Lovász Local Lemma **Date:** 27 January, 2006
Scribe: Ragesh Jaiswal **Instructor:** Prof. Mohan Paturi

2.1 Motivation

To be able to appreciate the power of Lovász Local Lemma (LLL), let us look at the following example problem:

Consider a boolean formula in CNF such that each clause contains exactly k literals (a k-CNF formula) and that the frequency *of each variable is* $\leq \frac{2^k}{e \cdot k}$. *Show that any such formula is satisfiable. (The frequency of a variable is the number of occurrences of the variable or its negation in the formula.)*

To see that the frequency bound influences the satisfiability of a boolean formula, consider the following simple exercise suggested by Chris.

Exercise 1: Show that a 3-CNF formula with a frequency bound of 3 is satisfiable. *(Hint: Use Hall's theorem to show that each clause has a unique variable that can be set to make the clause true)*

Let φ denote such a boolean formula with n variables and m clauses. Let us analyze the simple randomized algorithm which assigns each variable 0 or 1 with equal probability. Let E_i be the event that the i^{th} clause is *not* satisfied. Then we know

$$\forall i, Pr[E_i] = 2^{-k} \tag{2.1.1}$$

We want to show that there is at least one satisfying assignment. So it will be sufficient to show that

$$Pr[\bigwedge_{i=1}^{m} \bar{E}_i] > 0 \tag{2.1.2}$$

Let make an attempt to do this by using a familiar tool from the previous lecture, *union bound.* We have

$$\begin{aligned} Pr[\bigwedge_{i=1}^{m} \bar{E}_i] &= 1 - Pr[\bigvee_{i=1}^{m} E_i] \\ &\geq 1 - \sum_{i=1}^{m} Pr[E_i] \quad \text{(using union bound)} \\ &= 1 - m \cdot 2^{-k} \end{aligned}$$

So the union bound fails if $m \geq 2^k$. The reason that this fails is due to the fact that the union bound does not use the dependence of the events. This tells us that we need to incorporate the interdependence of these events in our analysis. The information we have is that one variable can appear in at most $\frac{2^k}{ek}$ clauses and the formula is a k-CNF. This implies that any event E_i is independent of all but at most $\frac{2^k}{e}$ events. This, in some sense is the degree of dependence of the events and the important question we need to answer is whether this information is sufficient to show 2.1.2. This leads us to Lovász Local Lemma which we state and prove in the next section. We will revisit our example problem in the section 2.3.

2.2 Lovász Local Lemma

We are given events $E_1, \dots, E_m$ in some probability space. We are also given a directed graph $G(V, E)$ called the *dependency graph* in which there is a vertex representing each event and the edges are such that the events corresponding to two non-connected pair of vertices are mutually independent. It is important to note that for two mutually independent events E_i, E_j there may be an edge going between i, j. This answers Vadim's question regarding the dependency graph being directed. We can now state and prove the most general version of the Lemma.

Lemma 2.2.1 (Lovász Local Lemma) *Let $G(V, E)$ be a dependency graph for events $E, \dots, E_m$ in a probability space. Suppose that there exist $x_i \in [0, 1]$ for $0 \leq i \leq m$ such that*

$$Pr[E_i] \leq x_i \prod_{(i,j)\in E} (1 - x_j) \tag{2.2.3}$$

Then

$$Pr[\bigwedge_{i=1}^{m} \bar{E}_i] \geq \prod_{i=1}^{m} (1 - x_i) \tag{2.2.4}$$

Proof: Let S be a subset of indices $\{1, \dots, m\}$. We first prove by induction on the size of S that for any S and i such that $i \notin S$

$$Pr[E_i | \bigwedge_{j\in S} \bar{E}_j] \leq x_i$$

<u>Base case</u>: $S = \phi$, this is trivially true from 2.2.3.

<u>Inductive step</u>: For any i and $S \subseteq \{1, \dots, m\}$ $(i \notin S)$, let $S_1 = \{j \in S : (i, j) \in E\}$ and $S_2 = S \setminus S_1$. Also let $F_S = \bigwedge_{j\in S} \bar{E}_j$, $F_{S_1} = \bigwedge_{j\in S_1} \bar{E}_j$ and $F_{S_2} = \bigwedge_{j\in S_2} \bar{E}_j$. We need to show that $Pr[E_i|F_S] \leq x_i$. Using conditional probabilities we get

$$\begin{aligned} Pr[E_i|F_S] &= \frac{Pr[E_i \wedge F_S]}{Pr[F_S]} \\ &= \frac{Pr[E_i \wedge F_{S_1}|F_{S_2}]}{Pr[F_{S_1}|F_{S_2}]} \end{aligned} \tag{2.2.5}$$

We can upper bound the numerator in the following manner:

$$\begin{aligned} Pr[E_i \wedge F_{S_1}|F_{S_2}] &\leq Pr[E_i|F_{S_2}] \\ &= Pr[E_i] \quad \text{(since } E_i \text{ is mutually independent of all events in } F_{S_2}) \\ &\leq x_i \prod_{(i,j)\in E} (1 - x_j) \quad \text{(from 2.2.3)} \end{aligned}$$

Let $S_1 = \{j_1, \ldots, j_r\}$. We can lower bound the denominator as

$$\begin{aligned} Pr[F_{S_1}|F_{S_2}] &= Pr[\bigwedge_{i=1}^{r} \bar{E}_{j_i}|F_{S_2}] \\ &= Pr[\bar{E}_{j_1}|\bigwedge_{i=2}^{r} \bar{E}_{j_i} \wedge F_{S_2}] \cdot Pr[\bar{E}_{j_2}|\bigwedge_{i=3}^{r} \bar{E}_{j_i} \wedge F_{S_2}] \ldots Pr[\bar{E}_{j_r}|F_{S_2}] \\ &\geq \prod_{i=i}^{r}(1 - x_{j_i}) \quad \text{(using the induction assumption)} \end{aligned}$$

Now we plug the bounds on the numerator and denominator in Equation 2.2.5 to get

$$\begin{aligned} Pr[E_i|F_S] &\leq \frac{x_i \prod_{(i,j)\in E}(1 - x_j)}{\prod_{i=1}^{r}(1 - x_{j_i})} \\ &= x_i \prod_{(i,j)\in E \wedge j \notin S_1} (1 - x_j) \\ &\leq x_i \end{aligned}$$

Finally we use the induction assertion to show the conclusion of the Lemma.

$$\begin{aligned} Pr[\bigwedge_{i=1}^{m} \bar{E}_i] &= (1 - Pr[E_1|\bigwedge_{i=2}^{m} \bar{E}_i]) \cdot (1 - Pr[E_2|\bigwedge_{i=3}^{m} \bar{E}_i]) \ldots (1 - Pr[E_m]) \\ &\geq \prod_{i=1}^{m}(1 - x_i) \end{aligned}$$

(Note that in the inductive step we need to make sure that $Pr[F_S] > 0$ to be able to express $Pr[E_i|F_S]$ as $\frac{Pr[E_i \wedge F_S]}{Pr[S]}$. This can be done using induction which essentially follows the last step shown above.) ■

The following simpler version of LLL can be more directly applicable to problems in randomized algorithms.

Corollary 2.2.2 *Let $E_1, \ldots, E_m$ be events in a probability space, with $Pr[E_i] \leq p$ for all i. If each event is mutually independent of all other events except for at most d, and if $ep(d+1) \leq 1$, then $Pr[\bigwedge_{i=1}^{m} \bar{E}_i] > 0$.*

Proof: For each $i \in \{1, \ldots, m\}$ we set $x_i = \frac{1}{d+1}$ and show that the precondition 2.2.3 of the Lemma 2.2.1 is satisfied. For any i

$$\begin{aligned} x_i \prod_{(i,j)\in E} (1 - x_j) &\geq \frac{1}{d+1} \cdot \left(1 - \frac{1}{d+1}\right)^d \\ &\geq \frac{1}{(d+1) \cdot e} \\ &\geq p \end{aligned}$$

Since $Pr[E_i] \leq p$ we get that $Pr[E_i] \leq x_i \prod_{(i,j)\in E}(1 - x_j)$. ■

Exercise 2: Show that setting $x_i = \frac{1}{d}$ above does not satisfy the precondition.

2.3 Applying LLL to k-SAT

In this section we apply the simpler version of the LLL to obtain an answer for our original problem. Two clauses which do not share a variable are mutually independent. Since one variable is allowed to appear in at most $\frac{2^k}{ek}$ clauses, the degree of dependence is at most $\frac{2^k}{e} - (k-1)$. So for applying Corollary 2.2.2 we need to make sure that the precondition is satisfied. We have

$$e \cdot 2^{-k} \cdot \left(\frac{2^k}{e} - (k-1) + 1 \right) \leq 1 \quad \text{(since } k \geq 3\text{)}$$

This implies that $Pr[\bigwedge_{i=1}^{m} \bar{E}_i] > 0$. So there is at least one satisfying assignment or in other words the formula is satisfiable.

2.3.1 Finding a Satisfying Assignment

Using LLL we showed the existence of a satisfying assignment. A natural question to ask at this point is can we find one? More formally

Given a k-CNF boolean formula φ such that no variable appears in more than $2^{\alpha k}$ clauses, for a small constant α. Find a satisfying assignment for the formula.

Note how the frequency bound has changed from $\frac{2^k}{ek}$ to $2^{\alpha k}$ in the problem definition. This is because of the solution that we have at hand and will be presenting in this lecture. Getting a randomized algorithm for constructing a satisfiable assignment for much larger value of the frequency bound is left open. The algorithm that we present here works in the following two phases:

Phase I: *(Setting a subset of variables)*

1. Select an ordering of variables
2. Select the next variable x in order
   (a) If x is not marked as *deferred*, set it randomly
   (b) Plug x in the current formula
   (c) For all clauses of width $k/2$, mark all the variables in the clause as deferred

Phase II: *(Exhaustive search for a satisfiable assignment)*

1. Find a satisfying assignment using exhaustive search

For the algorithm to work we need to make sure that the formula obtained after phase I has a satisfying assignment. We show this using LLL. Each clause has width at least $k/2$ which means that the probability that a random assignment makes a clause false is $\leq 2^{-k/2}$. The degree of dependency in the worst case remains the same, which is $d = k2^{\alpha k}$. So the precondition of the LLL is satisfied since $e \cdot 2^{-k/2} \cdot (k \cdot 2^{\alpha k} + 1) \leq 1$ for large values of k and small α.

A clause which is reduced to have width $k/2$ during phase I is called a *dangerous* clause. All clauses that remain unsatisfied after phase I are called *surviving clause.* Note that due to the manner in which we assign values to variables in the first phase, a clause is a surviving clause if either

(a) it is a dangerous clause, or

(b) it shares a deferred variable with a dangerous clause.

Let G denote the initial dependency graph and G' denote the dependency graph induced by the vertices corresponding to the surviving clauses. Note that for two surviving clauses C_i, C_j, there may not be an edge between i and j in G' though they may be connected in G. We will try to show that with high probability all the connected components in G' are of size $O(\log m)$. Since different connected components do not share variables, this would mean that the exhaustive search finds a satisfiable assignment in time polynomial in m.

So we are interested in the probability that there is a large connected component R in G'. The probability that a clause becomes dangerous is $2^{-k/2}$ (note that we do not have a factor of $\binom{k}{k/2}$ since we work with a fixed order of the vertices). So the probability that a clause survives is equal to the probability that there is a dangerous clause in its vicinity and which is $\leq (d+1) \cdot 2^{k/2}$. Since the event that clause survives might not be independent of the event that some other clause survives(e.g. neighboring clauses in G'), we cannot get a bound on the probability that a connected component survives by simply multiplying the bounds on the individual survival probabilities of the clauses in the component. If we can find a subset of nodes within the connected component such that the events of each of the original clauses corresponding to these nodes surviving, are mutually independent, then we can get a bound on the probability that such a subset survives (by taking a product of the survival probabilities of clauses in the subset). This gives a bound on the probability that the connected component survives. In this regard, we define a graph object called rooted 4-Tree T with respect to a connected component R. T has the following properties:

(a) T is rooted tree

(b) Every pair of nodes are at distance at least 4 in G

(c) There can be an edge between two nodes which are at a distance exactly 4 in G

(d) Any node of R is either in T or is at a distance ≤ 3 from a node in T (the distance being with respect to G)

What is the advantage of a 4-Tree? The following exercise would help to answer this question.

Exercise 3: Show that the events that the clauses corresponding to the nodes in a 4-Tree survive, are mutually independent. *(Hint: For nodes $i, j \in T$ consider the minimum distance between any two dangerous clauses which are at a distance ≤ 1 from them)*

So the probability that a given 4-Tree of size t survives $\leq [(d+1)2^{-k/2}]^t$. Given a connected component R with r nodes consider the maximal 4-Tree T with respect to R. Since any node of R is at most at a distance of 3 from a node of T and the degree of each node in G is d, we get that $3d^3|T| \geq r$. This gives that a maximal 4-Tree with respect to R has size at least $r/3d^3$. Using Exercise 3, the probability that such a T survives is $[(d+1)2^{-k/2}]^{r/3d^3}$. So the probability that there is a such a tree that survives is $[(d+1)2^{-k/2}]^{r/3d^3}$ times the number of such 4-Trees. The following exercise helps to get a upper bound on the number of 4-Trees in a graph with bounded degree d.

Exercise 4: Show that the number of 4-Tree of size s in a graph with bounded degree d is at most nd^{8s}. *(Hint: consider enumerating a 4-Tree by in-order traversal of G^4)*

The above exercise gives us that the number of 4-Trees of size $r/3d^3$ is at most $md^{8r/3d^3}$. The probability that there exists a 4-Tree of size $r/3d^3$ upper bounds that probability that there exists a connected component of size $\geq r$ in G'[1]. Hence the probability that G' has a connected component of size $\geq r$ is bounded by

$$md^{8r/3d^3}\left[(d+1)2^{-k/2}\right]^{r/3d^3} \leq m2^{rk/3d^3(8\alpha+2\alpha-1/2)} = o(1)$$

for $r \geq c\log m$ for some constant c and small constant α. So with probability $1-o(1)$ there are no connected components of size $\geq c\log m$.

Finally we need to show that the algorithm outputs a a satisfying assignment with high probability in time polynomial in m. With probability $1-o(1)$, all the components of G' have size $O(\log m)$. We can repeat phase I to obtain G' with small sized connected components (repeating twice should suffice in expectation). Since no two connected components share a variable, exhaustive search runs in time polynomial in m.

References

[MU05] M. Mitzenmacher, E. Upfal.: Probability and Computing. Cambridge University Press, pp. 138–148, 2005.

[MR95] R. Motwani, P. Raghavan.: Randomized Algorithms. Cambridge University Press, pp. 115–120, 1995.

[1]For events A, B if $A \Rightarrow B$, then $Pr[A] \leq Pr[B]$