HEURISTIC APPROACHES FOR SOLVING N-QUEENS PROBLEM

Aftab Ahmed[1], Ali Kamran[2], Mazhar Ali[1], Abdul Wahid Shaikh[3]

[1]Department of Computer Science, Balochistan University of Information Technology, Engineering & Management Sciences, Quetta; [2]Institute of Space Technology, Islamabad; [3]Department of Computer Science, SALU, Khairpur

Abstract

Abstract: *The research article examines the three distinguished heuristics approaches for solving the N-Queens problem. The problem is widely recognized as constraint satisfaction problems (CSP) in the domain of Artificial Intelligence. The N-Queens problem demands the non-attacking placements of finite number of queens over chessboard. So that, two or more queens cannot share the horizontal, vertical and diagonal positions in a straight line. In this research work, improved version of Backtracking Recursive Algorithm, modified Min-Conflicts Algorithm and classic Genetic Algorithm are applied to address the problem. The comparative results validate the efficiency of research direction.*

Keywords: *N-Queens problem, heuristics, Constraints;*

Corresponding Author's email: aftab.ahmed@buitms.edu.pk

INTRODUCTION

The N-Queens Problem is a well known constraints satisfaction problem (CSP) which can be defined by a set of variables Q_1, Q_2 … Q_n, where each variable can have its own finite domain of values, such as D_1, D_2, … D_n and third component is a set of constraints which can be represented as C_1, C_2…C_n. Classic solving approaches may produce quite satisfactory results in reasonable time in case of small chess-boards however computing time & complexity exponentially increases with size of problem. The N-Queens problem belongs to the group of Non-deterministic Polynomial-time•Hard (NP-hard) problems; therefore, finding its optimal solution is a significant issue of operation research as well as in Artificial Intelligence. The prime task of optimization is to obtain reasonable results in a polynomial time. In fact, N-Queens problem stands in combinatorial optimization so perfect results along with all situations cannot be expected.

This article inspects the efficiency and accuracy of Constraints Logic Programming in order to obtain feasible solutions for N-Queens problem. The aim of CLP implication over the N-Queens problem is to provide evidence of accurateness of adopted approach for identical real world/benchmark Problems such as University Scheduling Problem, Salesman travelling Problem etc.

The successive sections are arranged as follows:

The Section II provides a brief but up-to-date literature review over Constraints Logic Programming and N-queens problem. Section III is specifically focused on formulating the problem and describing its basic ingredients. Section IV illustrates the adopted solving approach while the section V discusses the experimental outcome and finally Section VI concludes with future research direction.

Background

The key advantage (Ligierski, 2002) of CLP is defend as *"a straightforward statement of the constraints serves as part of the program. This makes the program easy to modify, which is crucial in real-world problems".* Furthermore, when problems (White and Zhang, 1998) are formulated as constraint logic problems they become known as

Constraint Satisfaction Problems, (CSP). Constraint is a logical relation among variables where each variable represents its particular domain. Generally, a constraint is kind of filter that extracts out the acceptable state of solution among various available options. A good benchmark problem for evaluating CSP is a 8-Queens problem (Russess and Norvig, 2003) with *64* squires and maximum eight queens placement availability. The key criterion is as no any of queens supposed to attack other queen. There may have plenty of solution instances can be found out with different deployment of queens. The problem imitates the hard constraints satisfaction of scheduling problem, but in queens problem no ultimate feasible solution is acceptable. A CSP approach takes out values from domains according to prescribed constraints and return with exact solution(s) or optimal solution(s) which is very much required for NP-complete problem just as timetabling. Zhang and Lau, 2005 distinguishes; "*there are two approaches to solving CSP. One is using the search algorithms and the other is using the consistency technique. Constraints logic programming deals with plenty of problem including famous NP-hard set of problems. CSP combined different techniques to produce optimal solutions which would be acceptable on some certain measures".* In their research work (Arash, 2011) elaborated, cooperative PSO was used in quest to get better results than classical PSO. The chosen method provided parallel searching. Takeshi Onomi, 2011 have proposed a neural network using coupled–SQUIDs for solving the N-Queens problem and obtained promising results. Genetic Algorithm is always remained a prime attraction of many researchers due to its robustness and computing maturity. Bozikovic, 2003 genuinely applied the GA over N-Queens problems. The research work revealed the potential of GA for solving Queen Problem. Bespoke chromosome representation, fitness function and other genetic operators are defined and used in quit pretty way. The CSP solving approach is differentiated than other in terms of prices coding, accuracy in results. A single page of program is fully capable to solve dynamic size of problem instances.

Problem Description

Queen is strongest piece of chess board that can move across the line in horizontal, vertical or diagonal directions so that very few positions remain secured.

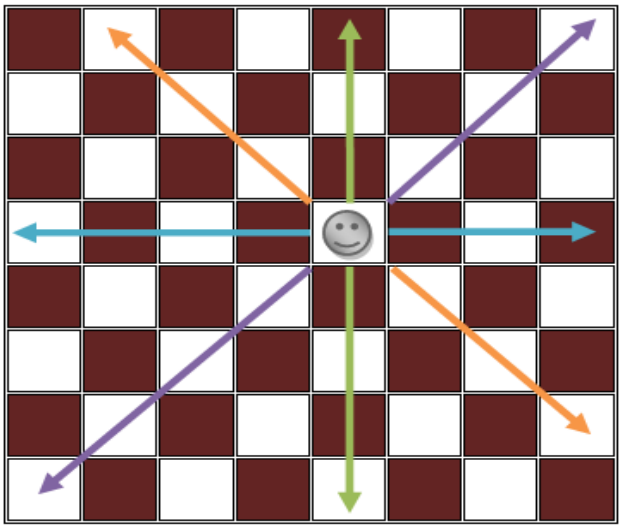

Figure 1: Queen Attacking Search Space.

The Classically N-Queen problem is a placement of predefined number of Queens over *n×n* chessboard in such way no queen can come into attacking range of other. In other words, two or more Queens cannot share the same row, column or both diagonal. Figure 1 is illustrating the state space of problem.

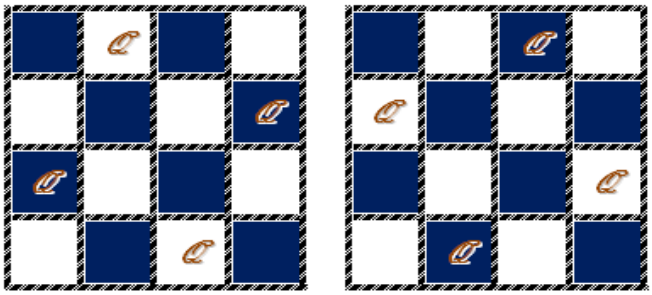

Figure 2: Two possible solutions of 4x4 chess-boards

The various placements of queens may provide multiple outcomes for example a chessboard of *8x8* possibly provide total 92 accurate results. Figure 2 is exemplifying the two possible results of small size of board.

Goal Formulation

N-Queen problem is a prominent combinatorial problem which contains a set

of hard constraints. In this article, the problem is simultaneously addressed by the Backtracking Recursive Search algorithm and Min-Conflict Algorithm.

Improved Backtracking recursive search

Backtracking recursive search is an intelligent use of recursion. The scope of BTS algorithm implies in scenario when one option is to be preferred among a group of interwoven choices and in consequent of such action, descendant options further are unfolded. The procedure is iterated continually until it reaches the appropriate state. Entire sequence of selection options gradually formulates the goal state of problem. The BTS process shapes a logical tree of decision making. Each option metaphors a node, starts with the root of tree deepening into leave nodes. If the logic gets an improper options it revoke from the recent option and rollbacks from the track or prune that part of search tree and go for another node.

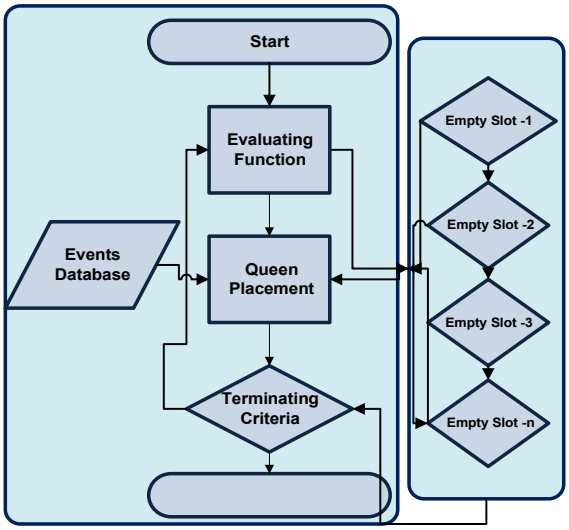

Figure 3: Logical Flow of Backtracking Recursive Search

Logical flow of significant steps is illustrated in figure 3.

Procedure Evaluation (Q, *i*):

1. *n* = Length(Q) [Distance between rows]

2. **For** j = 1 To Q **Do** where Q = $\{q_1, q_2, q_3 \ldots q_n\}$

3. **If** Q_j == *i*: [Same column]

4. **ElseIf** $Q_j + n == i$: [Horizontal diagonal]

5. **ElseIf** $Q_j - n == i$: [Vertical diagonal]

6. **Return True**

7. *n* = *n* - 1

8. **Return False**

The *Evaluation Function* performs the prune test and perceives constraint clashes of attacking queens on each other. The process repeats and ensures the validity before each placement of new entry. The mechanism proceeds in-depth- first order, in case of any violation backtracking mechanism rollbacks from pruned session and gets way to neighboring session.

Algorithm 1: Improved Backtracking Recursive Search

1. Procedure BTS
2. Repeat
3. IF All The Placement are Assigned Then Exit
4. For All the Periods
5. IF Evaluation (QList, New) == TRUE
6. THEN
7. Place the New Queen
8. Call BTS(QList[Counter + 1])
9. ELSE
10. Select Succeeding Period
11. End For
12. Until True

Although the list of placements is an array of strings but backtracking method traverse as tree in recursive order along with pruning test. The BTS algorithm used here is improved with “looking ahead” capability.

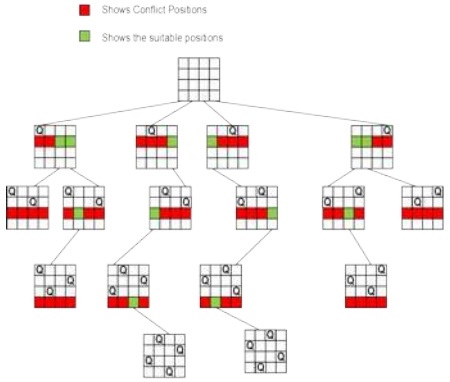

Figure 4: Backtracking Logic Flow

The Algorithm-1 navigates the nodes down in the depth-first manner. Each failure in event placement extends the sub-tree one step further until leaves of the search tree come in reach. Likewise, every event from the list is supposed to move back and forth in a search tree and avail the validate placement in sessions. The Backtracking search method is very accurate and fast for most of the datasets; Error! Reference source not found. depicts the overall mechanism.

Modified Min-Conflict Algorithm

Min-conflicts• algorithm randomly picks out any conflicting Queen and then chooses a squire of same row which is comparatively violated with small number of constraints. In case of no such available position, it chooses the random placement ensuing penalty does not increase with move. Min-conflict is an effective solver working on the pattern of Detect and Repair. The evaluation function analyzes status of chessboard and stamps out the conflicting queens. In this work, the algorithm selects the highest conflicting queen. The key idea is to slide the conflicting queen on minimum conflicting cost raised on the row.

Figure 5 and Algorithm 2 are showing the logical flow of Min-Conflict Algorithm. In its modified version acceptace criteria is added in classical algorithm which provides resistances against local optima. The acceptance criterion is very similar to Simulated Annealing technique.

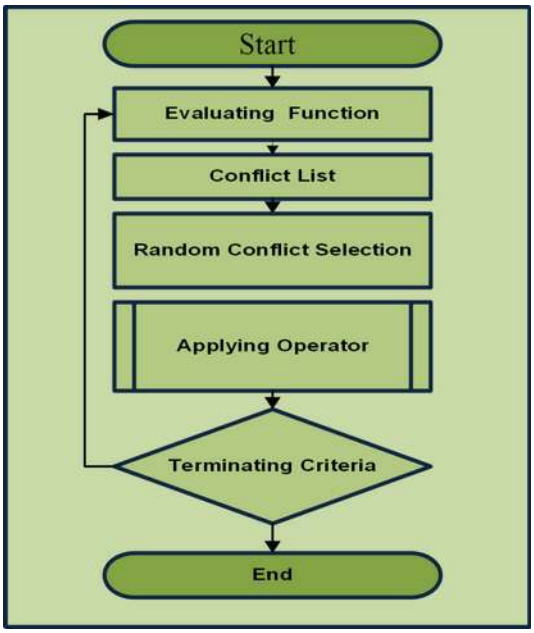

Figure 5: Min-Conflict Logic Flow

Algorithm 2: Min Conflict Method

1. Procedure MinConflict
2. Conflicted-List = Evaluation()
3. Repeat
4. Max_Conflicted _Queen = Max_Penalty (Conflicted-
5. For Column = 1 To EntireRow
6. Placement = Min_Penlaty(Check Placment (Column))
7. EndFor
8. IF Current_State > Privous_State Than
9. Accept NewState: Move Queen (Placement)
10. Else IF Current_State < Privous_State
a. IF Acceptance_Criteria Than
b. Accept NewState: Move Queen Placement)
11. Else Rollback Previous_State.
12. EndFor
13. Until (Maximum-Iterations).

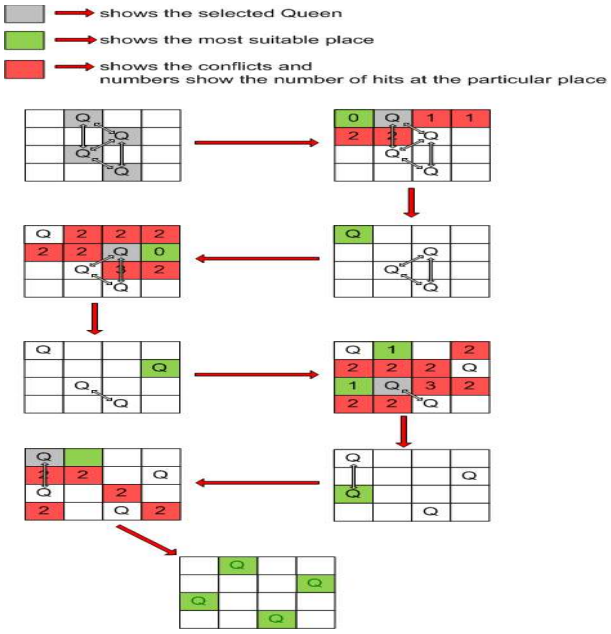

Figure 6: Classical Logical flow of Min-Conflict Algorithm

RESULTS

The empirical results are presented here, many of the parameters and operators have been examined in the experiments. All the tests are performed on Lenovo® Intel CORE ™ i3, 2.27 GHz, 2.00 GB RAM. The Python language version 2.6 is adapted to code the research work.

Table 1 Backtracking experimental Results

Chessboard Board Size N x N	Number of Possible Solutions	Execution Time
1 x 1	1	Less than 01 Sec
2 x 2	0	Less than 01 Sec
3 x 3	0	Less than 01 Sec
4 x 4	2	Less than 01 Sec
5 x 5	10	Less than 01 Sec
6 x 6	4	Less than 01 Sec
7 x 7	40	Less than 01 Sec
8 x 8	92	Less than 01 Sec
9 x 9	352	0.53 Sec
10 x 10	724	1.01 Sec
11 x 11	2680	1.40 Sec
12 x 12	14200	5.2 Sec
13 x 13	73712	31.6 Sec
14 x 14	365596	3.40 min
15 x 15	2279184	26.45 min

Table 1: is illustrating the possible number of solutions of *NxN* board in finite time.

Table 2 all four solutions of 6x6 chessboard

	R_1	R_2	R_3	R_4	R_5	R_6
S_1	2	4	6	1	3	5
S_2	3	6	2	5	1	4
S_3	4	1	5	2	6	3
S_4	5	3	1	6	4	2

Table 2 exemplify the possible solutions of 6x6 chessboard. The variables R_n demonstrates the specific row and the integer given it bellow shows the column number of Queen Placement. S_n depicts the set of single solution.

Table 3 Outcome of 20x20 chessboard using Min-Conflict

Iterations No	Cost	Acceptance	Improvement	Time
10000	48.00			0:00:00
9000	40.00	99.97%	38.07%	0:00:16
8000	35.00	99.84%	38.09%	0:00:32
7000	30.00	99.27%	37.84%	0:00:48
6000	21.00	96.66%	36.38%	0:01:04
5000	16.00	85.30%	30.06%	0:01:20
4000	12.00	52.00%	14.08%	0:01:36
3000	08.00	11.37%	0.87%	0:01:53
2000	06.00	05.12%	0.48%	0:02:13
1000	0.00	01.37%	0.17%	0:01:33

Table 3 is depicting the a solution of 20x20 chessboard, at the last stage of iterations lower ratio of acceptance can be noticed due to saturation of suitable placements.

CONCLUSIONS

The research article investigated the Constraints Logic Programming approach that proved to be an exceptionally capable of providing numerous feasible solutions of N-Queens Problem. The objective of study was to measure and record the effectiveness and efficiency of CLP approaches so they can be proposed to other identical combinatorial optimal Problems. The promising experimental results substantiate the research direction. In future, a comparative study on N-Queen problem is planned to conduct using various prominent solving approaches including Genetic Algorithms, Particle Swarm Optimization and Simulated annealing etc testing over the parameters of time complexity and space complexity.

REFERENCES

- Arash A. (2011). "Presenting a new method based on cooperative PSO to solve permutation problems: A case study of n-queen problem," in International Conference on Electronics Computer Technology (ICECT). 218 - 222.
- Bozikovic M. (2003), "Solving n-Queen problem using global parallel genetic algorithm." 104 - 107.
- Legierski W. (2002). "Constraint-based reasoning for timetabling," in AI-METH 2002, Gliwice, Poland.
- Russell S and Norvig P. (2003). Artificial Intelligence A Modern Approach Prentice Hall.
- Takeshi O. (2011). "Superconducting Neural Network for Solving a Combinatorial Optimization Problem," IEEE Transactions on Applied Superconductivity.1: 701 - 704.
- White GM and Zhang J. (1998). "Generating Complete University Timetables by Combining Tabu Search with Constraint Logic," in The Practice and Theory of Automated Timetableing PATAT 1997. 187-198.
- Zhang L and Lau S. (2005). "Constructing university timetable using constraint satisfaction programming approach," in International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce. 55-60.