[CS1961: Lecture 12] Algorithmic Lovász Local Lemma

Instructor: Chihao Zhang;
Scribed by Kangrui Cen, Yunran Yang, Yuchen He

1 Constraint Satisfaction Problem

Let $q \geqslant 2$ be an integer and consider n variables $x_1, x_2, \ldots, x_n \in [q]$. We have m constraints $C_1, C_2 \ldots, C_m$ where C_j acts on ℓ_j variables $x_{j_1}, x_{j_2}, \ldots, x_{j_{\ell_j}}$ and $C_j\left(x_{j_1}, x_{j_2}, \ldots, x_{j_{\ell_j}}\right)$ is either 0 or 1. SAT is a special case of this *constraint satisfiation problem* (CSP) when $q = 2$ and all C_j are disjunctive.

$C_j\left(x_{j_1}, x_{j_2}, \ldots, x_{j_{\ell_j}}\right) = 1$ means that C_j is satisfied by the given assignment while $= 0$ means it is not satisfied.

For example, the q-coloring problem can be viewed as an instance of CSP. Given a graph $G = (V, E)$, regard $V = \{v_1, \ldots, v_n\}$ as the variable set. There is a constraint for each edge $e = (v_i, v_j)$ requiring v_i and v_j are colored differently.

The q-coloring problem can be generalized to hypergraphs. Given a hypergraph[1] $G = (V, E)$, we color each vertex with one of the q colors. A coloring σ is proper if for any $e \in E$, there exists $v, w \in e$ such that $\sigma(v) \neq \sigma(w)$.

[1] A hypergraph is a generalization of a graph in which a hyperedge $e \in E$ can be any subset of V.

Consider a special instance: k-uniform hypergraph with max degree Δ [2]. For every edge e, define the bad event B_e as "e is monochromatic". Then $\mathbf{Pr}[B_e] = \frac{1}{q^{k-1}}$. The max degree of the dependency graph $d \leqslant k(\Delta - 1) < k\Delta$. By the Lovász local lemma, if $e\frac{1}{q+1}k\Delta < 1$, i.e., $q = \Omega(\Delta^{\frac{1}{k}})$, there exists a proper coloring of G.

[2] A hypergraph is k-uniform if each hyperedge contains exactly k vertices. The max degree of a hypergraph is Δ if a vertex appears in at most Δ edges.

Note that if we can determine whether a CNF ϕ is satisfiable in polynomial time, we can find the solution also in polynomial time by trying each value of the variables in turn and checking the satisfiability of the remaining CNF. However, this process does not apply to k-CNF under the condition of local lemma ($e2^{-k}\Delta k < 1$) since the regime is not closed under *pinning the value of a variable*. Is it possible to find such a solution in polynomial time under similar conditions? This is a long standing open problem in theoretical coputer science and has been resolved by Moser and Tardos by an extremely simple algorithm with an elegent analysis.

2 Algorithmic Lovász Local Lemma

Consider a k-CNF $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ where $C_j = x_{j_1} \vee x_{j_2} \vee \cdots \vee x_{j_k}$. If the probability of selecting a solution in a random trial is $\frac{1}{\mathrm{poly}(m)}$, then we can find a solution by sampling polynomial times in expectation. However, under the condition of local lemma, the probability might be exponentially small.

Moser and Tardos considered the following algorithm that keeps resampling the unsatisfied clauses: Let V be the set of all variables. At first, the

algorithm picks a random assignment $\sigma : V \to \{0,1\}$. While σ does not satisfy ϕ, pick a violated C_j and resample all the variables in C_j. This is a Las Vegas algorithm which terminates when a satisfiable assignment is found. We now analyze its running time, or equivalently the number of times a clause is resampled.

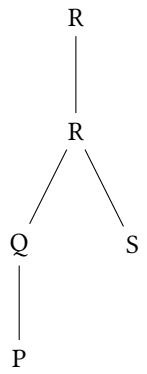
Theorem 1 (Algorithmic Lovász Local Lemma) *If there exists* $x : C \to (0,1)$ *s.t.* $\mathbf{Pr}[C_i \text{ is not satisfied}] \leqslant x(i)\prod_{j\sim i}(1-x(j))$*, then for each* C_j*, the algorithm resample* C_j *at most* $\frac{x(j)}{1-x(j)}$ *times in expectation before terminates.*

2.1 Witness Tree

We record the resampled clause at each step. This gives a log of execution $C\colon \mathbb{N} \to \{C_j\}_{j\in[m]}$ where $C(i)$ is the chosen clause at step i.

For each step t, we introduce a *witness tree* $\tau_c(t)$. We label each vertex v in the witness tree by a clause and denote the corresponding label as $[v]$. The root of $\tau_c(t)$ is labeled as $C(t)$. We construct the witness tree in an anachrony manner. For $i < t$, if there exists v such that $[v]$ is either $C(i)$ or a neighbor of $C(i)$ in the dependency graph, then add a neighbor to the deepest such v and label it as $C(i)$ in the $\tau_c(t)$. Otherwise we just skip this step and go to $i-1$. For a fixed witness tree τ, we say τ appears in the execution log C if there exists some t such that $\tau_c(t) = \tau$. For example, the witness tree $\tau_c(6)$ corresponding to the execution log P, S, Q, R, P, R with $N(P) = \{Q\}$, $N(Q) = \{P, R\}$, $N(R) = \{Q, S\}$ and $N(S) = \{R\}$ is as follows[3].

[3] Here $N(P)$ denotes the neighbor of P in the dependency graph.



For a fixed witness tree τ, we perform τ-*check* in an order of decreasing depth. We traverse the tree from bottom to root. For each clause $[v]$, assign random values for all the variables in $[v]$. If none of the clauses in τ is satisfied, then the τ-check process returns PASS.

Lemma 2 *For any witness tree* $\tau = (V_\tau, E_\tau)$*,*

$$\mathbf{Pr}[\tau \text{ appears in } C] \leqslant \prod_{v\in V_\tau} \mathbf{Pr}[[v] \text{ is not satisfied with a random assignment}].$$

Proof. We perform τ-check on this witness tree. Note that

$$\prod_{\nu \in V_\tau} \mathbf{Pr}\left[[\nu] \text{ is not satisfied with a random assignment}\right] = \mathbf{Pr}\left[\tau\text{-check returns } \textsc{Pass}\right].$$

We only need to prove that $\mathbf{Pr}\left[\tau\text{-check returns } \textsc{Pass}\right] \geqslant \mathbf{Pr}\left[\tau \text{ appears in } C\right]$.

Suppose that the resampling algorithm uses randomness $\mathcal{R}\colon [n] \times \mathbb{N} \to \{0,1\}$. That is, when variable x_i is resampled for the j-th time, the result of its assignment is $\mathcal{R}(i,j)$. $\mathcal{R}$ is called the resampling table and we assume that $\mathcal{R}$ is fixed beforehand.

We couple the τ-check process and the resampling algorithm by using the same $\mathcal{R}$ when performing τ-check. Assume τ appears in C. Suppose x_i is resampled when $\nu_j \in V_\tau$ is visited by τ-check. Let $\mathrm{rec}(x_i, \nu_j)$ be the set of indices of resampling x_i before visiting ν_j. That is

$$\mathrm{rec}(x_i, \nu_j) = \{k \in [j-1] \colon x_i \in \mathrm{var}([\nu_k])\}$$

where $\mathrm{var}([\nu_k])$ is the set of variables in clause $[\nu_k]$. Let $\mathrm{time}(\nu_j)$ be the time when ν_j is added to τ. For time $t = 1, 2, \ldots, \mathrm{time}(\nu_j) - 1$, x_i is resampled at step t iff $t = \mathrm{time}(\nu_k)$ for some $k \in \mathrm{rec}(x_i, \nu_j)$. Therefore, $x_i = \mathcal{R}\left(i, |\mathrm{rec}(x_i, \nu_j)| + 1\right)$ at $\mathrm{time}(\nu_j)$. This indicates that every clause in τ is not satisfied in the τ-check process since τ appears in C. In other words,

$$\mathbf{Pr}\left[\tau\text{-check returns } \textsc{Pass}\right] \geqslant \mathbf{Pr}\left[\tau \text{ appears in } C\right].$$

□

2.2 *Proof of Algorithmic Lovász Local Lemma*

We use the Galton-Watson process to generate a random τ. First we fix a randomly chosen clause A^* as the root of τ. In the i-th round, for each vertex ν born in the $(i-1)$-th round and each $B \in N^+([\nu])$,[4] attach a new child with label B to V with probability $x(B)$. All the random choices are independent in this process.

[4] $N^+([\nu]) = N([\nu]) \cup \{[\nu]\}$.

Let p_τ be the probability that the Galton-Watson process yields exactly τ. Let $W_\nu = \{A \in N^+([\nu]) \mid \text{no child of } \nu \text{ is labeled by } A\}$ and $x'(A) = x(A) \prod_{B \in N(A)} (1 - x(B))$ Then

$$p_\tau = \frac{1}{x(A^*)} \prod_{\nu \in V_\tau} \left(x([\nu]) \prod_{A \in W_\nu} (1 - x(A)) \right)$$

where $\frac{1}{x(A^*)}$ accounts for the fact that the root is always born. Furthermore,

$$p_\tau = \frac{1 - x(A^*)}{x(A^*)} \prod_{\nu \in V_\tau} \left(\frac{x([\nu])}{1 - x([\nu])} \prod_{A \in N^+([\nu])} (1 - x(A)) \right),$$

where $\prod_{A \in N^+([\nu])}(1 - x(A))$ is the probability that no child is born and the term $\frac{x([\nu])}{1-x([\nu])}$ says that in fact $[\nu]$ is born. Therefore, we have

$$\begin{aligned} p_\tau &= \frac{1 - x(A^*)}{x(A^*)} \prod_{\nu \in V_\tau} \left(x([\nu]) \prod_{A \in N([\nu])} (1 - x(A)) \right) \\ &= \frac{1 - x(A^*)}{x(A^*)} \prod_{\nu \in V_\tau} x'([\nu]). \end{aligned}$$

Let $\mathcal{T}_A$ be the set of proper witness trees whose root is labeled by A. Let N_A be the number of times that clause A is resampled during the algorithm, i.e., $N_A = \sum_{\tau \in \mathcal{T}_A} \mathbf{1}[\tau \text{ appears in } C]$. Then by Lemma 2,

$$\mathbf{E}[N_A] = \sum_{\tau \in \mathcal{T}_A} \mathbf{Pr}[\tau \text{ appears in } C] \leqslant \sum_{\tau \in \mathcal{T}_A} \sum_{\nu \in V_\tau} \mathbf{Pr}[[\nu] \text{ is not satisfied}].$$

Since $\mathbf{Pr}[[\nu] \text{ is not satisfied}] \leqslant x'([\nu])$, we have

$$\begin{aligned} \mathbf{E}[N_A] &\leqslant \sum_{\tau \in \mathcal{T}_A} \sum_{\nu \in V_\tau} x'([\nu]) \\ &= \frac{x(A)}{1 - x(A)} p_\tau \\ &\leqslant \frac{x(A)}{1 - x(A)}, \end{aligned}$$

which completes the proof of algorithmic Lovász local lemma.