

CSL 857: Randomized Algorithms

Ragesh Jaiswal
CSE, IIT Delhi

LLL: Statement

- Lovász Local Lemma (LLL): Let $G = (V, E)$ denote the dependency graph for events $E_1, \dots, E_m$. Suppose there exists $x_i \in [0, 1]$ for all $0 \leq i \leq m$ such that

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

then

$$\Pr[\bigwedge_{i=1}^m \neg E_i] \geq \prod_{i=1}^m (1 - x_i).$$

- Dependency graph: A dependency graph for dependent events $E_1, \dots, E_m$ is defined as follows: there is a node i for each event E_i and there is an edge (i, j) for every pair of dependent events (E_i, E_j) .
 - Note that there may be an edge between two independent events.

LLL: Proof

- Claim: Let $S \subset \{1, \dots, m\}$ and $i \notin S$, then

$$\Pr[E_i | \bigwedge_{j \in S} \neg E_j] \leq x_i.$$

- Proof of LLL (using above claim):

$$\begin{aligned} \Pr[\bigwedge_{i=1}^m \neg E_i] &= \\ (1 - \Pr[E_1 | \bigwedge_{i=2}^m \neg E_i]) \cdot \\ (1 - \Pr[E_2 | \bigwedge_{i=3}^m \neg E_i]) \dots (1 - \Pr[E_m]) \\ &\geq \prod_{i=1}^m (1 - x_i). \end{aligned}$$

LLL: A useful corollary

- LLL Corollary: Let $0 < p < 1$. Let $E_1, \dots, E_m$ be some events in some probability space such that $\forall i, \Pr[E_i] \leq p$. If each event is mutually independent of all other events except for at most d events, and if $e \cdot p \cdot (d + 1) \leq 1$, then $\Pr[\bigwedge_{i=1}^m \neg E_i] > 0$.

LLL: A useful corollary

- LLL Corollary: Let $0 < p < 1$. Let $E_1, \dots, E_m$ be some events in some probability space such that $\forall i, \Pr[E_i] \leq p$. If each event is mutually independent of all other events except for at most d events, and if $e \cdot p \cdot (d + 1) \leq 1$, then $\Pr[\bigwedge_{i=1}^m \neg E_i] > 0$.
- Problem: Consider boolean formulas in k -CNF (i.e., each clause has exactly k literals), and that the frequency of each variable is $\leq \frac{2^k}{ek}$. Show that any such formula is satisfiable.
- Let E_i denote the event that clause i is not satisfied (for random assignment). The degree of dependence is at most $\frac{2^k}{e} - (k - 1)$ and we have $e \cdot 2^{-k} \cdot \left(\frac{2^k}{e} - (k - 1) \right) \leq 1$ for $k \geq 3$. So, $\Pr[\bigwedge_{i=1}^m \neg E_i] > 0$.

Constructive LLL

Constructive LLL

- Problem: Consider boolean formulas in k -CNF (i.e., each clause has exactly k literals), and that the frequency of each variable is $\leq \frac{2^k}{ek}$. Show that any such formula is satisfiable.
- Using LLL, we have argued that there exists a satisfying assignment for the above boolean formula.
- Question: Is it possible to find a satisfying assignment efficiently?
 - Answer: We can when the frequency is $\leq 2^{\alpha k}$ for some small constant $\alpha < 1$.

Constructive LLL

- Phase I:
 - Select an ordering of the n variables.
 - Select the next variable x in order:
 - If x is not marked as *deferred*, set it randomly.
 - Plug x in the current formula.
 - For all clauses of width $k/2$, mark all the variables in the clause as *deferred*.
- Phase II:
 - Find a satisfying assignment using exhaustive search.

Constructive LLL

- Phase I:
 - Select an ordering of the n variables.
 - Select the next variable x in order:
 - If x is not marked as *deferred*, set it randomly.
 - Plug x in the current formula.
 - For all clauses of width $k/2$, mark all the variables in the clause as *deferred*.
 - Phase II:
 - Find a satisfying assignment using exhaustive search.
-
- Observation 1: Phase I is efficient.
 - Claim 1: The residual boolean formula after Phase I has a satisfying assignment.

Constructive LLL

- Phase I:
 - Select an ordering of the n variables.
 - Select the next variable x in order:
 - If x is not marked as *deferred*, set it randomly.
 - Plug x in the current formula.
 - For all clauses of width $k/2$, mark all the variables in the clause as *deferred*.
 - Phase II:
 - Find a satisfying assignment using exhaustive search.
-
- Observation 1: Phase I is efficient.
 - Claim 1: The residual boolean formula after Phase I has a satisfying assignment.
 - Observation 2: Phase II returns a satisfying assignment.

Constructive LLL

- Phase I:
 - Select an ordering of the n variables.
 - Select the next variable x in order:
 - If x is not marked as *deferred*, set it randomly.
 - Plug x in the current formula.
 - For all clauses of width $k/2$, mark all the variables in the clause as *deferred*.
 - Phase II:
 - Find a satisfying assignment using exhaustive search.
-
- Observation 1: Phase I is efficient.
 - Claim 1: The residual boolean formula after Phase I has a satisfying assignment.
 - Observation 2: Phase II returns a satisfying assignment.
 - Claim 2 (informal): Phase II is efficient w.h.p.

Constructive LLL

- Claim 2 (informal): Phase II is efficient w.h.p.
 - Proof idea: Let G be the original dependency graph and let G' be the dependency graph after Phase I. We will argue that the w.h.p. G' has small ($O(\log m)$) connected components. So, the exhaustive search can be made to run in time $\text{poly}(n)$.
- Definitions:
 - Dangerous clause: A clause is called dangerous if its width is reduced to $k/2$ after Phase I.
 - Surviving clause: A clause that survives after Phase I.
- Claim 3: The probability that a clause becomes dangerous is $_\?_\$.

Constructive LLL

- Claim 2 (informal): Phase II is efficient w.h.p.
 - Proof idea: Let G be the original dependency graph and let G' be the dependency graph after Phase I. We will argue that the w.h.p. G' has small ($O(\log m)$) connected components. So, the exhaustive search can be made to run in time $\text{poly}(n)$.
- Definitions:
 - Dangerous clause: A clause is called dangerous if its width is reduced to $k/2$ after Phase I.
 - Surviving clause: A clause that survives after Phase I.
- Claim 3: The probability that a clause becomes dangerous is $2^{-k/2}$.

Constructive LLL

- Claim 2 (informal): Phase II is efficient w.h.p.
 - Proof idea: Let G be the original dependency graph and let G' be the dependency graph after Phase I. We will argue that the w.h.p. G' has small ($O(\log m)$) connected components. So, the exhaustive search can be made to run in time $\text{poly}(n)$.
- Definitions:
 - Dangerous clause: A clause is called dangerous if its width is reduced to $k/2$ after Phase I.
 - Surviving clause: A clause that survives after Phase I.
- Claim 3: The probability that a clause becomes dangerous is $2^{-k/2}$.
- Claim 4: The probability that a clause survives is at most $__?$.

Constructive LLL

- Claim 2 (informal): Phase II is efficient w.h.p.
 - Proof idea: Let G be the original dependency graph and let G' be the dependency graph after Phase I. We will argue that the w.h.p. G' has small ($O(\log m)$) connected components. So, the exhaustive search can be made to run in time $\text{poly}(n)$.
- Definitions:
 - Dangerous clause: A clause is called *dangerous* if its width is reduced to $k/2$ after Phase I.
 - Surviving clause: A clause that *survives* after Phase I.
- Claim 3: The probability that a clause becomes *dangerous* is $2^{-k/2}$.
- Claim 4: The probability that a clause *survives* is at most $(d + 1) \cdot 2^{-k/2}$.

Constructive LLL

- Definitions:
 - Dangerous clause: A clause is called *dangerous* if its width is reduced to $k/2$ after Phase I.
 - Surviving clause: A clause that *survives* after Phase I.
- Claim 3: The probability that a clause becomes *dangerous* is $2^{-k/2}$.
- Claim 4: The probability that a clause *survives* is at most $(d + 1) \cdot 2^{-k/2}$.
- Question: What is the probability that the surviving clauses form a large connected component?
 - Issue: We cannot multiply the probabilities in Claim 4 since the events are not independent.

Constructive LLL

- Definitions:
 - Dangerous clause: A clause is called *dangerous* if its width is reduced to $k/2$ after Phase I.
 - Surviving clause: A clause that *survives* after Phase I.
- Claim 3: The probability that a clause becomes *dangerous* is $2^{-k/2}$.
- Claim 4: The probability that a clause *survives* is at most $(d + 1) \cdot 2^{-k/2}$.
- Question: What is the probability that the surviving clauses form a large connected component?
 - Issue: We cannot multiply the probabilities in Claim 4 since the events are not independent.
 - Consider two surviving clauses that are at a distance at least 4 in G . What is the probability that both clauses survive?

Constructive LLL

- Claim 5: Any connected component of surviving clauses is of size $O(\log m)$ w.h.p.
- Proof:
 - Consider any connected component R of size r of the surviving clauses.
 - Consider a maximal subset T of clauses in this component such that
 1. Each pair of clauses in T are at a distance at least 4 in G .
 2. If we create a new graph G'' with a vertex for each clause in T and there is an edge between C_i and C_j if they are at a distance of 4 in G , then G'' is connected.
 - Claim 6: $|T| \geq \frac{r}{3d^3}$.
 - Definition (4-tree): Any subset of clauses that follow property (1) and (2) above is called a 4-tree.

Constructive LLL

- Claim 5: Any connected component of surviving clauses is of size $O(\log m)$ w.h.p.
- Proof:
 - Consider any connected component R of size r of the surviving clauses.
 - Consider a maximal subset T of clauses in this component such that
 1. Each pair of clauses in T are at a distance at least 4 in G .
 2. If we create a new graph G'' with a vertex for each clause in T and there is an edge between C_i and C_j if they are at a distance of 4 in G , then G'' is connected.
 - Claim 6: $|T| \geq \frac{r}{3d^3}$.
 - Definition (4-tree): Any subset of clauses that follow property (1) and (2) above is called a 4-tree.
 - Claim 7: There are at most $\left(m \cdot d^{\frac{8r}{3d^3}}\right)$ 4-trees of size $r/3d^3$.

Constructive LLL

- Claim 5: Any connected component of surviving clauses is of size $O(\log m)$ w.h.p.
- Proof:
 - Consider any connected component R of size r of the surviving clauses.
 - Consider a maximal subset T of clauses in this component such that
 1. Each pair of clauses in T are at a distance at least 4 in G .
 2. If we create a new graph G'' with a vertex for each clause in T and there is an edge between C_i and C_j if they are at a distance of 4 in G , then G'' is connected.
 - Claim 6: $|T| \geq \frac{r}{3d^3}$.
 - Definition (4-tree): Any subset of clauses that follow property (1) and (2) above is called a 4-tree.
 - Claim 7: There are at most $\binom{m \cdot d^{\frac{8r}{3d^3}}}{r}$ 4-trees of size $r/3d^3$.
 - Claim 8: The probability that there is a surviving 4-tree of size $r/3d^3$ after phase I is at most $m \cdot d^{\frac{8r}{3d^3}} \left[(d+1)2^{-\frac{k}{2}} \right]^{\frac{r}{3d^3}} \leq m \cdot \frac{r^k}{2^{3d^3(8\alpha+2\alpha-\frac{1}{2})}} = o(1)$ for $r \geq c \cdot \log m$.

End
