COMPARATIVE STUDY OF DIFFERENT ALGORITHMS TO SOLVE N QUEENS PROBLEM

Soham Mukherjee[1], Santanu Datta[1], Pramit Brata Chanda[2] and Pratik Pathak[1]

[1]Computer Science & Engineering, M.Tech, Academy of Technology, Hooghly, India
[2]Computer Science & Engineering, M.Tech, Kalyani Govt. Engg. College, Nadia, India

ABSTRACT

This Paper provides a brief description of the Genetic Algorithm (GA), the Simulated Annealing (SA) Algorithm, the Backtracking (BT) Algorithm and the Brute Force (BF) Search Algorithm and attempts to explain the way as how the Proposed Genetic Algorithm (GA), the Proposed Simulated Annealing (SA) Algorithm using GA, the Backtracking (BT) Algorithm and the Brute Force (BF) Search Algorithm can be employed in finding the best solution of N Queens Problem and also, makes a comparison between these four algorithms. It is entirely a review based work. The four algorithms were written as well as implemented. From the Results, it was found that, the Proposed Genetic Algorithm (GA) performed better than the Proposed Simulated Annealing (SA) Algorithm using GA, the Backtracking (BT) Algorithm and the Brute Force (BF) Search Algorithm and it also provided better fitness value (solution) than the Proposed Simulated Annealing Algorithm (SA) using GA, the Backtracking (BT) Algorithm and the Brute Force (BF) Search Algorithm, for different N values. Also, it was noticed that, the Proposed GA took more time to provide result than the Proposed SA using GA.

KEYWORDS

Tractable and Intractable Problems, N Queens Problem, Genetic Algorithm, Simulated Annealing Algorithm, Backtracking Algorithm, Brute Force Search Algorithm, Fitness, No. of Solutions, Time.

1. INTRODUCTION

Depending on the classification of functions into polynomial and exponential, we can divide computational problems into two types-tractable and intractable. So, the tractability of a problem depends on how difficult the problem is w.r.t. the amount of time it takes to successfully solve the problem. It has very close link with the time complexity of a problem. If a problem has given solution in a small amount of time, then it can be easily solved in polynomial time and named as tractable problem. But, there are some problems, which can only be solved by some algorithms, whose execution time grows very quickly in case of larger input size and these problems cannot be solved in polynomial time by a conventional algorithm. These problems are named as intractable [24, 29, 23]. The N Queens Problem is a classical intractable problem, which is often used in case of discussing about various types of searching problems. In general, the goal is to place N number of queens on the (N*N) matrix, so that no two queens can threaten one another. According to the rule of this problem, a queen can move in either along a row, or along a column, or along a diagonal. In an (N*N) matrix, each of the N queens will be located on exactly one row, one column, and two diagonals [4, 25]. The rest of the paper is organized as follows. The

DOI:10.5121/ijfcst.2015.5202

Overview of Genetic Algorithm is discussed in Section 2. The Overview of SA Algorithm is narrated in Section 3. The Overviews of Backtracking and Brute Force Search Algorithms are given in Section 4 and 5 respectively. A Theoretical Comparison among the Used Algorithms is made in Section 6. The Proposed Work and Algorithms are explained in Section 7. The Results are shown in Section 8 with Analysis and Discussion in Section 9 and ultimate Conclusion is made in Section 10, followed by Future Works in Section 11, Acknowledgement and finally, References.

	Q1		
			Q2
Q3			
		Q4	

Fig. 1: Four Queens' Position in a (4*4) Chess Board

2. OVERVIEW OF GENETIC ALGORITHM

Genetic Algorithm is an optimization algorithm, which follows the concept of natural selection. It is a probabilistic search method, based on the ideas of evolution. It follows the Darwinian's Survival of the Fittest Principle. Unlike the traditional techniques, it is suitable for many real world problems, in which the goal is to find optimal solution. It is a method, which completely works on chromosomes. It differs from classical search algorithms. It searches among a group of points, rather than a single point; and operates with a coding of parameter set, not the parameters themselves. The method of genetic algorithm is probabilistic; whereas, traditional algorithms use deterministic methods. Because of these characteristics of genetic algorithm, it is widely used as a very much general optimization algorithm. It also searches in irregular search areas and hence, it is applied to various function optimizations. In case of GA, a population of strings is used, and these strings are named as chromosomes. Genetic Algorithm can reach to a solution, close to the global optima. It has advantages over traditional algorithms, which cannot always achieve a global or close to global optima. But, it does not always guarantee optimal solution, because of its randomness [5, 19]. The steps of Genetic Algorithm are given in Fig. 2.

Step 1: Randomly generate an initial population of solutions having a pool size (Parent Pool).
Step 2: Evaluate each solution in the population and calculate its fitness value.
Step 3: Select better solutions (chromosomes) based on fitness values and reject the rest of the solutions (chromosomes).
Step 4: If suitable solution(s) is/are found in the current generation or maximum number of generations has been completed, then stop.
Step 5: Otherwise, Change the population using crossover and mutation to generate a new population of solutions, having a pool size (Child Pool).
Step 6: Copy the Child Pool to the Parent Pool.
Step 7: Go to step 2 and continue in this way, until reach the desired solution [27].

3. OVERVIEW OF SIMULATED ANNEALING ALGORITHM

Simulated Annealing (SA) is a probabilistic search and optimization method. It is very much used, when the search space is discrete. It follows "Annealing", which is a process, in which metals are cooled gradually to make them reach a state of less energy, where they are very much strong. Simulated Annealing technique is a meta-heuristic optimization algorithm. The random movement occurs at high temperature and at low temperature, there is very less randomness. Simulated Annealing, at each step, chooses a variable randomly; and then, it also chooses a value randomly. If giving that value to the variable does an improvement or keeps the conflicts in same number as earlier, then this algorithm accepts the allotment and there comes a new current allotment. Otherwise, it allows the allotment with some probability, depending on temperature and how much bad the allotment is than the current allotment. If the change to the system is unaccepted, then the current allotment is totally unaltered [3, 15]. It is better than Hill Climbing Algorithm, where bad states are not accepted at any cost and global optima may not be reached in case of most of the problems. The steps of SA Algorithm are given in Fig. 3.

Step 1: Start with a random initial placement. Initialize a high temperature.
Step 2: Modify the placement through a defined change.
Step 3: Calculate the difference in the energy due to the change made.
Step 4: Depending on the difference in energy, accept or reject the change.
Step 5: Update the temperature value by reducing the temperature very slowly.
Step 6: Go back to Step 2 [16, 18].

Fig. 3: Steps of Simulated Annealing Algorithm

4. OVERVIEW OF BACKTRACKING

It is a very general algorithm for searching and finding the solutions of some problem, which use the concept of partial candidate solution and does a quick test to check, if it can be ended to a proper solution or not. When it is applicable, backtracking is very much quicker than brute force search, as it can eliminate large number of candidates with just a single test. It is a very much important tool for solving problems like Crosswords, Sudoku and many kinds of puzzles etc. It is often a very much effective process for solving the knapsack problem and some other optimization problems. It depends on the user-defined procedures, those defining the problem; the partial candidates' nature and how they are ended into complete solution. It is, therefore, not a specific algorithm – although, unlike many other non-specific algorithms, it is guaranteed to find all possible solutions of a problem in a comparatively less amount of time [20].

5. OVERVIEW OF BRUTE FORCE SEARCH

Brute Force Search, which is also known as Generate and Test, is a very well known algorithm, which examines all possible candidates for the solution and checks, whether each candidate fulfils the problem's criteria or not. A brute force search algorithm for the n queen's problem will examine all the possible placements of n number of queens on the (n*n) matrix, and, for each placement, check, whether each queen can threaten another queen or not. The basic idea of the

brute force search algorithm for n queen's problem is to place each of the n queens on all possible positions and check regularly, whether the queens threaten each other or not. If this does not occur, then it has reached a proper solution. It is very much simple to implement, and it always finds a solution, if the solution has proper existence. Its complexity grows very quickly, as increment occurs in the problem size. So, it is mostly used, when the problem size is relatively small. It can also be used, when simplicity is more crucial than speed [21, 30].

6. THEORITICAL COMPARISON AMONG THE USED ALGORITHMS

Genetic Algorithm (GA) and Simulated Annealing (SA) Algorithm are both *Optimization* Algorithms, while Backtracking (BT) and Brute Force (BF) Search Algorithms are both *not at all Optimization* Algorithms. Their methodology is quite different. Let us make a brief comparison among GA, SA, BT and BF Algorithms. In SA, we discuss about *solutions*, *temperature*, *neighbours*, *moves* etc. But with GA, we discuss about *chromosomes*, *fitness*, *selection*, *crossover*, *mutation*, *elitism* etc. SA makes a new solution by modifying a solution by making a move. But, GA makes solutions using the combination of two or three different solutions [22]. GA is a *heuristic* Algorithm, while SA is a *meta-heuristic* Algorithm. In GA, a few probabilities like *crossover probability*, *mutation probability* are there; while in SA, there is only one probability used; i.e., *probability to accept a bad state*. Unlike GA, a term *Entropy* is used in SA. It is an extremely important term in SA. Both GA and SA have some randomness. In case of BT and BF Algorithms, there is no randomness. They are very much similar, but little bit different in their methodologies. BF Algorithm uses the concept *Generate and Test*. It generates a solution and then tests that solution to check, whether it is correct or not. But, BT Algorithm builds and checks each and every partial solution and discards wrong partial solutions. BT Algorithm takes less time to solve problems than BF Algorithm. But, both these two algorithms are not at all efficient and effective to solve problems, when the input size becomes large. These algorithms are good for solving problems, when the input size is relatively small [20, 21].

7. PROPOSED WORK AND ALGORITHMS

In both of the Proposed Genetic Algorithm (GA) and Proposed Simulated Annealing (SA) Algorithm using GA, our entire concentration was focused on the fitness of the solutions; as in case of Genetic Algorithm, fitness is the most important factor, which signifies the goodness and optimality of the solution. We also had a good and conscious look on the execution time. Here, some modified approaches were applied in the conventional algorithms, to make both of the proposed algorithms. The proposed algorithms have both- some conventional steps and some modified approaches. The brief descriptions of the two algorithms are as follows. Firstly, in both algorithms, the initial population (Parent Pool) of chromosomes was randomly generated and therefore, some duplicate valued chromosomes may be present in that population. So, initially, there may be column conflicts in that population, which are completely eliminated, when mutation was done on the child population (Child Pool), which was generated after the two point crossover operation in the Proposed GA, which was performed after the tournament selection operation and this crossover may also increase the column conflicts in the population of chromosomes. Also, the same mutation operation was done on the parent population (Parent Pool), after the evaluation operation in the Proposed SA using GA. The temperature loop operation was also used as the generation loop operation; in the Proposed SA Algorithm using GA. Most important of all; as mentioned earlier, our concentration was always hard on the fitness

of the solutions. Hence, in the Proposed GA, just copying the Child Pool to the Parent Pool was not only done, but also; the best fitness of the Parent Pool and the Child Pool was taken and comparison was made between them. Then, copying the corresponding best fitness' chromosome (between the two best fitness' chromosomes) to the 0^{th} location of the New Parent Pool and showing the corresponding fitness as the "GA Fitness" in the Result (8) Section was performed. In both the proposed algorithms, the solutions were evaluated by using the formula ($^{n}c_2$ – the no. of diagonal conflicts), where $^{n}c_2$ is the maximum number of non-attacking queen pairs. The Proposed Genetic Algorithm (GA) and the Proposed Simulated Annealing (SA) Algorithm using GA are given in Fig. 4 and Fig. 5 respectively.

7.1 ALGORITHM N_QUEEN_SOLUTION BY PROPOSED GA

Input: A Population (Group) of Solutions (Chromosomes), each representing the placement of N number of Queens in (N*N) Chessboard.

Output: The Optimal Solution (Fitness), representing the placement of N number of Queens in the (N*N) Chessboard according to the rule of the N Queens Problem.

Step 1: Randomly generate the Initial Population of Queens (Parent Pool).
Step 2: Evaluate the chromosomes of the Parent Pool as:
  2.1: Fitness function is the no. of non-attacking queen pairs=$^{n}c_2$ (maximum value).
  2.2: Calculate the fitness value of each of the chromosomes as: ($^{n}c_2$) - the no. of diagonal conflicts).
Step 3: Find the best fitness' chromosome of the Parent Pool.
❖ Loop (generation)
Step 4: Perform Tournament Selection as the selection procedure.
Step 5: Now, generate offsprings (Child Pool) by performing Two Point Crossover Operation.
Step 6: Mutate the offsprings by replacing the duplicate bits of the Child Pool chromosomes by the unused ones.
Step 7: Perform Elitism operation as:
  7.1: Copy the Child Pool to the Parent Pool.
  7.2: Find the best fitness' chromosome of the New Parent Pool.
  7.3: Compare the fitnesses of the best fitness' chromosomes of the Parent Pool and the New Parent Pool.
  7.4: The Better Fitness' Chromosome is copied in the 0^{th} location of the New Parent Pool.
  7.5: If their fitnesses are equal, then the best fitness' chromosome of the New Parent Pool is copied in the 0^{th} location of the New Parent Pool.
❖ End Loop (generation)

7.1 ALGORITHM N_QUEEN_SOLUTION BY PROPOSED SA USING GA

Input: A Solution (Chromosome), representing the placement of N number of Queens in (N*N) Chessboard.

Output: The Optimal Solution (Fitness), representing the placement of N number of Queens in the (N*N) Chessboard according to the rule of the N Queens Problem.

Step 1: Firstly, Initialize 2 Temperatures-Initial Temperature (high) and Final Temperature (low).

Step 2: Randomly generate the Parent Chromosome of Queens.

- Loop (temperature T)

Step 3: Evaluate the Parent Chromosome as:

3.1: Fitness function is the no. of non-attacking queen pairs=$^{n}c_{2}$ (maximum value).

3.2: Calculate the fitness value of the chromosome as: ($^{n}c_{2}$) - the no. of diagonal conflicts).

Step 4: Mutate the Parent Chromosome by replacing the duplicate bit(s) of it by the unused one(s), to generate the Child Chromosome.

Step 5: Compute the fitness of the Child Chromosome.

Step 6: Perform Elitism operation as:

6.1: If fitness of the Child is better than or equal to that of the Parent, make it Parent for the next generation.

6.2: If Child fitness is not better than Parent fitness, calculate the probability of accepting the Child as- p=e^ ((-delta E)/T) - where- delta E is the change of energy (Parent fitness-Child fitness). Then, make the Child as the Parent for the next generation, depending upon this probability p.

- End Loop (temperature T)

Fig. 5: Proposed Simulated Annealing Algorithm using GA to solve N Queens Problem

7.3 ALGORITHM N_QUEEN_SOLUTION BY BACKTRACKING

Input: The number of Queens (N).

Output: The Number of Solutions (Placements) of that very number of Queens' Problem, according to the rule of the problem.

Step 1: Firstly, place a queen in the top row and take a note of the column and diagonals it occupies.

Step 2: Now, place another queen in the next row, such that, it is not in the same column or diagonal with the first one. Keep note of the occupied columns and diagonals and proceed to the next row to place the next queen.

Step 3: If no position is available open in the next row, get back to the previous row and place that queen to the next available position in its row and the process starts over again, until finding the correct arrangements for all the queens [28].

Fig. 6: Backtracking Algorithm to solve N Queens Problem

7.4 ALGORITHM N_QUEEN_SOLUTION BY BRUTE FORCE SEARCH

Input: The number of Queens (N).

Output: The Number of Solutions (Placements) of that very number of Queens' Problem, according to the rule of the problem.

Step 1: At first, place a queen in the top row.
Step 2: Then, place a queen in the next row down
Step 3: Check, if it is sharing the same column or same diagonal with the first one. If yes, then place the queen in the next available position in that row. Otherwise, move on to the next row to place the next one.
Step 4: If no position is open in the next row, move back to the previous row and move the queen over to the next available place in its row and the process starts over again and it will continue, until having the proper solution [28].

Fig. 7: Brute Force Search Algorithm to solve N Queens Problem

8. RESULTS

For experiment, some different numbers of queens have been taken as input in the implementations of the four algorithms. The Fitness and Execution Time obtained by implementing the Proposed GA and the Proposed SA using GA techniques are given in Table II. The No. of Solutions and Execution Time obtained by implementing the Backtracking and the Brute Force Search Algorithms are mentioned in Table III. The Graphical Representations of Fitness, Execution Time, No. of Solutions and Execution Time are also shown in Fig. 6, Fig. 7, Fig. 8 and Fig. 9 respectively. And, the System Specification is given in Table I.

Table I: - System Specification

Hardware Used	256-512 GB Hard Disk and 1-2 GB RAM
Processor Type	Dual Core, Core2Duo Processors
CPU Speed	2.4-2.93 GHz.
Operating System and Software Used	Windows XP, Windows 7, Linux and Turbo C++, Linux C

Table II: - Comparison of Obtained Fitness and Execution Time between GA and SA

No. of Queens	GA Fitness	GA Time (Sec)	SA Fitness	SA Time (Sec)
10	45	0.219780	40	0.054945
20	190	0.769231	175	0.109890

30	428	1.373626	414	0.164835
40	775	2.087912	753	0.219780
50	1214	3.186813	1198	0.274725
60	1752	4.395604	1731	0.329670
70	2394	5.769231	2368	0.384615
80	3139	7.527473	3098	0.409276
90	3977	9.615385	3934	0.439560
100	4919	11.483516	4886	0.494505
110	5959	14.365604	5913	0.604396
120	7102	16.593407	7069	0.659341
130	8342	19.450549	8292	0.769231
140	9684	24.505495	9629	0.824176
150	11126	27.967033	11070	0.872395
160	12669	31.538462	12595	0.934066
170	14300	37.197802	14259	1.043956
180	16047	44.065934	15989	1.098901
190	17886	49.109890	17807	1.157549
200	19810	57.604376	19754	1.208571

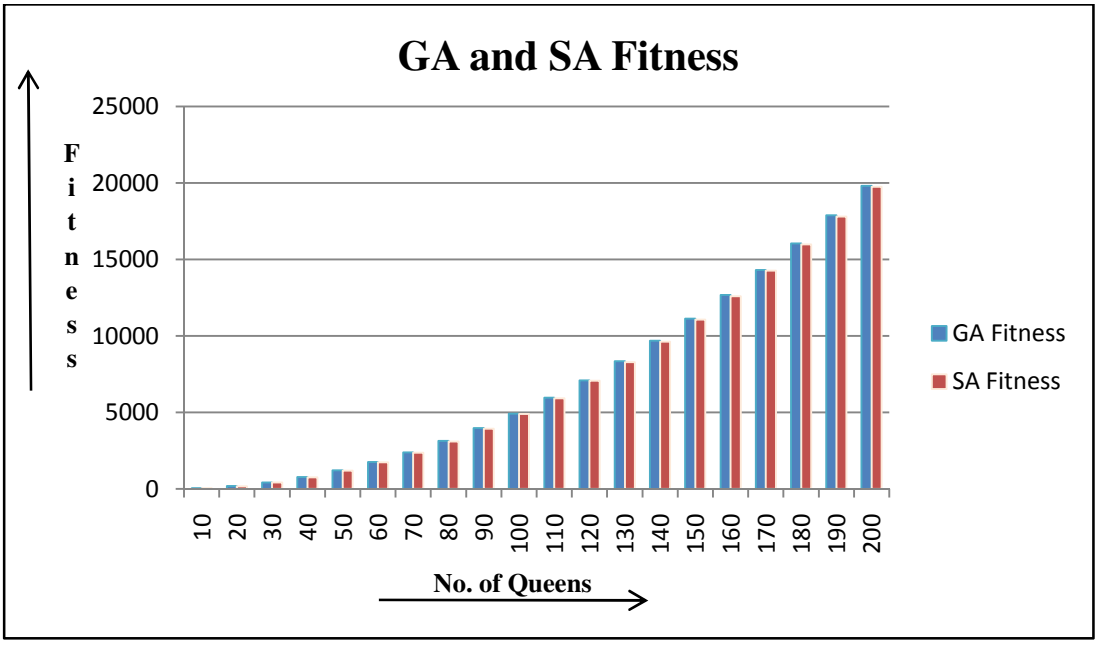

Fig. 6:- The Graphical Representation of Comparison of Obtained Fitness between GA and SA

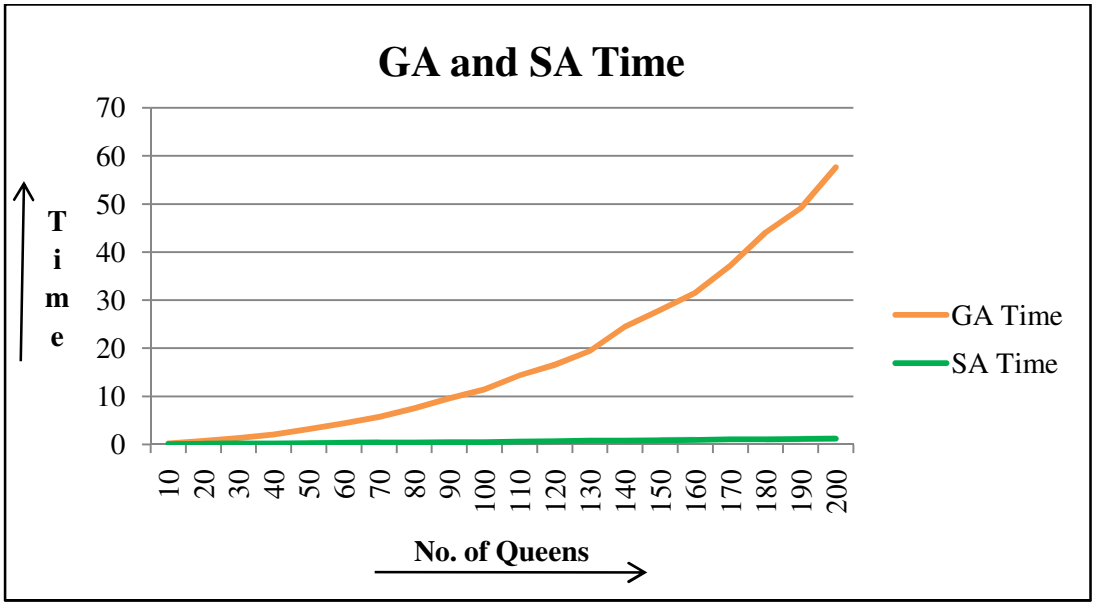

Fig. 7:- The Graphical Representation of Comparison of Execution Time between GA and SA

Table III: - Comparison of Obtained No. of Solutions and Execution Time between BT and BF

No. of Queens	BT No. of Solns.	BT Time (Sec)	BF No. of Solns.	BF Time (Sec)
1	1	0.000000	1	0.000000
2	0	0.000000	0	0.000000
3	0	0.000000	0	0.000000
4	2	0.000000	2	0.000000
5	10	0.000000	10	0.000000
6	4	0.000000	4	0.000000
7	40	0.054945	40	0.054945
8	92	0.109890	92	0.164835
9	352	0.384615	352	0.714286
10	724	1.153846	724	1.978022
11	2680	5.164835	2680	9.450549
12	14200	32.252747	14200	58.736264

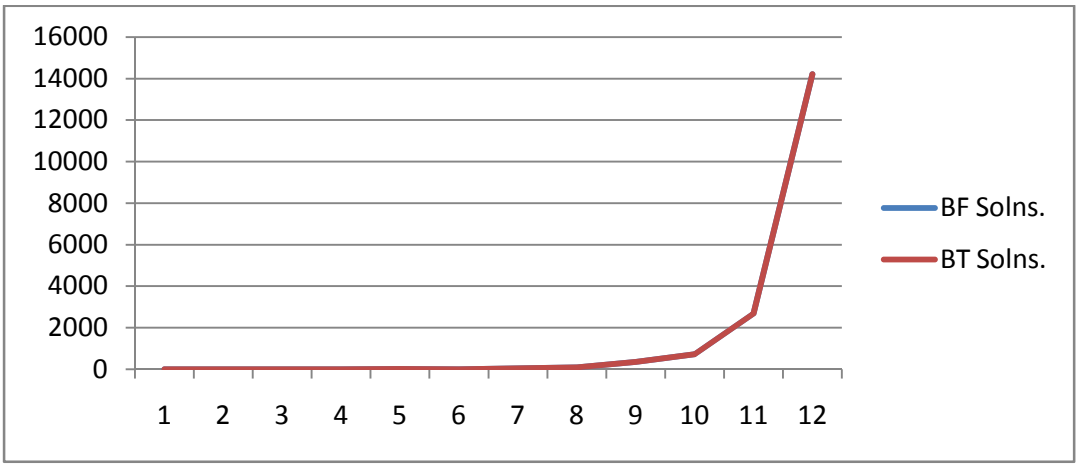

Fig. 8:- The Graphical Representation of Comparison of No. of Solutions between BT and BF

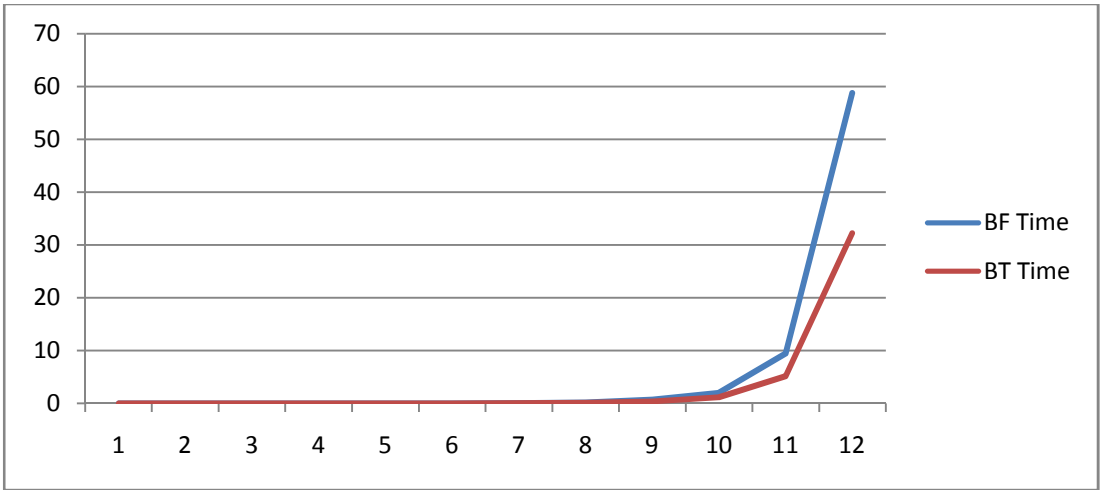

Fig. 9:- The Graphical Representation of Comparison of Execution Times between BT and BF

9. ANALYSIS AND DISCUSSION

The Proposed Simulated Annealing Algorithm using GA takes less time than the Proposed Genetic Algorithm, as it works on only one chromosome (solution) at a time and processes it to get the result. The Proposed Genetic Algorithm (GA) gives better solution than the Proposed Simulated Annealing (SA) Algorithm by GA, as it works on a population of chromosomes (solutions) and processes them to get the result. Thus, it explores the search space, which helps to get the solution close to the global optima. But, it is a very important and noticeable fact that, Genetic and SA using GA Algorithms do not guarantee global optima, as these are random search and optimization algorithms. Brute Force Search algorithm for N Queens Problem will examine all possible placements of N number of Queens on the (N*N) matrix, and, for each placement, check, whether each Queen can attack another Queen or not. It is very much simple to implement. It is mostly used, when the problem size is comparatively less. This algorithm can also be used, when the simplicity is more crucial than speed. It should not be confused with Backtracking Algorithm, where large sets of solutions can be eradicated without enumeration. Backtracking Algorithm will take less time than Brute Force Search Algorithm to solve N Queens Problem. Also, Brute Force (BF) Search and Backtracking (BT) Algorithms can provide exact result for N Queens Problem [20, 21]. But, these two algorithms cannot provide solutions in ample time, when N values are high. Therefore, these two algorithms are not at all efficient and effective to solve N Queens Problem; whereas, Genetic Algorithm (GA) and Simulated Annealing (SA) Algorithm by GA can provide good solutions to higher valued Queens.

10. CONCLUSION

In this paper, the performance of the Proposed Genetic Algorithm in terms of Fitness is enhanced, except for some larger values of queens. The Proposed Genetic Algorithm has of score of improvements over the Proposed Simulated Annealing Algorithm using GA and the other two algorithms; i.e., Backtracking and Brute Force Search. Therefore, it can undoubtedly be concluded that, the Proposed Genetic Algorithm is very much better than the Proposed Simulated

Annealing Algorithm by GA, w.r.t. Fitness of the Solutions obtained in the Result (8) Section, in which our entire concentration was focused upon. But, as per Execution Time is concerned, the Proposed Simulated Annealing by GA performed better than the Proposed Genetic Algorithm. Also, there is no confusion to conclude that, the Proposed Genetic Algorithm (GA) and the Proposed Simulated Annealing (SA) Algorithm using GA are very much better than the Brute Force Search (BF) and Backtracking (BT) Algorithms. Finally, one thing we must say that, this paper is entirely nothing but a review based work.

11. FUTURE WORKS

This paper work can be extended by adding a few algorithms like Dynamic Programming, Greedy, Hill Climbing, Tabu Search, Ant Colony Optimization, Swarm Optimization etc. to solve N Queens Problem and make a comparative study of these algorithms and thus, making the whole task more efficient and effective. Also, the Proposed Genetic Algorithm can be well modified, so that, it can provide solution to higher values of N. Besides these, in the Proposed Genetic Algorithm, some other selection/crossover/mutation methods can also be applied and comparative analysis can be made among those methods to check, which method is better and this also will be an efficient and effective work. Finally, although several modifications will have to be made, yet this approach can be tried to be applied to solve 3D Queens Problem also.

ACKNOWLEDGEMENT

Firstly, we, the authors of this paper, offer devotion to the lotus feet of God for giving us courage and blessings for doing this work successfully. Also, we thank our Parents and Friends for giving us support. We are very much grateful to our Colleges, Departmental Faculties and Other Departmental Staffs for giving us opportunity, enthusiasm, working environment, facilities, help and support to complete this work in time. We are also very much thankful to the Authors of various Research Papers, which provided us good enough concepts and confidence for performing this work. We also truly believe that, the different study materials of the internet provided us good enough knowledge and concepts about this topic, the used algorithms etc. and thus, helped us in performing this work. Without these helps and supports, we couldn't have completed this paper successfully at all.

REFERENCES

[1] S.N.Sivanandam and S.N.Deepa, "Principles of Soft Computing", 2nd Edition, Wiley India Pvt. Ltd.
[2] David E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", 4th Edition, Dorling Kindersley (India) Pvt. Ltd.
[3] http://en.wikipedia.org/wiki/Simulated_annealing.
[4] http://en.wikipedia.org/wiki/Eight_queens_puzzle.
[5] http://www.civil.iitb.ac.in/tvm/2701_dga/2701-ga-notes/gadoc/gadoc.html.
[6] Marko Božiković, Marin Golub and Leo Budin , "Solving n-Queen problem using global parallel genetic algorithm".
[7] http://www.cs.bc.edu/~alvarez/ML/GA/nQueensGA.html.
[8] http://kursinfo.himolde.no/forskningsgrupper/papers/Chapter%203%2 %20Genetic%20Algorithms.pdf.
[9] http://geneticalgorithms.ai-depot.com/Tutorial/Overview.html.
[10] Eric Cantú-Paz, "A survey of parallel genetic algorithms", Computer Science Department and The Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.

cantupaz@illigal.ge.uiuc.edu.
[11] Ivica Martinjak, Marin Golub, "Comparison of Heuristic Algorithms for the N-Queen Problem" (Citations: 5). International Conference on Information Technology Interfaces - ITI, pp. 759-764, 2007.
[12] "Comparative Study on Genetic Algorithm and Backtracking Algorithm to Solve N-Queen Problem", Santanu Datta, Pramit Brata Chanda, Soham Mukherjee, Sritama Bisi. Student Paper Contest, IEM IEEE Student Chapter.
[13] S. Pothumani, "Solving N Queen Problem Using Various Algorithms - A Survey", Department of CSE, Bharath University, India. International Journal of Advanced Research in Computer Science and Software Engineering. pp. 247-250, 2013, ISSN: 2277 128X.
[14] Anita Thengade and Rucha Dondal , "Genetic Algorithm-Survey Paper", IJCA Proceedings on National Conference on Recent Trends in Computing, 7-8April, 2012, pp25-29, ISSN: 0975-8887.
[15] http://artint.info/html/ArtInt_89.html.
[16] http://www.cas.mcmaster.ca/~cs777/presentations/2_GO_Doron_Simulated_Annealing_and_Tabu.pdf
[17] http://www.cs.nott.ac.uk/~nza/G52PAS/lecture4.pdf.
[18] www.ecs.umass.edu/ece/labs/vlsicad/ece665/presentations/SimulatedAnnealing-Oregan.ppt.
[19] http://love1for4to3.blogspot.in/2011/02/genetic-algorithm.html.
[20] http://en.wikipedia.org/wiki/Backtracking.
[21] http://en.wikipedia.org/wiki/Brute-force_search.
[22] Jukka Kohonen, "A brief comparison of simulated annealing and genetic algorithm approaches", Term paper for the "Three Concepts: Utility" course. 15.12.1999. Department of Computer Science, University of Helsinki. http://www.cs.helsinki.fi/u/kohonen/papers/gasa.html.
[23] http://www.multiwingspan.co.uk/a23.php?page=types.
[24] http://www.cs.ucc.ie/~dgb/courses/toc/handout29.pdf.
[25] http://www.codeproject.com/Articles/682129/Solving-N-Queen-Problem-by-DFS-and-BFS-and-Show-Go.
[26] Vikas Thada, Shivali Dhaka, "Performance Analysis of N-Queen Problem using Backtracking and Genetic Algorithm Techniques", Asst.Professor (CSE), ASET, Amity University, Gurgaon, India, Asst.Prof. (CSE), ASET, Amity University, Gurgaon, India. International Journal of Computer Applications (0975 – 8887), Volume 102– No.7, September 2014, pp. 26-29.
[27] https://www.sellyourtime.in/register/uploads/1420613023.doc.
[28] www.dcc.fc.up.pt/~ines/aulas/1213/SI/Queens-formula-of-placement.ppt.
[29] http://openstudy.com/updates/4fa01a53e4b029e9dc311b8d.
[30] https://sites.google.com/site/nqueensolver/home/algorithms/1brute-force-algorithm.