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.

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