Artificial Intelligence (CSCI-6511)

Project 2: Constraint Satisfaction Problem

Spring 2024

Author: Dursun Dashdamirov

Date: 11.03.2024

Introduction

This paper explains the approach and code utilized to solve N-queens problem. This problem includes placement of N queens in N-by-N chess board. The queens should not attack each other after placements. The following sections include the path to the code and implementation details.

How to use the code

In order to run the code, Python should be installed on your computer. Subsequent steps to download the application and steps to run it are explained in READ.ME file. The file and necessary codes to run and test the application can be found in the following GitHub repository *(https://github.com/DDursun/AI_NQuenns.git).*

Design of algorithm

The code is written in Python programming language. In order to maintain the modularity of the code, the processes are split into functions. There are 9 functions in total (excluding *__init__*):

1. **input_reader**: Function reads an input from either a file or directly from the user, if keyword is input instead of path to file, function proceeds with random board initialization.
2. **check_consistency**: Function checks if assigning value to a variable maintains consistency with the current assignment.
3. **select_unassigned**: Function selects unassigned variables.
4. **prioritize_domain_values**: Determines the possible values for a given variable using the Least Constraining Value (LCV) heuristic. This method prioritizes values that impose the fewest constraints on neighboring variables.
5. **revise_domains**: Functions iteratively revises the domains of variables x and y to ensure arc consistency is maintained through iterations. The values which are not consistent are removed at each iteration.
6. **apply_arcconsistency_algorithm**: Utilizes AC3 algorithm to achieve arc consistency across all variables.
7. **attempt_solution**: Function recursively searches for a solution to a Constraint Satisfaction Problem (CSP) by backtracking. It iterates through possible variable assignments, ensuring consistency and exploring potential solutions until either a valid solution is found, or the search space is exhausted. If successful, it returns the solution; otherwise, it returns None.
8. **visualize_board**: Creates visualization of two boards; one for initial positions of the queens and one for found solution.
9. **resolve_nqueens**: Main function which tackles the N-Queens challenge employing Constraint Satisfaction Problem (CSP) strategies.

Implementation details

Some of the technical implementation highlights about the code are:

Constraint Satisfaction Problem (CSP) Approach: The code employs CSP tactics to solve the N-Queens problem, engaging techniques like backtracking search and constraint propagation through the AC3 algorithm to effectively explore the solution space.

Heuristic Implementation: It incorporates heuristics such as Least Constraining Value (LCV) to prioritize domain values, enhancing the efficiency of variable assignment selection and pruning of solution paths, leading to faster convergence to a solution.

Visualization and Time Statistics: The code not only finds solutions but also presents visualization using Matplotlib, displaying initial and final queen positions on a chessboard grid. Additionally, it reports execution time, offering awareness into the algorithm's efficiency.

Conclusion

The N-queens problem is successfully solved using Python code which gives flexibility of providing pre-determined set up of queens of random initialization. In both settings, algorithms effectively achieve the convergence within reasonable time.