CS 276 Lecture 16: The Goldreich Levin Theorem

Scribe: Daniel Chen

March 9, 2006

1 Introduction

The Goldreich-Levin Theorem states that there exists an efficient probabilistic algorithm to solve the following problem: Given a secret string $s \in \{0,1\}^n$, an oracle $f(\cdot)$ such that:

$$p = \Pr[x \leftarrow \{0,1\}^n; f(x) = s \cdot x] = \frac{1}{2} + \frac{\epsilon}{2}$$

and an oracle $EQ(\cdot)$ such that:

$$EQ(y) = \begin{cases} 1 & \text{if } y = s; \\ 0 & \text{otherwise.} \end{cases}$$

we want to be able to guess what s is with probability greater than $\frac{1}{2}$. Naturally, we are seeking an algorithm that runs in time polynomial in n and $\frac{1}{\epsilon}$. Note that an important parameter is the probability p. During the lecture, we will start with the case where $p = 1$, but gradually relax the lower bound of p to eventually get an algorithm for the Goldreich-Levin problem.

2 Case $p = 1$

Here is an algorithm that finds s given $p = 1$:

1. For $i = 1, \ldots, n$, compute $f(e_i)$ where e_i is the string with 1 at the ith position and 0 everywhere else.

2. Then $f(e_i) = e_i \cdot s$ which is the ith bit of s. Clearly, we then just need to output $(f(e_1), \ldots f(e_n))$.

3 Case $p = 1 - \frac{1}{10n}$

The above algorithm does not work anymore because it may be the case that $f(e_i) \neq e_i \cdot s$. Furthermore, it may be the case that $f(e_i) \neq e_i \cdot s$ for all e_i. Therefore, we need to introduce new tricks to solve this problem. Firstly, lets say a string x is *good* if $f(x) = s \cdot x$, and *bad* if $f(x) \neq s \cdot x$. If a string x is selected uniformly at random, then the probability that it is good is $1 - \frac{1}{10n}$ and the probability that it is bad is $\frac{1}{10n}$.

Then, we introduce a randomized algorithm $SC(x)$ such that $\Pr(SC(x) \neq x \cdot s) \leq \frac{1}{5n}$ for all x:

SC(x):

1. $r \leftarrow \{0,1\}^n$
2. Return $f(x+r) - f(r)$

Note that if $f(x+r) = (x+r) \cdot s$ and $f(r) = r \cdot s$, then $SC(x) = x \cdot s$. Therefore, we have:

$$\Pr(SC(x) \neq x \cdot s) \leq \Pr(x+r \text{ is bad}) \cup \Pr(r \text{ is bad}) \leq \frac{1}{10n} + \frac{1}{10n} = \frac{1}{5n}$$

Now, we have an algorithm that outputs s with high probability:

1. For $i = 1, \ldots, n$, let $s_i = SC(e_i)$.
2. Output $y = (s_1, \ldots, s_n)$.

Now, the probability that this algorithm does not return the correct y is bounded by $\sum_i \Pr[s_i \text{ is wrong}] \leq \frac{n}{5n} = \frac{1}{5}$, so this algorithm works.

4 Case $p = \frac{3}{4} + \frac{\delta}{2}$

In this case, the $SC(x)$ algorithm above does not work anymore, because the probability that a string is bad is no longer proportional to n^{-1}. However, we can define a new algorithm Majority-$SC(x)$:

1. For $i = 1, \ldots, k$, let $a_i = SC(x)$.
2. Return majority$(a_1, \ldots, a_k)$.

To analyze the success probability of this algorithms, its useful to think of each invocation of $SC(\cdot)$ as a Bernoulli trial. In this direction, we define:

$$X_i = \begin{cases} 1 & \text{if } SC(x) = s \cdot x; \\ 0 & \text{otherwise.} \end{cases}$$

Then, by similar analysis to the that of the previous section, we have:

$$\Pr[X_i = 1] \geq 1 - \left(\frac{1}{2} - \delta\right) = \frac{1}{2} + \delta$$

Then, we use the Chevyshev Inequality to bound the error probability of Majority-$SC(x)$. First, we note that:

$$E[X_i] = \frac{1}{2} + \delta$$

and

$$Var[X_i] = \left(\frac{1}{2}+\delta\right)\left(\frac{1}{2}-\delta\right) = \frac{1}{4}-\delta^2 \leq \frac{1}{4}$$

Let $X = \sum_i X_i$. Then, by linearity of expectations, we have:

$$E[X] = k\left(\frac{1}{2}+\delta\right)$$

and by pairwise independence, we have:

$$Var[X_i] \leq \frac{k}{4}$$

Then, we have:

$$\begin{aligned}
\Pr(\text{Majority-}SC(x) \neq s \cdot x) &\leq \Pr\left(X < \frac{k}{2}\right) \\
&\leq \Pr(|X - E[X]| > \delta k) \\
&\leq \frac{Var[X]}{\delta^2 k^2} \leq \frac{1}{4\delta^2 k}
\end{aligned}$$

Therefore, if we take $k = \frac{3n}{4\delta^2}$, then the error probability of Majority-$SC(x)$ is at most $\frac{1}{3n}$ so we can use the technique in the previous section to come up with an algorithm that finds s with probability at least $\frac{2}{3}$.

5 The General Case $p = \frac{1}{2} + \frac{\epsilon}{2}$

The general case involves even more tricks. Suppose we know $r_1 \cdot s, \ldots, r_m \cdot s$ for $r_1, \ldots, r_m$ random strings. Then, let $b_i = r_i \cdot s$. Also, let $I = \{1, \ldots, m\}$. Then, we can define an exponential (in m) family of functions $g_J(x)$ where J is a non-empty subset of I:

$$g_J(x) = f\left(x + \sum_{j \in J} r_j\right) - \sum_{j \in J} b_j \tag{1}$$

It is easy to verify that each function in this family returns $x \cdot s$ with probability $\frac{1}{2} + \frac{\epsilon}{2}$. We can also see that this is a pairwise independent family of functions, because each subset of I differs by at least one element. Then, we have the following algorithm returns $x \cdot s$ with a bounded probability of error:

Strong-SC$(x, r_1, \ldots, r_m, b_1, \ldots, b_m)$:

1. For $J \subseteq I$, let $a_J = g_J(x)$.
2. Return majority of all a_J.

Again, in the spirit of a Cheybshev Inequality bound, we let

$$X_J = \begin{cases} 1 & \text{if } a_J = x \cdot s; \\ 0 & \text{otherwise.} \end{cases}$$

Then, $\Pr(X_J = 1) = E[X_J] = \frac{1}{2} + \frac{\epsilon}{2}$, and $Var[X_J] \leq \frac{1}{4}$. Then, let $X = \sum_J X_J$. By linearity of expecatations, $E[X] = 2^m \left(\frac{1}{2} + \frac{\epsilon}{2}\right)$ and by pairwise independence we have $Var[X] \leq \frac{2^m}{4}$. Then, using the Chebyshev Inequality, we have:

$$\begin{aligned} \Pr(\text{Strong-}SC(x) \neq s \cdot x) &\leq \Pr\left(X < \frac{2^m}{2}\right) \\ &\leq \Pr\left(|X - E[X]| > \frac{\epsilon}{2} 2^m\right) \\ &\leq \frac{Var[X]}{\left(\frac{\epsilon}{2} 2^m\right)^2} \leq \frac{1}{\epsilon^2 2^m} \end{aligned}$$

By judicious choice of m such that $2^m \geq 3n/\epsilon^2$, we can bound the error probability of Strong-SC by $\frac{1}{3n}$. This value of m is logarithmic in n and poly-logarithmic in $1/\epsilon$. Now, we can present the algorithm that solves the Goldreich-Levin Problem:

1. Pick $r_1, \ldots, r_m$ randomly from $\{0,1\}^n$
2. For each $(b_1, \ldots, b_m) \in \{0,1\}^m$:
   (a) For $i = 1, \ldots n$, let $y_i = \text{Strong-SC}(e_i, r_1, \ldots, r_m, b_1, \ldots, b_m)$
   (b) Let $y = (y_1, \ldots, y_n)$.
   (c) If $EQ(y) = 1$, output y.

Since one of the $(b_1, \ldots, b_m)$ is correct, in that iteration, we have an error probability bounded by $1/3$ using a union bound. Therefore we have at least $2/3$ chance of getting it right. Clearly, this algorithm runs in polynomial time, because m is logarithmic in n. Therefore, we have the following theorem:

Theorem 5.1 (Goldreich-Levin Theorem). *Given f, EQ as oracles, we can solve the Goldreich-Levin Problem in time polynomial in n and ϵ^{-1}.*