP (Constraint Satisfaction Problem), and non-trivial as such.

## III. Backtracking

Backtracking is a general algorithm for finding solutions to some computational problems. It try to solve problem recursively by trying to build a solution incrementally. If candidate fail to satisfy the constraints of the problem at any point of time, is abadoned.

## IV. Difference between sequential, parallel and MPI mode

Before we start discusing about logic and implementation of program we will describe how work any type of program.

**Sequential version:** Sequential or serial program is the most basic mode. It is executed once through, from start to finish. It uses a single thread all the time. So in our practical example we are checking only one matrix at time. If matrix is one of solutions, it is saved.

**Parallel version:** Parallel algorithm divide large problems in the smaller one, which can then be solved at the same time. In particular problem we devide matrices with a previously set queen on position [i][0] to workers, so that when worker get task he need to check only n-1 columns.

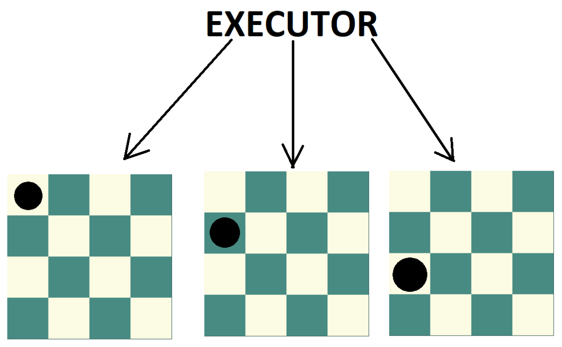
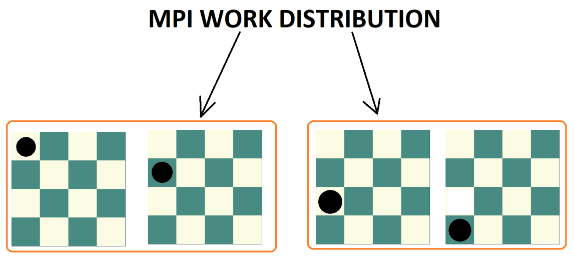


Fig. 1. Work distribution in parallel version

**Distributed version:** Distributed version is designed to run on computer hardware constructed from interconnected processors. In distributed mode we devide matrices with a previously set queen on position [i][0] to workers and as in parrallel version they check n-1 columns. The difference betwen parallel and distributed version is that worker in parallel get only one matix, on the other hand, the process in distributed version get few matrices.



Work distribution in distributed version

## V. How the number of results increases

Number of solutions

| chessboard dimensions | number of solutions |
|---|---|
| 1x1 | 1 |
| 2x2 | 0 |
| 3x3 | 0 |
| 4x4 | 2 |
| 5x5 | 10 |
| 6x6 | 4 |
| 7x7 | 40 |
| 8x8 | 92 |
| 9x9 | 352 |
| 10x10 | 724 |
| 11x11 | 2,680 |
| 12x12 | 14,200 |
| 13x13 | 73,721 |
| 14x14 | 365,596 |
| 15x15 | 2,279,184 |
| 16x16 | 14,772,512 |
| 17x17 | 95,815,104 |
| 18x18 | 666,090,624 |
| 19x19 | 4,968,057,848 |
| 20x20 | 39,029,188,884 |

As we can see the number of results is growing very fast, so we can say that the problem will be too demanding for our personal computer in just few magnifications of chessboard.

## VI. Results

When we finished with programming of all three versions we execute every program three times for chessboards of size from 7x7 to 16x16, and we made the table of average time. The 17x17 was not executed in time of ten minutes so we stopped the program.

Average execution time

| Matrix | Sequential | Parallel | MPI |
|---|---|---|---|
| 7x7 | 0 | 0 | 0,66 |
| 8x8 | 1,33 | 2 | 0,66 |
| 9x9 | 3 | 2,66 | 1,66 |
| 10x10 | 8,66 | 4,667 | 4,66 |
| 11x11 | 37,66 | 6,33 | 17,33 |
| 12x12 | 188 | 33,66 | 70 |
| 13x13 | 1025,33 | 168,333 | 397,66 |
| 14x14 | 6343,33 | 816,33 | 2623,33 |
| 15x15 | 40920,66 | 4886 | 15495 |
| 16x16 | 285531,66 | 37414,33 | 107185 |

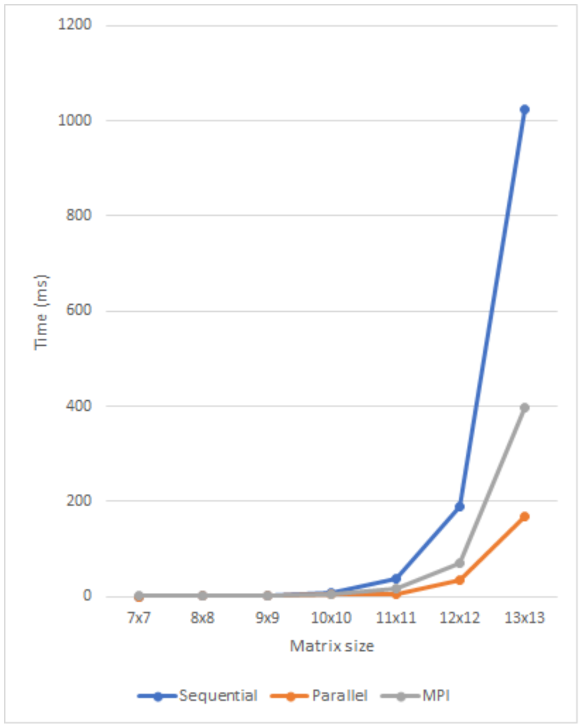


Fig. 2. Time spent depending on the size of the chessboard (7x7 - 13x13)

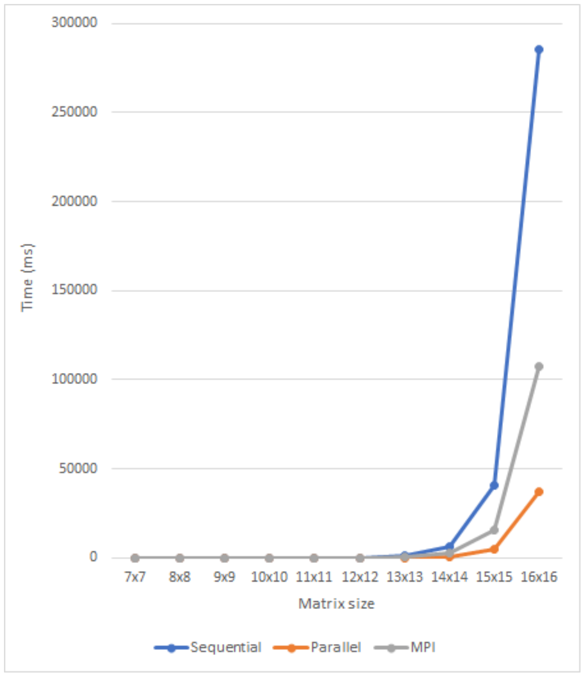


Fig. 3. Time spent depending on the size of the chessboard (7x7 - 16x16)

## VII. Conclusions

As we can see from table or graphs the fastest version was parallel and the second one was MPI. This result was expected, because we are doing same task over and over again. So when we do this task in sequential, where only one thread working, this thread goes over all matrices, on the other hand in parallel we check few matrices at once.