Cpt S 515 Homework #6

No late homework!

1. Google uses a Markov chain to mimic a person browsing webpages. Actyually, markov chains have many other applications as well. However, a difficult part is, at first place, a markov chain is hard to obtain for a practical application. Suppose that I treat a coffee maker as a markov chain (to simulate how one would use the machine). Notice that a modern coffee maker is quite complex and full of buttons to control almost everything. Suppose now that I think the coffee maker as a progam and I want to test it. Can you give me a "random" test sequence (of buttons)? This i a difficult program. Open your mind.

2. Show that the following problem is NP-complete:
   Given: a Boolean formula F
   Question: Does F have even number of satisfying assignments?

3. Show that #3SAT is #P-complete.

4. (easy) Consider a 3SAT Boolean formula F which has variables $x_1, \cdots, x_n$. Recall that the F is a conjunction of clauses, and each clause is a disjunctions of three literals. A Boolean variable a shows up in a clause if either a or $\bar{a}$ is in the clause. A set C of variables is a *cover* of F if for each clause in F, there is a variable in C that shows up in the clause. Let k be the size of a cover C of a given F. Prove that the problem of 3SAT is fixed parameter tractable wrt k.

Assignment 6
Sheryl Mathew (11627236)
December 2, 2018

1. Question 1

We approach the problem in two ways. Assuming that it is possible to obtain a "typical" behavior of a coffee maker in terms of sequences of buttons on it, we can obtain a Markov chain and generate random words from there.

Alternatively, it might be the case that we don't have enough information to obtain a "typical" specification or such an specification is partial, incomplete or missing or the coffee maker is too complex to obtain a "typical" behavior.

I provide approaches for both assumptions, starting with assumption of obtaining Markov chain first. Both approaches are ones specified by Cui and Dang (2017).

In the first case, the sequence of buttons on the coffee maker can be modeled as a DFA.

Using the following method, we can generate a "random" test sequence:

1. Model the minimal and strongly connected DFA from the possible sequence of operation of buttons. Call this M'.
2. Convert M' to a weighted graph G_m which is also strongly connected.
3. Obtain transition probabilities on the weighted graph G_m by the following approach:
    a. Build a weight matrix by populating it with the weights on the edges from the adjacency matrix of graph G_m.
    b. Conduct eigen decomposition on the matrix W.
    c. Obtain the right eigenvector of the matrix W and find the Perron number (the largest eigenvalue).
    d. Obtain the transition probability matrix as in the algorithm shown in Dang (2017).

4. Using the transition probabilities in the transition probability matrix obtain a probabilistic DFA $\widehat{M}$ where each transition in the DFA is assigned a probability. This can be considered a Markov chain albeit possibly an imperfect one.

5. Test sequences can be obtained by running $\widehat{M}$ as a probabilistic program.
6. Run the following algorithm on $\widehat{M}$ to obtain "random" words or "random" test cases:

    a. For every state in $\widehat{M}$, generate the cumulative transition probability matrix.
    b. Use a random number generator to generate a random number between 0 and 1 (including 1).
    c. For each edge (transition between states):
        i. Depending on values of current cumulative transition probability, the next transition cumulative probability and the random number, output the symbol on the edge or not.
    d. End at accepting state.

7. Once "random" sequences are generated, the words generated asymptotically with probability 1 are taken to be "typical" test sequences (since a finite automaton is a probabilistic program to generate words in a regular language). Whether a walk is typical depends upon the free energy rate of the graph and is calculated as follows:

    a. Take the Markov chain constructed from minimal DFA with given transition probabilities.
    b. Treat the Markov chain $\widehat{M}$ as a probabilistic program and start walk from initial node.
    c. When program hits a specified length N, then if the walk till then satisfies the formula for typicality (depending on free energy rate of walk and free energy rate of graph) given in Dang (2017), then return the word as a typical word. Else no.

In case, it is too complex or not possible to obtain a specification for "normal" behavior of a coffee maker, or we may only have a partial, incomplete specification of behavior using the buttons on a coffee maker, we can still generate typical/atypical test sequences using the approach by Cui and Dang (2017).

In this case we have no DFA or regular language to model the coffee maker, we can still obtain "random" test sequence as follows:

1. Consider that we have a finite number of sets of all weighted activity sequences of the coffee maker.
2. Consider one specific set of weighted activity sequences. For each sequence within the set, let us assign a random but unique number that falls in in the range of 1 through the size of the set.
3. Concatenate all the weighted sequences in a given set to a single long sequence.
4. For each long sequence, we calculate the probability of occurrence of each symbol by counting the occurrences.
5. Run Lempel-Ziv algorithm on each of the long sequences to obtain the reciprocal of the compression ratio, which is a measure of entropy.

6. Calculate free energy rate λ_i for each of particular set of weighted sequences and calculate total free energy rate λ.
7. For given ε, if $|\lambda - \lambda_i| < \varepsilon$, then return that the particular set S_i of weighted activity sequences is typical, else report abnormal.

2. Question 3

We must show that #3SAT is #P-Complete. We can show this by applying Cook-Levin reduction on any #P-problem.

- Using the same R that defines 3SAT (and input length 'n'), first we construct a circuit C such that $(x, y) \in R \text{ } iff \text{ } C(x, y) = 1$. This maintains the number of solutions because the same 'y' that satisfies the relation R will make the circuit value 1.

- Convert C into a boolean formula $\phi(x, y, z)$ such that if $C(x, y) = 1$ then we know there exists a unique z such that $\phi(x, y, z) = 1$ (the number of satisfying assignments for ϕ is equal to the number of inputs for which C outputs 1). Thus x and y are again the same and z is the variables whose satisfying values are uniquely determined by x and y.

    - If C has inputs $(x_1, x_2 ... x_n)$, gates 1....m and ϕ has inputs $(x_1, x_2 ... x_n)$ and $(g_1 ... g_n)$ with g_i being the gate output.
    - Thus each gate has two input variables and one output variable. Therefore each gate can be be output of 4 possible inputs and thus each of it can be simulated using 4 boolean clauses.
    - Thus circuit C has been converted to a boolean formula with n+m variables and 4m clauses. Thus we maintain number of solutions from #CIRCUITSAT to #3SAT.
    -

- Since we have shown #3SAT $\leq$#CIRCUITSAT, #3SAT is #P-Complete.

3. Question 4

We are given a $3SAT$ Boolean Formula 'F' which has variables $(x_1, x_2 ... x_n)$. F is a conjunction of clauses and each clause is a disjunction of three literals. For example it could be $(x_1 \vee \overline{x_3} \vee x_6) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_6})$

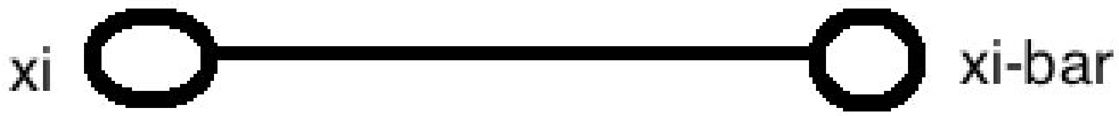
Let U be the set of all variables in F such that = $\{x_1, x_2 ... x_n\}$

We have to show that the 3SAT problem in this case which is whether there are values of $x_1, x_2 ... x_n$ that cause the given 3SAT Boolean formula 'F' to be TRUE.

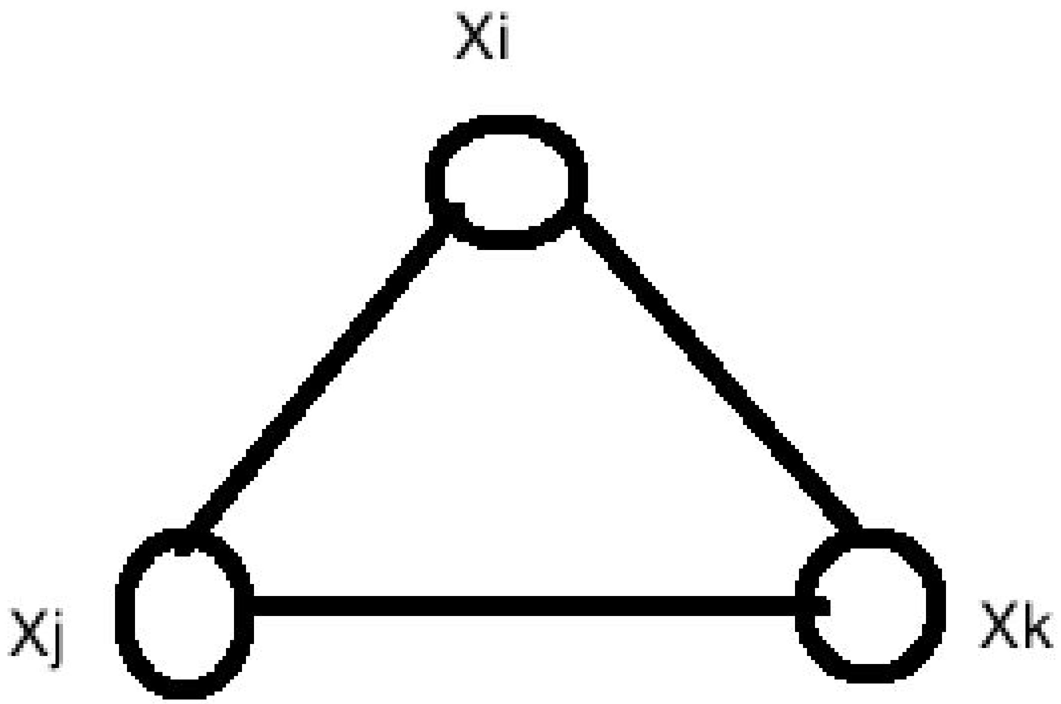
The 3SAT problem can be reduced to the VERTEX COVER problem. That is, the 3SAT problem is equivalent to the problem that given an arbitrary graph G, can we find a subset (cover) of 'k' nodes in the graph such that every edge in the graph touches one of the nodes in this subset.

The reduction can be done in the following way:

1. Every variable x_i in F can be replaced by 2 nodes in a graph.

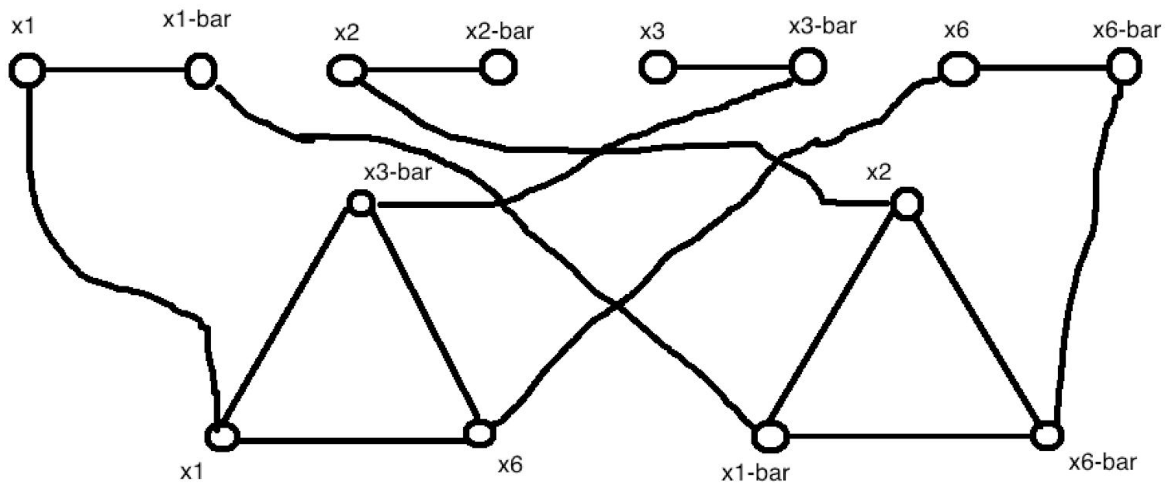


2. Each clause in F can be replaced by a clique of 3 nodes in the form of a triangle.



The above clique would represent $(x_i \vee x_j \vee x_k)$.

For example for F1= $(x_1 \vee \overline{x_3} \vee x_6) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_6})$ we can reduce it to a graph G as shown below:



If a boolean formula F has ‘n’ variables and ‘c’ clauses, the graph G would have 2n+3c vertices.

Such a graph reduced from a boolean formula ‘F’ would have a vertex cover of size n+2c.

1. We see from inspection that every satisfying TRUE assignment gives a vertex cover and every vertex cover gives a satisfying TRUE assignment.

2. We see that vertex cover (the subset of ‘k’ nodes in the graph such that every edge in the graph touches one of the nodes in this subset) is equivalent as the definition of cover in the problem (for each clause, there is a variable belonging to the cover (of size k) that shows up in clause).

3. Thus we have shown 3SAT problem in this case can be reduced to a Vertex Cover problem with the same vertex cover size ‘k’.

4. We know that the vertex cover problem wrt fixed parameter k is tractable as shown in lecture.

5. Thus the 3SAT is fixed parameter tractable wrt k since 3SAT $\leq$ Vertex cover.

<u>References:</u>

“Random Words in a (Weighted) Regular Language: a Free Energy Approach” by Cewei Cui and Zhe Dang, 2017.