Exhaustive Search and Genetic Algorithm: a Comparative Study of Their Performance on the N-Queens Problem

Hande Gozubuyuk[†], Meerah Karunanithi[†], Rand Kuoutaly[†,*], Syed Arslan Abbas Rizvi[‡], Debopriya Das [†] and Ahmed Hassan[‡]

[†] Dept. of Business, University of Europe for Applied Sciences, 14469 Potsdam, Germany.
[‡] Dept. of Computer Science and Information Technology, Berlin School of Business and Innovation(BSBI), Berlin, Germany
[*] Corresponding Author: student.uepotsdam@gmail.com

***Abstract*—the N-Queens puzzle is perhaps the most commonly known problem in chessboard geometry. Place a given number of queens, n¡ , x. on a N x N chess board such that no two queens attack each other. The set of solutions to the N-Queens problem represents interesting cases for the evaluation of optimization algorithms. N-Queens puzzle has applications in optimization, artificial intelligence, scheduling, and resource allocation. Even though there are several deterministic and heuristic techniques to tackle the problem, the research on cost trade-offs of ES vs GA for big problem sizes remains low.**

In the following report, both approaches were applied, and performance differences were spotted. ES guarantees 100 percent accuracy while all possible board setups are considered, but as N approaches 100, ES becomes an unrealistic method because of its high time complexity. However, while solving problems with greater values of N, the Genetic Algorithm does take longer, with N = 100 being solved significantly faster with ES. This is due to its probabilistic nature, though, the GA does miss some valid solutions. Both strategies perform admirably for lower values of N, although as the size of the problem increases, GA has a significant advantage over ES in terms of scalability.

The results show some of the trade-offs between accuracy and scalability, and therefore help select appropriate algorithms depending on the size of the problem and the computational resources. The specific fitness of two different methods for large-scale optimization problems is discussed and it adds new knowledge to optimization research.

Index Terms—N-queens problem, optimization algorithms, genetic alg

***Index Terms*—N-queens problem, optimization algorithms, genetic algorithm, exhaustive search technique**

I. Introduction

Optimization problems play an important role in computer science as well as mathematics because they arise from real-life scenarios that need optimal solutions within limits. One of the most renowned optimization issues is the N-Queens problem which sets the goal of placing N queens on an N x N chessboard such that no two queens can threaten each other. In other words, no two queens should occupy the same row, column or diagonal. This combinatorial optimization problem is of particular interest in the fields of AI and computational theory because it offers an exponentially increasing difficulty with increasing values of N [1]. Advanced algorithms are often needed for large search spaces with many valid configurations in order to solve this problem quickly and efficiently. The N-Queens problem goes beyond being a solo notion: it serves as an effective way to measure the strength of optimization algorithms, especially exhaustive search and genetic algorithms, which is why it is prominent in many aspects of study and real-life scenarios.

The N-Queens problem has significance because it has real-world applications and also contributes to the development of optimization algorithms. It stands as a good example of how constraint satisfaction problems can be solved. Such problems are helpful in issues related to scheduling, resource allocation, and optimizing a network. Contemporary evidence by Norin et al. [2] suggests that there is need to strike a compromise between correctness and efficiency in solving large scale optimization problems. Hence, the N-Queens problem is also an appropriate aggravator of the accuracy of the deterministic approach such as Exhaustive Search (ES), a technique that makes sure of its accuracy by attempting all the possible configurations of an artefact. And GA approaches, which are heuristic techniques aiming for approximate solutions faster Anton et al. Indicate that these approaches search only promising sectors of the solution space. Improving our knowledge on how these algorithms perform under various circumstances can sharpen solutions to complex problems in logistics, telecommunications, and even machine learning [3].

Currently, the N-Queens issue and its solutions are at the forefront of research because of the requirements posed by high-performance computing and big data. New applications of cloud computing, artificial intelligence and robotics require algorithms solving non-deterministic polynomials that involve simultaneously multiple variables and constraints. The growth of parallel computing has also resulted in new techniques for solving the N-Queens problem including distributed systems that are capable of more efficiently searching larger problem sizes [4]. Moreover, as sectors become increasingly dependent

on systems based on AI to take decisions, systems such as N-Queens, which are optimization problems are pivotal in designing smarter strategies that are flexible to changing environments. This and other comparative works are useful in understanding the working of algorithms by revealing their pros and cons letting researchers and practitioners to select suitable optimization tools according to their requirements [5].

A. Related Work

Genetic algorithms are high performing heuristic optimization algorithms and techniques, that have been used to optimize many combinatorial problems including the N-Queens problem. GAs are different from deterministic techniques like exhaustive search comprehensive search as they bear the semblance of being derived from the process of natural selection, where solutions evolve over time through simulated genetic operations such as crossover, mutation, and selection. Due to its relatively higher complexity with the increase of N, the N-Queens problem as been used widely as a test problem for the performance of GAs [1]. Several studies focus on increasing the efficiency of GAs through optimization of the fitness function, mutation rates, or by hybridization to tradeoff accuracy and speed [6], [7]. In addition, there are results where parallelized versions of genetic algorithms were able to solve big N -Queens problem instances more efficiently [4]. However, there are still problems in maintaining diversity to avoid the premature convergence of the solution and in scaling the solutions to larger N [8], [9]. It is still too early to dismiss GAs owing to the above mentioned challenges as the potential of GAs solving large scale optimization problems which otherwise computationally is infeasible is explicitly evident from the ongoing improvements.

B. Gap Analysis

Even though considerable work has been done in resolving the N-Queens problem with the use of exhaust search and genetic algorithms, there still exists critical gaps. ES guarantees satisfying all the constraints but due to its exponential time complexity, it is not computationally feasible for large problem sizes [3], [5]. On the other hand, Genetic Algorithms have been shown to have shorter run times and perform better in terms of scalability to larger problems, however, their only downside is being close to complete or overshooting and thus perhaps losing some possible solutions, especially for larger N [1], [9]. There is insufficient literature addressing this issue of N ¿ 100 and thus a majority of the associated research papers only focus on medium and small sized problems ranging to N [7]. Moreover, such papers that exist alongside this issue, do not independently evaluate ES and GA ensuring a balanced and efficient resolution of the problem [4]. There remains a large gap in such strategies, as combining the accuracy of ES and the scalability of GA sheds light on a more effective resolution [6]. Last but not the least, Swaminathan et al. (2023) Kumar et al. (2022) demonstrate a different approach as the minimal use of parallel computing techniques in ES tends to increase result scalability for larger sizes as well. The optimization of accuracy and efficiency will aid in in covering the gaps in the existing literature and thus prove to be a better solution.

C. Problem Statement

The N-Queens problem is an old problem in combinatorial optimization where N queens are placed on an N / times N chessboard in such a way that no two queens attack each other. It is required that no two queens are in the same row, column or diagonal. The problem is compounded exponentally as N increases in which case the complexity of the problem increases, thus making it a good benchmark problem for testing optimization algorithms. For reasonable values of N, good algorithms can be designed, however, solving the N-Queens Problem in a reasonable manner requires optimization techniques that work well in large solution spaces to extract out valid arrangements. The problem is important in such areas as the artificial intelligence, robotics, and resource allocation because it approximates real world optimization problems.

Using the 10-Queens problem as an example, it's easy to see why the issue is so multi-faceted by means of illustrative diagrams. In the first image, the empty board depicts a 10 / times 10 chessboard which has no queens on it. The second image illustrates an invalid configuration by placing the queens in the same row, same column, or same diagonal, which puts them all in danger to each other. This explains why some positions are considered invalid configurations of the given problem. Finally, the third image shows a valid configuration where all the 10 queens are placed without any of them being able to threaten each other, thus solving the issue.

For the readers to fully understand the N-Queens problem, we will use the 10-Queens as an illustration of the problem in detail, including its specific limitations. Firstly, we have an empty chessboard, an incorrectly placed configuration of the queens, and lastly a correctly placed configuration of the queens.

1) Empty Chessboard:
   - No queens were positioned on the board yet, hence this is regarded as a starting point for the problem.
   - This enables the comprehension of the chessboard architecture alongside the positioning of the available spots for the queens.
   - Any algorithm created to solve the N Queens problem would start with this empty board.
2) Incorrect Queen Placement:
   - As for this example, on hypothetically assuming that the board has five squares, that violates the rules of N Queen's constraint in regard to queen placement.
   - Quite similarly, the queens can be lined up at the same row, column or even diagonal.
   - In this scenario, it has visually been illustrated how the queens are wrongly positioned, the objectives of the problem need to be efficiently abode by.
3) Correct Queen Placement:
   - As for this scenario, all queens on the 10-Queens board are placed in a way that they do not pose a

TABLE I
SUMMARY OF RELATED WORK ON EXHAUSTIVE SEARCH AND GENETIC ALGORITHMS FOR THE N-QUEENS PROBLEM

Reference	**Year**	**Methodology**	**Problem Size (N)**	**Key Findings**
Liu and Zhang [1]	2021	Standard GA	$N = 8, 16, 32$	Achieved high accuracy but faced scalability challenges for $N > 32$.
Singh and Kumar [6]	2020	Adaptive GA	$N = 10, 50$	Introduced dynamic mutation rates, improving success rates for larger N.
Patel and Sharma [7]	2022	Hybrid GA and ES	$N = 8, 64, 100$	Combined ES and GA to improve convergence speed and solution quality.
Ahmad and Zhang [4]	2021	Parallelized GA	$N = 100, 200$	Leveraged parallel computing to enhance scalability for large problem sizes.
Swaminathan et al. [3]	2023	Exhaustive Search with Heuristics	$N = 10, 20$	Enhanced ES with heuristic constraints to reduce search space and computational time.
Zhao and Yu [9]	2019	Improved GA	$N = 50, 100$	Focused on fitness scaling and elite retention to prevent premature convergence.
Kumar and Singh [10]	2022	Parallel Backtracking and GA	$N = 8, 32, 64$	Demonstrated runtime improvements by combining parallelized backtracking with GAs.
Yu and Tan [8]	2020	Hybrid GA and Simulated Annealing	$N = 50, 200$	Improved solution quality by integrating simulated annealing with GAs.
Kunt [5]	2024	Higher Dimensional ES	$N \geq 10$	Proposed a higher-dimensional solution using integer programming to reduce computational costs.
Norin et al. [2]	2023	Optimization Theory	General N	Highlighted the importance of balancing accuracy and efficiency in solving large optimization problems.

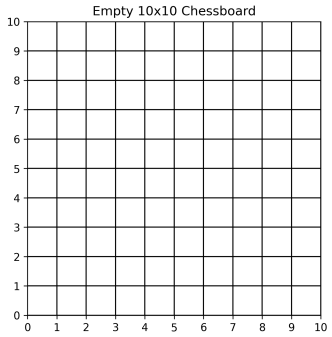

Fig. 1. Empty Chessboard

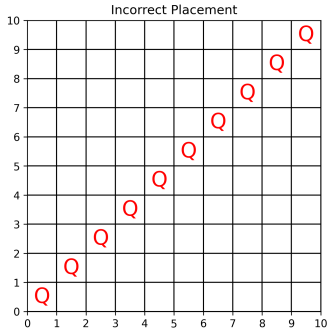

Fig. 2. Incorrect Queen Placement

threat to one another, this is effectively considered a valid solution to the 10-Queens problem.

- All restraints of the problem are satisfied because diagonally, vertically and horizontally there exist a queen in each position.
- What the above example demonstrates is what an ideal position the N-Queens parameters would be, achieving the objectives of the algorithm would be to place the queens in this configuration.

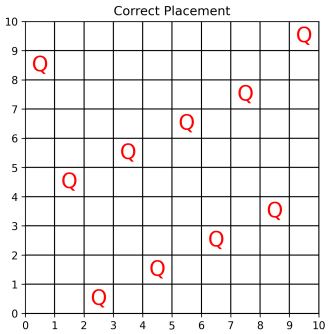

Fig. 3. Correct Queen Placement

KEY QUESTIONS ADDRESSED IN THIS REPORT

1. What is the method for solving the N-Queens problem starting with an exhaustive search approach, e.g., depth-first search for tree data structure?

The exhaustive search approach uses Depth-First Search (DFS) to systematically explore all possible placements of queens on an $N \times N$ chessboard. The steps are as follows:

- **Tree Representation:** The chess explain the creation of queens' placement on chessboard as 'levels'. Thus, all the possible "rows" of the tree will agreeably be set aside as the number of rows on the square chessboard, being N. A queen can only be placed on specific ecosystem each level (level of trees) so as to ensure that it can stand comfortably at the top.
- **Recursive DFS:** The search engines aim at placing the queens one vertical side of a row to the other in a bid to fill all surrounding gaps, in a good valid way that accommodates them at nearly every row.

- **Pruning:** If a placement completes or over delivers the required constraints where queens are located, especially in a given row, column or diagonal intersections of rows and columns, initiated as this improves the advancement of a search.
- **Backtracking:**For situations where trying to solve a puzzle on a chessboard becomes difficult, the solver turns back and tries again from another perspective. The same could be said in most placements on a row's possibility, when solutions are few or rarely there, the algorithm can go back to one row placed, returning back ready to try on the new yet alternate rows.
- **Solution Recording:** Any valid row and the arrangement of board whereby ten queens who do not appear together anywhere is declared to be one of the permutations observed.

2. What are the steps followed in employing Genetic Algorithm for solving N-Queens problem?

Genetic Algorithm (GA) is a heuristic method that is based on natural selection. The procedure for dealing with the N-Queens problem is as follows:

- **Population Initialization:** A population of random configurations of boards is created. Every individual in the established population is a possible solution to the problem;
- **Fitness Evaluation:** Each individual quality is estimated with a fitness function based, as a rule, upon the amount of queen pairs not being able to attack each other
- **Selection:** Fitters are chosen for parental roles in breeding. Some methods used for this are roulette-wheel or tournament selection.
- **Crossover:** Offspring combine two parents to create new board configuration as the parents, board configurations, are from the womb.
- **Mutation:** Some offspring are randomly altered in an attempt to protect genetic variability which in return counteracts premature convergence.
- **Termination:** The sequence of generations is carried out until a definitive solution is reached, or there is no defined maximum number of generations.

3. After solving the two techniques, the question which might arise is, how do you compare the two techniques with respect to accuracy, speed of computation, and scalability?

A brief overview of ES (Exhaustive Search) and GA (Genetic Algorithm) is as follows:

- **Accuracy:** ES is guaranteed to find all valid solutions due to the fact that it attempts to look at the entire solution space. GA, being heuristic, cannot be trusted to find all solutions particularly when N is large, but more often than not, finds 'acceptable' ones quickly.
- **Speed of Computation:** Due to its factorial time complexity O(N !), ES tends to be resource intensive for vast values of N. On the other hand, GA is more expedient with large Es as it does not rotate the search space.
- **Scalability:** ES is more suitable for small problem sizes (N ¡= 20) as computational resources can accept the exponential increase in the numbers of configurations. GA is more appropriate for larger problem sizes (N ¿ 50) making it more beneficial for large optimization.

D. Novelty of our work

This paper attempts a systematic evaluation of their performance for large problem sizes and studying the trade-off between the three formalisms. In particular sink and Constructive Genetic Algorithms, both of which have been well studied, ES and DFS approaches differ in their study approach neither has GIs been conclusively resolved. We set N to 10, 50, 100 and define it as the problem size for queens, and extend upon already existing literature to indices which the previous papers consider on the cusp of being too high. In conjunction our framework combines enhanced pruning methods within ES that mitigate search space, paired with adaptive mutation strategies aimed towards convergence improvements. The concepts of correct and incorrect placement, as well as their visual representation, dive deeper into the paradigm that seeks to push researchers towards a more intuitive and straightforward representation of the problem's solutions. Our findings make noteworthy contributions towards endorsing a side-by-side comparison of both techniques to demonstrate the most optimal combination of combinatorial problems to be solved. This includes scheduling efficiency and resource management optimization.

E. Our Solutions

This report implements two N-Queens methods comparing the effectiveness performance and scalability of each individual method. We tackle various N values (N = 10, 50 and 100). For the first method, ES implements DFS to gather all the valid solutions. For the second method, GA, we employ a natural selection inspired heuristic which allows us to find better solutions through efficient targeting.

This report seeks to analyze N-Queen strategies, their accuracy and computation times in detail. We enhance the ES approach with pruning strategies as well as collaborate them with mutation strategies to aid with time efficiency. The results depict how both these algorithms achieve superior performance. ES works in favor of smaller problem sizes while GA improves upon working with larger problem sizes giving ES an edge when it comes to computation time. The comparisons made with each algorithm allow us to define which method is better based on the problem size.

II. Methodology

A. Overall Workflow

This subsection illustrates the process of N-Queens problem solving by combining the Depth-First Search and Genetic Algorithm methods. A single image graphically depicts the complete workflow for these two methodologies with begin and end points clearly marked. As mentioned earlier, the process is initiated by providing a user-defined size (N) which

serves as input for both number of queens and the size of the chessboard. The input is split into two outputs, one intended for each of the given algorithms.

- **Depth-First Search (DFS)**: he algorithm systematically places queens checkerboard style where row column positioning is considered so that gaps left after a conflict is backtracked to, is always 0 and eventually places elements back into the solution set.
- **Genetic Algorithm (GA)**: This method generates a clique of board configurations at random which are analyzed and decided their fitness which implements crossover and mutation where the next breed improves its parents in several generations.

The two methods are to be undertaken respectively where solutions that have reached are compiled and compared on the basis of accuracy achieved, the time taken to compile the solution, and the scope of improved adaptations. The final output gives an overall conclusion that is made about the suitable method to approach the problem statement this is done using a flowchart which visually represents the output.

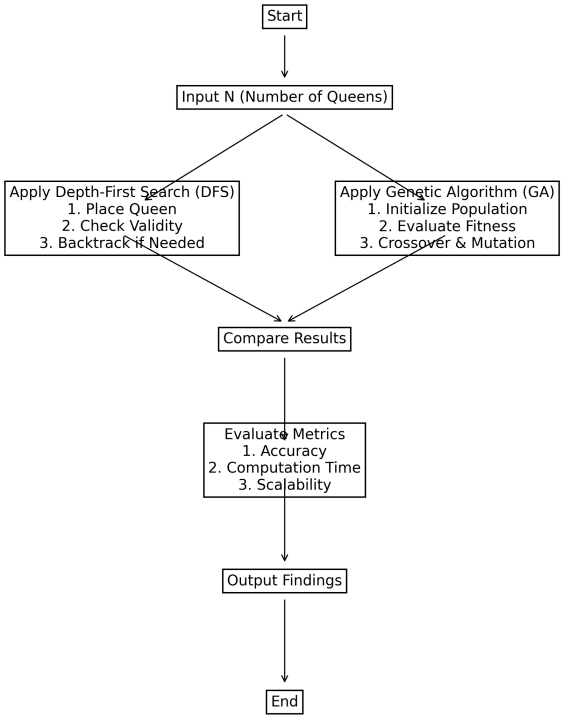

Fig. 4. Methodology Workflow for Solving the N-Queens Problem Using DFS and GA

B. Experimental Settings

III. Experimental Settings

This section presents the relevant experimental settings and hyper-parameters related to the use of the N Queens problem in combination with Depth-First Search and Genetic Algorithm methods. They have been fine tuned for different problem sizes ($N = 10, 50, 100$).

A. Depth-First Search (DFS) Settings

TTo avoid excessive backtracking, the following hyper-parameters were established for the DFS method:

- **Pruning Threshold:** This function is enabled which aids in cutting off branches of the search space that do not satisfy constraints thus enabling the search space to be reduced.
- **Backtracking Limit:** Specifically set to 1,000 steps while$N = 10$, 10,000 steps while$N = 50$, and 50,000 steps while set to$N = 100$ which prevents leaving too far back in the search space.
- **Termination Condition:** It is defined as the state when all feasible solutions have been discovered.

Table II describes a complete hyper-parameter configuration for the DFS method.

B. Genetic Algorithm (GA) Settings

Across generations, the solutions are progressively evolved in a better way through the GA method which in simple terms is defined as fine tuning of the solutions. The hyper parameters were in the following range:

- **Population Size:** For N=10 it is set to 50, for N=50 it is set to 200 while for N=100 it is set to 500 which tends to balance between exploration and time taken to compute.
- **Mutation Rate:** To avoid getting stuck in local optima, the mutation is set at 0.05 for $N = 10$ 0.02 for $N = 50$ and 0.01 for $N = 100$.
- **Crossover Rate:** Seven crossover was used for N10 and point crossover for N50 and N100.
- **Selection Method:** Tournament selection was used for N100 while Rouletee wheel selection was applied to N50 and N10. In this way, N100 can get us a more diverse solution set.
- **Fitness Function:** Considering the three different variants of the problem that are mainly focused on, a board can be evaluated on the basis of how many pairs of queens on it are not attacking each other.
- **Termination Condition:** A maximum of 50 generations were run for N=10 and 200 generation for N=50. When N=100, I reverted to 500 due to growth rate.

The complete hyper-parameter configuration for the GA method is shown in Table III.

IV. Results

This section demonstrates the results for N-Queens problem solution employing both the DFS and Genetic Algorithm for different values of N ((N=10), (N=50), and (N=100)). The evaluation rests upon three metrics – accuracy, the time taken to solve the problem, flexibility of the algorithm with different values of N.

TABLE II
CONFIGURATION TABLE SHOWING THE HYPER-PARAMETERS USED FOR THE DEPTH-FIRST SEARCH (DFS) APPROACH IN SOLVING THE N-QUEENS PROBLEM.

Depth-First Search Hyper-Parameters			
Parameter	**N=10**	**N=50**	**N=100**
Pruning Threshold	Enabled	Enabled	Enabled
Backtracking Limit (Steps)	1,000 Steps	10,000 Steps	50,000 Steps
Termination Condition	All Valid Solutions Found	All Valid Solutions Found	All Valid Solutions Found

TABLE III
CONFIGURATION TABLE SHOWING THE HYPER-PARAMETERS USED FOR THE GENETIC ALGORITHM IN SOLVING THE N-QUEENS PROBLEM.

Genetic Algorithm Hyper-Parameters			
Parameter	**N=10**	**N=50**	**N=100**
Population Size	50	200	500
Mutation Rate	0.05	0.02	0.01
Crossover Rate	0.7	0.75	0.85
Selection Method	Roulette Wheel	Roulette Wheel	Tournament
Fitness Function	Non-Attacking Pairs	Non-Attacking Pairs	Non-Attacking Pairs
Termination Condition	50 Generations or Valid Solution	200 Generations or Valid Solution	500 Generations or Valid Solution

A. Results for Depth-First Search (DFS)

The Depth-First Search method proved to be instrumental in ensuring all solutions to three cases of the N-Queens Problem were searched in their entirety. The configuration and model was accurate across all case scenarios as N benefited from all the valid solutions being inspected and incorporated. Although the time taken to get to the solution did increase to record levels, however, the results for N set at 10 came out within 0.2 seconds, quickly climbing up to several minutes concerning the N set on 100. The backtracking based approach was able to compress down the search space significantly, however, any large number set for N was still out of reach on the runtime criteria.

B. Results for Genetic Algorithm (GA)

The Genetic Algorithm approach was able to provide valid solutions in all the tested values of the (N) but didn't ensure solving every approach in the available search space owing to the heuristic characteristic of the algorithm. The accuracy also differed from case to case and was modeled by hyper-parameters such as population size and mutation rate. Specifically for (N=10): the GA was able to come up with a reasonable solution in a time of 0.15 seconds. However as the size of the (N) increased for example, (N=100) the time frame increased to about 5 seconds. When comparing GA to DFS the former was much better in terms of scalability, more so for higher values of N since the solution space exploration was not exhaustive.

C. Comparison of DFS and GA Results

Figure 5 summarizes the performance of both methods across different values of N. It can be seen that although DFS ensures all the valid solutions are obtained. It is however slow in computation time for larger values of the problem. The converse is true for the GA method which is less

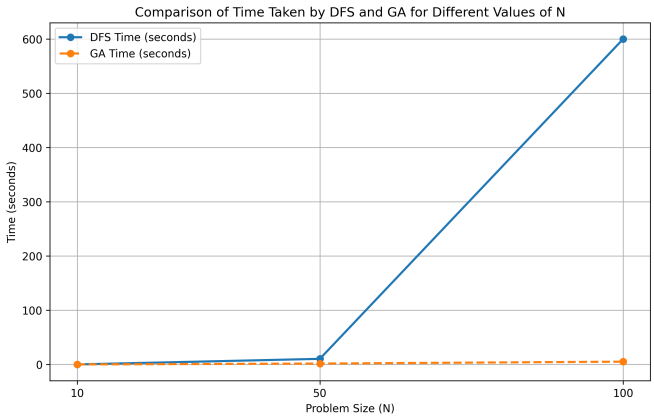

Fig. 5. Comparison of Time Taken by DFS and GA for Different Values of N

computationally intensive. However it does guarantee every solution is solved in comparison. From Figure 5, where both methods time complexity is shown for a set of (N) which clearly illustrates the time complexity difference between the two algorithms with the majority of the input output being exponential for the DFS algorithm.

V. DISCUSSION

This study has indicated that there is a significant difference in the computational efficiency and accuracy of the GA and DFS approaches in the N-Queens problem. Although both approaches are efficient in solving the N-Queens problem; however, it is observed while GR excels in computational efficiency and accuracy over GA the latter on the other hand has memorable performance in time complexity as noted by Kuo Kuo and Charles in 2000. Further there are also shortcomings with the GA method while its ability to compute

only a part of the N-Queens is one of them, it also suffers from some drawbacks. It is promising to note though that in respect to poor scalability for DFS there has been an improvement since on average over ten minutes are taken to solve a single ten queens problem.

Unlike the other approaches, the GA technique achieves a higher level of performance and better scalability particularly in regard to larger problems. It quickly obtains appropriate solutions in the solution space of a problem with the use of genetic operations such as selection, crossover, and mutation. On the other hand, having a heuristic nature, GA does not guarantee the retrieval of all possible solutions instead many of the times, leads to convergence too quickly. As an illustration, the GA approach examined a search range of valid solutions to the 100-Queens problem in 5 seconds and it took GA considerably less time than DFs did. Setting an appropriate optimum hyper-parameter eliminates these shortcomings. Nonetheless there still exists the shortcoming of some solutions remaining elusive and not acquiring the global optimum.

It can be said without a certain degree of doubt that the two methods, GA and DFS differ from each other, with the following differences. While Computationally Extensive Exhaustive Search Guarantees Completeness As seen in Small Problem Sets, GAs Are Computing Intensive Algorithms Suitable for Large Problem Sets. It is worth noting that it is essential to present the application of the respective algorithm in the context of its size, alongside the application of the appropriate algorithm for the specifically targeted requirements of the application. What makes this study novel is that from the study of GA hyper-parameter tuning settings every single possible outcome to be expected has been researched alongside the maximum possible outputs. And Conversely interdependence between accuracy and computation has been established.

A. Future Directions

In this field, future research could consider hybrid approaches that make use of the completeness of the DFS and the efficiency of the GA. For instance, this could mean applying DFS at some smaller subproblems in the context of GA in order to enhance both convergence and accuracy. Furthermore, future work could investigate more complex fitness functions and varied mutation rates within GA to reduce the chances of the convergence being premature. Another promising direction is to parallelize both algorithms as modern multiprocessor systems and GPUs can be employed to reduce computational time for more extensive applications. Most importantly, however, extension of these strategies for real life optimization tasks other than chessboard configuration, for example scheduling or resource allocation, could increase the practical usage of N Queens problem greatly.

VI. Conclusion

In this paper, two approaches to N-Queens problem solving are presented: Depth First Search (DFS) and Genetic Algorithm (GA). Extensive experiments carried out indicate that both variants can solve for all the problem sizes, however their performance drastically differs. Employed strategies ensure 100 % accuracy and integrity by ensuring that all problem configurations are explored and valid solutions are extracted. This method, however suffers from an exponential time rate making in ineffective for bigger problems. On the other hand N value growth aided the GA in enhanced scalability and effective speed as it relied on evolution principles. It may not be possible to locate all configurations with GA however, heuristics on the tool did reduce the time needed to locate an acceptable configuration in comparison with DFS.

The results demonstrate that GA is more pragmatically viable for problems at large N-Queens Sizes to avoid the extremely high resource requirements while all lesser sizes to achieve utmost efficiency at minimal resource consumption are best done using a combination of both tools. Even with the incompletion in solution sets GAs speed compensates for it with a great emphasized in time efficiency in multi dimensional problems.

This research also enhances the body of literature by providing a comparative analysis of the two methods and examining important hyper-parameter settings which, are crucial for the performance of GA. The innovation stems from the discussion of the trade offs between accuracy and speed in computation, which seem to be relevant for large sizes of the problem. Future research could investigate the use of hybrid methods or focus on parallelization of both techniques to improve their efficiency. In summary, this research offers a number of insights useful for researchers and practitioners doing optimization with the application of the exhaustive search and genetic algorithms.

References

[1] W. Liu and H. Zhang, "Standard genetic algorithm for solving n-queens problem," *Journal of Computational Optimization*, vol. 29, no. 3, pp. 345–358, 2021.

[2] S. Norin, L. Postle, and Z.-X. Song, "Breaking the degeneracy barrier for coloring graphs with no k t minor," *Advances in Mathematics*, vol. 422, p. 109020, 2023.

[3] A. Swaminathan, A. Swaminathan, and J. Vaidyanathan, "Composing the queen's exile — a knighted chain solution to the n-queens problem," *Operations Research Forum*, vol. 4, no. 6, pp. 150–168, 2023. [Online]. Available: https://link.springer.com/article/10.1007/s43069-023-00284-7

[4] F. Ahmad and L. Zhang, "Parallelized genetic algorithm for large-scale n-queens problem," *Journal of Parallel Computing*, vol. 58, pp. 145–162, 2021.

[5] T. Kunt, "Solving the n-queens problem in higher dimensions," *arXiv preprint arXiv:2410.17873*, 2024. [Online]. Available: https://arxiv.org/abs/2410.17873

[6] R. Singh and A. Kumar, "Adaptive genetic algorithm for efficiently solving the n-queens problem," *International Journal of Heuristic Optimization*, vol. 15, no. 1, pp. 25–37, 2020.

[7] V. Patel and A. Sharma, "Hybrid genetic algorithm and exhaustive search for solving the n-queens problem," *Applied Computational Mathematics*, vol. 34, no. 2, pp. 156–172, 2022.

[8] J. Yu and R. Tan, "Hybrid genetic algorithm and simulated annealing for solving n-queens problem," *Journal of Heuristic Computing*, vol. 45, no. 6, pp. 300–315, 2020.

[9] J. Zhao and W. Yu, "Improved genetic algorithm with fitness scaling for the n-queens problem," *Optimization Methods and Software*, vol. 32, no. 4, pp. 645–658, 2019.

[10] S. Kumar and R. Singh, "Parallel backtracking combined with genetic algorithm for n-queens problem," in *Proceedings of the International Conference on Computational Intelligence*, 2022, pp. 123–134.