

## ASSIGNMENT 2: GRAPH SUBSET MAPPING

**Goal:** The goal of this assignment is to take a complex new problem and formulate and solve it as a SAT problem. Formulation as SAT is a valuable skill in AI that will come in handy whenever you are faced with a new problem in NP class. SAT solvers over the years have become quite advanced and are often able to scale to decently sized real-world problems.

**Scenario:** You are an investigative agency working on uncovering a drug mafia. You have got telephone records of various telephone numbers which are believed to be associated with this mafia. You have also got a set of emails related to the mafia. However, you do not know which telephone number corresponds to which email address. The goal is to automatically figure out the mapping between emails and phones if it exists. To solve this problem (for our assignment), you make a few assumptions.

1. Some people are net savvy and use emails. All people know how to use phones. People who use emails regularly use both emails and phones to communicate with each other.
2. If a person X emailed a person Y, he also called Y on phone at some point.
3. Each person has exactly one email address and exactly one phone.

You abstract out the problem by creating two graphs – G<sub>phone</sub> and G<sub>email</sub>. There exists a directed edge between two nodes in G<sub>phone</sub> (or in G<sub>email</sub>) if the first phone number (or email address) called (or emailed) the second. Your goal is find a mapping from emails to phone numbers. G<sub>email</sub> is the smaller graph because fewer people are net-savvy.

**Problem Statement:** There are two directed graphs  $G$  and  $G'$ . Find a one-one mapping  $M$  from nodes in  $G$  to nodes in  $G'$  such that there is an edge from  $v_1$  to  $v_2$  in  $G$  if and only if there is an edge from  $M(v_1)$  to  $M(v_2)$  in  $G'$ . Sample cases are shown here [here](#).

We will use miniSAT, a complete SAT solver for this problem. Your code will read two graphs in the given input format. You will then convert the mapping problem into a CNF SAT formula. Your SAT formula will be the input to miniSAT, which will return with a variable assignment that satisfies the formula (or an answer "no", signifying that the problem is unsatisfiable). You will then take the SAT assignment and convert it into a mapping from nodes of  $G$  to nodes of  $G'$ . You will output this mapping in the given output format.

You are being provided a problem generator that takes inputs of the sizes of  $G$  and  $G'$  and generates random problems with those parameters. Study the scaling of MiniSat with problems of increasing size.

**Input format:**

Nodes are represented by positive integers starting from 1. Each line represents an edge from the first node to the second. Both graphs are presented in the single file, the larger first. The line with "0 0" is the boundary between the two. The input file that represents the last example in the [slide](#) is:

1 2

1 3

1 4

2 4

3 4

0 0

1 2

3 2

**Output format:**

The mapping will map each node of  $G$  into a node id for  $G'$ . The first numbers on each line represent a node as numbered in the smaller graph  $G$ , and the second number represents the node of the larger graph  $G'$  to which it is mapped. The output of the same problem is

1 2

2 4

3 3

If the problem is unsatisfiable output a 0.

**Code**

Your code should compile, if necessary, and run on `attu.cs.washington.edu`. Please supply a `compile.sh` script if compiling is necessary. Also supply two shell scripts `run1.sh`, `run2.sh`:

1. Executing the command `./run1.sh test` will take as input a file named `test.graphs` and produce a file `test.satinput` – the input file for `minisat`. You can assume that `test.graphs` exists in the present working directory.

2. Executing the command `./run2.sh test` will use the generated `test.satoutput`, `test.graphs` (and any other temporary files produced by `run1.sh`) and produce a file `test.mapping` – the mapping in the output format described above. You can assume that `test.graphs`, `test.satoutput` (and other temp files) exist in the present working directory.
3. The TA will execute your scripts as follows:  
`./run1.sh test`  
`./minisat test.satinput test.satoutput`  
`./run2.sh test`

When we call `./run1.sh test`, you can assume that `test.graphs` exists in the present working directory. When we call `./run2.sh test`, you can assume that `test.graphs`, `test.satinput` and `test.satoutput` exist in the present working directory, along with any other temporary files created by `./run1.sh test`.

Please note that you are NOT allowed to call `minisat` within `run1.sh` or `run2.sh`. The TA will call `minisat` and the `minisat` process will be killed after the given problem cutoff time, so that the cutoff time is only relevant for the time `minisat` takes to process your output, making the testing independent of time requirements for I/O.

### Useful resources

1. <http://minisat.se/MiniSat.html>: The MiniSat page
2. <http://www.dwheeler.com/essays/minisat-user-guide.html>: MiniSat user guide

### What is being provided?

A problem generator for outputting  $G$  and  $G'$  where  $G$  is a subset of  $G'$  and therefore a mapping between the two exists. A check function to test whether your output is accurate or not, i.e., the mapping is indeed an accurate graph mapping.

### What to submit?

1. Submit a zipped folder (not tar.gz) containing your code and writeup. Make sure that when we run `unzip yourfile.zip` on attu the following files are produced in the present working directory:  
`compile.sh`  
`run1.sh`  
`run2.sh`  
`Writeup.pdf`

You will be penalized for any submissions that do not conform to this requirement.

We will run your code on a few sample problems and verify the ability of your code to find solutions within a cutoff limit (as mentioned earlier, the cutoff time only measures the time required for minisat to run). The cutoff limits will be problem dependent and your translation does not need to depend on the cutoff limit, therefore it is not part of the input format. Of course, better translations will scale better.

2. Your writeup should contain the following:
  - a. Your algorithm for translation into SAT.
  - b. Some experimental results reporting the scalability of your code.
  - c. Any analysis of what worked and what worked better.

### **Evaluation Criteria**

1. [3 points] Soundness of your code, i.e., its ability to output accurate mappings for small problems.
2. [4 points] Scalability of your code, i.e., its ability to solve harder problems within cutoff times.
3. [3 points] Your writeup. It will be assessed for its clarity and analysis of your algorithm.
4. [2.5 points] Discretionary points awarded to top performers, highest quality write-ups, and anything else beyond the normal.

### **What is allowed? What is not?**

1. You may work in teams of two or by yourself. We do not expect a different quality of assignment for 2 people teams. At the same time, please spare us the details in case your team cannot function smoothly. Our recommendation: this assignment may be a little hard for students with limited prior exposure to logic; work in teams if you can find a workable partner.
2. You can use any programming language for translation into and out of miniSAT.
3. You must not discuss this assignment with anyone outside the class. Make sure you mention the names in case you discuss with anyone in the class outside your team.
4. Please do not search the Web for solutions to the problem.
5. Your code will be automatically evaluated against another set of benchmark problems. There will be negative penalty if your output is not automatically parsable and the TA has to go back and forth with your team to get your code/output working.