Exhaustive Study of Essential Constraint Satisfaction Problem Techniques based on N-Queens Problem

Md. Ahsan Ayub*, Kazi A Kalpoma†
*Department of Computer Science
*American International University-Bangladesh
†Department of Computer Science and Engineering
†Ahsanullah University of Science and Technology
Dhaka, Bangladesh
Email: {md.ahsanayub*, kalpoma†}@gmail.com

Humaira Tasnim Proma‡, Syed Mehrab Kabir§,
Rakib Ibna Hamid Chowdhury¶
Department of Computer Science
American International University-Bangladesh
Dhaka, Bangladesh
Email: {humairatasnimproma‡, mehrab.kabir§}
@outlook.com, rakib101025@gmail.com¶

***Abstract*—Constraint Satisfaction Problem (CSP) is observed in various applications, i.e., scheduling problems, timetabling problems, assignment problems, etc. Researchers adopt a CSP technique to tackle a certain problem; however, each technique follows different approaches and ways to solve a problem network. In this exhaustive study, it has been possible to visualize the processes of essential CSP algorithms from a very concrete constraint satisfaction example, N-Queens Problem, in order to possess a deep understanding about how a particular constraint satisfaction problem will be dealt with by the studied and implemented techniques. Besides, benchmark results - time vs. value of N in N-Queens - have been generated from the implemented approaches, which help understand at what factor each algorithm produces solutions; especially, in N-Queens puzzle. Thus, extended decisions can be made to instantiate a real life problem within CSP's framework.**

Keywords*—Arc Consistency (AC); Backjumping Algorithm (BJ); Backtracking Algorithm (BT); Constraint Satisfaction Problem (CSP); Forward Checking (FC); Least Constrained Values (LCV); Maintaining Arc Consistency (MAC); Minimum Remaining Values (MRV); N-Queens Problem

I. Introduction

Over the years of research, it has been possible to design and/or model many real world hard problem as a finite domain of constraint satisfaction problem. Therefore, the development of effective solution techniques for CSPs is an important research problem. Besides, a finite domain constraint satisfaction problem can be expressed into three components: a set of variables X, a set of domains D, and a set of constraints C. Here, states are defined by a set of variables X_i with values come from a set of domains D_i, and constraints C_i consists of a pair $(scope, rel)$, where $scope$ is a tuple of variables and rel is a relation that defines the values that those variables can take on. On the other hand, goal test is defined by a set of constraints that specifies allowable combinations of values for subset of variables. [1]

In this extensive study, an illustration is presented of a set of essential CSP techniques using a very concrete example of CSP, N-Queens Puzzle - where the desired goal state is easy to state. However, only one solution was considered with no preference as to which one.

A. N-Queens Puzzle: A Classic CSP Example

The N-Queens puzzle can be modeled as a CSP. Here, N is a positive set of integer values, and the problem is to place N number of queens on N number of distinct squares in an $N \times N$ chess board, by satisfying a certain set of constraints. In addition, two or more Queens cannot be placed in a same row as well as two or more Queens cannot be placed in a same column. Also, it is also not permissible to place Queens in diagonally: from upper left to lower right diagonal and lower left to upper right diagonal, as shown in Fig. 1.

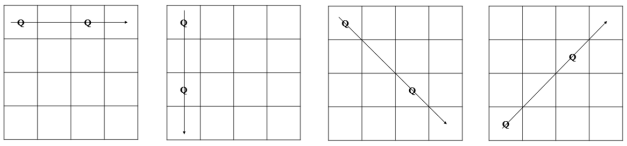

Fig. 1: All possible constraints or attacks in N-Queens Puzzle

II. Implementation of the Algorithms

In this work, as stated, it has been possible to successfully implement essential CSP techniques based on N-Queens Problem. In order to find solution by legally placing Queens in a $N \times N$ board, the chess board has been buffered from row to row, and a Queen will be tried to place in a particular column's single cell at each iteration. To summarize, the goal is to achieve any solution while starting from buffering first row of the board till the last row, and for each buffering, column's cells will be checked to place a Queen by satisfying all the constraints given in N-Queens puzzle.

A. Naive Techniques

The initial approach was to implement algorithms in the domain of Naive Techniques. However, it has been to possible to successfully implement and obtain desired result of Backtracking Algorithm (BT) and Backjumping Algorithm (BJ).

1) Implementation of BT: To start with, Queen is placed in the empty assigned board at $board[0][0]$ cell, as shown in Fig. 2(a). Then, 2nd column will be pointed in the next iteration at step 2, and as mentioned for each buffered row's index at the pointed column, constraints will be checked from the

978-1-5386-1150-0/17/$31.00

implemented modular function that checks all the constraints of N-Queens for the pointed cell. Thus, it is not possible to place Queen at 1st & 2nd cell at 2nd pointed column; however, for the 3rd cell, no constraints will be violated, resulting a safe placement for Queen, as illustrated in Fig. 2(b).

However, in the next iteration, Queen cannot be placed at any cells of the pointed 3rd column because of constraints, as shown in Fig. 2(c). Therefore, the process will backtrack to the previous assigned Queen's cell and reassign it to a new buffered cell at the same column, as shown in Fig. 2(d). In this way, if a Queen is failed to be placed at any pointed column, then it will backtrack and reassign the conflicted Queen to a new valid position and continue the operation likewise.

This is how the process will find a solution of 4-Queens problem by satisfying all the constraints, and it has taken 12 steps to complete, as shown in Fig. 2.

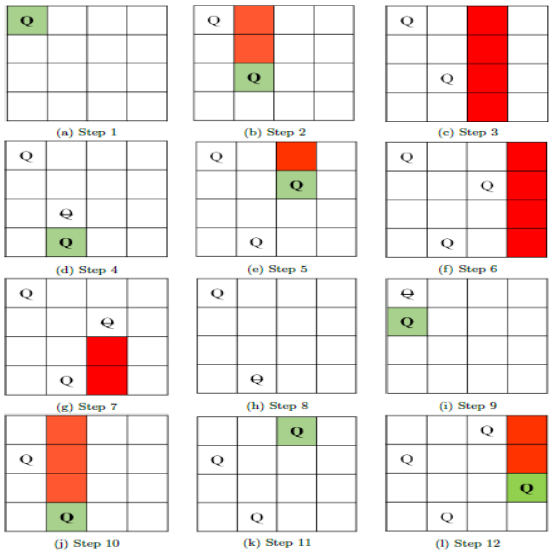

Fig. 2: Illustrative view to visualize Backtrack Algorithm for 4-Queens Problem

2) Implementation of BJ: To illustrate implementation procedures of this algorithm, a visualization is delineated as a demonstration. Like BT, 1st Queen is assigned to an empty board, as shown in Fig. 3(a). Then, while trying to find a permissible position at 2nd pointed column, conflicted recordings of Queens is being recorded in a Vector written in C language, as visualized in Fig. 3(b). As per Fig. 3(c), no Queen can be placed at 3rd pointed column, it is needed to decide - from the record stored in Vector - where to jump back.

In addition, instead of backtracking to the 2nd column - previously Queen assigned column, it should jump back to the 1st column. The answer is derived from the implemented modular function that returns which column should the buffering jump back. Thus, the conflicted Queen is reassigned to a newly valid position at column 1, as shown in Fig. 3(d). In this manner, the final successful result has been eventually achieved of 4-Queens puzzle using BJ algorithm, as illustrated in 3.

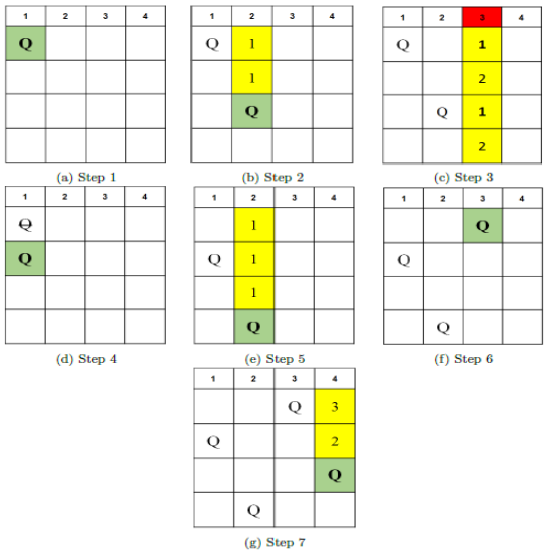

Fig. 3: Illustrative view to visualize Backjump Algorithm for 4-Queens Problem

B. Filtering Techniques

From the literature review, it was seen that applying filtering Techniques will lead to gain better results, and the implemented essential Filtering Techniques are: Forward Checking (FC), Arc Consistency (AC) and Maintaining Arc Consistency (MAC).

1) Implementation of FC: With a view to demonstrating how the implemented approach works, again the visualization part of FC algorithm has been given, as illustrated in Fig. 4. As previous, the first Queen will be placed in an empty board, as shown in Fig. 4(a). Then, it will call the implemented modular function to create FC consistency to other variables. At step 3, Fig. 4(c), it can be seen, a Queen cannot be placed at any cells at the pointed 3rd column, and thus the buffering needs to backtrack. So, calling to the implemented function is responsible to clear the constraints which was created by the conflicted Queen at 2nd column and then reassign it to a new valid position.

In this way, the process can traverse through the full CSP network. Finally, a successful state can be obtained where all the Queens are safely placed without violating any constraints. [2], [3]

2) Implementation of AC: There have been various techniques proposed to implement Arc Consistency (AC) algorithm so far; however, in this proposed work, the focused was on the AC-3 algorithm.

Now, to demonstrate the approach of AC-3, an illustration has been provided to depict. At first, the first Queen will be assigned to the first index of row and column. Then, it will create ARC consistency by calling the modular created function, as shown in Fig. 5(a). Followed by, backtracker will avoid constrained cell in the next pointed column, and it will check whether it is safe to place a Queen in the unconstrained cells or not, as illustrated in Fig. 5(b).

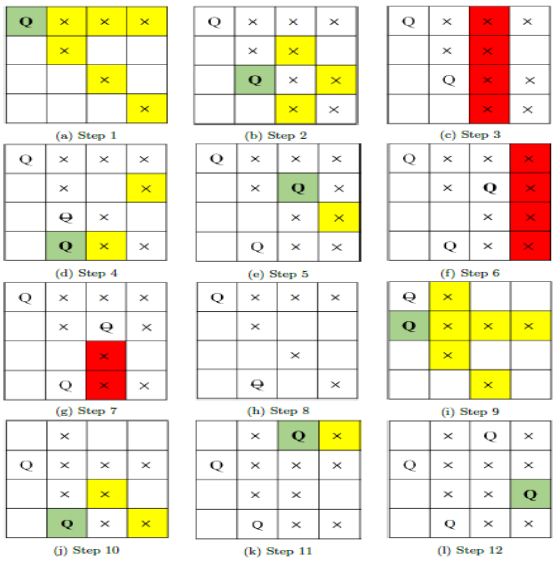

Fig. 4: Illustrative view to visualize Forward Checking Algorithm for 4-Queens Problem

Conversely, if it does not become possible to place a Queen at the pointed column, like in Fig. 5(c), backtracker will backtrack the previous assigned Queen's column to reassign the conflicted Queen to a newly safe cell at the same column. Then, it will, at first, clear the constraints it created by calling the implemented modular function.

In this way, the whole CSP network will be traversed until it reaches to a solution, and in the given visualization, it has taken 12 steps, as visualized in Fig. 5. [4], [5]

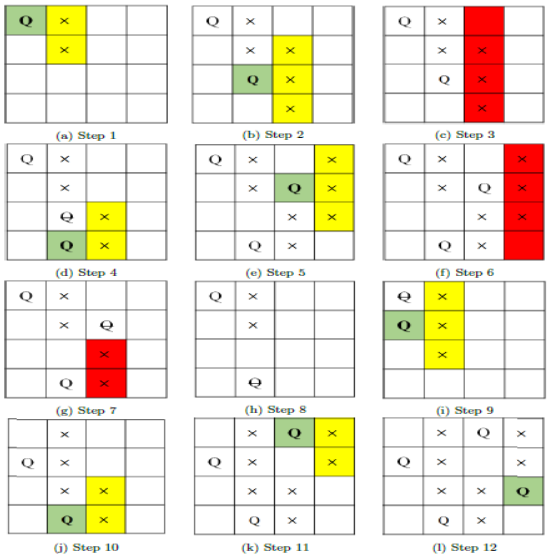

Fig. 5: Illustrative view to visualize AC-3 Algorithm for 4-Queens Problem

3) Implementation of MAC: In this study, MAC algorithm has been implemented by enforcing the previously implemented AC-3 algorithm. However, it is possible to implement MAC with AC-4, AC-5, AC6, AC-7 and AC-2001. [6]

To visualize the simulation in Fig. 6, at first, there will be an assignment step in MAC. Then, it will enforce arc consistency in the next pointed column. To follow, it will find a safe cell at just created constraints' column and again create arc consistency to the next column, as shown in Fig. 6(b). However, if a followed pointed column's unconstrained cell returns an unsafe placement for a Queen, then it will re-point to the previous column and create arc consistency again, as shown in Fig. 6(d). This process will continue till buffering process in the board returns a success state. In Fig. 6(h), *mac* subroutine return a false result. So, backtracker will reassign the only assigned Queen to a newly safe position, and then it calls the subroutine again.

In this assignment, as shown in Fig. 6(i), *mac* subroutine can place all the Queens to the board successfully. Thus, it returns a true result, as illustrated in Fig. 6(l). By this way, MAC algorithm finds a satisfied solution of 4-Queens problem in just 4 assignment steps. [6]–[8]

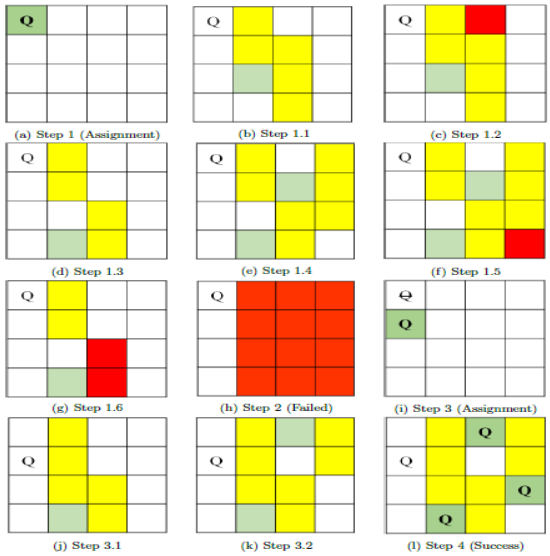

Fig. 6: Illustrative view to visualize Maintaining Arc Consistency Algorithm for 4-Queens Problem

C. Ordering Techniques

In this proposed study, it has been possible to successfully implement two essential ordering techniques: Minimum Remaining Values (MRV) and Least Constrained Values (LCV). Both can be added to different CSP algorithms, but MRV and LCV have been implemented with Forward Checking algorithm. [9]

1) Implementation of FC-MRV: 6-Queens problem's illustration view has been demonstrated in order to have a wider idea of how this heuristic is implemented; full visualization is shown in Figure 7.

At first, FC's restrictions will be added to all columns' cells for the placed Queen, and total number of constraints of each column is recorded below, as shown in Fig. 7(a). In this manner, the process will continue till step 4, as shown in Fig. 7(d). Here, the pointed column is 4, but the process can detect early that all the cells of 6th column have

already been exhausted. So, without traversing to the next iteration, the implemented modular function will will backtrack at 4th column's conflicted Queen's cell from the detection. Consequently, it saves many unwanted iteration(s).

In this way, the process will travel throughout the CSP network and provide early detection if a dead-end occurs to any column before even reaching there.

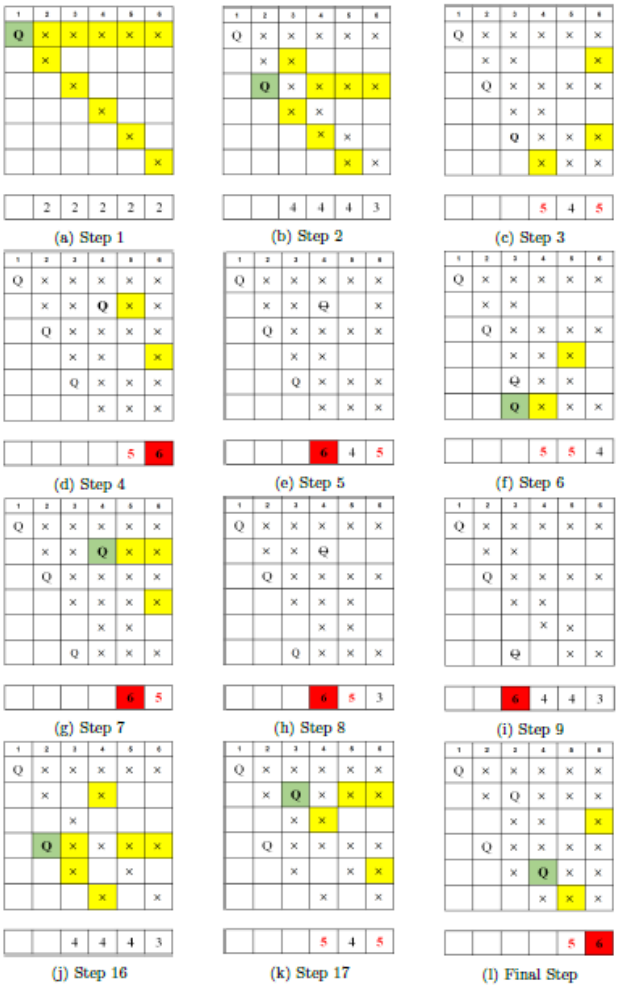

Fig. 7: Illustrative view to visualize FC-MRV for 6-Queens Problem

2) Implementation of FC-LCV: To illustrate the approach to implement FC-LCV, a step-by-step simulation of 4-Queens problem has been demonstrated in Figure 8.

To begin with, it starts with an empty assigned board, and then places a Queen at the first row, first column. Then, it creates FC constraints while storing the records in an array's indexes of the corresponding columns, as shown in Fig. 8(a). In the next step, at first the process pointed at the unconstrained cell of the 2nd column, and then it checks whether or not any failure will be observed by the help of the record array. In Fig. 8(b), it is shown that the pointed cell will be the reason of a failure attempt later. Thus, it travels to the next unconstrained cell of the 2nd column, which ensure a safe place for Queen's placement. So, the Queen assignment is observed in Fig. 8(d).

In general, this is how FC-LRV places a Queen before actually assigning to a cell, which results avoiding unwanted iteration(s). For this reason, it can be seen assignment steps in Fig. 8(i) & Fig. 8(k). So, FC-LCV takes only 9 steps to successfully place all the Queens in 4×4 board.

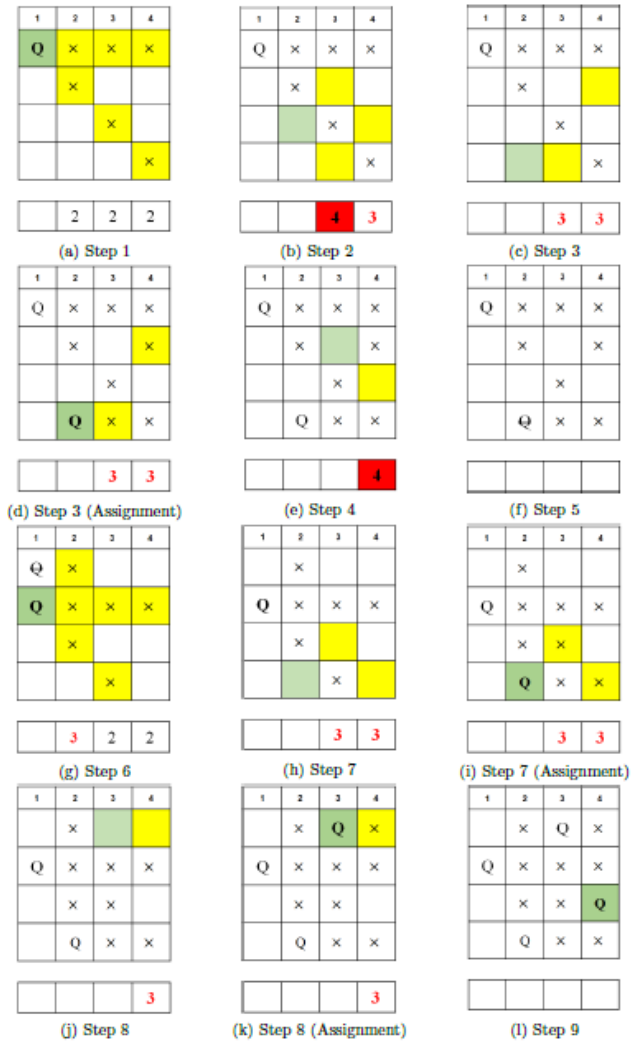

Fig. 8: Illustrative view to visualize FC-LCV for 4-Queens Problem

III. Experimental Result and Analysis

To begin with, Backjumping Algorithm failed to generate result after the value of N becomes 16, whereas, Backtracking Algorithm produced result till $N = 33$ and then failed. It happened because BJ used to fall in to an infinite loop when N's value gets large, and thus it cannot decide the main conflicted Queen to jump back in board, see comparison graph in Fig. 9.

Then, AC-3 and MAC should be compared between themselves, and the implemented AC-3 approach obtain results faster than MAC algorithm. Also, it is to note, both algorithms failed to perform better compared with another implemented Filtering Technique, Forward Checking (FC), as shown in Table I. Besides, MAC provides solutions for larger values of

TABLE I: Performance of Implemented Filtering Techniques, showed till time = 30 minutes (* = Exceeding time limit)

N	Time (in seconds)		
	FC	AC	MAC
26	0.6	25.7	123.9
28	5.4	358.5	811.9
30	59.3	1475.7	1524.2
32	105.5	*	*

N where AC-3 failed, but FC provides the best result compared in the domain of Naive and Filtering Techniques.

TABLE II: Performance of FC & Implemented Ordering Techniques with FC, showed till time = 30 minutes
(* = Exceeding time limit)

N	Time (in seconds)		
	FC	FC-MRV	FC-LCV
28	5.4	1.2	2.3
30	59.3	23.9	42.7
32	105.5	39.8	63.2
34	*	1009.2	1702.6

On the other hand, FC with heuristics has been added which actually brought more fruitful results. Both FC-MRV & FC-LCV can generate almost the same result; however, FC-MRV provides results at less time cost as well as FC's flaws have been eliminated while adding it with the heuristic approaches, see significant comparisons in Table II.

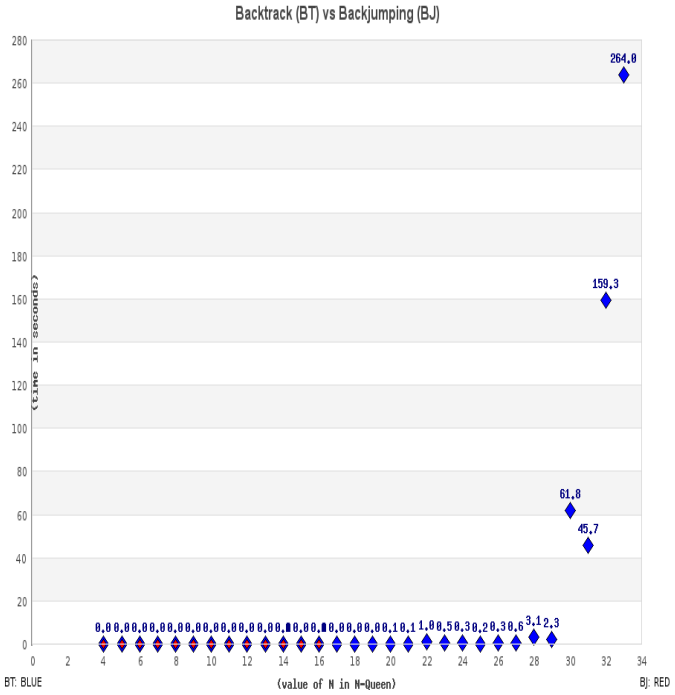

Fig. 9: Time vs Value of N in N-Queen graph for Backtrack Algorithm vs. Backjump Algorithm

However, every algorithm that was described above compiled and executed in C, and then results, that is, performance of each algorithm for corresponding value of N is N-Queens problem, were derived. In addition, overall result of the study

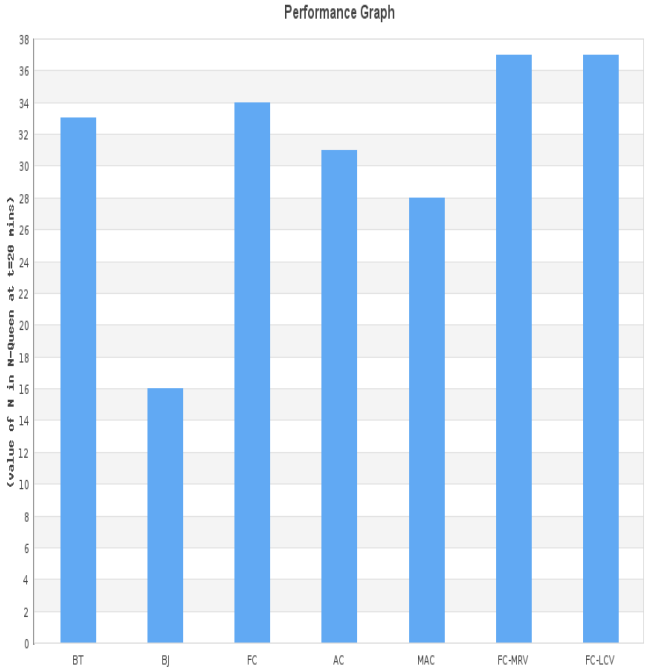

Fig. 10: Time vs Value of N in N-Queen graph for all algorithms at time = 20 mins.

can be plotted to a single bar graph restricted at run time, t = 20 minutes, as visualized in Fig. 10.

IV. Discussion

It was anticipated from the exhaustive literature review that algorithms of Naive Techniques - as the name suggests - will fail to handle larger value of N at reasonable time cost, which inspired us to implement algorithms of Filtering and Ordering Techniques. Such algorithms try to avoid the dead-ends that were observed in Naive Techniques. By doing so, better results are derived compared with Backtracking algorithm and Backjumping algorithm.

On the other hand, heuristics approach has surpassed all the other techniques, which is also agreeable. Although all of the implemented algorithms in filtering and ordering techniques could provide results for more value of N in N-Queens puzzle than what it is showed in Fig. 10 and Table I & II, all the results had been neglected after 30 minutes of run time taken by each value of N.

All the implemented algorithms have been executed and compiled several times, and then the mean value of the run time has taken to provide the benchmarking. Additionally, every implementation is implemented in C language, and then the source file is compiled in Linux OS Distribution, Ubuntu 16.04 LTS, using GCC - GNU Compiler Collection - command. Besides, by the time of implementation, solutions have been derived from an infinite loop of N's value starting from 4 because it is not possible to achieve a solution for $N = 2$ and $N = 3$ in N-Queens problem. In addition, at first the time vs value of executed N written to text file from

which using PHP libary, named JPGraph 4.0.1, graphs were generated.

V. Conclusion

A varieties of different real life problems can be formulated as instance of CSP. Additionally, each problem domain contains various sets of constraints. N-Queens problem contains $N \times N$ variable, and the larger the value of N gets, the more complex it becomes. Thus, in this exhaustive study, one of the purpose is to find out the performance of essential CSP techniques by benchmarking so that it can be possible to possess an idea about which technique should be used at certain scenario(s). In this way, by taking a fixed problem domain, all the time vs value of N in N-Queens results have been derived to compare.

It is also shown in this study how each algorithm works in order to obtain solution of a CSP; especially, in N-Queen Puzzle. However, the overall idea will still remain the same. So, visualization of each technique provides step-by-step guideline to reach the goal state of CSP with a set of constraints.

However, the proposed work had not been able to combine every CSP technique that has been practiced and offered by researchers to solve such disciplinary problems, which is aimed to amalgamate in future. In this manner, adding more advanced comparison and in-depth analysis is highly expected.

References

[1] S. J. Russell and P. Norvig, *Artificial Intelligence A Modern Approach*, 3rd ed. Upper Saddle River: Prentice hall, 2003.

[2] C. Bessire, P. Meseguer, E. C. Freuder, and J. Larrosa, "On forward checking for non-binary constraint satisfaction," *Artificial Intelligence*, vol. 141, no. Issue 1-2, pp. 205–224, 2002.

[3] F. Bacchus, "Extending forward checking," in *International Conference on Principles and Practice of Constraint Programming*, 2000, pp. 35–51.

[4] Lecoutre, C. and Hemery, F., "A study of residual supports in arc consistency," *IJCAI*, vol. 7, pp. 125–150, 2007.

[5] Dib, Mohammad and Abdallah, Rouwaida and Caminada, Alexandre, "Arc-consistency in constraint satisfaction problems: A survey," in *2010 Second International Conference on Computational Intelligence, Modelling and Simulation*, 2000, pp. 291–296.

[6] Rgin, J.C., "Maintaining arc consistency algorithms during the search without additional space cost," in *International Conference on Principles and Practice of Constraint Programming*, 2010, pp. 520–533.

[7] B. Neveu and P. Berlandier, "Maintaining arc consistency through constraint retraction," in *Tools with Artificial Intelligence, 1994. Proceedings., Sixth International Conference on*. IEEE, 1994, pp. 426–431.

[8] D. Sabin and E. C. Ereuder, "Understanding and improving the mac algorithm," in *International Conference on Principles and Practice of Constraint Programming*. Springer, 1997, pp. 167–181.

[9] M. Mouhoub and B. Jafari, "Heuristic techniques for variable and value ordering in csps," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 2011, pp. 457–464.