1 Overview

In this lecture, we derive

1) Deterministic algorithm for the Lovász Local Lemma
2) Parallel algorithm for the Lovász Local Lemma

from the Moser-Tardos algorithm.

2 Review

2.1 Limited Independence

In lecture 9, we learned how to generate random bits with limited independence using fewer truly random bits. In fact, some randomized algorithms do not need full independence, and limited independence suffices for them. Suppose we use $\ell$ truly random bits to generate $n$ random variables with limited independence. If $\ell = O(\log n)$, then $2^\ell$ is polynomial in $n$. That means we can check every single seed from those $\ell$ truly random bits in polytime. This will derandomize your randomized algorithm where limited independence is sufficient.

Note the following definition for $k$-wise independence.

Definition 1 ($k$-wise independence) Let $X_1,\ldots,X_n$ be random variables, with respective ranges $A_1,\ldots,A_n$. We say that they are $k$-wise independent if for all subsets $S = \{s_1,\ldots,s_\ell\}$ of $[n]$ with $\ell \leq k$ and for all $a_i \in A_i$

$$Pr[\bigwedge_{i=1}^\ell X_{s_i} = a_i] = \prod_{i=1}^\ell Pr[X_{s_i} = a_i].$$

In lecture 9, we showed that we can generate $\mathbb{F}_q$-valued $k$-wise independent random variables $X_1,\ldots,X_n$ for some prime $q = O(n)$ using only $O(k \log n)$ truly random bits. The following shows that we can generate $n$ binary $k$-wise independent random variables $X_1,\ldots,X_n$ using $O(k \log n)$ truly random bits.

Claim 1 We can generate binary $k$-wise independent random variables $X_1,\ldots,X_n$ using $O(k \log n)$ bits.
Proof With $O(k \log n)$ bits, we can generate $\mathbb{F}_q$-valued $k$-wise independent random variables $Y_1, \ldots, Y_n$ for some prime $q = O(n)$. Let’s define a random variable $X_i$ for each $i$ as $X_i := Y_i \pmod{2}$. It is sufficient to show that $X_1, \ldots, X_n$ are $k$-wise independent.

Note that for all $S = \{s_1, \ldots, s_\ell\} \subseteq [n]$ with $|S| = \ell \leq k$ and $b_1, \ldots, b_n \in \{0, 1\}$,

$$\Pr[\bigwedge_{i=1}^\ell X_{s_i} = b_i] = \Pr[\bigwedge_{i=1}^\ell Y_{s_i} \equiv b_i \pmod{2}]$$

$$= \Pr[\bigwedge_{i=1}^\ell \bigvee_{t_i \equiv b_i \pmod{2}} Y_{s_i} = t_i]$$

$$= \sum_{(t_1, \ldots, t_\ell) \equiv (b_1, \ldots, b_\ell)} \Pr[\bigwedge_{i=1}^\ell Y_{s_i} = t_i]$$

$$= \prod_{i=1}^\ell \sum_{t_i \equiv b_i} \Pr[Y_{s_i} = t_i]$$

$$= \prod_{i=1}^\ell \Pr[X_{s_i} = b_i]$$

By Claim 1, we are able to generate $\text{poly}(n)$ number of binary random variables using $O(k \log n)$ bits. In the analysis of the Moser-Tardos algorithm, binary random variables are used to fill in the randomness table. We can replace them with our $k$-wise independent random variables to fill in the randomness table. Since there are $2^{O(k \log n)}$ seeds from $O(k \log n)$ bits, we need to check $2^{O(k \log n)}$ corresponding tables.

Later, we will show that we need $C \log n$-wise independence for some constant $C$ in order to derandomize the Moser-Tardos algorithm. Then we need to use $O((\log n)^2)$ random bits. However, using $O((\log n)^3)$ random bits is not appropriate for generating a deterministic algorithm since $2^{O((\log n)^2)} = n^{O(\log n)}$. Instead of using $O(\log n)$-wise independence, we can take advantage of $\delta$-approximate $O(\log n)$-wise independence. When $\delta = 1/\text{poly}(n)$, we can also get some high probability results. Note the following.

Definition 2 (\(\delta\)-approximate $k$-wise independence) Let $X_1, \ldots, X_n$ be random variables, with respective ranges $A_1, \ldots, A_n$. We say that they are $\delta$-approximate $k$-wise independent if for all subsets $S = \{s_1, \ldots, s_\ell\}$ of $[n]$ with
\( \ell \leq k \) and for all \( a_i \in A_i \)

\[
\left| \Pr_{\ell} \left[ \bigwedge_{i=1}^{\ell} X_{s_i} = a_i \right] - \prod_{i=1}^{\ell} \Pr[X_{s_i} = a_i] \right| < \delta.
\]

**Claim 2** We can generate binary \( \delta \)-approximate \( k \)-wise independent random variables \( X_1, \ldots, X_n \) using \( O(k + \log n + \log \frac{1}{\delta}) \) bits.

The above claim implies that we can generate polynomially many binary \( 1/\text{poly}(n) \)-approximate \( O(\log n) \)-wise independent random variables using only \( O(\log n) \) bits. Since \( 2^{O(\log n)} = O(n^c) \) for some constant \( c \), there are only polynomially many randomness tables to check to get a deterministic algorithm.

### 2.2 The Moser-Tardos Algorithm

In lectures 15, 16, and 17, we learned the Moser-Tardos algorithm which provides a constructive proof of the Lovász Local Lemma. Here, we restate some technical details which are used in the analysis of the algorithm.

The following is setting for the symmetric version of the Lovász Local Lemma. We have \( n \) binary random variables \( X_1, \ldots, X_n \) and \( m \) events \( A_1, \ldots, A_m \). Let \( vbl(A_i) \) denote the set of random variables that \( A_i \) depends on for each \( i \). Let \( k_{\min} := \min_{1 \leq i \leq m} |vbl(A_i)| \) and \( k_{\max} := \max_{1 \leq i \leq m} |vbl(A_i)| \). We can assume that \( k_{\max} \) is a constant, i.e., each event \( A_i \) depends on only a constant number of random variables. Then we can draw the corresponding dependency graph. Let us assume that the maximum degree of the dependency graph is \( \Delta \) and \( \Pr[A_1] = \ldots = \Pr[A_m] = p \) for some constant \( p \in [0, 1] \). By the Lovász Local Lemma, if \( (\Delta + 1)p < 1 \), we know that \( \Pr[A_1 \wedge \cdots \wedge A_m] > 0 \). In other words, there is an assignment of \( X_1, \ldots, X_n \) such that all of \( A_1, \ldots, A_m \) are violated.

Note the following claims.

**Claim 3** If \( (\Delta + 1)p < 1 - \epsilon \), then there is no consistent witness tree of size greater than \( \frac{c \log n}{\epsilon} \) with high probability where \( c \) is a constant.

**Proof** Note that the probability that there is a witness tree of size at least \( K = \frac{c \log n}{\epsilon} \) is at most

\[
\sum_{i=1}^{m} \sum_{s=K}^{\infty} \Pr[\exists \text{consistent witness tree rooted at } A_i \text{ of size } s].
\]

by the union bound. We also know that the number of consistent witness trees rooted at \( A_i \) of size \( s \) is at most \( \binom{\Delta s}{s-1} \). Let \( \tau \) be a witness tree rooted at \( A_i \) of size \( s \). The probability that \( \tau \) is consistent with the table is equal to the probability that the event in each node of \( \tau \) is violated by the assignment given in the table. Since each entry in the randomness table is independent of the other entries, one node is independent from the other nodes. The probability that the event
corresponding to each node is violated is \( p \). Therefore, the probability that \( \tau \) is consistent with the table is \( p^s \). Therefore, by the union bound, we get

\[
\sum_{i=1}^{m} \sum_{s=K}^{\infty} P[\exists \text{consistent WT rooted at } A_i \text{ of size } s] \leq \sum_{i=1}^{m} \sum_{s=K}^{\infty} \left( \frac{\Delta s}{s-1} \right) p^s \\
\leq \sum_{i=1}^{m} p \sum_{s=K}^{\infty} (\Delta pe)^{s-1}
\]

Note that \( \sum_{s=K}^{\infty} (\Delta pe)^{s-1} \leq \frac{1}{(1-\Delta pe)\Delta pe} (\Delta pe)^K \). In addition, we can also prove that \( \sum_{i=1}^{m} p < n/e \). The proof for that is following. For each random variable \( X_i \), consider the subgraph \( C_i \) of the dependency graph induced by the nodes where their corresponding events depend on \( X_i \). Clearly, \( C_i \) is a clique. Besides, \( C_1, \ldots, C_n \) cover all the nodes of the dependency graph. Therefore,

\[
\sum_{i=1}^{m} p(A_i) \leq \sum_{j=1}^{n} \sum_{A \in C_j} p(A) \leq \sum_{j=1}^{n} (\Delta + 1) p < n/e.
\]

since the size of each clique is at most \( \Delta + 1 \) and we assumed that \( (\Delta + 1)pe < 1 \).

Since \( K = \frac{c \log n}{\epsilon} \), we also get that

\[
\frac{1}{(1-\Delta pe)\Delta pe} (\Delta pe)^K < \frac{1}{C'}(1-\epsilon)^{\frac{c}{k_{\min}}} \log n = \frac{1}{C'} n^{\frac{c}{k_{\min}} \log(1-\epsilon)} \leq \frac{1}{C'} n^{-c/2}
\]

where \( C' \) is a constant. Then, \( \sum_{i=1}^{m} p \sum_{s=K}^{\infty} (\Delta pe)^{s-1} \leq \frac{1}{C'} n^{1-c/2} \). We can choose a sufficiently large \( c \) so that there is no consistent witness tree of size at least \( c \frac{\log n}{\epsilon} \) with high probability.

**Claim 4** If \( (\Delta + 1)pe < 1 - \epsilon \), the total number of resampling is \( O(n \log n / \epsilon k_{\min}) \) with high probability.

**Proof** By Claim 3, we know that no random variable is resampled more than \( \frac{c \log n}{\epsilon} \) times with high probability. Otherwise, there exists a consistent witness tree of size greater than \( \frac{c \log n}{\epsilon} \). Let \( r \) denote the number of total resampling steps. Since each event depends on at least \( k_{\min} \) random variables, at least \( k_{\min} \) variables are resampled in each resampling step. Then,

\[
rk_{\min} \leq \frac{cn \log n}{\epsilon}.
\]

Hence, \( r \leq \frac{n \log n}{\epsilon k_{\min}} \).
3 Deterministic Algorithm for the Lovász Local Lemma

We proved in Claim 3 that there is no consistent witness tree of size greater than $C \log n$ with high probability for some constant $C$ when we use full independence. To fill in the randomness table, you need $\Omega(n^c) \log n$ random variables for some constant $c$. However, you cannot derive an efficient deterministic algorithm out of it in this case, because there are $2^{O(n^c)}$ tables to consider.

In order to get a deterministic algorithm for the Lovász Local Lemma, we will use $\delta$-approximate $k$-wise independent binary random variables. When $\delta$ is $1/poly(n)$ and $k = O((\log n)^c)$, we need only $O(\log n)$ random bits to generate polynomially many $\delta$-approximate $k$-wise independent random variables. That means there are $2^{\Omega(n^c)} = n^{O(1)}$ seeds, so we need to check polynomially many tables to get a satisfying assignment. However, the problem is Claim 3 does not hold in this case. If a witness tree $\tau$ has size $\omega(\log n)$, then we cannot directly compute the probability of $\tau$ being consistent. In other words, we cannot apply the union bound to bound the probability that there exist consistent witness tree of size greater than $K$.

Fortunately, we have another strategy. In fact, we can prove that there is a table among those $n^{O(1)}$ tables where there is no consistent witness tree of size greater than $C \log n$ for some constant $C$. The following two lemmas show that.

**Lemma 5** If $\tau$ is a consistent witness tree, then there exists a consistent witness tree of size in the interval $[|\tau| - 1, |\tau| - 1]$.

**Proof** Note that there are at most $\Delta + 1$ child nodes of the root of $\tau$, since the root is allowed to have a child node with the same event. Let $v_1, \ldots, v_\ell$ be the child nodes of the root where $\ell \leq \Delta + 1$. Consider the subtree of $\tau$ rooted at $v_i$ for each $i$, and say it is $\tau_i$. By the Pigeonhole principle, the maximum of $|\tau_1|, \ldots, |\tau_\ell|$ is at least $|\tau| - 1$. Without loss of generality, we may assume that $|\tau_1|$ is the maximum. Let us generate the witness tree rooted at $v_1$. Then each node that was in $\tau_i$ for $i \neq 1$ are in the witness tree. Thus, the size of the witness tree is at least $|\tau_1|$.

By Lemma 5, if there is no consistent witness tree of size in the interval $[C \log n, C(\Delta + 1) \log n]$, then there is no consistent witness tree that has size greater than $C(\Delta + 1) \log n$ for any constant $C$. Thus, there is no consistent witness tree of size greater than $C \log n$ in this case.

**Lemma 6** We fill in the randomness table using $\delta$-approximate $k$-wise independent random variables where $\delta = 1/n^c$ and $k = (\Delta + 1)Ck_{\max} \log n$ for some constants $C$ and $c$. Then, there is a table such that there is no consistent witness tree of size in $[C \log n, C(\Delta + 1) \log n]$.

**Proof** It is sufficient to show the probability that there is a consistent witness tree of size in $[C \log n, C(\Delta + 1) \log n]$ is less than 1. Let $\tau$ be a witness tree of
size \( s \in [C \log n, C(\Delta + 1) \log n] \). Note that

\[
|Pr[\tau \text{ is consistent with the randomness table}] - p^*| < \delta,
\]

since we use \( \delta \)-approximate \( k \)-wise independent random variables and \( \tau \) depends on at most \( sk_{\max} \) entries of the randomness table. Then, we get the following.

\[
\sum_{i=1}^{m} \sum_{s=C \log n}^{C(\Delta + 1) \log n} Pr[\exists \text{ consistent WT rooted at } A_i \text{ of size } s] \leq \sum_{i=1}^{m} \sum_{s=K}^{\infty} \binom{\Delta s}{s-1} (p^* + \delta).
\]

To complete the proof, we need to argue that there are constants \( C \) and \( c \) such that

\[
\sum_{i=1}^{m} \sum_{s=K}^{\infty} \binom{\Delta s}{s-1} p^* \leq \sum_{i=1}^{m} \sum_{s=C \log n}^{C(\Delta + 1) \log n} (\Delta e)^{s-1}
\]

\[
\leq n \cdot (C \Delta (\log n)) (\Delta e)^{C \log n - 1}
\]

\[
= \frac{C}{pe} n \log n \cdot (n^{C \log (\Delta e)}
\]

\[
= \frac{C}{pe} n^{1-C \log(1/(\Delta e))} \log n
\]

Secondly, consider the following.

\[
\sum_{i=1}^{m} \sum_{s=K}^{\infty} \binom{\Delta s}{s-1} \delta \leq \frac{\delta}{p} \sum_{i=1}^{m} \sum_{s=C \log n}^{C(\Delta + 1) \log n} (\Delta e)^{s-1}
\]

\[
\leq \frac{\delta}{p} n \cdot (C \Delta (\log n)) (\Delta e)^{C \log n - 1}
\]

\[
= \frac{C}{pe} n^{1-C \log(1/(\Delta e))} \log n
\]

\[
= \frac{C}{pe} n^{1-c-C \log(1/(\Delta e))} \log n
\]

Therefore, we can choose both \( c \) and \( C \) such that \( \sum_{i=1}^{m} \sum_{s=K}^{\infty} \binom{\Delta s}{s-1} (p^* + \delta) \) is less than 1.

By Lemmas 5 and 6, there is a table such that there is no consistent witness tree of size greater than \( C \log n \) when we fill in the randomness table using \( (1/n^c) \)-approximate \( (C(\Delta + 1)k_{\max} \log n) \)-wise independent random variables. Then, the total number of resampling steps is at most \( \frac{Cn \log n}{k_{\min}} \), so \( \frac{Cn^2 \log n}{k_{\min}} \) entries of the table are used. Hence, we can use \( (1/n^c) \)-approximate \( (C(\Delta + 1)k_{\max} \log n) \)-wise independent binary random variables \( Y_1, \ldots, Y_{Cn^2/k_{\min}} \) to fill in the randomness table. In this case, we use

\[
O\left( (C(\Delta + 1)k_{\max} \log n + \log(Cn^3/k_{\min}) + \log(n^c) \right) = O\left( (C(\Delta + 1)k_{\max} + 3C/k_{\min} + c) \log n \right)
\]
truly random bits, and thus there are $n^{\theta(\Delta k_{\text{max}})}$ tables to consider. The deterministic algorithm for the Lovász Local Lemma is as follows.

**Algorithm 1:** Deterministic LLL algorithm

**Input:** $n$ variables $X_1, \ldots, X_n$ and $m$ events $A_1, \ldots, A_m$ where $vbl(A_i)$ denotes the set of random variables that $A_i$ depends on for each $i$;

**Generate** all $n^{\theta(\Delta k_{\text{max}})}$ possible tables that contain only $\frac{Cn^2}{k_{\text{min}}}$ resampling steps by using $(1/n^c)$-approximate $(C(\Delta + 1)k_{\text{max}} \log n)$-wise independent binary random variables $Y_1, \ldots, Y_{\frac{Cn^3}{k_{\text{min}}}}$;

**Run** the Moser-Tardos algorithm on each table;

**Return** the current assignment;

Since there are polynomially many entries in each table, the running time of the above algorithm is $n^{\theta(\Delta k_{\text{max}})}$. If both $\Delta$ and $k_{\text{max}}$ are constants, then the algorithm terminates in polyme. Since we proved that there is a table of size $n \times \frac{Cn^3}{k_{\text{min}}}$ which gives a satisfying assignment, the algorithm is correct.

### 4 Parallel Algorithm for the Lovász Local Lemma

In this lecture, we also give a parallel algorithm for the Lovász Local Lemma. The original Moser-Tardos algorithm picks one violated event and resamples variables that the event depends on. In the parallel algorithm, we select a maximal independent $S$ set in the subgraph of the dependency graph induced by the nodes corresponding to all the violated events. Since each event in $S$ does not share any variable with the others in $S$, we can resample all the variables that the events in $S$ depend on in parallel. Consider the below algorithm.

**Algorithm 2:** Parallel LLL algorithm

**Input:** $n$ variables $X_1, \ldots, X_n$ and $m$ events $A_1, \ldots, A_m$ where $vbl(A_i)$ denotes the set of random variables that $A_i$ depends on for each $i$;

**Start** with a random assignment;

**while** there exists a violated event $A$ **do**

- **Take** a maximal independent set $S$ in the subgraph of the dependency graph induced by all the violated events;
- **Resample** all the variables in $\bigcup_{A \in S} vbl(A)$;

**end**

**Return** the current assignment;

Suppose that the above parallel algorithm terminates after $\ell$ iterations. Let $S_1, \ldots, S_\ell$ denote the respective maximal independent sets. Let $A$ be an event which is chosen in the $i$th iteration. Then the next lemma shows that the witness tree rooted at $A$ has depth exactly $i$.

**Lemma 7** Let $A$ be an event included in the maximal independent set $S_i$ chosen in the $i$th iteration. Then the witness tree rooted at $A$ has depth exactly $i$.

**Proof** Since $S_i$ is a maximal independent set, then any node in $S_i$ cannot be attached to $A$. Suppose that there is no node in $S_{i-1}$ that is attached to witness
tree. Then we get that $vbl(B) \cap vbl(A) = \emptyset$ for all $B \in S_{i-1}$, so $\{A\} \cup S_{i-1}$ is an independent set. Besides, when we resample all the variables in $\cup_{B \in S_{i-1}} vbl(B)$, the values in the variables that $A$ depends on do not change at all. That means $A$ was also violated in the $i-1$th iteration, and thus $\{A\} \cup S_{i-1}$ is an independent set which appears in the $i-1$th iteration. However, this contradicts the assumption that $S_{i-1}$ is maximal. Therefore, there is a node $A_{i-1}$ in $S_{i-1}$ attached to $A$. Again, notice that the other nodes in $S_{i-1}$ cannot be attached to $A_{i-1}$ and they can only be attached to $A$. Then the depth of the witness tree before attaching nodes in $S_{i-2}$ is exactly 2. Likewise, there is a node $A_{i-2}$ in $S_{i-2}$ attached to $A_{i-1}$ and the depth increases by exactly 1 after looking at $S_{i-2}$. This proves the lemma.

In fact, we can use Lemma 7 to prove that the number of iterations of the parallel algorithm is $O(\log n)$ with high probability.

**Theorem 8** The number of iterations of the parallel algorithm for the Lovász Local Lemma is $O(\log n)$ with high probability.

**Proof** Note that a tree of depth $k$ contains at least $k$ nodes. If the number of iterations is at least $K$, then there exists a consistent witness tree of size at least $K$. Therefore, we can bound the probability that the number of iterations is at least $K$ by the probability that there is a consistent witness tree of size at least $K$. Note that

$$\sum_{i=1}^{m} \sum_{s=K}^{\infty} Pr[\exists \text{ consistent WT rooted at } A_i \text{ of size } s] \leq \sum_{i=1}^{m} \sum_{s=K}^{\infty} \left(\frac{\Delta s}{s-1}\right) p^s$$

$$\leq \sum_{i=1}^{m} p \sum_{s=K}^{\infty} (\Delta pe)^{s-1}$$

$$\leq \frac{1}{\Delta pe(1-\Delta pe)} n \cdot (\Delta pe)^K$$

If $K = C \log n$ for some constant $C$, then we get that

$$\frac{1}{\Delta pe(1-\Delta pe)} n \cdot (\Delta pe)^K = \frac{1}{\Delta pe(1-\Delta pe)} n^{1-C \log \frac{1}{\Delta pe}}.$$

If we assume that both $\Delta$ and $p$ are constant, then there exists a sufficiently large constant $C$ such that $1 - C \log \frac{1}{\Delta pe} = -C'$ for another positive constant $C'$. Therefore, we just proved that the number of iterations is at most $C \log n$ with high probability.

5 Conclusion

In this lecture, we derived deterministic and parallel algorithms for the Lovász Local Lemma from the Moser-Tardos algorithm. Note that the parallel algorithm works on just one randomness table, while the deterministic one works
on polynomially many randomness tables. In fact, we can also combine those two algorithms so that we derive another deterministic algorithm which runs in $O(\log n)$ iterations to compute an LLL assignment. In lecture 19, we showed that Luby’s algorithm finds a maximal independent set of a graph in $O(\log m)$ iterations in expectation where $m$ is the number of nodes in the graph. With $m$ processors for the $m$ nodes in the dependency graph, we can find a maximal independent set.

References
