CS575: Final Project Report

Project Title: 8-Queens problem using Backtracking & Branch-and-Bound Algorithms

Team Member : Tanmayee Kulkarni

I. Problem

The eight queens' problem is a non-trivial problem of placing eight queens on 8x8 chessboard such that none of them attack one another (i.e. no two queens are in the same row, column or diagonal). More generally, the N-queens problem places N queens on a NxN chessboard. It can be solved using Backtracking and Branch-and-Bound algorithms.

II. Algorithms

To solve the 8-queens algorithm, we are going to use following two algorithms -

A. *Backtracking*

B. *Branch and Bound*

Backtracking

The basic idea is to place queens one by one in different columns, starting from the leftmost column till we hit a dead end.

When we place a queen in a column, we check for the clashes with already placed queens in the same column and row. In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution. If we do not find such a row due to clashes then we backtrack and return false.

Branch and Bound

The branch-and-bound approach suggests that we create a partial solution and use it to ascertain whether we should continue in a particular direction or not.

III. Software Design and Implementation

A. *Software Design*

In the 8-queens problem, we ensure the following –

1. No two queens share same row.
2. No two queens share same column.
3. No two queens share the same left diagonal.
4. No two queens share the same right diagonal.

Backtracking approach –

Algorithm :

1. Start from the leftmost column.
2. If all queens are placed
   return true.
3. Try all rows in the current column.
   Do following for every tried row -

   a. If the queen can be placed safely in this row then mark this [row, column] as part of the solution and recursively check if placing queen here leads to a solution.
   b. If placing the queen in [row, column] leads to a solution then return true.
   c. If placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and repeat step (a) to try other rows.

4. Return false to trigger backtracking if all rows have been tried and nothing worked

Branch-and-Bound approach –

We have already resolved column condition via Backtracking solution.

For the other rest of the three conditions, we will create three Boolean arrays that tell us which rows and which diagonals are occupied.

Pre-processing required –

1. Create two NxN matrices, one for top-left to bottom-right diagonal, and other for top-right to bottom-left diagonal.
2. Fill these in such a way that two queens sharing same top-left bottom-right diagonal will have same value in slashDiagonal and two queens sharing same top-right bottom-left diagonal will have same value in backSlashDiagonal.
3. For placing a queen *i* on row *j*, check the following:
   a. whether row 'j' is used or not
   b. whether slashDiagonal 'i+j' is used or not
   c. whether backSlashDiagonal 'i-j+7' is used or not
4. If the answer to any one of the following is true, we try another location for queen **i** on row **j**, mark the row and diagonals; and recur for queen **i+1**.

B. *Implementation and Tools Used*

Programming language - Java

Tool– Visual Studio Code editor

C. *Performance Evaluation (Optional)*

Time complexity :

Backtracking – O(N!)
Branch-and-Bound – O(N!)

Branch-and-Bound is preferred because

IV. Project Outcomes

Presentation link –

https://docs.google.com/presentation/d/1VuyARlWioDSkMtpF7yzOVOaOFViCYNdwToI6NW6e5z0/edit#slide=id.g126706fca62_2_76

YouTube video link -

https://www.youtube.com/watch?v=Ivd0DdGJfU8

References

[1]http://www.alphr.com/artificial- intelligence/1006866/win-million-eight-queens- puzzle#

[2] "The 8-Queens Puzzle". www.claymath.org. Clay Mathematics Institute. September 2, 2017. Retrieved September 7, 2017

[3] Gent, Ian P.; Jefferson, Christopher; Nightingale, Peter (August 2017). "Complexity of n-Queens Completion". Journal of Artificial Intelligence Research. 59: 815–848. doi:10.1613/jair.5512. ISSN 1076-9757. Retrieved September 7, 2017.

[4] https://github.com/VanHakobyan/8-queen-chess-problem

[5] Backtracking Algorithms in MCPL using Bit Patterns and Recursion by Martin Richards mr@uk.ac.cam.cl