Programming Assignment 4: Constraint Satisfaction Problem

Problem Description

Consider a problem of scheduling meetings of students with eminent personalities who all happen to visit your institute almost at the same time during fourth week of October this year. Let us denote these eminent personalities as a set of variables $X = \{ N_i : 1 \le i \le m\}$, where m is the number of these eminent personalities coming to the institute. Students are eager to meet many of them and you must schedule meetings such that no student has to miss any meeting due to clash with another scheduled meeting with another expert. The students who want to meet one set of people are grouped together, e.g. Group 1 wants to meet N3, N5, N8, N9, N12, N18 and N19. The details of all groups are given as follows

Group 1: N3, N5, N8, N9, N12, N18, N19
Group 2: N8, N9, N12, N19, N2
Group 3: N3, N5, N4, N16, N8, N9, N19
Group 4: N8, N9, N12, N15
Group 5: N15, N16, N17, N18, N19, N20
Group 6: N3, N5, N7, N11, N14, N20
Group 7: N3, N5, N12, N2, N18, N19, N20, N1
Group 8: N3, N5, N8, N9, N10, N18, N19, N20
Group 9: N3, N13, N8, N9, N7, N19, N20
Group 10: N1, N8, N9, N13, N20
Group 11: N18, N19, N20
Group 12: N33, N11, N8, N18, N19, N20
Group 13: N3, N8, N10, N12, N4, N20
Group 14: N3, N5, N11, N9, N10, N17, N19, N20
Group 15: N2, N8, N12, N18, N19, N20

The persons N_i $1 \le i \le 20$ have their choices of possible slots for meetings based on their availability.

N1: 2,5,7
N2: 1,4,6,2
N3: 2, 5, 6,1
N4: 2, 4, 6, 8
N5: 2, 6, 5
N6: 1, 5, 3
N7: 2, 4, 6,1, 8
N8: 1, 3, 4
N9: 4, 1, 5, 8, 6
N10: 8
N11: 2, 3
N12: 1, 2, 3, 4, 7
N13: 7, 1, 8
N14: 5, 3, 6, 1

N15: 2, 5
N16: 2, 5, 1, 4
N17: 1, 4, 5, 6
N18: 5, 4
N19: 1, 3, 6, 8
N20: 6

Schedule meeting of all students with the persons they want to meet under the constrained availability of slots. Draw the constraint graph by hand on a paper neatly which you will be asked to upload during submission. Ensure the scalability of your solution to problems with m, the number of persons, with values of m ranging from 6 to 30. 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 3.7 (Windows 10), Turtle, PyQT and matplotlob 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. **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 m×2 matrix holding the values in its cells as shown in graphics section. 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 N1 to Nm.
2. **DFS _BT_ Constraint_Propagation (CSP, variable_list, constraint_graph) returns assignment**: The implementation uses constraint propagation using AC-3 algorithm 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 N1 to Nm.

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 (AC3) 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

The driver must integrate all functionalities and execute the functions appropriately using these options. Create an appropriate interface for user to input the details of slots if it is needed for changing the data, else show on the screen the previously fed data (as given in this document). Prompt for choice of algorithm between DFS + BT with and without constraint propagation. Also give a way for the user to select an appropriate heuristic such as degree or MRV. Create user interface to show the partial assignments after every 100 nodes generated. Mention on the screen your variable ordering technique and show the variables in the selected sequence. For example, following sequence of variables displays partial assignment as 1, 2, 5, 7 for the slots for N2, N6, N14 and N13. The partial assignment must not violate the constraints.

N2	**N6**	**N14**	**N13**	N1	N10	N4	N12	N3	N20	N5	N9	N7	N8	N15	N11	N16	N18	N17	N19
2	**3**	**5**	**7**																

Also, display the pre-computed values R1-R8 on the screen appropriately. The appropriate display of the work is student's responsibility to showcase the work in the best possible way.

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 1, 2019 (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. Inform me any discrepancy in the document immediately.

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

Vandana

October 22, 2019