

CSL 857: Randomized Algorithms

Ragesh Jaiswal
CSE, IIT Delhi

Constructive LLL: Moser and Tardos

Matching bounds of non-constructive LLL

Constructive LLL: Moser and Tardos

- Solve(ψ)
 - Pick a random assignment
 - While there is an unsatisfied clause A
 - Re-sample the variables in clause A
- Observation 1: When the algorithm terminates, it returns a satisfying assignment.
- Claim (informal): The algorithm Solve is efficient.
- Claim (formal): The expected number of resampling steps in the above algorithm is at most $\sum_A \frac{x(A)}{1-x(A)}$.

Constructive LLL: Moser and Tardos

- Claim (formal): The expected number of re-sampling steps in the algorithm is at most $\sum_A \frac{x(A)}{1-x(A)}$.
- Definitions:
 - Computation Log: This is the sequence of events for which re-sampling was done.
 - Witness Tree: A finite rooted tree along with a label function $\sigma: V \rightarrow \Delta$. The tree has the following property:
 - Children of a vertex u receive labels from the set of neighbors of $\sigma(u)$.
 - Proper witness tree: A witness tree is called proper if distinct children of a vertex receive distinct labels.

Constructive LLL: Moser and Tardos

- Given a computation log $\mathcal{C}$ and a re-sampling step t , we consider the following tree $\tau_{\mathcal{C}}(t)$:
 - The root of the tree is $\mathcal{C}(t)$.
 - We consider events $\mathcal{C}(t-1), \mathcal{C}(t-2), \dots$
 - Suppose we are considering event $\mathcal{C}(i)$ and let T denote the current tree. If $\mathcal{C}(i)$ has dependency on any $\sigma(v)$ for v in T , then we attach a vertex with label $\mathcal{C}(i)$ to the deepest such v . Otherwise, we skip $\mathcal{C}(i)$.

Constructive LLL: Moser and Tardos

- Given a computation log $\mathcal{C}$ and a re-sampling step t , we consider the following tree $\tau_{\mathcal{C}}(t)$:
 - The root of the tree is $\mathcal{C}(t)$.
 - We consider events $\mathcal{C}(t-1), \mathcal{C}(t-2), \dots$
 - Suppose we are considering event $\mathcal{C}(i)$ and let T denote the current tree. If $\mathcal{C}(i)$ has dependency on any $\sigma(v)$ for v in T , then we attach a vertex with label $\mathcal{C}(i)$ to the deepest such v . Otherwise, we skip $\mathcal{C}(i)$.
- We say that a witness tree τ occurs in log $\mathcal{C}$ if there exists a t such that $\tau_{\mathcal{C}}(t) = \tau$.

Constructive LLL: Moser and Tardos

- Lemma 1: Let τ be a fixed witness tree and $\mathcal{C}$ be the computation log produced by the algorithm. The following holds true:
 1. If τ occurs in $\mathcal{C}$, then τ is proper.
 2. The probability that τ occurs in $\mathcal{C}$ is at most $\prod_{v \in V(\tau)} \Pr[\sigma(v)]$.
- Proof:
 - (1) is true by construction.
 - For (2) consider the following argument: We define a randomized procedure called τ -check. We consider vertices of τ in reverse BFS order and consider random assignment to variables in $\sigma(v)$ in that order.

Constructive LLL: Moser and Tardos

- Lemma 1: Let τ be a fixed witness tree and $\mathcal{C}$ be the computation log produced by the algorithm. The following holds true:
 1. If τ occurs in $\mathcal{C}$, then τ is proper.
 2. The probability that τ occurs in $\mathcal{C}$ is at most $\prod_{v \in V(\tau)} \Pr[\sigma(v)]$.
- Proof:
 - (1) is true by construction.
 - For (2) consider the following argument: We define a randomized procedure called τ -check. We consider vertices of τ in reverse BFS order and consider random assignment to variables in $\sigma(v)$ in that order.
 - The probability that τ -check succeeds (i.e., makes all the $\sigma(v)$'s happen) is given by $\prod_{v \in V(\tau)} \Pr[\sigma(v)]$.
 - When τ occurs in $\mathcal{C}$ and we run the τ -check on the same random source as was used by the algorithm, τ -check succeeds.

Constructive LLL: Moser and Tardos

- For any event A , let r.v. N_A denote the number of times A occurs in the random execution $\log \mathcal{C}$.
- Claim 1: The number of distinct proper witness trees with root labeled A that occurs in $\mathcal{C}$ is N_A .
- We now define a randomized process that generates proper witness trees with fixed label of the root. This is known as the *Galton-Watson* process:
 - Let A denote the label of the root.
 - Repeat:
 - For each vertex v added in the previous round, do the following:
 - Let $N(\sigma(v))$ be the neighboring events of $\sigma(v)$. Consider the set $P = N(\sigma(v)) \cup \{v\}$. For each event A in P : add a vertex u with label A with probability $x(A)$.

Constructive LLL: Moser and Tardos

- We now define a randomized process that generates proper witness trees with fixed label of the root. This is known as the *Galton-Watson* process:
 - Let A denote the label of the root.
 - Repeat:
 - For each vertex v added in the previous round, do the following:
 - Let $N(\sigma(v))$ be the neighboring events of $\sigma(v)$. Consider the set $P = N(\sigma(v)) \cup \{v\}$. For each event A in P : add a vertex u with label A with probability $x(A)$.
- Lemma 2: Let τ be a fixed proper witness tree with its root labeled A . The probability p_τ that the Galton-Watson process yields exactly tree τ is $p_\tau = \frac{1-x(A)}{x(A)} \cdot \prod_{v \in V(\tau)} x'(\sigma(v))$. Here $x'(B) = x(B) \prod_{D \in T(B)} (1 - x(D))$.

Constructive LLL: Moser and Tardos

- Lemma 1: Let τ be a fixed witness tree and C be the computation log produced by the algorithm. The following holds true:
 1. If τ occurs in C , then τ is proper.
 2. The probability that τ occurs in C is at most $\prod_{v \in V(\tau)} \Pr[\sigma(v)]$.
- Lemma 2: Let τ be a fixed proper witness tree with its root labeled A . The probability p_τ that the Galton-Watson process yields exactly tree τ is

$$p_\tau = \frac{1-x(A)}{x(A)} \cdot \prod_{v \in V(\tau)} x'(\sigma(v)).$$
 Here

$$x'(B) = x(B) \prod_{D \in T(B)} (1 - x(D)).$$
- Let $W(A)$ denote the set of all proper witness trees with root labeled A .
- $$\begin{aligned} E[N_A] &= \sum_{\tau \in W(A)} \Pr[\tau \text{ occurs in } C] \\ &\leq \sum_{\tau \in W(A)} \prod_{v \in V(\tau)} \Pr[\sigma(v)] \end{aligned}$$

Constructive LLL: Moser and Tardos

- Lemma 1: Let τ be a fixed witness tree and $\mathcal{C}$ be the computation log produced by the algorithm. The following holds true:
 1. If τ occurs in $\mathcal{C}$, then τ is proper.
 2. The probability that τ occurs in $\mathcal{C}$ is at most $\prod_{v \in V(\tau)} \Pr[\sigma(v)]$.
- Lemma 2: Let τ be a fixed proper witness tree with its root labeled A . The probability p_τ that the Galton-Watson process yields exactly tree τ is $p_\tau = \frac{1-x(A)}{x(A)} \cdot \prod_{v \in V(\tau)} x'(\sigma(v))$. Here $x'(B) = x(B) \prod_{D \in T(B)} (1 - x(D))$.

- Let $W(A)$ denote the set of all proper witness trees with root labeled A .

- $$\begin{aligned} E[N_A] &= \sum_{\tau \in W(A)} \Pr[\tau \text{ occurs in } \mathcal{C}] \\ &\leq \sum_{\tau \in W(A)} \prod_{v \in V(\tau)} \Pr[\sigma(v)] \\ &\leq \sum_{\tau \in W(A)} \prod_{v \in V(\tau)} x'(\sigma(v)) \\ &= \frac{x(A)}{1 - x(A)} \cdot \sum_{\tau \in W(A)} p_\tau \leq \frac{x(A)}{1 - x(A)}. \end{aligned}$$

End
