

CSC2512 Constraint Satisfaction Problems

Assignment 1

Handed out Feb 14th. Due March 12th.

1 Social Golfer Problem

The social golfer problem $\langle w, g, s \rangle$ is to schedule a golf tournament for $g \times s$ golfers that runs over w weeks. In each week the golfers must be divided into g groups of size s . Every golfer must play only once in every week, and every pair of golfers can meet at most once over the course of the tournament.

Normally, we would be interested in knowing the schedule, but to make things easier in the assignment we will only be concerned with whether or not a schedule exists for various settings of $\langle w, g, s \rangle$.

The aim of the assignment is to use this problem to compare CSP and SAT solving from various points of view (e.g., ease of modeling, efficiency of solving) In particular, you will have to formalize the problem as a CSP problem, and as a SAT problem. Then you will use existing CSP and SAT solvers to solve each formulation, and compare the CSP and SAT approaches to this problem.

In more detail your tasks are as follows:

1. Decide on an underlying SAT solver that you would like to use (you can write your own, but given the timeframe I don't recommend this unless you are a very efficient programmer). There are a couple of reasonable possibilities:

Solver	Download Location
zchaff	http://www.princeton.edu/~chaff/zchaff.html
MiniSat	http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat/

In my opinion MiniSat is by far the superior peice of software. There might also be a few other SAT solvers available for public use.

Almost all SAT solvers take as input DIMACS formated SAT problems. The format is very simple:

- Lines starting with ‘c’ are comment lines discarded by the solver.
- The first non-comment line starts with the string “p cnf” followed by two numbers. The first being the number of variables, the second being the number of clauses.
- A sequence of lines specifying the clauses of the problem. Each clause is specified by a sequence of numbers with i being the i -th variable and $-i$ being the negation of the i -th variable. Variable numbers range from 1 up. Each clause ends 0.

For example a CNF file with the following contents

```
c
c start with comments
c
c
p cnf 5 3
1 -5 4 0
-1 5 3 4 0
-3 -4 0
```

Specifies the CNF formula $\{(1, -5, 4), (-1, 5, 3, 4), (-3, -4)\}$ that has 3 clauses and 5 variables.

2. Decide on an underlying CSP solver you would like to use (the same comment as above applies if you are thinking of writing your own CSP solver). Here the choice is more complex, but some reasonable options include

Solver	Download Location
Minion	http://minion.sourceforge.net/
Choco	http://choco.sourceforge.net/
Geocode	http://www.geocode.org/

Each solver has its own way of specifying a CSP problem and then solving it. For example, Minion uses a file to specify the CSP problem utilizing a “matrix model.” Geocode and Choco provide more of a programming interface where calls are made to specify the variables and constraints of the problem. Minion and Choco have “getting started” documentation, while for Geocode the best place to start might be the publication “Views and Iterators for Generic Constraint Implementations” available at <http://www.geocode.org/paper.html?id=SchulteTack:Advances:2006>.

3. Come up with a formulation of the problem as a CSP, you will have to write some code that can take the problem parameters $\langle w, g, s \rangle$ as input and create the appropriate CSP.

4. Come up with a SAT formulation. The easiest method might be to automatically convert your CSP formulation into a SAT formulation. This can be accomplished by realizing that every multi-valued CSP variable V can be converted to a collection of binary-valued SAT variables each of which represent assigning V one of the values in its domain along with clauses to ensure that one and only one of the corresponding SAT “assignment” variables is true in any solution. Then each constraint of the CSP can be converted to a collection of clauses: every set of assignments that falsifies the constraint yields a clause blocking that set of assignment variables.

Alternatively you might want to write some code that inputs $\langle w, g, s \rangle$ and generates a DIMACS formatted CNF that is satisfiable if and only if this problem has a solution. This would allow you to utilize a SAT model different from the CSP model. The downside of this is that a direct comparison of the SAT approach and the CSP approach now becomes more difficult.

5. Perform an empirical comparison of the performance of your SAT and CSP solvers by varying the parameters $\langle w, g, s \rangle$. Make sure you experiment with both satisfiable and unsatisfiable problems.

2 What to Hand In

Write a paper describing your work. You should write this as you would a paper submission. In other words, you need to introduce the problem, specify the details of what you did (e.g., the details of your SAT and CSP formalization), present and analyze the empirical data you generated, and draw some relevant conclusions. Be sure to mention any special details particular to your solution, e.g., if you used or designed a special propagator for your CSP formulation, used a special heuristic for variable branching, or used symmetry breaking constraints.

You must write the paper using the LNCS (Lecture notes in Computer Science) format. See

<http://www.springer.de/comp/lncs/authors.html>

(the easiest way to format your paper in this style is to utilize the Latex style files provided at this web site input files, the site also provides Word templates).

Your paper must be no longer than 12 pages including the bibliography and any appendices (if you have them—typically appendices are not included in short papers). You do not need to hand in any of the code you wrote, but you can discuss features of your implementation if you feel that this is a useful use of space in your paper.