CS 4244 Project – Stage 2

Due on the last day of classes; i.e. April 15, 2021

1 Objective

The objective of this project is to demonstrate learning of the material in CS 4244. To this end, you have an option to choose among following four projects. I have marked every project with my ratings of difficulty but remember, your mileage may vary.

Programming Language

You are free to choose your favorite programming language.

Honor Code

You are allowed to consult internet but the analysis (theoretical or empirical) as well as the code must be your own. In other words, you are not allowed to copy a code or paraphrase text on the internet in your report.

2 Behavior of Random CNF Formulas: (Difficulty: Easy)

Let $X = \{X_1, \cdots, X_n\}$ be a set of propositional variables and let F be a formula defined over X. A *satisfying assignment* or *witness* of F is an assignment of truth values to the variables in X such that F evaluates to true. Let $\#F$ denote the number of satisfying assignments of F. We say that F is *satisfiable* (or *sat.*) if $\#F > 0$ and that F is *unsatisfiable* (or *unsat.*) if $\#F = 0$.

A k-clause is the disjunction of k literals out of $\{X_1, \cdots, X_n\}$, with each variable possibly negated. For fixed positive integers k and n and a nonnegative real number r, let the random variable $F_k(n, rn)$ denote the formula consisting of the conjunction of $\lceil rn \rceil$ k-clauses, with each clause chosen uniformly and independently from all $\binom{n}{k}2^k$ possible k-clauses over n variables.

The goal of this project is to conduct an empirical study followed by an analysis of the behavior of the satisfiability of random variable $F_k(n, rn)$. You can fix value of $n = 150$. You should experiment with $k \in [3, 5]$ and $r \in (0, 10)$ with different values of r at the intervals of 0.2. You can use any state of the art SAT solvers available. (For starters, you can use CryptoMiniSAT). Once you fix k and r, you should sample $F_k(n, rn)$, i.e., generate several formulas and check if the generated formulas are satisfiable or not. For every fixed k and r, generate 50 formulas and calculate the probability of satisfiability of $F_k(n, rn)$ for fixed k,n,r as the number of satisfiable formulas divided by the total number of formulas. For every k, plot a curve of satisfiability of $F_k(n, rn)$ vs. r. Explain the behavior that you observe. Try to be as rigorous as possible.

3 Encoding of Bayesian Networks (Difficulty: Between Easy and Moderate)

The goal of this project is to encode probabilistic inference query over a Bayesian network into a weighted counting problem. Your program should take the Bayesian network (*model*) and query (*evidence*) as input and return a weighed counting instance. Below are the formats for inputs and outputs:

1. Model: http://www.hlt.utdallas.edu/~vgogate/uai14-competition/modelformat.html The link here describes how to encode Bayesian networks but we will treat every Markov network as Bayesian network by assuming that every function table is a conditional probability such that the last column specifies probability of event $Pr[X_n|X_1, ...X_{n-1}]$ where X_n is the variable in the second last column. For example, the table corresponding to Y and Z (on the url above) should be treated as CPT that describes $Pr[Z|Y]$.

2. Evidence: http://www.hlt.utdallas.edu/~vgogate/uai14-competition/evidformat.html

3. Weighted Formulas: You should output a CNF formula in DIMACS format and another "weight file" that maps weights of all the literals. The header of "weight file" should be p $\langle \#ofvariables \rangle$ followed by every row that starts wtih w, followed by literal and its weight 0 to end the line. Here is a toy example:

p 2
w 1 0.1 0
w -1 0.2 0
w 2 .8 0
w -2 1 0

This file specifies that there are two variables and each of the line specifies the weights of literals.

4 Proof of Unsatisfiability (Difficulty: Moderate)

Modify the SAT solver you have implemented so that if the formula is unsatisfiable, the solver should return a proof of unsatisfiability using resolution. The solver should print out the proof in a file; where the first line should being with "v <# of clauses>" where <# of clauses> specify the number of clauses in your resolution proof. Then the next <# of clauses> should list down all the clauses that are used in resolution proof. An empty clause is also a clause, you can use the symbol -1 to specify empty clause. Note that this ordering also assigns unique id to every clause.

Following listing of all the clauses, you should list down the actual proof in the following format. Every line should first list the id two clauses used in resolution and then the clause that was generated. (E.g., "1 2 3" would represent clause 1 and clause 2 were used in resolution and clause 3 was produced). The last line should end with empty clause being generated.

5 Construction of ROBDD (Difficulty: Hard)

The goal of this project is to construct a Reduced Ordered Binary Decision Diagram (ROBDD) representation of a given CNF formula, and to perform *apply* operation on a given pair formulas. In particular, you need to do the following two tasks:

Task 1 Given a formula F, construct the ROBDD representation of F, say F_R. You are allowed to choose any variable order you want.

Task 2 Given two formulas F and G, you need to perform *apply* operation on them; *apply* $(\odot, F, G)$ computes a ROBDD representing $F \odot G$, where $\odot$ represents the binary operations $\vee, \wedge, \oplus$. You need to perform only *apply* $(\vee, F, G)$ operation for this project.
Your program should have two modes: (i) Construct, (ii) Apply. Construct should perform task 1 and Apply should perform task 2.

Input Format Input formula(s) would be in DIMACS format.

Output Format The output should be in two files:

1. The ".robdd" file is an adjacency list representation of ROBDD F_R. Every non-leaf vertex in the graph representing F_R has an unique integer id, *vid*. Every ROBDD F_R has a true and a false sink, and we will use the special symbols T and F to denote the sinks. Every non-leaf vertex of F_R has exactly two children, *1-child* and *0-child*, where *1-child* (resp. *0-child*) is the child node when *vid* is set to 1 (resp. 0). Therefore, each line in .robdd file will have exactly three elements.

$$\langle vid\ 1\text{-}child\ 0\text{-}child \rangle$$

2. The ".map" file is the mapping from variables of F to vertices of F_R. Each variable v of F appears k_v times in F_R with k_v different *vid*s, where $k_v \geq 0$. Each line in the .map file will have exactly two elements.

$$\langle v, list \rangle$$

   Where, *list* represents the *vid*s that variable v maps to.

Toy Example: Let the given formula F be $x_1 \oplus x_2$ represented in DIMACS as follows:

```
p cnf 2 2
1 2 0
-1 -2 0
```

Figure 1 represents the ROBDD representation F_R corresponding to F, and Listing 1 represents the .robdd file, and listing 2 represents the corresponding .map file.

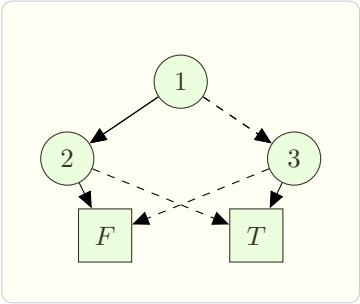

Figure 1: ROBDD representation F_R

Listing 1: .robdd file

```
1 2 3
2 F T
3 T F
```

Listing 2: .map file

```
1 [1]
2 [2,3]
```

6 Final Submission

Your final submission should include:

1. The source code along with verbal explanation of your design choices. Remember, when you are working as a software engineer, your code will never make to production if your peers can not review it.

2. In addition, you should submit a written report of at most 10 pages (excluding references), including:

   (a) You should include a written report of a verbal summary and an analysis of your findings. Your analysis can be intuitive, but you need to come up with an explanation of your findings.

   (b) I believe that you will be a leader in your field in a few years. As a leader, you are expected to evaluate yourself. you should include what letter grade would you have assigned to yourself if you were the instructor. Finally, your report should end with a paragraph explaining what you learned from this project.