Experimental comparison of AI algorithms to solve N-Queens problem

Ashwini Karappa
Masters Student, Department of Computer Science.
University of Texas at Dallas
Richardson, Texas 75080, USA.
Email: axk142031@utdallas.edu

***Abstract*—This report discusses different techniques known for solving constraint satisfaction problems. I implemented few of these techniques which are namely backtracking, forward checking, variable ordering using minimum remaining values and minimum conflicts for solving N-Queens puzzle. N-Queens problem has some practical applications such as in traffic control, deadlock prevention, statistics, and parallel memory storage schemes. It is often used as an example of constraint satisfaction problem. I also present an experimental comparison of the techniques that I implemented for solving this puzzle.**

***Keywords*— Constraint satisfaction problem, N-Queens, Backtracking, Forward checking,Variable ordering and Value ordering, Minimum Conflicts**

I. Introduction and Background [1,2]

We come across various puzzles based on chessboards such as counting the number of squares, counting the number of rectangles and finding the best moves. The n-queens problem is one of the chess puzzles. This problem was originally known as the 8-Queens problem. It has been of interest for many mathematicians. It was generalized to N-Queens for N X N boards in 1850 by Franz Nauck. According to the chess rules, queens can attack by moving forward or backward in the same row, the same column or the same diagonal. N-Queens puzzle asks, 'How to place N queens on N X N chess board, such that no two of them can attack each other?'. Figure 1, shows an example of 4-Queens problem. Part a), in figure 1, shows the wrong placement of queens, because the queens in second row and third row can attack each other since they lie on the same diagonal. Part b), in figure 1, shows the correct placement of 4 queens where no two queens can attack each other.

N-Queens puzzle has many practical applications like parallel memory storage schemes, VLSI testing, traffic control and deadlock prevention. This problem is an example of backtracking algorithms, permutation generation, the divide and conquer paradigm, program development methodology, constraint satisfaction problems, integer programming, and specification. Thus, with the rapid development of computer science, N-Queens problem has gained a lot of attention and solving it with optimal techniques has gained importance.

This report initially explains different AI techniques people experimented on to solve this puzzle in related work. Further, I explain constraint satisfaction problem solving techniques namely: backtracking, forward checking, forward checking with MRV and minimum conflicts that I implemented for solving N-Queens puzzle. I also present the experimental results of these algorithms and my key observations. Finally, I conclude the paper.

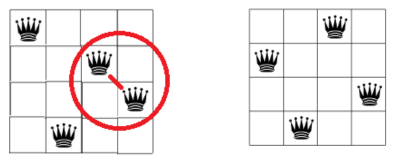

(a) Wrong placement (b) Correct placement

Fig. 1. Example of 4 Queens puzzle

II. Related work

Previously, lots of work is done on N-Queens problem. This problem can be solved using different techniques. Kuruvilla Mathew and Mujahid Tabassum in [3], compared the performances of search strategies namely: Breadth First Search, Depth First Search, A* Search, Greedy Best First Search and the Hill Climbing Search to solve 8 queens problem. They concluded that A* is the best strategy amongst the others listed and it achieves convergence faster, at cheaper cost, and therefore they also suggest that A* is best suited for solving issues of this nature and also has the possibilities to be applied towards issues of growing dimensionalities. K. D. Crawford in [4], applied Genetic Algorithm and have discussed two ways to solve n-Queen problem. Marko Boikovi, Marin Golub, Leo Budin in [5], show how genetic algorithms can be used to solve n-Queen problem. In this paper, they have also shared their experimental results for several large values of n and their conclusions for NP problems with genetic algorithms.

III. Constraint satisfaction problems

Although, there are many ways of solving N-queens puzzle, it mainly falls under and is well known as constraint satisfaction problem. Constraint satisfaction problems are basically the problems in which there is a set of variables and a set of values. Each variable is to be assigned a value, such that, it satisfies all the constraints for the problem. N-Queens puzzle, thus, is a constraint satisfaction problem, where queens are to be placed such that no two can attack each other.

IV. Constraint Satisfaction problem solving techniques implemented[6,7]

Soling N-Queens Problem

In N-Queens problem, no two queens should be on the same row or the same column or the same diagonal. I have implemented Backtracking, Forward checking, Forward checking with minimum remaining values and Minimum conflicts techniques for solving N-Queens puzzle. In the following section, I have explained in brief these techniques and illustrated them with 4 Queens example. Backtracking and forward checking examples are taken from [7]. I have created a similar example for forward checking with minimum remaining values and the example for minimum conflicts is taken from the book [6]. In the example figures, queens are placed column wise. There are 'N' columns and 'N' rows in a 'N x N' chess board. First Column has 'N' cells, because of 'N' rows, Second column also has 'N' cells and so on. These techniques, place queens column wise. In each column, one queen will be placed on any of the 'N' cells by selecting the row such that, it satisfies the constraints for N-Queens puzzle. For figures 2, 3, and 4 we have 4 x 4 chess boards. And red dot denotes the queen and cross mark denotes the places where queens cannot be placed. In figure 5, we have 8 x 8 chess board for minimum conflicts.

A. *Backtracking*

Backtracking is a simple depth first search approach. It starts initially with any variable, assigns a value from the domain of values and then goes deeper into the tree to assign values to other variables. Later on, at some level, if it finds out that there are no values left for a variable, it backtracks and assigns different value to the parent variables. Figure 2, shows an example to understand backtracking approach for solving 4-queens problem.

Explaining the example-

1. First queen is placed in the first column and the first available row. Initially for the first queen it is first column and first row, then it goes down into the search tree and check assignments for other queens.

2. Second queen is placed in the second column, initially it checks whether placing second queen in the second column and first row is possible, then the algorithm understands that this is not the valid place. Thus, it goes to the second row, again it understands that this is not the valid place, then it places the queen in the third row.

3. It again goes down deeper into the tree and does the same thing for other queens.

4. But if for some queen, it does not find a cell to place the queen, it backtracks on the previous level and checks other places for the previously placed queens.

5. This algorithm will go on until the last queen is placed.

B. *Forward Checking*

Generally, in most of the constraint satisfaction problems, as soon as any variable is assigned a value, this value is no longer valid for some of the other variables. Forward checking is the technique which is nothing but backtracking, but it also saves the effect of variables assigned with values in the previous step and grows the search tree only for the assignments which are valid.

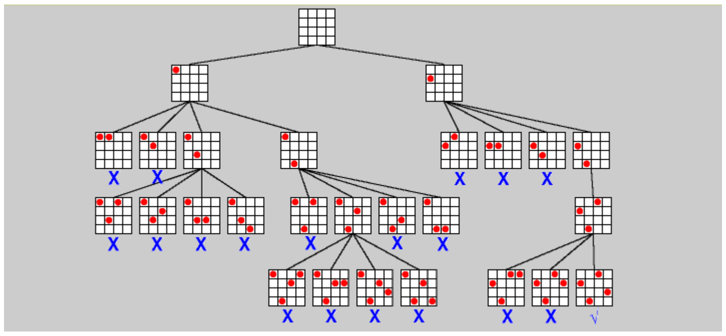


Fig. 2. Backtracking approach

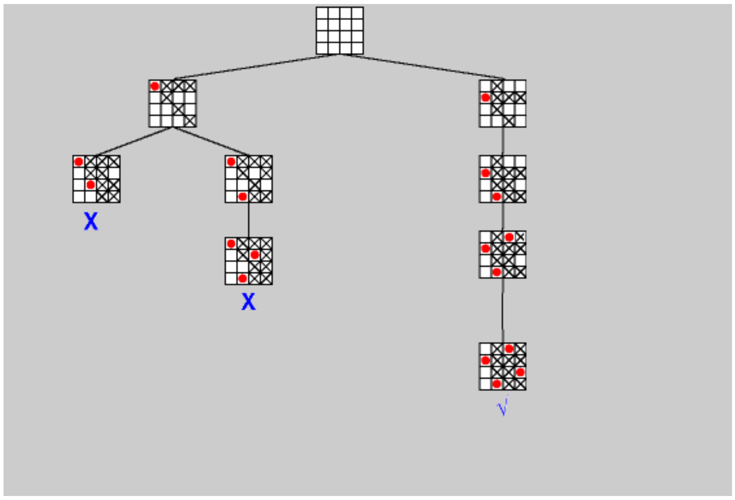


Fig. 3. Forward Checking approach

Explaining the example-

Figure 3, shows an example of forward checking approach for solving 4 Queens problem. In this example, we can see that, once a queen is placed, all the cells in its row, column and diagonals where other queens cannot be placed are marked with a cross. So, when the search tree goes deeper, it knows which cells cannot be considered for the placement of the current queen. Thus, the search tree that grows is smaller as compared to the backtracking.

In my implementation of this algorithm, for every recursive call, I maintain an old and a new boolean N X N chessboard. As soon as I enter into the recursive call, I make a new chessboard from the old chessboard. I update the new one with false values for the cells which are no longer valid for other queens because of the place assigned for the current queen. When algorithm backtracks, it has the old chessboard, which it can use for checking other places for previously assigned queens.

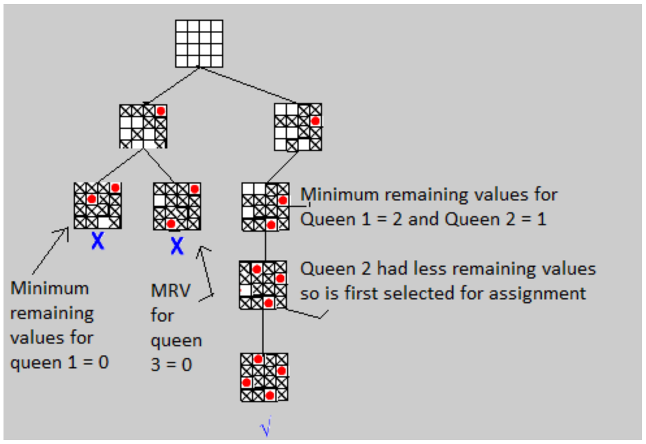

Minimum remaining values for Queen 1 = 2 and Queen 2 = 1
Queen 2 had less remaining values so is first selected for assignment
X
X
Minimum remaining values for queen 1 = 0
MRV for queen 3 = 0
√

Fig. 4. Forward checking with minimum remaining values approach

C. Variable Ordering and Value Ordering

Backtracking and Forward checking both these techniques do not use any special order for assigning values to the variables. Variable ordering technique is based on an intuitive idea of giving preference to the variable which is left with the fewest possible legal values. This is also called as ”Minimum Remaining Values”, (MRV) heuristic. Value ordering heuristic known as ”Least Constraining Values”, (LCV) heuristics, prefers the values that rules out the fewest choices for the neighboring variables in the constraint graph.

We can apply MRV and LCV heuristics, both as a combination or individually on backtracking as well as forward checking. For experimental analysis, I have implemented MRV heuristic for forward checking. Figure 4 shows an example for the same.

Explaining the example-

In this example, initially every queen has minimum remaining values (MRV) equal to 4, since none of the queens have been placed. As MRV is same for all the queens, algorithm selects any queen randomly for the assignment. In the example shown, randomly 4th queen is selected for placement. Then, it marks the the cells with a cross which are ruled out for other queens. Similarly it repeats the process when going down further for other queens. If MRV becomes zero for any queen, algorithm backtracks as shown on the left branch of the tree. On the right branch of the tree, it can be seen on 3rd level, that MRV for queen 1 = 2, and queen 2 = 1, thus on the next step, algorithm first selects queen 2 for the placement.
Algorithm terminates when all the four queens are placed with a valid placement.

In my implementation of this algorithm, I am maintaining a minimum priority queue, which after every assignment of any queen, gives me the queen with the minimum remaining legal values for placement. I select this queen and assign a place to it and repeat the process of adding remaining queens to the queue with updated remaining values. Also, when the algorithm backtracks, I again update the priority queue for queens with minimum remaining values.

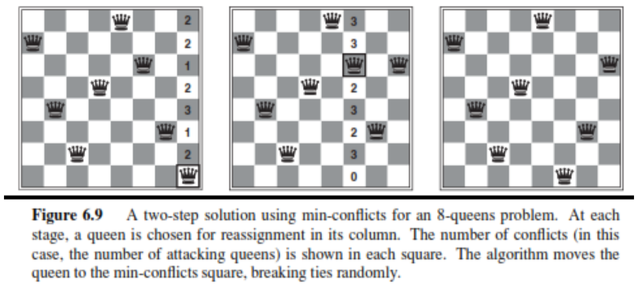


Figure 6.9 A two-step solution using min-conflicts for an 8-queens problem. At each stage, a queen is chosen for reassignment in its column. The number of conflicts (in this case, the number of attacking queens) is shown in each square. The algorithm moves the queen to the min-conflicts square, breaking ties randomly.

Fig. 5. Minimum Conflicts approach

D. Minimum Conflicts

In this technique, a complete random assignment is taken for all the variables. Then the variables that are under conflicts, i.e, the ones which violates the constrains are computed. Randomly any variable under conflict is reassigned to some other value by computing the value which will have least number of conflicts. Again conflicts for all the variables are computed, and the same process of reassigning the values to the variables in conflicts is repeated until there are no conflicts.

Figure 5, shows an example for minimum conflicts [6].

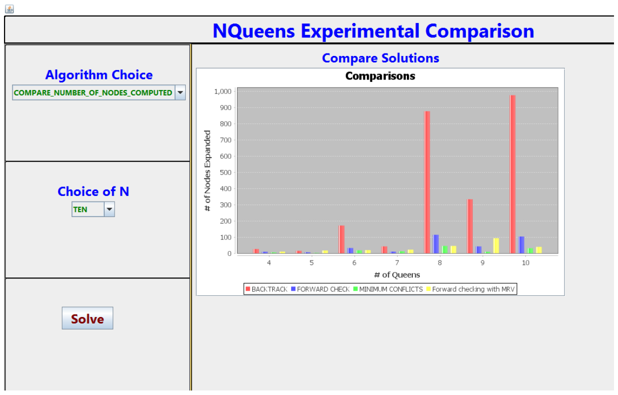
In my implementation of this algorithm, I am maintaining a max priority queue which after every complete random assignment of the problem, gives me the queen with the maximum number of conflicts and I select this queen for reassignment to the values which will have least number of conflicts. Again, conflicts are computed for all the queens and the process of reassigning the maximum priority queens (those with max number of conflicts) repeats until there are not conflicts.

V. Project Application Information

I have developed a project for solving N-Queens problem with the help of four techniques discussed above. Figure 7, shows a snap shot of my application. On left side, there is a dropdown to select the algorithm, and a dropdown to select the ’N’ value and when you click on solve, you will see all the solutions on the right hand side. And if you select any particular solution from the dropdown of solutions on right side, you will see the chessboard with N-Queens placed for that solution.

In the dropdown on the left side for inputs, if you select the COMPARE_NUMBER_OF_NODES_COMPUTED, you will see a comparison graph for the number of nodes computed in the search tree in order to get the first solution each, using all the four techiques. Example is shown in figure 6.

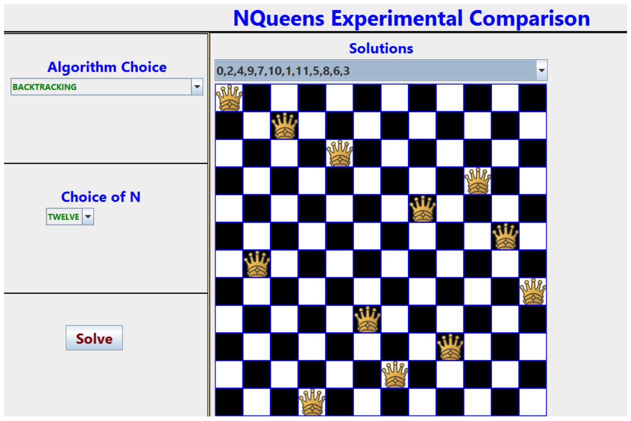
In the dropdown, if you select the COMPARE_TIME_REQUIRED, you will see a comparison graph for the the time required in order to get first solution each, using all the four techiques.



Algorithm Choice
COMPARE_NUMBER_OF_NODES_COMPUTED

Choice of N
TEN

Fig. 6. Application image for showing comparison



Algorithm Choice
BACKTRACKING

Choice of N
TWELVE

0,2,4,9,7,10,1,11,5,8,6,3

Fig. 7. Application image for showing NQueens solutions

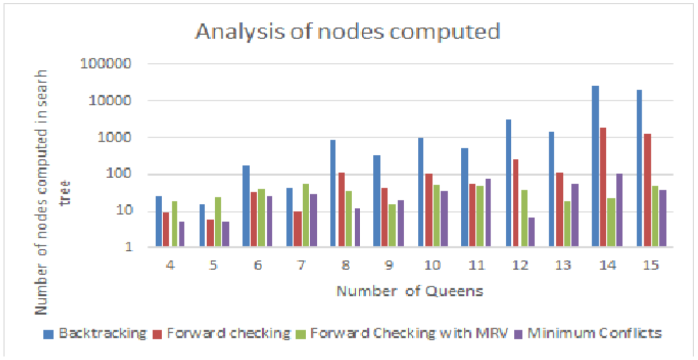


Fig. 8. Analysis of nodes computed

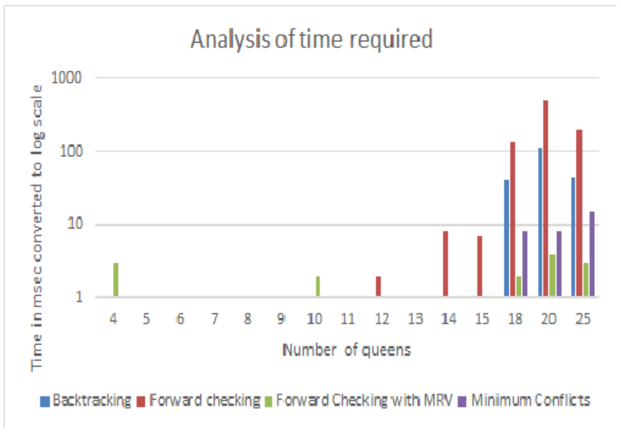


Fig. 9. Analysis of time required

VI. Experimental Results

I have analyzed the performance of the mentioned four algorithms based on the number of nodes computed and time required to compute the solutions. Backtracking and forward checking techniques generate a search tree, where nodes represent the states of the chessboard. At each level a new queen is placed and on the same level sibling nodes denote different cells checked for placing the queen.

Performance based on nodes computed

Backtracking checks all the places for every queen. That is, if there are 'd' domain values and 'N' variables then the search tree generated will be of n!d leaves, which is very huge. Thus the number of nodes computed for backtracking is the highest.

Forward checking, rules out the values for next queen to be placed based on the previous queens assignment. Thus, the number of nodes computed in this case is less as compared to the backtracking approach. But, when we compare it with forward checking with MRV, nodes computed would be less or more. Reason being, Forward checking with MRV heuristic, randomly selects the queens, when every queen has the same minimum remaining values. Thus, if the queen selected randomly, gives a node which is closer to the solution, then forward checking with MRV heuristics will compute lesser nodes as compared to the forward checking.

Minimum Conflicts starts with a random complete assignment. Thus, it eliminates the nodes that are computed for initial queen placements. This improves the performance to a great extent. I found out that even for N=1000, this algorithm solves N-Queens puzzle on an average just within 350 steps. Thus, this approach performes the best as compared to the other three techniques.
Figure 8, shows the analysis of nodes computed for N = 4 to N = 15.

Performance based on time required

As analyzed for N=4 till N=15, Backtracking requires very less time as compared to forward checking as well as forward checking with MRV. Reason for this behavior is because, backtracking requires very less computations as compared to forward checking and and forward checking with MRV though its search tree is larger. Forward checking and forward checking with MRV have to compute unsafe places for next all the queens based on the previous queen's assignment which eats up more time. Minimum conflicts is as compared to the other three techniques, is very fast, since it eliminates initial queen placements and starts directly with a random complete assignment.

VII. Conclusion

Based on my experimental analysis, minimum conflicts approach proves to be the best approach for solving N-Queens problem as compared to backtracking, forward checking, and forward checking with MRV. It is better, both in terms of number nodes computed, i.e, equal to the number of steps in minimum conflicts as well as the time required to get to the solution. One by one computing the values for the variables recursively in the search tree does not help in the case of N-Queens. In fact, such techniques just need more time and huge memory for computing the solutions.

References

[1] Information on the n Queens problem http://theory.cs.uvic.ca/amof/e_queeI.htm

[2] The N-queens Problem https://developers.google.com/optimization/puzzles/queens#overview

[3] Kuruvilla Mathew and Mujahid Tabassum, Experimental Comparison of Uninformed and Heuristic AI Algorithms for N Puzzle and 8 Queen Puzzle Solution

[4] K. D. Crawford, Solving the N-Queens Problem Using GA. In Proceedings ACM/SIGAPP Symposium on Applied Computing, Kansas City, 1992, pages 1039- 1047.

[5] Marko Boikovi, Marin Golub, Leo Budin, Solving n-Queen problem using global parallel genetic algorithm

[6] Book by Stuart J. Russell and Peter Norvig, Artificial Intelligence, A Modern Approach, Third Edition

[7] Constraint Propagation http://ktiml.mff.cuni.cz/ bartak/constraints/propagation.html#compare