CS 421/820
Assignment 3
Total: 100pts
Dr. Malek Mouhoub

Write a program that generates a random consistent binary CSP and solves it using the techniques listed in Section 2.

1 Binary CSP Instances

Binary CSP instances should be randomly generated using the model RB proposed in [1]. The choice of this model is motivated by the fact that it has exact phase transition and the ability to generate asymptotically hard instances. More precisely, you need to randomly generate each CSP instance as follows using the parameters n, p, α and r where n is the number of variables, $p\,(0 < p < 1)$ is the constraint tightness, and r and $\alpha\,(0 < r, \alpha < 1)$ are two positive constants used by the model RB [1].

1. Select with repetition $rn \ln n$ random constraints. Each random constraint is formed by selecting without repetition 2 of n variables.
2. For each constraint we uniformly select without repetition pd^2 incompatible pairs of values, where $d = n^\alpha$ is the domain size of each variable.
3. All the variables have the same domain corresponding to the first d natural numbers $(0 \ldots d-1)$.

According to [1], the phase transition pt is calculated as follows: $pt = 1 - e^{-\alpha/r}$. Solvable problems are therefore generated with $p < pt$.

2 Solving Techniques

The following backtrack search strategies should be implemented.

BT Standard Backtracking.

FC Forward Checking.

FLA Full Look Ahead (also called MAC).

In addition, the user should be given the option to run Arc Consistency (AC) before the actual backtrack search.

3 Input and Output

3.1 User Input

- n, p, α and r (to generate the CSP instance).
- The chosen solving strategy (BT, FC or FLA) with or without AC before the search.

3.2 User Output

- The CSP instance.
- The solution to the CSP instance.
- The time needed to solve the instance.

4 Example of an RB Instance

4.1 Input Parameters

- Number Of Variables (n): 4
- Constraint Tightness (p): 0.33
- Constant α: 0.8
- Constant r: 0.7

4.2 Generated RB Instance

- Domain Size (n^{α}): 3
- Number Of Constraints ($rn \ln n$): 4
- Number of Incompatible Tuples (pd^2): 3

```
Variables : {X0, X1, X2, X3}
Domain : {0, 1, 2}
Constraints: Incompatible Tuples
(X2,X3): 1,2  2,2  2,0
(X1,X2): 1,0  2,0  2,1
(X3,X1): 2,2  0,2  2,0
(X0,X2): 2,2  2,1  1,2
```

5 Marking scheme

1. Readability : 10pts
2. Compiling and execution process : 10pts
3. Correctness : 80pts

6 Hand in

6.1 Single file submission

Using URCourses, submit the file containing the C++ (or others) code of the programming part **assign3.cpp**. At the top of this file add the following comments :

1. Your first name, last name and ID #,
2. the compiling command you have used,
3. an example on how to execute the program,
4. and other comments describing your program.

6.2 Multiple files submission in C++

Submit all files required by the programming part:

1. README (file including your name and ID #; and explaining the compilation and execution of your program including the format of input and other details)
2. headers (.h)
3. implementations (.cpp)
4. the Makefile :
    - should be named "**makefile**". In the makefile, the generated executable should be named : "**assign3**"

You can give any name to your source files. The marker will only have to run "**make**" to compile your program and "**assign3**" to execute it.

References

[1] K. Xu and W. Li. Exact Phase Transitions in Random Constraint Satisfaction Problems. *Journal of Artificial Intelligence Research*, 12:93–103, 2000.