PROJECT REPORT
ON N-QUEENS

Submitted by

Ankit Kumar - Regn NO: 2241011104
Aakash Jena - Regn NO: 2241010013
Suvam Muduli - Regn NO: 2241013373
Baivab Mishra - Regn NO: 2241013140
Subhendu Mohapatra - Regn NO: 2241013345
Subhankar Mishra - Regn NO: 2241013187

Supervised By

Mr. Shiva Agarwal
Assistant Professor
Computer Science and Engineering
Faculty of Engineering and Technology

Centre For Artificial Intelligence and Machine Learning
Institute of Technical Education And Research(ITER)
SIKSHA 'O' ANUSANDHAN (DEEMED TO BE) UNIVERSITY
Bhubaneswar-751030, Odisha, India

_____
Shiva Agarwal
(Supervision)

1.ABSTRACT:-

The N-queens trouble is a famous classic puzzle where numbers of queen were to be placed on an n x n matrix such that no queen can attack some other queen. The Branching component grows in a roughly linear manner, that's an crucial consideration for the researchers. however, many researchers
have noted the issues with help of synthetic intelligence seek patterns sa y DFS, BFS and backtracking algorithms. we've conducted an intensive take a look at on this problem and realized that no algorithms to this point designed with simple formulation based.

2.INTRODUCTION:-

The N-queens puzzle is the trouble of placing N chess queens on an N × N chessboard such that none of them is able to capture some other using the same old chess queen’s moves. The color of the queens is meaningless in this puzzle, and any queen is assumed in order to assault every other. therefore, a solution calls for that no queens share the equal row, column, or diagonal.

In a nutshell, the N-Queen trouble is a chess hassle. The objective is to find all feasible solutions for placing N variety of queens on an N x N chessboard in order that not one of the queens can assault (or “clash”) with each other because queens can pass in all feasible instructions, i.e. diagonally and orthogonally, that is a
reasonably complex and interesting hassle to solve.
The origins of this nicely-studied problem are
debated, but similar puzzles were distinct as early because the eighth century. The current hassle as we realize it, but, have become famous after the nineteenth century. on account that the arrival of computer technological know-how in the 70s,
the trouble has received traction, and lots of algorithms have been designed in an try to remedy the hassle maximum optimally.

3.LITERATURE SURVEY:-

1848: The Problem's Debut:
German chess composer Max Bezzel first posed the N-Queens problem in a chess magazine. It quickly captured mathematicians' and puzzle enthusiasts' attention.

Early 20th Century: AI's Dawn:
The N-Queens problem gained significance with AI's emergence as a field.It became a valuable test-bed for developing and evaluating AI algorithms.

1950s: Pioneering Backtracking:
Researchers like Newell, Shaw, and Simon explored backtracking as a solution approach. Their work contributed to the development of general problem-solving strategies in AI.

1960s-1970s: Expanded Approaches:
Branch and bound, constraint programming, and genetic algorithms emerged as alternative solution methods. These techniques expanded the problem-solving toolkit for AI researchers.

1980s-1990s: Computational Advancements:
The growth of computing power enabled exploration of larger N values and more complex problem variations. Researchers developed specialized solvers and optimized algorithms for the N-Queens problem.

21st Century: Continued Relevance and Innovation:
The N-Queens problem remains a benchmark for evaluating AI algorithms and exploring new approaches.
Recent advancements include hybrid algorithms, heuristics, meta-heuristics, parallelization, and quantum computing approaches.

Key Research Contributions:
W. W. Bledsoe and Herbert A. Simon (1966): "Generalization of a Theorem Proving Machine to a Complete Strategy for a Chess Endgame" introduced a backtracking algorithm for the N-Queens problem.

P. Gent and T. Walsh (1999): "The N-Queens Problem" surveyed solution approaches and computational complexity.
J. R. Rice (2006): "The Algorithm Selection Problem" explored the use of heuristics and meta-heuristics for the N-Queens problem.
The N-Queens problem is a classic combination problem in the field of Artificial Intelligence (AI). It involves placing N queens on an N x N chessboard such that no two queens can attack each other directly (diagonally, horizontally, or vertically). This seemingly simple problem has fascinated researchers for centuries and served as a benchmark for various AI algorithms and techniques.

Significance:
Benchmark for AI algorithms: The N-Queens problem presents a well-defined problem space with increasing complexity as N grows. It's used to evaluate the efficiency and effectiveness of different search algorithms, constraint satisfaction techniques, and optimization methods. Fundamental concepts in AI: The problem exemplifies core AI concepts like backtracking, search space exploration, state space representation, and constraint satisfaction. Solving it provides valuable insights into these concepts and their practical application.
Applications beyond chess: Although rooted in chess, the N-Queens problem has applications in various domains like scheduling, resource allocation, and data analysis. Its solution techniques can be adapted to solve similar problems in different contexts.

Solution Approaches:

Backtracking:
A classic approach that explores all possible placements of queens recursively, backtracking if a conflict arises. Efficient for smaller boards but becomes time-consuming for larger N. Branch and Bound: Builds a search tree systematically, pruning branches that cannot lead to a solution based on bounds on the number of solutions. More efficient than backtracking for larger boards. Genetic Algorithms: Simulate natural selection to evolve populations of potential solutions, iteratively improving them until a valid solution is found. Suitable for parallel computation and finding near-optimal solutions.

Constraint Programming:
Formulates the problem as a set of constraints and uses specialized solvers to find solutions that satisfy all constraints simultaneously.
Efficient for problems with complex constraints.

Recent Advancements:

Hybrid algorithms:
Combine multiple approaches, like backtracking with constraint satisfaction, to leverage their strengths and improve efficiency.
Heuristics and meta heuristics: Develop heuristics to guide search algorithms towards promising solutions and meta heuristics to escape local optima in complex search spaces.

Parallelization:
Utilize parallel computing architectures to speed up search and solution finding for large N cases.
Quantum computing: Explore the potential of quantum algorithms to solve the N-Queens problem with exponential speedup compared to classical algorithms.

4.METHODOLOGY:-

Problem Representation:

Chessboard: Model the board as an N x N matrix, where each cell can hold a single queen.

Queen Placement: Use a data structure, such as an array or list, to store the queen's positions in each row.

Constraints: Represent the non-attacking conditions as rules to check for valid placements.

Common Solution Approaches:
Backtracking:
Place queens recursively, row by row. If a placement conflicts with constraints, backtrack to the previous row and try a different placement.

Efficient for smaller boards, but can become time-consuming for larger N.

Constructive Approaches:

One-row-at-a-time:
Place queens row by row, checking for conflicts in the current and previous rows.

Permutation-based:
Generate all possible permutations of queen placements and check for validity.

Constraint Satisfaction Programming (CSP):
Model the problem as a set of constraints on queen positions.
Use specialized CSP solvers to find solutions that satisfy all constraints.

Genetic Algorithms:
Simulate natural selection to evolve populations of potential solutions.
Iteratively improve solutions until a valid arrangement is found.

Key Considerations for Ethical and Responsible Use:

Algorithm Bias:
Be mindful of potential biases in algorithms and data that could lead to unfair or discriminatory outcomes.

Explainability and Transparency:
Ensure that the algorithms and solutions are understandable and explainable to users.

Fairness and Accountability:
Consider the potential social and ethical impacts of the problem and its solutions.

Optimization and Efficiency Techniques:

Symmetry Breaking:
Exploit symmetries in the problem to reduce the search space.

Pruning:
Eliminate branches of the search tree that cannot lead to valid solutions.

Heuristics:
Use rules of thumb to guide the search process towards promising solutions.

Parallelization:
Distribute the search process across multiple processors or computers for large N values.

Choosing the Right Methodology:

Problem Size:
Consider the size of the board (N) when selecting an algorithm.

Desired Number of Solutions:
Determine if you need a single solution or all possible solutions.

Constraints:
Identify any additional constraints beyond the non-attacking condition.

Computational Resources:
Evaluate the available computational resources and time constraints.
I'm ready to provide more specific examples, code implementations, or further discussions on these methodologies based on your interests and needs. Feel free to ask any further questions you may have.

5.RESULT:-

Existence of Solutions:

Guaranteed Solutions:
For every board size (n), there exists at least one valid solution where n queens can be placed on an n*n chessboard without attacking each other. This demonstrates that solutions are always attainable.

Number of Solutions:

Variety and Growth:
The number of possible queen arrangements that solve the puzzle varies depending on the board size. For smaller boards, there might be only a few solutions, but as the board gets larger, the number of solutions increases exponentially, showcasing the problem's complexity.

Algorithms for Solving:

Diverse Approaches:
Various algorithms have been developed to tackle this problem, each with unique strategies for exploring the solution space.

Backtracking:
Systematically explores potential solutions, backtracking when a path proves infeasible.

Branch and bound:
Divides the problem into smaller subproblems, discarding branches that can't lead to solutions.

Constraint programming:
Models the problem as a set of constraints and employs specialized algorithms to find solutions.

Benchmarking and Evaluation:

Performance Assessment:
The n*n queen problem is widely used to benchmark and evaluate the performance of AI algorithms. It enables researchers to compare different algorithms in terms of their efficiency in finding solutions, time consumption, and memory usage.

Insights for AI:

Challenges of Constraint Satisfaction:
The problem highlights the challenges of solving constraint satisfaction problems efficiently, especially as the problem size grows. This

motivates ongoing research into better search techniques and more efficient algorithms.

Algorithm Design and Selection:
It demonstrates the importance of choosing the right algorithm for a specific problem, as different algorithms have varying strengths and weaknesses.

Search Strategies:
It provides a model for understanding how computers explore large problem spaces to find solutions, which is a fundamental concept in AI problem-solving.

Applications and Pedagogical Value:

Educational Tool:
While the problem itself has limited direct real-world applications, it serves as a valuable pedagogical tool for teaching core AI concepts, including:

a) Search algorithms
b) Constraint satisfaction
c) Algorithm design and evaluation

Advantages:

1. Algorithm Development and Testing:

Benchmark Problem:
The N*N Queen problem serves as a well-defined benchmark for evaluating different AI algorithms, allowing for direct comparison of their performance and efficiency.

Algorithm Refinement:
It challenges researchers to develop new and improved algorithms for solving constraint satisfaction problems, advancing the field of AI.

2. Understanding Constraint Satisfaction:

Core Concepts:
It provides a clear example of constraint satisfaction problems (CSPs), a fundamental concept in AI, involving variables, domains, and constraints that must be satisfied.

Problem-Solving Techniques:
It helps develop and test general-purpose AI techniques for solving CSPs, applicable to various real-world problems.

Disadvantage:

1. Computational Complexity:

Exponential Time Complexity:
As the board size (n) increases, the number of possible queen arrangements grows exponentially. This makes finding solutions for larger boards extremely time-consuming, even for powerful computers.

Limited Scalability:
This limits its practical application to smaller board sizes, hindering its use in real-world scenarios that might involve larger problem spaces.

2. Narrow Scope:

Specific Problem Domain:
The n*n queen problem focuses on a very specific constraint satisfaction problem. It doesn't directly address broader AI challenges like learning, reasoning, or understanding natural language.

Limited Generalization:
The techniques used to solve the n*n queen problem may not be easily transferable to other AI domains with different constraints and requirements.

6.CONCLUSION:-

Finding all solutions to the n queen puzzle is a good example of a simple but nontrivial problem. For this reason, it is often used as an example problem for various programming techniques, including nontraditional approaches such as constraint programming, logic programming. Most often, it is used as an example of a problem that can be solved with a recursive algorithm, by phrasing the n queens problem inductively in terms of adding a single queen to any solution to the problem of placing n-1 queens on an n-by-n chessboard.

7.Future scope :

- Developing more efficient algorithms for larger N.
- Exploring hybrid approaches combining different techniques.
- Applying AI techniques to other constraint satisfaction problems.

Here are future directions that encompass broader constraint satisfaction problems:

1. Algorithmic Advancements:

Parallel and Distributed Computing:
Leveraging multi-core processors and cloud computing to distribute the search for solutions, potentially accelerating the process for large-scale problems.

Approximate Solutions: Developing algorithms that find acceptable solutions within reasonable time constraints, even if not exact, for scenarios where optimality is less critical than speed.

2. Hybrid Approaches:

Combining AI Techniques:
Merging different AI paradigms like constraint programming, evolutionary algorithms, and neural networks to create more robust and adaptive problem-solving systems.

Incorporating Domain Knowledge:
Integrating expert knowledge into AI algorithms to guide the search process and improve efficiency in specific domains.

3. Theoretical Advances:

Complexity Analysis:
Refining our understanding of the computational complexity of constraint satisfaction problems to identify potential boundaries and guide algorithm development.

Unifying Frameworks:
Developing overarching frameworks to encompass diverse constraint satisfaction problems, enabling the transfer of knowledge and techniques across domains.

4.Real-World Applications:

Scheduling and Resource Allocation:
Optimizing resource allocation in healthcare, manufacturing, logistics, and other industries to improve efficiency and reduce costs.

Design and Engineering:
Optimizing the design of complex systems such as aircraft, power grids, and communication networks to meet performance and safety requirements.

Finance and Economics: Modeling financial markets and economic systems to better understand and predict their behavior.

5. Emerging Research Directions:

Explainable AI:
Developing AI algorithms that can explain their reasoning and decision-making processes, fostering trust and understanding in real-world applications.

Fairness and Bias:
Addressing potential biases in AI algorithms for constraint satisfaction problems to ensure fair and equitable outcomes.

Adversarial Robustness:
Building AI systems that are resilient to adversarial attacks, ensuring their reliability in sensitive domains.

References:

[Blu1928] L.M. Blumenthal. Discussions: An extension of the Gauss problem of eight queens. *The American Mathematical Monthly*, 35(6):307–309, 1928. doi>

[BM1999] A.P. Burger and C.M. Mynhardt. Queens on hexagonal boards. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 31:97–111, 1999.

[BM2000a] A.P. Burger and C.M. Mynhardt. Properties of dominating sets of the queens
graph *Q*4*k*+3. *Utilitas Mathematica*, 57:237–253, 2000.

[BM2000b] A.P. Burger and C.M. Mynhardt. Small irredundance numbers for queens graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 33:33–43, 2000.

[BM2000c] A.P. Burger and C.M. Mynhardt. Symmetry and domination in queens' graphs. *Bulletin of the Institute of Combinatorics and its Applications*, 29:11–24, 2000.

[Bea1989] J.D. Beasley. The mathematics of games. In *Recreations in Mathematics, volume 5*. The Clarendon Press - Oxford University Press, 1989.

[Beh1910] H. Behmann. Das gesamte Schachbrett unter Beachtung der Regeln des Achtk¨oniginnenproblems zu besetzen. *Mathematisch-Naturwissenschaftliche Bl¨atter.*