Programming Assignment 4: Constraint Satisfaction Problem

Problem Description

Consider a problem of magic square of size n×n. The values for cells in n×n square are stored in a square matrix and the values range from 1 to n^2. The cells contain unique values such that the sum of the numbers in its any row, column or diagonal is equal to the sum of numbers in any other row, column or diagonal. Formulate the problem as a constraint satisfaction problem and solve it using Depth first search with backtracking. When n is large, the number of constraints tends to increase exponentially. Define two general constraints named as ***Alldiff*** and ***sumConstraint*** and implement as functions which return a boolean true value if these constraints are not violated. Use a construct appropriately to store the partial assignments. Handle the complex situation of broader DFS tree by reducing the domain size using constraint propagation and use appropriate heuristics such as MRV (Minimum remaining value) and degree heuristic to decide upon the variable ordering. An example magic square of size 3×3 is shown below.

2	**9**	**4**
7	**5**	**3**
6	**1**	**8**

Ensure the scalability of your solution to problems with sizes n×n, with values of n ranging from 3 to 10. While the problem is represented as CSP, the constraints should be represented as constraint graph. The data structure of constraint graph has variables as nodes and constraints as edges. Also variable nodes have their value domains associated with them. You have the flexibility of design to suit the needs.

Implementation

Use Python version 2.7.13 (Windows 10) for implementing your solution. Only standard Python Libraries should be used. Support from external sources or libraries such as github will not be accepted in your submissions. Each student must design own solution and write own code. [Refer handout to understand the malpractice policies.]

Modules

Implement the following techniques.

1. Constraints Alldiff() and sumConstraint() returning boolean values. You can design the list of input parameters according to the need. A partial assignment can always be one parameter.
2. **DFS_BT (CSP, variable_list, constraint_graph) returns assignment**: Takes as input the CSP problem description, a list of variables and the constraint graph and returns an assignment of values. An Assignment is a n×n matrix holding the values in its cells. Constraint graph represents the constraints as edges and nodes as variables. A constraint graph is used for accessing the neighboring nodes for verifying whether or not is any constraint violated. Use a prompt for choice of heuristic for value ordering and implement accordingly. The default variable ordering is simply the variables in sequence 1 to n^2.
3. **DFS _BT_ Constraint_Propagation (CSP, variable_list, constraint_graph) returns assignment**: The implementation uses constraint propagation to reduce the value domain and implements DFS with backtracking. Use a prompt for choice of heuristic for value ordering and implement accordingly. The default variable ordering is simply the variables in sequence 1 to n^2.

Analysis Module

Produce the following analyses and display the resultant values.

(a) DFS+backtracking algorithm based analysis

i. Compute the number of nodes generated till the problem is solved. **[R1]**
ii. Compute the amount of memory allocated to one node. **[R2]**
iii. Compute the maximum growth of the implicit stack (if recursion is used) or of explicit stack used with the search tree. **[R3]**
iv. Compute the total time to compute the values. **[R4]**
v. Use an appropriate heuristic such as MRV or degree heuristic and compute the number of nodes generated till the problem is solved. **[R5]**

(b) DFS+ backtracking using constraint propagation algorithm based analysis

i. Compute the number of nodes generated till the problem is solved. **[R6]**
ii. Compute the ratio (R1 - R6)/R1 as saving using constraint propagation. **[R7]**
iii. Compute the total time to compute the values. **[R8]**

(c) Comparative analysis

i. Compare R4 and R8.

Graphics

Divide the screen into two parts - one to display the magic square and the other to display the values R1 to R8.

Driver

The driver must integrate all functionalities and execute the functions appropriately using these options

Option 1: Execute DFS+BT with and without the heuristic. Prompt for the choice of use of heuristic first. Show the output magic square computed by your program on the console.

Option 2: Execute DFS+BT+Constraint_Propagation with and without the heuristic. Prompt for the choice of use of heuristic first. Show the output timetable in the format specified above on the console.

Writeup, evaluation and submission

Write up details will be made available two days before the submission. Evaluation will be out of 16 marks (8% weight). Students are advised to inform me immediately, if any discrepancy exists in this document. The assignment is due for submission on November 2, 2018 (Friday) by 6:30 p.m. Students should clarify all their doubts pertaining to the problem specification, explanations given above, conceptual understanding, doubts related to constraint graph or the data structure to be used, and the doubts relating other aspects of implementation.

Please feel free to write me an email for a mutually convenient time for discussion.

Vandana
October 22, 2018
