CSCI-GA 3033-114/CS-GY 9223

Spring 2025

Homework 2

This homework is due **Thurs. Feb. 27 at 11:59 P.M.** You do not have to typeset your answers if you don't want to, but make sure they are legible.

Please do your homework together with one other person in the class. You and your partner should hand in only one copy of your solutions, through Gradescope, with both of your names on it. Use the group submission option when uploading.

POLICY ON CONSULTING REFERENCES: Please try to solve the problem yourselves without using any external references (you will learn a lot more that way). If you do consult external references, please write your solutions in your own words and cite any references you have used.

---

1. **Read-Once Formulas**. A read-once Boolean formula is a Boolean formula using the operators $\wedge$ and $\vee$ such that each variable x_i appears only once in the formula. For example, $f(x_1, x_2, x_3, x_4) = x_1 \vee ((x_2 \wedge x_3) \vee x_4)$ is a read-once formula. However, $f(x_1, x_2, x_3, x_4) = (x_1 \wedge x_2) \vee ((x_2 \wedge x_3) \vee x_4)$ is not a read-once formula, because x_2 appears twice. The SBFE problem for read-once formulas was first studied in the 1970's, and it is still not known whether the problem is NP-hard or has a polynomial time algorithm.

   Consider instead the SBFE problem for a simple subclass of read-once formulas, namely read-once DNF formulas. A read-once DNF formula is a Boolean formula of the form $f(x_1, \ldots, x_n) = T_1 \vee T_2 \vee \ldots \vee T_n$ where each T_j is the conjunction ($\wedge$) of a subset of the variables x_i, and each variable x_i appears in at most one T_j. The T_j are called the terms of the formula. For example, $f(x_1, x_2, x_3, x_4, x_5, x_6) = (x_1 \wedge x_2 \wedge x_6) \vee x_4 \vee (x_3 \wedge x_5)$ is a read-once DNF formula, and $(x_3 \wedge x_5)$ is one of its terms.

   Solve the SBFE problem for read-once DNF formulas in the unit-cost case, i.e., in the special case where all $c_i = 1$. You will get partial credit if you solve it with the additional assumption that all the p_i are equal to $1/2$.

2. **k-of-n functions**.

   (a) The optimality of the strategy presented in class for evaluating k-of-n functions relied on the following lemma: Let $f(x_1, \ldots, x_n)$ be a k-of-n function. Let S_1 be the evaluation strategy that performs the tests in increasing order of c_i/p_i until the value of f can be determined. Then S_1 is an optimal strategy for verifying that $f(x) = 1$, i.e., its expected cost is equal to $\min_S E[\text{cost}(S, x) | f(x) = 1]$, where the minimization is over all strategies S for evaluating k-of-n functions S.

   Prove this lemma.

(b) The optimal strategy we presented in class for evaluating k-of-n functions was an adaptive strategy. Show that there is a non-adaptive strategy for evaluating k-of-n functions whose expected cost is at most twice that of the optimal adaptive strategy.

(Recall that a non-adaptive strategy is specified by a permutation of the tests. The strategy performs tests in the order specified by the permutation until there is enough information to detemine the value of the function.)

3. **Submodular Goal Functions**. Let $f(x_1, \ldots, x_n)$ be a Boolean function. A submodular goal function for f is a utility function $u : \{0, 1, *\}^n \to \mathbb{Z}^{\geq 0}$ such that $u(*, \ldots, *) = 0$, u is monotone and submodular, and there exists a value Q such that for all $b \in \{0, 1, *\}^n$, $u(b) = Q$ iff b is a 1-certificate or a 0-certificate for f. The value Q is called the "goal value" of u. One approach to solving an SBFE problem is to reduce it to Stochastic Submodular Cover through the construction of a submodular goal function u.

   Consider the not-all-equal function $f : \{0, 1\}^n \to \{0, 1\}$ such that $f(x) = 1$ if there exist $i, j \in \{1, \ldots, n\}$ such that $x_i = 1$ and $x_j = 0$, and $f(x) = 0$ otherwise. Show that there exists a submodular goal function u for the not-all-equal function whose goal value Q is $O(n^2)$.

4. **Latency-based arguments**. Another approach to proving approximation bounds for testing strategies, not covered in the lectures, is to use latency-based arguments. These compare the progress of the proposed testing strategy after it has performed some number of tests with the progress of an optimal strategy after it has performed a related number of tests.

   Consider an SBFE problem for a class of Boolean functions, in the unit-cost case. Since we are in the unit-cost case, the goal is to minimize the expected number of tests.

   Suppose that we have a proposed evaluation strategy S for solving the SBFE problem for some class, and S^* is an optimal evaluation strategy. A latency-based argument for proving an approximation bound for S might be based on a lemma such as this one:

   Lemma: For all $j \in \{0, \ldots, \lceil \log n \rceil\}$,

$$|R_{\alpha 2^j}| \leq \beta |R_{\alpha 2^{j-1}}| + |R^*_{2^j}|$$

   where α and β are constants such that $0 < \beta < 1/2$ and $\alpha > 1$. Here R^*_t is the probability that on a random x, S^* has not finished testing within its first t tests. Similarly, R_t is the probability that on a random x, S has not finished testing within its first t tests.

   The lemma says roughly that between its first $\alpha 2^{j-1}$ tests and the first $\alpha 2^j$ tests, either the probability that S has not terminated goes down significantly, or the probability it hasn't terminated within $\alpha 2^j$ tests isn't much more than the probability that S^* hasn't terminated within its first 2^j steps.

   Prove that this lemma implies that $E[\text{cost}(S, x)] = O(E[\text{cost}(S^*, x)])$. That is, prove that, assuming the lemma, S achieves a constant-factor approximation relative to the optimal strategy.