Constraint Satisfaction Problems

B. Nebel, C. Becker-Asano, S. Wölfl
Wintersemester 2014/15

University of Freiburg
Department of Computer Science

Project: Part 4

Due: 6.2.2015

The Competition We will evaluate your solvers in a small competition. Working on this project sheet is not mandatory; there are no points. But the authors of the top three ranked solvers will get small prizes.
If you want to participate, please hand in your code by Thu, Feb 5, in your git repository in a new branch `competition`. We will start with first tests on Fri, Feb 6. The last chance to update your code is on Fri, Feb 6, 23:59. The results will be presented on Wed, Feb 11.

What You Should Prepare You already have implemented a selection of different algorithms and heuristics. Of course, you may implement further algorithms (e.g. some backjumping technique). Thus, you have to decide on a reasonable default selection of constraint solving methods. To this end, implement an `--auto` command line switch that sets the heuristics and algorithms as you want (and maybe tailored to the input instance); your selection of preprocessing, maintaining arc consistency/backtracking algorithm, heuristics, etc. We will invoke your solver with:

```
python3 solver.py --auto instance.xml
```

Further, we require you to print the found solution in case of satisfiable problem instances. The output format for satisfiable problems must be:

```
SAT
+Solution: variable_name=value, variable_name=value, ...
+Info: ...
```

The last line should be a single info line (less than 80 chars) that provides information about the methods used by your solver (if you want to log more information use the Python logging module). For example, for $V = (var1, var2, var3)$ and solution $(1, 2, 3)$ the output could look as follows:

```
SAT
+Solution: var1=1, var2=2, var3=3
+Info: preproc=gac3, hvar=lexico, search=gaschnig
```

For unsatisfiable problems the output must have the form:

```
UNSAT
+Solution: None
+Info: ...
```

Further, make sure that nothing else is printed to the output.

Competition Rules The evaluation will be done on around 60 instances *without* giving you access to these (blind evaluation). We will use a time-out of 5 minutes per solver and instance. For each correctly solved instance, the solver will get *points*. Each solver accumulates the points over the all instances. Solvers will be ranked according to the number of points they have accumulated.
To calculate the number of points for a solver that successfully solves an instances we will use a *purse-based method*:

- "solved purse": there are 100 points distributed *equally* among all solvers that correctly solve the instance.
- "speed purse": there are 100 points for each problem which are distributed relative to the speed (i.e., runtime) of each solver that correctly solves this instance:

$$F_{solver} := \frac{1}{1 + time(solver)}$$

$$points(solver) := 100 \cdot \frac{F_{solver}}{\sum_{s \in solvers} F_s}$$

Note, the "solved purse" gives points for solving instances, and rewards success on hard problems. The "speed purse" rewards fast solvers.

Example 1. *If 4 out of 6 solvers correctly finish on instance 1 with running times* 20.5, 30.4, 80.8,140.0 *seconds:*

- *each of them will be given* $\frac{100}{4} = 25$ *points for the solution.*
- *the "speed-purse" will give* 47.6 *points to the first,* 32.6 *to the second,* 12.5 *to the third, and* 7.3 *to the forth.*

For this instance we have the following points (and preliminary ranking):

1.	*solver-1*	72.6
2.	*solver-2*	57.6
3.	*solver-3*	37.5
4.	*solver-4*	32.3