DIFFERENT PERSPECTIVES OF THE N-QUEENS PROBLEM

CENGIZ ERBAS[1], SEYED SARKESHIK[2], MURAT M. TANIK[3]

INTRODUCTION

The N-Queens problem is a commonly used example in computer science. In the problem you look for a placement of N queens on an NxN chessboard, such that no two queens threaten one another. The solution for the N-Queens problem can be represented by a permutation.

Algorithms for the N-Queens problem can be classified into 3 categories. The algorithms that produce every possible answer for a given N fall under the first category. Only the fundamental solutions are generated by the algorithms in the second category. The last group of algorithms produce only one or very few solutions.

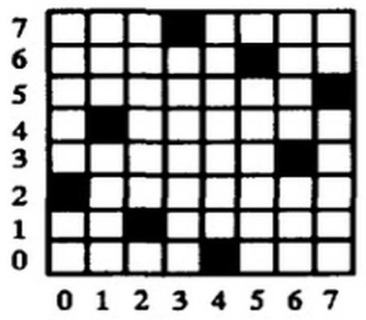

Figure 1. A solution to 8-Queens problem

[0 1 2 3 4 5 6 7
2 4 1 7 0 6 3 5]

Figure 2. A solution to 8-Queens problem represented in a permutation

ALGORITHMS GENERATING ALL SOLUTIONS

These algorithms locate all solutions for N. We may employ 3 sets of algorithms to create all possible solutions:

Brute-Force Trial and Error Algorithms: have a time complexity of O(N^N); therefore are very inefficient;

Backtracking Algorithms: build up the solution vector one component at a time and test it according to criteria. The time complexity is O(N^2);

Permutation Generation Algorithms: generate all permutations of N different elements, however, they do not test the partial arrangements. The complexity of this approach is O(N!)

N	# of solutions	N	# of solutions
1	1	10	724
2	0	11	2,680
3	0	12	14,200
4	2	13	73,732
5	10	14	365,596
6	4	15	2,279,184
7	40	16	14,772,512
8	92	17	95,815,104
9	352	18	666,090,624

Figure 3. The number of solutions for the N-Queens problem

ALGORITHMS GENERATING FUNDAMENTAL SOLUTIONS

The number of fundamental solutions rises exponentially, as seen in Figure 4. Several of the algorithms that locate fundamental solutions for N are shown in this section:

Generate and Test Algorithms: utilise algorithms that produce all possible solutions. They evaluate the solution to see if it is fundamental. It is eliminated if it can be transformed into a fundamental solution. Otherwise, it gets included in the fundamental solutions list.

Naur's Algorithm: uses central rows to eliminate some of the symmetries, while also using generate and testing the algorithm on other rows to discard the isomorphic solutions.

Topor's Algorithm: employs orbits of squares under a group, which are a collection of squares to which members of the group can map a given square. Thus, placing a queen on any square in the orbit leads to an equivalent solution.

N	# of solutions	N	# of solutions
1	1	9	46
2	0	10	92
3	0	11	341
4	1	12	1,787
5	2	13	9,233
6	1	14	45,752
7	6	15	285,053
8	12	16	1,846,955

Figure 4. The number of fundamental solutions for the N-Queens problem

ALGORITHMS GENERATING SOME SOLUTIONS

The algorithms in this category aim to find only one or very few solutions to the problem. 4 examples are shown:

Nondeterministic Algorithms: select one square from each column, being careful not to select two squares from the same row or diagonal. Move on to the following column after writing down the selected row.

Backtracking Algorithms: only look for the lexicographically first answer. However, empirical studies reveal that first-solution backtracking algorithms have progressively longer run times.

Probabilistic Algorithms: frequently used in conjunction with a backtracking method, this algorithm randomly distributes a few queens over the board before using backtracking to add the remaining queens to the original configuration.

Divide-and-Conquer Algorithms: introduce a solution to N-Superqueens; N-Queens-related issue.

LIMITATIONS

Backtracking is the most often used technique for performing a systematic search, which includes N-Queens. However, it has a few flaws:

The first issue is **thrashing,** which indicates that the application is making little or no progress due to memory depletion. Intelligent backtracking can be used to avoid thrashing[4].

Another typical disadvantage is having to **perform redundant tasks.** Even if conflicting variable values are found during intelligent backtracking, they are not remembered in subsequent calculations. This disadvantage can be overcome by employing dependency-directed backtracking[5].

The last difficulty with backtracking is that it **discovers the disagreement too late.** This may be prevented by screening for any conflicts ahead of time. Additionally, setting a time restriction for the solver or a breakpoint after a predetermined number of solutions might also be beneficial for an efficient backtracking.

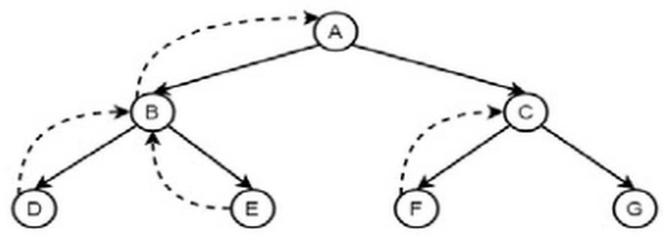

Figure 5. Backtracking Algorithm[6]

REAL-WORLD APPLICATIONS

The N-Queens problem is an SAT (satisfiability) problem since it is a logic problem for which a solution is desired. Although the N-Queens issue is generally handled in mathematical or scientific domains, it can lead to understanding methodologies such as search space reduction utilising symmetry, constraint programming, or evolutionary algorithms. Here are some instances from real life:

Deadlock prevention: It is demonstrated that given a solution to the N-Queens problem, a set of deadlock-free pathways may be found.[7]

Motion estimation: For both texture and edge characteristics, N-Queens can characterise spatial information in the vertical, horizontal, and diagonal directions.[8]

Other practical uses of the N-Queens issue include **VLSI testing** and **traffic control.**

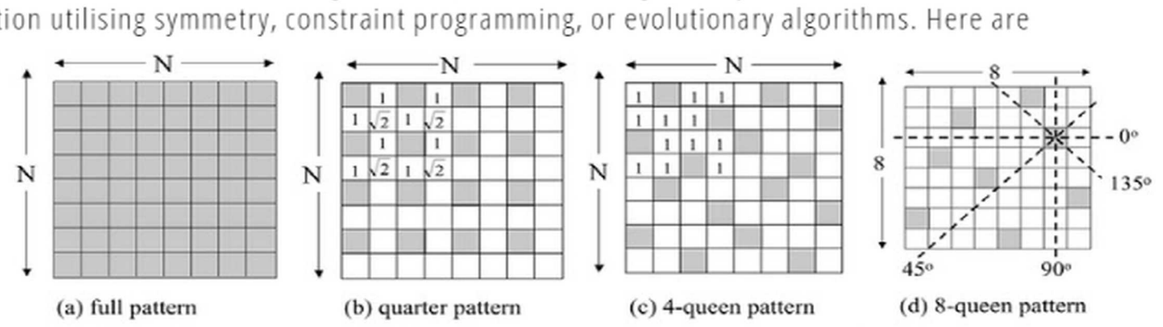

Figure 6. Fast Motion Estimation Using N-Queen Pixel Decimation[8]

CONCLUSION

The results show that certain methods for solving N-Queens are superior to others. Even though just a few algorithms' time complexities were presented, we can easily understand the advantages and downsides of the others.

Because N-Queens is a simple but also complex issue, it may illustrate not only the usual solution with backtracking but also approaches like search space reduction utilising symmetry, constraint programming, or evolutionary algorithms. All of these approaches have extensive applicability in a wider scope.

[1] Graduate student at CSE, Southern Methodist University, Texas

[2] Associate professor of CSE, Southern Methodist University, Texas

[3] Graduate student at the University of Texas at Arlington

[4] Ma, W. et al. (2021) Frontiers In Parallel Programming Models For Fog And Edge Computing Infrastructures, Arabian Journal for Science and Engineering. Available at: https://link.springer.com/content/pdf/10.1007/s13369-021-05984-2.pdf (Accessed: November 14, 2022).

[5] Udovičić, M. (2006) Chronological and dependency-directed backtracking. Serbian-Hungarian Joint Symposium on Intelligent Systems. Available at: http://conf.uni-obuda.hu/sity2006/55_Udovicic.pdf (Accessed: November 14, 2022).

[6] Backtracking (no date) Tutorials Point. Available at: https://www.tutorialspoint.com/prolog/prolog_backtracking.htm (Accessed: November 16, 2022).

[7] Panwar, P. et al. (2013) Load balancing using N-Queens. Available at: https://www.ijert.org/load-balancing-using-n-queens-problem (Accessed: November 15, 2022).

[8] Yang, S.-W. et al. (2001) Fast Motion Estimation Using N-Queen Pixel Decimation. Available at: https://link.springer.com/content/pdf/10.1007/3-540-45453-5_17.pdf (Accessed: November 12, 2022).

Oliver Alan Stafurik
2444536s