ISSN:0975-9646

Solving N-Queen Problem by Prediction

Lijo V. P.
School of Computing Science and Engineering,
VIT University, Vellore

Jasmin T. Jose
School of Computing Science and Engineering,
VIT University, Vellore

***Abstract*– The classic problems lead the researchers to innovate general solutions for similar class of problems. One of the well-known problems and its solution are the 8-Queen puzzle and back tracking. At many cases back tracking may consider as brute force, but it is not true. It is not necessary to find a solution from the point where we stand as sub-solution and proceed for final solution by tracking one of the branches optimistically, rather selecting a solution which is different completely. Backtracking is effective method to find solutions for a problem which is have more optional path to consider. But this is not computationally efficient. In this paper, I have proposed a novel method which is computationally efficient, and it is predicting the solution with more accuracy.**

***Index Terms:*– Backtracking, Classic Problem, Quadratic Algorithm**

I. INTRODUCTION

The classic problem, N-queen puzzle is an ever challenging problem in computer science. Current literature gives many solutions to this problem, but they suffer with its computational complexities. A well-known solution for this Queen puzzle is backtracking which is with very high time complexity. The complexity exponentially increases as n increases. An interesting fact is that there is no guarantee to increase the number of solutions as the n increases. For an instance the number solutions for 6-Queen puzzle is lesser than the number of solutions for 5-Queen problem.

In back tracking algorithm, procedure starts from a point of user choice and proceed with any one of the choice of next step. From the current step, try to find next level of solution from a list of choices. This process will be iterated until either a final solution arrived or no possible solution. Then start to backtracked to a previous step, and repeat its previous step of process by selecting another choice.

Naive Algorithm demonstrates the method where all the queen configurations are identified and display some of them which are satisfied the given constrains. The given constrain is the queens should not attack each other in this given configuration. If such a configuration is available then select and print. This algorithm gives a general view that it problem means. The challenge is laid under the complexity to find a configuration which follows the conditions. Trial and error method can be followed, but it is quite expensive in computation. The aim is to arrange the Queens one after other in different rows, user can begin this process leftmost row and when user put a queen in a raw, user search for any conflicts with available queens. In the current row, suppose the player find a column for which there is no conflict, then player take this column and row part of the solution. If player failed to find a column with no clash then back track. The right choice of the size n is eight in case of this classic problem. Because $n = 8$ is large enough to demonstrate the beauty, challenges and complexity of the puzzle. If you could find an efficient solution for this 8-queen problem, then it can be extended to N-queen.

Backtracking is a general algorithm to find complete solutions for some computational problems which have more number of solutions. This algorithm builds the solution by adding qualified candidates to solution one by one. This will immediately reject the candidate if it identified as this candidate cannot be a part of the solution. The partial candidate solution is the l queens arranged in firs l rows. The backtracking is only applicable if the solution is an incremental model and it is supporting a sudden way to find candidate solutions to reach the final solutions.

The sections of the paper are arranged as follows: the Section 2 is dedicated for literature survey. Section 3 gives insight in to the proposed algorithm and Section 4 describes the results of the proposed algorithm. The Section 5 depicts the conclusion and future work. Acknowledgements included in Section 6 and finally quoted the references.

II. LITERATURE SURVEY

Algorithm strategies are the approach to solve computational problems. It may combine many approaches together to solve a problem. We follow different strategies based on the problem. Algorithm can be implemented either iteration or recursive in structure. Dynamic Programming, Branch-and-Bound, Bruit Force, Backtracking, Greedy, Recursive, Heuristic, etc. are some the important algorithm strategies.

II.1 Backtracking Strategy

Backtracking algorithms are applicable for NP-Complete problems. Priestley and Ward [1] presented the details about the backtracking and its applications. They have clearly explained the preliminaries of the algorithm and gave a clear picture of the 8-queen puzzle. The solution for the problem could be achieved through tree structure representation of the choices. The proposed procedure could reduce the number of test cases to a sum of 15,720. The time complexity was reduced by reducing test cases. Pre –analysis was used to reduce the test cases. They used bush pruning technique for further improvement. Finally hybrid approach of pre-analysis and bust pruning gave better result.

Ginsberg [2] introduced dynamic backtracking algorithm but that does not solve constraint satisfaction problem dynamically. Gerald and Thomas [3] proposed some alteration to support a dynamic constraint satisfaction. But this method suffers due to heavy time complexities.

M M Noori and B T Razaie [4] expose and implemented an improved backtracking algorithm for identifying t-designs which is proposed by J. Combin. Des. According to Noori, the algorithm uses a systematic method to derive new useful equations from the initial equations which are useful in speeding up the classical backtracking algorithm. The reader can refer [5] to get brief description of backtracking algorithms and their applications.

Bessière et al. [6] proposed a asynchronous backtracking algorithm for distributed constraint satisfaction problems. This is based on distributed backtracking with storage of the previous results to reduce the total number of trials. M.A. Gutierrez-Naranjo et al. presented the N-queen problem in conjunctive normal form. They described the queen problem as a SAT problem by assuming the each P-systems send truth values as Yes or No. Pioneer solution have presented to the *N*-queens puzzle based on Membrane Computing [7]. Algorithm 1 gives an insight on back tracking algorithm which is based on depth-first recursive search.

Algorithm 1: *Backtracking general algorithm* [8]
Checks whether solution has been found
If found solution, return it
else for each choice that can be made
take that choice
Recurrence
If recursion gives a positive result, return it
If no choices remain, return failure

II.2 Computational Complexities

Vipin Kumar [9] gives a detailed survey on constraint satisfaction problem algorithms. Some of them solve the problems by constraint propagation and rest of them are solving problem by direct approach though backtracking. A survey on complexity analysis of space-bounded algorithms for constraint satisfaction problems is proposed by J Roberto et al.[10]. They cover unrestricted, size-bounded, relevance-bounded learning and their complexities.

J. H Patterson [11] presented the computational results of the minimized and maximized problems in a general way. They have considered both mainframe and personal computer experiences. Optimal solution for small problems is very easily attained in PCs but for large problems quite difficult to achieve result in PCs.

John Gaschnig [12] proposed a fast backtracking algorithm which is reduced its computational complexities by eliminating redundant tests. He tried to exploit space-time trade off maximum in his algorithm. According to John, it is hard to eliminate all redundant tests in computationally efficient manner. But it is possible to eliminate all redundant test cases with heuristic approach. In [13], Jordan Bell and Brett Stevens discussed the computational approaches of n queen's problem. In this survey paper they have given concentration for different approaches and results for the same problem.

Erbak and Tanik given a detailed study of algorithms used for n-queens problem in [14]. They grouped the total algorithms in different categories based on the outcome of the algorithms. As per their view they are three types of algorithms based on their outcomes. Some of them generate all solutions and others are produce fundamental or subset of total number of solutions. Brute-force trial and error and backtracking are examples of algorithms which generate complete solutions. The algorithms based on group properties of results, symmetric elimination and test-based produce only basic solutions.

III. PROPOSED ALGORITHM

In this section, discuss the details about the proposed algorithm for the N-queen problem. The N is a positive integer. In this algorithm, predicts the solution systematically and then check for the correctness of the solution by performing trial on the actual configuration. The probability to quality this as solution is approximately 73%. The efficiency of this algorithm is improved by reducing the time complexity. The prediction of the solution is done by mathematical progression approach.

Initially, designate each row of the N X N matrix with the value 0 to N-1. For instance, first row is designated as '0', second row is '1' and so on. The combinations of the value from 0 to N-1 represent the arrangement of the Queens on the board. For example, in 5 X 5 board, 13042 is representing the queens positions on the board, where queen in first column is at row 2, in second column- row 4, third column-row 1, fourth column-row 4 and fifth column-row 3.

These combinations of the numbers can be taken as the number with N digits with base N. In this case of this example 13042 is the number with 5 digits and base 5. The given 13042 is a solution for the 5 X 5 queen puzzle. Close analysis of this number is giving an insight that the adjacent digits of the numbers have difference with value 2. The digit 1 and 3 have difference 2, the digits 3 and 0 has difference 2, and so on. This point is an important clue in this algorithm that we can find next solution only after a distance of $2* 5^5$ if the first value to change to 3 or 4. By adding the value $2 * 5^5$ to the given solution 13042 will give next predicted value. For the correctness of the value place the queens in the respective rows and check for the clashes. If there is no clash then display it as a solution, else reject that candidate solution and predict next potential candidate.

Algorithm 2: *Candidate Selection.*

```
Function Candidate-Selection(N, Init)
 if Init >= N^N
   Return;

   else
  while k < N do
              check digits in init at k and at k+1
              if difference is greater than one
                    k=k + 1
             else
Init = Init + predict_value(N, Init)
             end if
  end while
Display Init;
```

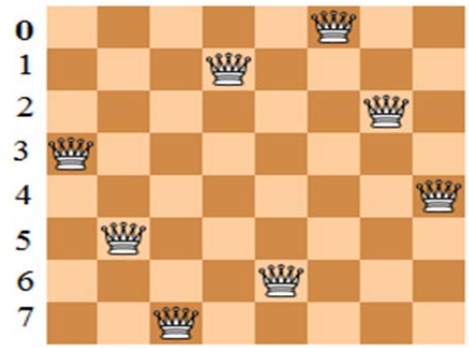

Fig. 1: The 3 5 7 1 6 0 2 4 placement of the 8 X 8 queen.

Algorithm 2, is giving the candidate selection for the solution. This algorithm receives the value N and Init as arguments. Where N is the number of columns and Init is the initial value. For instance, 0 2 4 1 3. This is the solution with minimum value, as in case base 5. The predict value function will find next possible value to be added with Init to get next candidate for solution. Possibly the next solution is 0 3 1 4 2. The difference of the values of first and second solutions is 03142 -02431= 711, which approximately 5^4. Because the digits with weight 5^3, 5^2,5^1 and 5^0 are changed in the first solution to get second solution. The value 5^2 is change as negative and others are positive. So while iterating by skipping this many steps is reducing the time complexity with an extend level.

Fig: 1 illustrates the 3 5 7 1 6 0 2 4 placement of the 8 X 8 queen. Fig. 2 shows the clashes in 8 Queen puzzle. Here, the queen in column 6 clash with other queens. There is no choice to place the queen in 6th column. So it is necessary to backtrack to the 5th position, i.e. 5th column's placement of queen. Then, try to find some other solution to place the queen in the 5th column. It is possible to place the queen in first row of 5th column. Then it is possible to place the queen in the 6th column-3rd row. After successful placement of 6th queen then proceed to place the queen. It is possible to find a place at 7th row of 7th column. Finally, 8th queen will be failed to find a place in 8th column without a class. So it is necessary to backtrack to 7th stage, then 6th and so on. But this is not necessary in this proposed algorithm. Because the prediction will reduce number of trials and give solution.

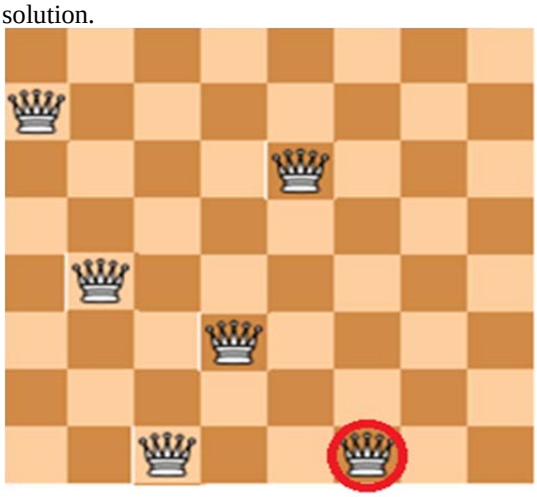

Fig. 2: A Clash in 8-Queen puzzle

Working Principle

The given algorithm follows predictive trial and error method. In brute-force trial and error, try all possible combinations to get solutions. In case N dimensional problem there with N! combinations can be prepared and it is mandatory to check each and every combinations. This will give the complexity of $O(N^{N+1})$. The prediction for the potential candidates will reduce number of combinations considerably. The reduction of the combinations is exponent of N. For an instant, 03142 -02431= 711, where the reduction is 711, which is approximate 5^4. Without checking each and every combinations try the combinations after a gap of exponent of N. The function call for Candidate-selection is initiated only when the combination contains no duplicate digits. The checking for the duplicates is very simple as checking sum of digits in the combination is exactly equal to the (N-1)*(N)/2 if N is even and ensure that all digits are available. If all the digits are not present in the combination then there is duplication occurred. Here we assume that *s* is -1 or +1 depends on the sign of the digits which is going to be changed and the *i* is the position of the from right and position starts from 0. And *d* is the difference between the existing value at *i* and value to be replaced. So predicted value from position P_j is as follows

$$P_j = s * d * n^i \qquad (1)$$

So predicted value for the potential candidate is *Cv*, previous candidate solution *PreCs* and Potential Candidate *Cp*.

$$Cv = \sum P_j \qquad (2)$$

And

$$Cp = preCs + Cv \qquad (3)$$

Cp gives the potential candidate and the system will check for its correctness.

If Cp qualify the test,

PrevCs = Cp
Retrn PrevCs

The *PrevCs* gives the next solution for the puzzle with given size N. The choice of the Init value determines the completeness of the solutions. So a careful selection of the Init value is necessary to get complete solutions with better efficiency. Some statistical approaches is applicable to determine the Init value. Suppose the Init value selected as a value greater than value of a solution, and then the algorithm never find that solution. The prediction permits to go in one direction from smaller value to higher only. No reverse check is allowing for maintaining efficiency. To avoid such a situation to miss the solutions, choose the Init value as zero. But this cause to increase the number of trial in a considerable manner.

IV. RESULTS

Performance Improvement

The proposed algorithm finds solution for the problem efficiently. Even though the complexity increases as the dimension increases, it is giving better result when compare with existing algorithms. The backtracking algorithm considers as the most effective algorithm for Queen puzzle, but it is computationally highly expensive. It is expressed as *O(n!)*. The proposed algorithm reduces time complexity by omitting much iteration. The iterations of the candidate selection process is reduced by predicting the next possible candidate. By prediction of the potential candidates, it is possible to reducing $O(n^3)$ iterations. The complexity of the algorithm is reduced by $O(n^3)$. And actual running time is very less when compared to backtracking algorithm. This is a great attainment in this case of branch type of problems.

Table 1: depicts the sample values of N-queen puzzle. It is clearly gives the performance improvement of the proposed algorithm. The number of trails for 13-queens gets reduced to 84,034,432 where the counterpart has 89,088,384 [1]. In this paper, I consider the results of a hybrid [1] algorithm which is running on single node.

The result of the algorithm based on the accuracy of the prediction. Even though prediction is more accurate but it needs to be improved to get 100% accuracy. Complete accurate prediction is not possible as the size of the problem increases exponentially. Computational complexity is reduced in a considerable manner. Single node computation is considered here. It is possible to execute this algorithm in parallel or distributed environment. By giving proper Init value to every node in the given environment we can reduce the running time in factor of n, where n is the number of nodes. The nodes can process the algorithm independently if getting proper *Init* value. So there should be a leader/ initiator to calculate and distribute Init values among different nodes in the system. Every nodes can finish their job independently and parallel.

Fig. 3 illustrates the comparison of number of trials in different approaches such as Brute Force Trial and Error, Pre-Analysis, Hybrid and Prediction. Fig. 4 shows CPU time *(in Seconds)* for various approaches for 13 Queens and 14 Queens puzzles.

Table 1: N queens puzzle sample results.

Method Used	No. of Trials		CPU Time	
	13 Queens	14 Queens	13 Queens	14 Queens
Without Pre-analysis	130,150,618	899,139,237	66.45	112.21
Pre-analysis	100,515,902	654,151,660	54.32	89.29
Hybrid	89,088,384	569,929,575	56.01	109.09
Proposed Algorithm	84,034,432	543,672,172	44.14	87.56

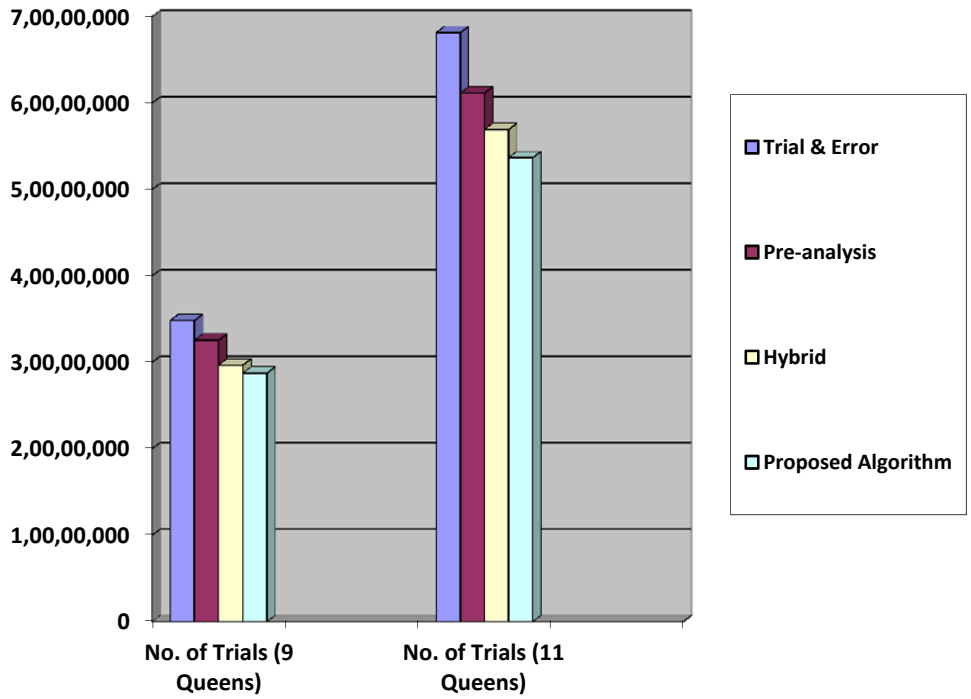

Fig. 3: Comparison of Number of Trails in Various Approaches.

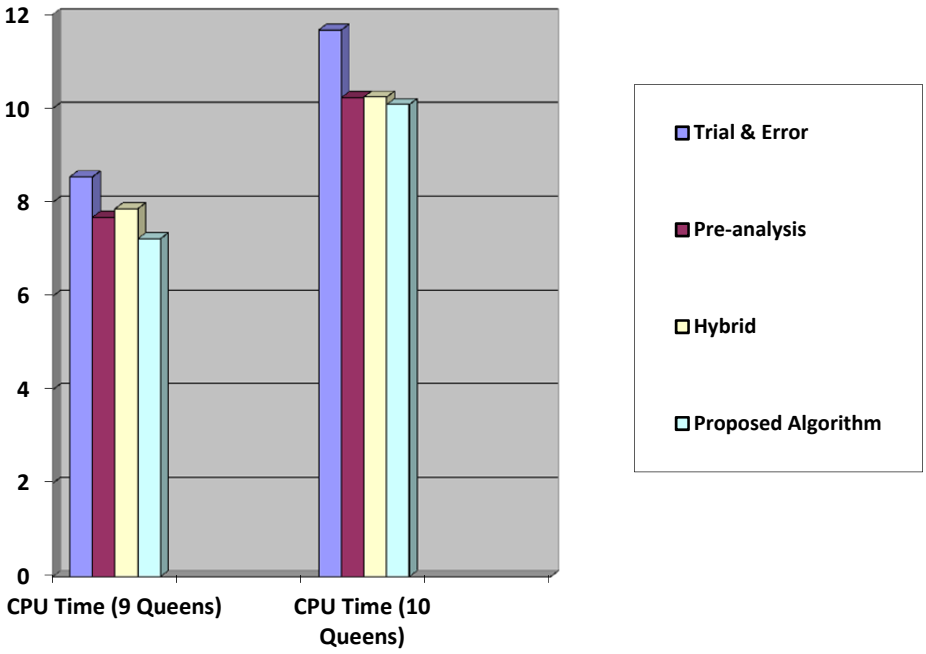

Fig. 4: Comparison of CPU time of Trials in Various Approaches

V. Conclusions And Future Work

In this paper I have proposed an algorithm which has less computational complexity when compare with backtracking algorithm to solve N-queen puzzle. The proposed algorithm works on prediction of potential candidate solution, based on arithmetic progression. I conducted the experiments on single machine and expecting improved results from a distributed computing environment. As future work, try to optimize this algorithm by improving the prediction technique and reduce the number of false prediction.

Acknowledgment

My thanks to Mr. Lijo V. P who has contributed towards completion of this work. I am extending my gratitude to my colleagues in VIT University Vellore and friends in MES College of Engineering, Kuttippuram for their valuable guiding and sharing.

References

1. Priestley, Hilary A., And Martin P. Ward. "A Multipurpose Backtracking Algorithm." *Journal Of Symbolic Computation* 18.1: 1-40, (1994).
2. Ginsberg, Matthew L., And David Mcallester. "Gsat And Dynamic Backtracking." *Principles And Practice Of Constraint Programming*. Springer Berlin Heidelberg, (1994).
3. Verfaillie, Gérard, And Thomas Schiex. "Dynamic Backtracking For Dynamic Constraint Satisfaction Problems." *In Proceedings Of The Ecai-94 Workshop On Constraint Satisfaction Issues Raised By Practical Applications*, (1994).
4. M. M. Noori And B. Tayfeh Rezaie. "A Backtracking Algorithm For Finding T-Designs." *Designs, Codes And Cryptography* 32.1-3: 185-191, (2004).
5. P. B. Gibbons, "Computational Methods In Design Theory, In: The Crc Handbook Of Combinatorial Designs" (C. J. Colbourn And J. H. Dinitz,Eds.), Crc Press Series On Discrete Mathematics And Its Applications, Pp. 718–740, (1996).
6. Bessière, Christian, Et Al. "Asynchronous Backtracking Without Adding Links: A New Member In The Abt Family." *Artificial Intelligence* 161.1: 7-24, (2005).
7. Gutiérrez-Naranjo, Miguel A., Et Al. "Solving The N-Queens Puzzle With P Systems." *Seventh Brainstorming Week On Membrane Computing* 1: 199-210, (2009).
8. Yan, Jun, And Jian Zhang. "Backtracking Algorithms And Search Heuristics To Generate Test Suites For Combinatorial Testing." *Computer Software And Applications Conference, 2006. Compsac'06. 30th Annual International*. Vol. 1. Ieee, (2006).
9. Kumar, Vipin. "Algorithms For Constraint-Satisfaction Problems: A Survey." *Ai Magazine* 13.1: 32, (1992).
10. Bayardo, Roberto J., And Daniel P. Miranker. "A Complexity Analysis Of Space-Bounded Learning Algorithms For The Constraint Satisfaction Problem." *Proceedings Of The National Conference On Artificial Intelligence*. (1996).
11. Patterson, James H., Et Al. "Computational Experience With A Backtracking Algorithm For Solving A General Class Of Precedence And Resource-Constrained Scheduling Problems." *European Journal Of Operational Research* 49.1: 68-79, (1990).
12. Gaschnig, John. "A General Backtrack Algorithm That Eliminates Most Redundant Tests." *Ijcai*. (1977).
13. Bell, Jordan, And Brett Stevens. "A Survey Of Known Results And Research Areas For N-Queens." *Discrete Mathematics* 309.1: 1-31, (2009).
14. C. Erbas, M.M. Tanik, "N-Queens Problem And Its Algorithms", Technical Report 91-Cse-8, Dept. Of Comp. Sci. And Eng., Southern Methodist University, (1991).