Spin Glass Ground State Problem

Spin glass models are widely used in physics to model the microscale properties of materials and other complex systems. Formally, an "Ising (+/-1) spin glass" consists of n variables that can take a value from the set $\{-1, 1\}$ and are called ("sites") $s_1, \ldots, s_n$. Each pair of sites i, j are connected by an "interaction coefficient" J_{ij}, which may take on values +1, -1, or 0. A given system of interactions J defines an "energy" function on the spin configurations as:

$$E_J(s_1, ..., s_n) = -\sum_{ij} J_{ij} s_i s_j$$

(note the constant factor -1). A "ground state" for a given system of interactions J is a configuration of the spin values achieving a minimum of the energy function E_J. (Intuitively, two positively interacting spins "want" to be in the same state, and two negatively interacting spins "want" to be in opposite states. The goal is to find a configuration of the spins satisfying as many of these "wishes" as possible.)

Assignment

Write a program that encodes the optimization problem instance (given in extended DIMACS format) as a linear program in the `lp_solve` input format (see the *General instructions for Assignment 2*). Note that this means that your program will take as input an instance of the spin-glass groundstate problem, as described above, and produce as output an integer linear program whose optimal solutions correspond to ground states of the given spin glass.

The choice of programming language may be either Java or Python 2. In the assignment package we provide a Java graph library and Python graph class that is able to process the input files in DIMACS format. Further, you find an example programs that use this library / class for encoding a different problem (`IndependentSetLP.java` and `IndependentSetLP.py`, respectively), which you may adapt for your solution. Also there is example Java and Python code available for accessing the spin glass instance from within the library (see `PrintSpinGlass.java` and `PrintSpinGlass.py`, respectively) See the document on general instructions for further information on the input format of the LP solver.

Test your solution with two sets of instances (the optimal energy value is included in the file name for your convenience): **dms_assignment2_spinset1** and **dms_assignment2_spinset2**, which can also be found in thr assignment package. Let `lp_solve` run for 10 minutes for each instance of both sets. For each set, test how many of the instances can be solved with `lp_solve` within the time limit.

Input

The program must be run with the command

```
java SpinGlass <graph>
        OR
python SpinGlass.py <graph>
```

where <graph> is the problem instance graph with each node representing a "site" An edge between two nodes has an attribute "interaction" which can have -1/0/+1 values and represents the interaction between the sites. **Do not use any third-party code or non-standard libraries apart from the code provided in the assignment package.**

Make sure that your program can be called using exactly the syntax above. This is because we are using an automated system to check submissions and a different syntax would cause it to fail.

You can assume that the following versions of Java and Python are installed:

```
java -version
java version "1.7.0_65"
OpenJDK Runtime Environment (IcedTea 2.5.3) (7u71-2.5.3-2~deb7u1)
OpenJDK 64-Bit Server VM (build 24.65-b04, mixed mode)

python --version
Python 2.7.3
```

Output

After running

```
java SpinGlass <graph> | lp_solve
        OR
python SpinGlass.py <graph> | lp_solve
```

the output should show an assignment for the variables n1, n2, ... The variable ni should be assigned the value -1 if site ni has value -1 and 1 if it has value 1. The output can of course have your own extra variables in addition to these.

BE SURE TO USE EXACTLY THESE VARIABLE NAMES FOR THE SPINS. Using different variable names will break our checker and thus lead to resubmission.

Example

Consider the following toyexample with three nodes which is also available in the assignment package under the name `spin.g`.

```
c f 3 3 " " "source dest interaction"
e 1 2 1
e 1 3 -1
e 2 3 -1
```

This spin glass has two ground states of energy -3. The output should include either one of the assignments:

```
n1 = 1;                                 n1 = -1;
n2 = 1;                                 n2 = -1;
n3 = -1;                                n3 = 1;
```

In the assignment package you also find example instances `spinglass17.g`, `spinglass43.g`, and `spinglass48.g` with minimum energy of -17, -43, and -48 respectively.

Deliverables

Your submission must contain two files, both are to be submitted through the Stratum system.

- *Commented* source code; and
- A short (max 3 A4 pages) report in pdf or ASCII text format that outlines your approach, lists your results for the two instance sets and briefly describes the computer (clock speed, memory) used. It is recommended to use the latex template provided in the assignment package.

The deadline for submissions is

April 12 (23:59).

Your code should be tidy and sufficiently commented to allow to easily understand what it does. Easy-to-read code and comments usually help troubleshooting problems in the submitted program.

Grading

A correct solution submitted in time (i.e., before the deadline) with a sufficient report earns 3 points. An additional point can be awarded for exceptionally good submissions (either in terms of an efficient encoding or an excellent report).

If the submitted solution does not work correctly, the source code is unreadable or not sufficiently commented, or the report is unclear or missing, you are asked to correct and resubmit your solution. Each re-submission is due within one week (in case a resubmission is needed, you will receive further instructions and a deadline for the resubmission in Stratum system) and decreases the maximum number of points achievable by 1.

The solutions submitted after the deadline (April 12, 23:59) will not get any points, i.e., maximum number of points for a late submission is 0.

Do remember that the points in the Stratum system do not correspond to the final points you will receive. In the Stratum system the check is two-phase (automated / manual) for both the code and report. The points have the following meaning (see Checker output for more information):

0 points: No submission / submission not yet graded / failed submission

1 points: Submission has passed automated check (code is working) / submission has the right format (pdf/txt for report), manual check not yet done (e.g. style and commenting for code)

2 points: Submission has passed both manual and automated checks.