Randomized Algorithms: Lab 6

February 19, 2018

1 SAT Problem

Boolean satisfiability problem is the problem of determining if there exists an interpretation that satisfies a given Boolean formula. In other words, it asks whether the variables of a given Boolean formula can be consistently replaced by the values TRUE or FALSE in such a way that the formula evaluates to TRUE. If this is the case, the formula is called satisfiable. On the other hand, if no such assignment exists, the function expressed by the formula is FALSE for all possible variable assignments, and the formula is unsatisfiable. For example, the formula "a AND NOT b" is satisfiable because one can find the values $a =$ TRUE and $b =$ FALSE, which make (a AND NOT b) = TRUE. In contrast, "a AND NOT a" is unsatisfiable.

A formula is in **conjunctive normal form (CNF)** or clausal normal form if it is a conjunction of one or more clauses, where a clause is a disjunction of literals; otherwise put, it is an AND of ORs. For example:

$$\neg A \wedge (B \vee C)$$

$$(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E)$$

***k*-SAT problem** is a boolean satisfiability problem for CNF where each clause has exactly k literals.

2 Algorithm

1. Start with an arbitrary assignment
2. Repeat m times, terminating with all clauses satisfied
   (a) Choose a clause that is currently not satisfied.
   (b) Choose uniformly at random one of the literals in the clause and switch.
3. If valid assignment found, return it.
4. Else, conclude that F is unsatisfiable.

3 Variables

Let's use next notation for CNF and the algorithm:

k: number of literals in clause
n: number of clauses
v: number of variables
m: maximum iterations
r: parameter value from the theorems below.

4 Theorems

2-SAT case:
Since the algorithm runs for $m = 2rn^2$ iterations, the 2-SAT algorithm answers correctly if the formula is unsatisfiable. Otherwise, with probability $\geq 1 - 1/2(r)^r$, it returns a satisfying assignment.

3-SAT case:
Since the algorithm runs for $m = crn^{0.5}(4/3)^n$ iterations, the 3-SAT algorithm answers correctly if the formula is unsatisfiable (c is a constant). Otherwise, with probability $\geq 1 - (1/2)^r$, it returns a satisfying assignment.

5 Task

1. Implement the SAT solver.
2. Select n, v, r. Generate 1000 formulas for 2-SAT case. How much of them are satisfiable? Adjust parameters so that there would be both satisfiable and unsatifiable formulas.
3. With the same 1000 formulas, repeat the experiment for different r values.
4. What is the dependency between n and percentage of satisfiable formulas?
5. What is the dependency between v and percentage of satisfiable formulas?
6. Repeat experiments for 3 literals formulas.