Problem Set 1 Problem 3

Anonymous

February 2021

0.1 A Brief Description

In this project, we were tasked with the implementation of a program which searches for a solution to 3 hierarchically organized constraint satisfaction problems. The way I approached this problem was with a combination of brute-force and a detailed mathematical analysis of the constraints.

0.2 Pseudo Code

```
Problem A
def problem_a_solver():
    for E in [E_lower, E_upper]:
        for F in [F_lower, F_upper]:
            if mathematical analysis of C3 for A is valid:
                set A to analysis result
                for B in [B_lower, B_upper]:
                    set C to analysis result
                    set D to analysis result
                    returns A, B, C, ..., nva_counter
    returns no result
```

```
Problem B
def problem_b_solver():
    ...
        // [took out return statement from part a]
        // part b starts right after
        if mathematical analysis of C6 for A is valid:
            set G to analysis result of C6
            set J to analysis result of C7
            for H in [H_lower, H_upper]:
                if mathematical analysis for C5 for I is valid:
                    set I to analysis result of C5
                    if remaining unused constraints are valid:
                        return A, B, C, ..., nva_counter
    returns no result
```

```
Problem C
def problem_c_solver():
    ...
        // [took out return statement from part b]
        // part c starts right after
        if mathematical analysis for C13 for O is valid:
            set O to analysis result of C13
            if mathematical analysis for C10 and C14 are valid:
                set K to analysis result of C10 and C14
                set M to analysis result of C10 and C14
                if mathematical analysis for C12 is valid:
                    set N to analysis result of C12
                    for L in [L_lower, L_upper]:
                        if L satisfies C15:
                            return A, B, C, ..., nva
    return no result
```

0.3 Pseudo Code Explained

The pseudo code above is split into three pieces of code, each with the respective problem in which they solve. Part b of the code uses the entirety of part a (with the exception of the return statements) and continues within the inner most loop. Part c does the same for part b.
The code starts off with a loop for the variable E, looping through from the lower to the upper bound of the variable E (which was given as a constraint, or mathematically analysed). This loop is used to determine the potential value for variable E. The very same concept was applied to the variable F loop in which comes right after it. Within the F loop, there is a if statement in which a certain constraint, this case the C3 constraint, was verified to fit with the variables that we currently have. A documentation of the mathematical analysis applied to these constraints can be found further down in this report. If the constraint is satisfied, we continue and set the variables based on the mathematical analysis of the constraints. The rest of the program operates on a very similar fashion. When the variables are verified by the final constraint of the specific problem (a, b, c), we return the variables as a solution to the problem. If the function finishes without finding a valid set of variables to satisfy the constraints, the function returns no result.

0.4 Strategies

To simply solve the three constraint satisfaction problems, all we had to do was use a brute force approach: looping through every single variable on each of their domain [1, 50] and checking whether the combination of variables satisfies the constraints. However, this will take far too long.
The strategy I employed to greatly reduce the time and increase the efficiency of solving these constraint satisfaction problems was to first try to reduce the number of constraints to check. Using mathematical analysis, there were many cases in which the constraint could be shaped in a way to give an equation that determines a specific variable using the variables that we have already set. This reduced the number of variable assignment because instead of having to loop through the entire [1, 50] domain, we could mathematically solve for the variable in question.
Mathematical analysis also allowed me to reduce the size of the bounds some variables had. Variables in some constraints overlapped and related to each other, thus one could find the relationship between some variables, which in many cases reduced the domain size. This shrinking of the bounds does not completely take out the need to loop through a variable as did the previous strategy, but it did reduce the number of times a variable had to loop. With the amount of nested loops used, we can see how this will greatly increase the efficiency of solving these constraint satisfaction problems.

0.5 Mathematical Pre-Analysis

```
Constraint 1 [C1]
A = B + C + E + F

B < A
C < A - B
E < A - B - C
F = A - B - C- E
```

```
Constraint 2 [C2]
D = E + F + 21

since we know D is at most 50...
51 > E + F + 21
30 > E + F
since we know E and F are at least 1...
30 > E + 1
29 > E

F < 29 - E
```

```
Constraint 3 [C3]
D**2 = E*E*A + 417

D = sqrt(E*E*A + 417)
```

```
Constraint 4 [C4]
E + F < A

No need for analysis [implied by C1]
```

```
Constraint 5 [C5]
H*J + E*12 = (G+I)**2

sqrt(H*J + 12*E) = G + I
I = sqrt(H*J + 12*E) - G
```

```
Constraint 6 [C6]
A + D = (F - G)**2 - 1

A + D + 1 = (F - G)**2
sqrt(A + D + 1) = F - G
G = F - sqrt(A + D + 1)
```

```
Constraint 7 [C7]
4*J = G**2 + 39

J = (G**2 + 39) / 4
```

```
Constraint 8 [C8]
(I - G)**9 = (F - H)**3

no mathematical analysis done...
```

```
Constraint 9 [C9]
(G - C)**2 = F*C*C + 1

no mathematical analysis done...
```

```
 Constraint 10 [C10]
 2*M = K**2 - 6

M = (K**2 - 6) / 2
```

```
 Constraint 11 [C11]
 (N - O)**3 + 7 = (F - I)*N
```

```
 Constraint 12 [C12]
N**2 = M**2 + 291

N = sqrt(M**2 + 291)
```

```
 Constraint 13 [C13]
O**2 = G*H*I*B + 133

O = sqrt(G*H*I*B + 133)
```

```
 Constraint 14 [C14]
M + O = K**2 - 10

M = K**2 - 10 - O
 using C10...
K**2 - 10 - O = (K**2 - 6) / 2
 2*K**2 - 20 - 2*O = K**2 - 6
K**2 = 14 + 2*O
K = sqrt(14 + 2*O)
```

```
 Constraint 15 [C15]
L**3 + I = (L + B)*K

no mathematical analysis done...
```

0.6 Concluding Remarks

There are many ways to solve these three constraint satisfaction problems, and the way I went about it was just one of many. My solution resulted in a final number of variable assignment for problem c to be 407 [problem a nva: 92, problem b nva: 400], with no noticeable time lag for the computation of the solution.

0.7 Appendix

To run the program, simply download the file and run as a normal python file either through an IDE or a command terminal by entering the directory in which the file is saved and then typing p3.py