

CSE 203A: Advanced Algorithms

Lecture Notes: The Lovász Local Lemma

Lecturer: Russell Impagliazzo

Scribe: Stefan Schneider

February 19, 2013

In earlier lectures we considered the “if any, then many” paradigm for randomized algorithm. In many cases, some event $E(x)$ is true for some input x only if it is true for a sizeable number of inputs. Then, the problem of finding such an input can be approached by sampling x randomly. Here we consider the case where the probability of $E(x)$ can be extremely small. Can we still use the probabilistic method to show that the probability is positive? And is it still possible to find an input x that satisfies the property?

The simplest case is an event $E(x_1, \dots, x_n)$ that is true if and only if the events $E_1(x_1), \dots, E_n(x_n)$ are all simultaneously true. If the probability of $E_i(x_i)$ is $\frac{1}{2}$, then the probability of $E(x_1, \dots, x_n)$ is 2^{-n} , but nonzero. Also, we can find an input that satisfies all events by finding inputs x_i independently from each other.

A slightly less simple example to consider is an event $E(x_1, \dots, x_{n+1})$ which is true if and only if events $E_1(x_1, x_2), E_2(x_2, x_3), \dots, E_n(x_n, x_{n+1})$ are true. In this case, it is not always possible to find an input that satisfies E , even if the events E_i can all be satisfied individually. For instance, if $E(x_1, x_2) = 1$ if $x_2 = 1$ and $E(x_2, x_3) = 1$ if $x_2 = 0$, then no assignment will satisfy both E_1 and E_2 .

On the other hand, if we have $P(E_i(x_i, x_{i+1})) \geq \frac{15}{16}$ for all i , then we can pick an a_1 such that $P(E_1(a_1, x_2)) \geq \frac{15}{16}$ as a first step, and as a second step we can pick an a_2 such that $E_1(a_1, a_2)$ is satisfied and $P(E_2(a_2, x_3)) \geq \frac{15}{16} - \frac{1}{16} = \frac{7}{8}$.

The Lovász Local Lemma generalizes above example.

Lemma 1 (Lovász Local Lemma). *Suppose $E_1, \dots, E_n$ is a system of events and for each event E_i there is a set $S_i \subseteq \{E_j\}$ with $|S_i| \leq d$ such that E_i is independent of all events in $\{E_j\} - S_i$. Furthermore each event has probability at least $1 - \frac{1}{ed}$. Then $P(E_1 \wedge \dots \wedge E_n) > 0$.*

Instead of proving the general result, we consider a corollary. A formula in conjunctive normal form is an AND of clauses, where a clause is an OR of literals (a variable or its negation). The k -SAT problem asks if a given CNF where each clause has at most k literals is satisfiable, that is, if there is an assignment to the variables such that the formula evaluates to true. We say a k -CNF is *pure*, if each clause has exactly k literals. We further say a k -CNF has *density* D if no variable appears in more than D clauses.

For a pure k -CNF φ with clauses $C_1, \dots, C_m$, we can apply the Lovász Local Lemma by defining E_i as the event that C_i is satisfied when choosing an assignment uniformly at random. Then $P(E_i) = 1 - 2^{-k}$ and since clauses can only depend on clauses it shares variables with, an event can depend on at most Dk other events. The corollary is then the following.

Corollary 1. *If φ is a pure k -CNF of density less than $\frac{2^k}{ek}$, then φ is satisfiable.*

Proof. We will prove a weaker result with density at most $\frac{2^k}{4k} - 1$. We give a probabilistic algorithm that finds a satisfying assignment with high probability.

Let $C_1, \dots, C_m$ be an arbitrary ordering of the clauses. As a general outline, the algorithm first finds an assignment that satisfies C_1 , then finds an assignment that satisfies both C_1 and C_2 and so on. We use I for the currently last clause we want to satisfy and we keep a stack S of clauses C_j for $j \leq I$ that might be unsatisfied.

```

Set all variables randomly
 $I \leftarrow 1$ 
while ( $I \leq n$ ) {
  if ( $C_I$  is unsatisfied) {
    S.push  $C_I$ 
    Resample variables of  $C_I$ 
  }
  while ( $S$  not empty) {
     $C_j \leftarrow S.top$ 
    if (there are unsatisfied clauses  $C_k$  intersecting with
     $C_j$ , including  $C_j$  itself) {
      S.push  $C_k$ 
      replace all variables of  $C_k$  with new random values
    } else {
      S.pop
    }
  }
   $I++$ 
}

```

We have the following invariant: Every unsatisfied clause C_j with $j \leq I$ intersects with some clause on the stack. We only delete a clause from the stack if it and all intersecting clauses are satisfied. At the same time, a clause can only get unsatisfied if we resample one of its variables, which we only do if we also push a clause containing that variable on the stack.

It remains to show that with high probability, we get to $I > m$ quickly and the algorithm terminates. We construct a history of the algorithm that uniquely determines a run of the algorithm, i.e. in what order we push, pop, or increment I , and which clauses we push to the stack. We then argue that long histories are unlikely. Let T be the number of push operations in a history. We show the following.

1. The number of syntactically correct histories with exactly T push operations is small
2. The probability of getting a particular history is small for large T

For 1), we observe that the number of pop operations is bounded by T , and the number of times we increment I is bounded by m . Since there are at most $2T + m$ steps in total, there are at most $\binom{2T+m}{m}$ possible positions for the increment steps. For the push and pop operations there are at most 2^{2T} configurations. Note that this includes partial histories that do not empty the stack. To make a history unique we further need to specify what clauses we push. Since there are at most Dk clauses intersecting with a clause C_j , the number of possible histories with exactly T

push operations is bounded by

$$\binom{2T+m}{m} 2^{2T} (Dk)^T \leq 2^{o(T)} (4Dk)^T$$

For 2), consider a function $\text{Verify}(\text{randomness}, \text{history})$ that returns true if the history is a correct history for the given random bits. For our purposes, it is sufficient to consider a simple procedure that only checks if clauses we push to the stack are unsatisfied. We observe that the algorithm indeed only pushes unsatisfied clauses to the stack, and then immediately resamples its variables. Hence whenever we push a clause to the stack, all its variables were recently sampled and are therefore random and independent of any previous time steps. The probability that the clauses that the history pushes are indeed unsatisfied is therefore exactly $(2^{-k})^T$.

The probability that the correct history has at least T push operations is therefore bounded by the number of partial histories of length T times the probability that any given history is the correct one. Using $D \leq \frac{2^k}{4k} - 1$ we get

$$\begin{aligned} P(\text{actual history has at least } T \text{ push operations}) &\leq \binom{2T+m}{m} (4Dk)^T (2^{-k})^T \\ &\leq \binom{2T+m}{m} (1 - 4k2^{-k})^T \end{aligned}$$

which is much smaller than 1 if $T > C2^k m$ for some constant C . Note that if $2^k \geq m^2$, then the initial assignment is satisfying with high probability using union bound. Hence we can assume $2^k \leq m$ and conclude $T \leq Cm^2$ with high probability. $\square$