Parallel Algorithms for the N-Queens Problem

Jing Wang
National Chiao Tung University
Hsinchu, Taiwan
jingwang.cs08g@nctu.edu.tw

Mu-Hua Yang
National Chiao Tung University
Hsinchu, Taiwan
n860404@gmail.com

Yu-Wun Tseng
National Chiao Tung University
Hsinchu, Taiwan
ienlie0513@gmail.com

1 INTRODUCTION

The n-queens problem is to place n queens on an $n \times n$ chessboard such that no two queens can attack each other. Thus, the solution requires that no two queens share the same row, column, or diagonal. Traditional approaches to the n-queens problem are based on backtracking. In our project, we will implement different algorithms and try to parallelize them to solve this problem. Additionally, we will analyze the maximum value of n under each algorithm in limited time and compare the performance of each algorithm under the same n. The classical 8-queens problem is shown in Figure 1.

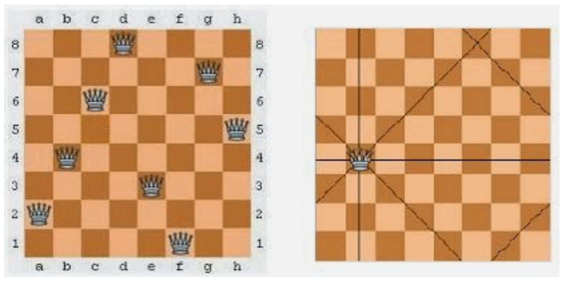

Figure 1: 8-Queens Problem

2 STATEMENT OF THE PROBLEM

The n-queens problem is a classical search problem. Traditional search methods involve backtracking. Because the complexity of backtracking usually rises exponentially with the size of the problem, it is desirable to develop alternatives to backtracking search. Therefore, many approaches have been proposed to research, e.g., hill-climbing, simulated annealing, genetic algorithm. In the artificial intelligence course, we have implemented these algorithms using Python. We want to try at least three algorithms and parallelize them to solve this problem. We will compare the execution time of different algorithms before and after the parallelization at different size of n.

3 PROPOSED APPROACHES

In this section, we will describe the algorithms which will be used for solving n-queens problem and how we will parallelize.

3.1 Backtracking Algorithm

Backtracking is the most common way to solve n-queens problem which can be described as a search tree. Each node of the tree corresponds to a partial solution. Going down the tree matches the progress of obtaining a complete solution, which is to place queens one by one in different columns and make sure there is no conflict occurred. While going up the tree matches the backtracking part, which means a conflict occurred and the node should returning a partial solution. In backtracking, we will use *master-worker* pattern [1] to parallelize search.

3.2 Hill Climbing Algorithm

Hill Climbing is a heuristic search algorithm. The idea is to move continuously in the direction of increasing value to find the peak of the mountain. It's also called greedy search because it only considers appropriate neighbors around but not beyond that. In Hill Climbing, we will parallelly do the hill climbing with different heuristic start points to find all solutions.

3.3 Genetic Algorithm

Genetic Algorithm is a search procedure to find optimized solution, the spirit of which is "Survival of the fittest". The three principles of genetic algorithm are selection, crossover and mutation. Each potential solution is represented as individuals that are evaluated using a fitness function. The fitness function determines how fit an individuals is and gives a fitness score. The basis process is shown in the following list:

(1) The individuals of first generation are randomly created and evaluated using the fitness function.
(2) Selecting the individuals using the selection operator. The individuals will survive into next generation which are viewed as "better" individuals.
(3) Selecting individuals act as parents and mate for generating the children using crossover operator.
(4) The mutation operator is applied on few individuals for randomly changes.
(5) Children and parents form the next generation.

Steps 2.-5. are repeated until the terminal condition are filled. In Genetic Algorithm, we will parallelize the selection, crossover and mutation part of each selected individual and the calculation of the fitness function.

4 LANGUAGE SELECTION

We will use C++ to implement these algorithms. Because if the program is written in Python, methods that can be used to accelerate the program will be limited.

5 RELATED WORK

There are many algorithms and approaches have been used to resolve n-queens problem. The most common approach to solve the problem is based on backtracking. General backtracking approaches could solve n-queens problems for n up to 100. By using a specialized heuristic, solutions n could up to 1000.

Vikas Aggarwal [3] parallelized the backtracking method on reconfigurable hardware with Handel-C and MPI. They added different levels of parallelism to improve performance: 1) Functional Unit replication. 2) Parallel column check. 3) Parallel row check. Their result shows that parallel backtracking for solving n-queens problem in certain hardware will improve the performance. Monire Taheri Sarvetamin [2] also delivered a new approach using Parallel Genetic Algorithm to solve this problem. The algorithm is much efficient with large number of *n* and in a type of comparison they can achieve super linear speedup.

6 STATEMENT OF EXPECTED RESULTS

We expect the time complexity is $O(N)$, in other words, the algorithms will yield linear speedup. Additionally, we wish that parallelized algorithms can be accelerated by at least 50% after parallelization.

7 TIMETABLE

Our project schedule is shown in Table 1.

Event	Estimated Finish Date
Group registration	October 4, 2019
Project proposal	October 29, 2019
Programming	December 20, 2019
Final report writing	January 10, 2020

Table 1: Project Schedule

REFERENCES

[1] Lorna E. Salaman Jorge. 2001. A Parallel Algorithm for the n-Queens Problem. (2001). http://www.ece.uprm.edu/crc/crc2001/papers/lorna_salaman.PDF

[2] Mohammad Hadi Zahedi Monire Taheri Sarvetamin, Amid Khatibi. 2018. A New Approach to Solve N-Queen Problem with Parallel Genetic Algorithm. *Journal of Advances in Computer Engineering and Technology* 4, 2 (2018), 69–78. arXiv:http://jacet.srbiau.ac.ir/article$_{0214c8b674cd37a7211fae820ec59dfd12443.pdf}$ http://jacet.srbiau.ac.ir/article_12443.html

[3] Ian Troxel Vikas Aggarwal and Alan D. George. 2004. *Design and Analysis of Parallel N-Queens on Reconfigurable Hardware with Handel-C and MPI.* Retrieved October 29, 2019 from https://ufdcimages.uflib.ufl.edu/UF/00/09/47/56/00002/MAPLD2004_Paper198.pdf