10/11/2017

Since I’m going to change the container for variables, should I make a separate class? For now, no, as I could not read the variables with a for properly. Actually, I will make a separate class and make the container public, since the for iteration is the same for sets and arrays.

!!! ArrayList and Set both use add and remove for access, so I actually don’t need to implement a separate container class.

I’m just going to use the Collection interface.

I’ll put all the read/write functionality in separate static functions so I can reuse the code if needed.

**QUESTION:** There is a serious question of whether to let constraints which refer to values that cannot be taken for the current assignment be or remove them for now and add them back on backtracking. !!! (For Variable Ordering based on Least/Most Constrained and Value Selection, it matters whether I have useless constraints or not for time consumption)

**QUESTION:** I think I’d rather model the Constraint in code as a class which holds a pair of values (var1, var2) and a pair of values they cannot take (val1, val2). This way I can index the constraints for a variable var1 based on the value val1 corresponding to it. At the moment I am maintaining constraints as (var1, var2) and an array of the pairs of values they cannot take. Other than the memory consumption increase, is there anything else you think I may need to take into account?

For faster constraint access based on value (which we will need for value selection), I have stored the values in a map indexed by the variable value they concern.

I will progressively remove values from domains such that assigning a value for a variable does never violate a constraint for current assignment.

13/11/2017

Due to a more efficient implementation of ConsistentAssignmentValueSelection, values in domains must now be nonnegative. (N/A actually)

Input format does not accept more than 1 constraint targeted at the same variables. Will be solved in input parsing.

23/11/2017

FileHandling does input into argument instead of return because, when reading the problem, we must read both variables and constraints, so we have 2 arraylists. It would be weird to output one as argument and the other one as return. (When reading just the assignment we leave it as argument to be consistent with the previous case)

At the moment, when generating CSP problems, the order of the variables is locked in the file because they are not named (they must be in order of index 0,1,2,3,…). I do have the facility of providing names for them as indexes (so they can have IDs). Should I integrate it? I don’t see any reason why not, although the project is working like this as well.

FileHandling also handles reading the NQueens problem solution into memory, but at that point we need to use int[][] as a return value, because we don’t know the size of the matrix before reading the solution. Passing a int[][] as a matrix and reassigning it inside the function does not reassign it outside the function.

Laptop died on this date due to rain in backpack. Did backup data on github in the morning, so loss was only one days worth. Recovered laptop on 25, with no loss in the end.

11/12/2017

There are multiple ways to generate Maps such as the DrunkWalk or the ForestGrow. At the moment, I have decided to use ForestGrow. The input is the number of zones desired (name it n), as well as an optional size of starting zone (name it w). The algorithm finds the least perfect square greater than n, name it k^2 and creates a map of k\*w lines by k\*w columns, which is split into k^2 zones of w lines \* w columns. Then, the n zones are seeded from n of the k^2 zones and grow outward until there are no empty spaces and their sizes are about k\*k\*w\*w/n.

I’m also modifying the format of the input and the output. From now on, we’ll also add the index of the variable (its “name” basically) when we refer to it (in input: before enumerating number of values and values for a variable; in assignment: before each value we’ll put the index of the variable it refers to).

Assignments also have to specify the number of variables which they are assigning.

12/12/17

There’s a lot of discussion with how the map is generated, ranging from which algorithm we’re using (RandomWalk, ForestGrow), to how the algorithm is implemented (when should a Forest stop growing? Is it allowed to “eat” from other forests?)

21/12/17

The FileHandler class is getting a bit too big as it has the file handling for the CSPs, the MapColoring and the NQueens. At the moment, I have the following packages: Auxiliary, LookBack, MapColoring, nQueens, Solver, Validator, ValueSelection, VariableOrdering. I intend to reorganize them to balance the weight of the code better.