Phuong Dinh

Solving Discrete Constraint Satisfaction Problems Report

* Through multiple trials and errors to optimize problems A, B, and C within the given constraints, I have come up with a solution that utilizes a combination of nested for loops and mathematical manipulation in order to achieve a solution within a reasonable timeframe. My overall strategy is iterating through at least variable as possible to decrease variable assignment. The first method is to calculate some variables based on other parameters instead of assigning a value. The second method is creating variable limits based on mathematical calculations before coding. The third method is to do the constraint check last in every problem in order to minimize redundancy in validation. The fourth method is to eliminate repetitive constraints.
* Pseudo code:

Csp(dict):

For b from 1 to 101:

Add 1 to counter

For c from 1 to 101:

Add 1 to counter

For e from 1 to 9:

Add 1 to counter

For f from 1 to 79 – e

Add 1 to counter

If(constraint is met):

Calculate d

Calculate a

Add 2 to counter

Assign a, b, c, d, e, f to the dictionary

Assign 0 to g, h, I, j, k, l, m, n, o

return

Csp2(dict):

Calculate h with problem A result

Add 1 to counter

For g from 1 to 21:

Add 1 to counter

For I from 1 to 101:

Add 1 to counter

If(constraint is met):

Calculate j

Add 1 to counter

Assign g, h, I, j to the dictionary

Assign 0 to k, l, m, n, o

Csp3(dict):

For k from 1 to 101:

Calculate m

Calculate n

Calculate l

Calculate o

Add 5 to counter

If(constraint is met):

Assign k, l, m, n, o, to the dictionary

Csp\_total(decision):

Create dictionary dict to hold all variables

If decision is a:

Call Csp(dict)

Else if decision is b:

Call Csp(dict)

Call Csp2(dict)

Else if decision is c:

Call Csp(dict)

Call Csp2(dict)

Call Csp3(dict)

Else:

Print: Please enter a valid option

For each value in the dictionary:

If the value is not equal to -1:

There is a solution

Else there is a value that is equal to 0:

There is no solution

If there is a solution:

Print solution

Else there is no solution:

Print no solution

Export dict to csv

Main():

While True:

Get input from user (a, b, or c)

Csp\_total(input)

* I separated the three problems A, B, and C into three different functions. Csp is for A, csp2 is for B, and csp3 is for C. Inside csp, I iterate through b, c, e, and f within the given range. After which, I check to see if condition C3 is met using the four variables. If it does, it means that the four variables are correct. After which, I calculated a and d with b, c, e, and f and add up the nva counter. Finally, I assigned a, b, c, d, e, f to the dictionary. Taking the result from problem A, I start csp2 by calculating h since h is dependent on all variables from A. Then, I iterate through g and I until condition C5 is met. After C5, I am certain that I have the correct solution for problem B. Thus, I calculate j, add up the counter, and assign g, h, l, j to the dictionary.
* In csp3, I iterate through k and then calculate the value of m, n, l, and o since all the required parameters are given. Lastly, I check to see if the constraints C0 and C13 are met. If yes, it means I have found the answer for problem C. Then, I add in the counter from all previous problems and assign l, m, n, and o to the dictionary. The function csp\_total is to call the previous 3 functions in one place and output whether or not there is a solution based on if all the variables inside the dictionary is assigned with a value that is different than -1. Everything is under a while true loop in main for the code to be ran continuously.
* The first mathematical pre-analysis is not iterating through variables that can be calculated. For example, a = b + c + e + f; thus, if I know b, c, e, and f, I can directly get the value of a. I used this method to get a, d, h, j, m, n, l, and o. The second mathematical method I used is limiting the range of variables based on the constraints. For instance, using C3, we know that the maximum value of D is 100 and the lowest is 1; thus 1 <= D2 <= 10000. Then we have 1 <= E2 \* A + 694 <= 10000. Since the min value for all variables is 1, we only consider the max value in this equation. Thus, E2 <= (10000 – 694)/100, and E <= 9.6. This means that the range of e can only be from 1 to 9. Moreover, since D = E + F + 21, assuming D is at maximum value of 100, F = 100 – 21 – E and is equal to 79 – E. I used the same concept to calculate the maximum value for G is 19.3 => G max is 19. Lastly, I took out unnecessary constraint such as C4 since it doesn’t give any new information (we have already established that a = b + c + e + f and all are positive values => a will always be larger than e + f) in order to cut down on processing time.
* My program does utilize hierarchical structure by solving problem A, using the result from A to solve problem B, and use both problem A and B to solve problem C. Thus, I add the nva counter throughout each problem.
* Unfortunately, the method I used for this problem is dependent on the pre-analysis of the math constraints. Thus, it is not ‘generic’ enough to be used with other mathematical constraint problems.