CSIS0351/8601: Randomized Algorithms
Lecture 9: Lovasz Local Lemma, Job Shop Scheduling
Lecturer: Hubert Chan
Date: 15 Nov 2010

These lecture notes are supplementary materials for the lectures. They are by no means substitutes for attending lectures or replacement for your own notes!

1 Limited Dependency

We have seen how the union bound is used in probabilistic method. Suppose $A_0, A_1, \ldots, A_{n-1}$ are *bad* events (not necessarily independent) such that each $Pr[A_i] \leq p$. If $np < 1$, then by union bound, we conclude that $Pr[\cup_i A_i] \leq np < 1$, and hence, with positive probability, none of the bad events occur. We see that if the events have limited dependency, we can have the same conclusion under a weaker condition.

Definition 1.1 (Dependency Graph) *Suppose $A_0, A_1, \ldots, A_{n-1}$ are events in some probability space. A dependency graph $H = (V, E)$ is a graph with vertex set $V = [n]$ such that for each $i \in [n]$, if $J := \{j : \{i, j\} \in E\}$ is the set of neighbors of i, then the event A_i is independent of all the events $\{A_j : j \notin J\}$.*

Formally, for any disjoint subsets $J_1, J_2 \subseteq [n] \setminus J$,

$Pr[A_i] = Pr[A_i | (\cap_{j \in J_1} A_j) \cap (\cap_{j \in J_2} \overline{A_j})]$.

Remark 1.2 Observe that the dependency graph is not unique. The complete graph is trivially a dependency graph, but not a very useful one.

1.1 Example: Monochromatic Subsets

Recall from the first lecture that $S_1, S_2, \ldots, S_m$ are l-subsets of U. We show that if $m < 2^{l-1}$, then it is possible to color each element of U BLUE or RED such that none of S_i is monochromatic. We show that if the subsets have limited intersection, then l does not have to depend on m.

Theorem 1.3 (Lovasz Local Lemma) *Suppose the collection $\{A_i : i \in [n]\}$ of events has a dependency graph with maximum degree $D \geq 1$. Suppose further that for each i, $Pr[A_i] \leq p$, and $4pD \leq 1$. Then, $Pr[\cup_i A_i] < 1$, i.e., with positive probability, none of the events A_i happens.*

Claim 1.4 *Suppose each subset S_i intersects at most 2^{l-3} other subsets. Then, it is possible to color each element of U BLUE or RED such that none of the subsets S_i is monochromatic.*

Proof: For each element in U, we pick a color uniformly at random. Let A_i be the event that the subset S_i is monochromatic. Then, $p := Pr[A_i] = \frac{1}{2^{l-1}}$.

Observe that the event A_i is independent of all events A_j's such that $S_i \cap S_j = \emptyset$. Hence, in the dependency graph $H = ([n], E)$, $\{i, j\} \in E$ *iff* $S_i \cap S_j \neq \emptyset$. The maximum degree is $D \leq 2^{l-3}$.

Hence, $4pD \leq 4 \cdot \frac{1}{2^{l-1}} \cdot 2^{l-3} \leq 1$. By Lovasz Local Lemma, $Pr[\cup_i A_i] < 1$. ■

2 Proof of Lovasz Local Lemma

We shall prove the following claim.

Claim 2.1 *If $S \subseteq [n]$ and $i \notin S$, then $Pr[A_i | \cap_{j \in S} \overline{A_j}] < \frac{1}{2D}$.*

The result follows from the claim because

$Pr[\cap_i \overline{A_i}] = \prod_i Pr[\overline{A_i} | \cap_{j<i} \overline{A_j}] > (1 - \frac{1}{2D})^n > 0.$

We next prove the claim by induction on the size of S.

Base Case. $|S| = 0$. In this case, $Pr[A_i | \cap_{j \in S} \overline{A_j}] = Pr[A_i] \leq p \leq \frac{1}{4D} < \frac{1}{2D}$.

Inductive Step. Suppose the result holds for all S such that $|S| < r$, for some $r \geq 1$. We now consider $|S| = r$.

Suppose $i \notin S$. Consider decomposition of S into two sets:

(1) $S_1 := \{j \in S : \{i, j\} \in E\}$;

(2) $S_2 := \{j \in S : \{i, j\} \notin E\}$.

If $S_1 := \emptyset$, then $Pr[A_i | \cap_{j \in S} \overline{A_j}] = Pr[A_i | \cap_{j \in S_2} \overline{A_j}]$. By the dependency assumption, the latter quantity equals $Pr[A_i] \leq p < \frac{1}{2D}$.

We next consider $S_1 \neq \emptyset$. Hence, $|S_2| < r$.

Observer that

$$Pr[A_i | \cap_{j \in S} \overline{A_j}] = \frac{Pr[A_i \cap (\cap_{j \in S_1} \overline{A_j}) | \cap_{j \in S_2} \overline{A_j}]}{Pr[\cap_{j \in S_1} \overline{A_j} | \cap_{j \in S_2} \overline{A_j}]}.$$

We first consider the numerator. First, $Pr[A_i \cap (\cap_{j \in S_1} \overline{A_j}) | \cap_{j \in S_2} \overline{A_j}] \leq Pr[A_i | \cap_{j \in S_2} \overline{A_j}] = Pr[A_i] \leq p$, where the equality in the middle follows from the dependency assumption.

We next consider the denominator. For $j \in S_1$, we have $Pr[A_j | \cap_{j \in S_2} \overline{A_j}] < \frac{1}{2D}$. We can apply the induction hypothesis because $j \notin S_2$ and $|S_2| < r$. By the union bound, we conclude that $Pr[\cup_{j \in S_1} A_j | \cap_{j \in S_2} \overline{A_j}] < \frac{|S_1|}{2D} \leq \frac{1}{2}$. Hence, $Pr[\cap_{j \in S_1} \overline{A_j} | \cap_{j \in S_2} \overline{A_j}] > \frac{1}{2}$.

Therefore, we have $Pr[A_i | \cap_{j \in S} \overline{A_j}] < \frac{p}{1/2} = 2p \leq \frac{1}{2D}$, finishing the inductive step of the proof.

3 Job Shop Scheduling

Problem Instance. We are given m jobs $J_1, J_2, \ldots J_m$ and n machines $M_0, M_2, \ldots, M_{n-1}$ with the following rules.

1. Each job J_j must be processed by some subset of machines in a specific given order. Each job is processed by a particular machine at most once. For example, job J_1 has to be processed by the machines in the order M_6, M_1, M_5, M_3.

2. It takes one unit of time for a machine to process a job during its turn; this is the same over all machines and jobs. A machine can only process at most 1 job at the same time.

Goal. Schedule the jobs among the machines so that the *makespan*, which is the time for the last job to be finished, is minimized.

Some Easy Lower Bounds.

In a problem instance, let C be the number of jobs performed by the machine processing the most number of jobs. Since each machine can only process at most one job in one time step, it cannot finish before time C.

On the other hand, let L be the number of machines required by the job having the longest machine sequence. Since each machine takes one time-step to perform a job, the job cannot finish before time L. Hence, $T := \max\{C, L\}$ is a lower bound on the makespan of the optimal schedule.

3.1 Random Delay

We will use randomness and Lovasz Local Lemma to show there exists a schedule whose makespan approaches the lower bound asymptotically.

Theorem 3.1 *There exists a schedule with makespan* $2^{O(\log^* T)}T$.

Remark 3.2 Given a positive integer n, $\log^* n$ is the smallest non-negative integer i such that $\log_2^{(i)} n < 2$, i.e., the number of times one can take logarithms before the number drops below 2.

The function $\log^*$ grows very slowly, and in practice it can be considered as a constant. For instance, $\log^* 2^{65536} = 5$, where 2^{65536} has more than 19,000 digits in base 10.

Relax Assumption. Suppose we relax the assumption and allow each machine to process more than 1 job at the same time. However, a job still takes one time-step to be processed by a machine in the required sequence. Then, there is a relaxed schedule S_0 with makespan L. We show how to convert this infeasible schedule to one that is feasible.

Definition 3.3 *In a schedule (not necessarily feasible), the relative congestion of a machine in some time window is the ratio of number of jobs performed in the window divided by the number of time steps in the window.*

Hence, it follows that if we consider the relaxed schedule S_0, the relative congestion of each machine in the time window of size T is at most 1. Moreover, a schedule is feasible if the relative congestion of each machine in all time windows with 1 time step is at most 1.

We define $T_0 := T$, and have the following invariant. In schedule S_i, each machine in any time window of size T_i or more has relative congestion at most 1. The goal is to decrease T_i at each step, at the cost in increasing the makespan.

Scheduling by Random Delay. We convert the schedule S_0 into S_1 in the following way. For each job J_j, pick an integer x_j uniformly at random from $\{0, 1, 2, \ldots, 2T - 1\}$ independently. Each job J_j delays for x_j time steps before starting as before. As before, we still allow machines to work on more than 1 job at the same time.

We next show that with positive probability, for some $T_1 < T$, all windows of size T_1 or more for each machine under schedule S_1 have relative congestion at most 1.

3.2 Applying Lovasz Local Lemma

Lemma 3.4 *There is some function $f : \mathbb{N} \to \mathbb{N}$ where $f(n) = \Theta(\log n)$ such that with $T_1 = f(T)$, there is a positive probability that, under schedule S_1, for every machine, every window of size T_1 or more has relative congestion at most 1.*

Proof: For each machine M_i, define A_i to be the event that there is some window with size at least T_1 for machine M_i that has relative congestion larger than 1. We specify the exact value of T_1 later. For the time being, think of $T_1 = O(\log T)$.

We next form a dependency graph $H = ([n], E)$ such that $\{u, v\} \in E$ *iff* both machines M_u and M_v process the same job. Observe that A_i is independent of all the A_j's for which M_i and M_j do not process any common job.

We estimate the maximum degree of H. Consider machine M_i. Observe that it can process at most $C \leq T$ jobs. Each of those jobs can go through at most $L \leq T$ machines. Hence, the maximum degree of H is $D \leq T^2$.

We next give an upper bound on $Pr[A_i]$. Consider a fixed window W of size $\tau \geq T_1$ for machine M_i. For each job J_j that is being processed by machine M_i, we define X_j to be the indicator random variable that takes value 1 if job J_j falls into the window W for machine M_i, and 0 otherwise.

Observe that X_j's are independent, because the random delays are picked independently. Moreover, $E[X_j] = Pr[X_j = 1] \leq \frac{\tau}{2T}$.

Define Y to be the number of jobs that fall into the window W for machine M_i. Then, Y is the sum of X_j's for the jobs J_j that are performed by machine M_i. Note that Y is a sum of at most T independent $\{0,1\}$-independent random variables, each of which has expectation at most $\frac{\tau}{2T}$.

Introducing Dominating Random Variable Z. We define Z to be a sum of T independent $\{0,1\}$-independent random variables, each of which has expectation exactly $\frac{\tau}{2T}$. Intuitively, Z is more likely to be larger than Y. (This can be proved formally. We will talk about this in details in the next lecture.) Observe that $E[Z] = \frac{\tau}{2}$.

Hence, $Pr[Y > \tau] \leq Pr[Z > \tau] = Pr[Z > 2E[Z]]$. By Chernoff Bound, this is at most $\exp(-\frac{E[Z]}{3}) = \exp(-\frac{\tau}{6}) \leq \exp(-\frac{T_1}{6})$. Observe that if we had not used Z to analyze Y, then since $E[Y] \leq \frac{\tau}{2}$, we would have obtained $\exp(-\frac{E[Y]}{3}) \geq \exp(-\frac{\tau}{6})$, i.e., the direction of the inequality is not what we want.

Note that there are trivially at most $(3T)^2$ windows. Hence, using union bound, we have $Pr[A_i] \leq 9T^2 \cdot \exp(-\frac{T_1}{6}) =: p$.

Hence, in order to use Lovasz Local Lemma, we need $4pD \leq 1$. Therefore, it is enough to have $36T^2 \exp(-\frac{T_1}{6}) \cdot T^2 \leq 1$. We set $T_1 := 6\ln(36T^4) = \Theta(\log T)$.

By the Lovasz Local Lemma, $Pr[\cap_i \overline{A_i}] > 0$. Hence, the result follows. ■

Conclusion. We begin with a schedule S_0 of makespan at most $P_0 := T$ such that every window of size $T_0 := T$ or more for each machine has relative congestion at most 1. After the transformation, we obtain a schedule S_1 of make span at most $P_1 = 3P_0$ such that every window of size $T_1 = 6\ln(4T_0^4)$ or larger for each machine has relative congestion at most 1.

3.3 Recursive Transformation

Observe that we can apply the same transformation to schedule S_1. In particular, we divide the total time into windows of size T_1, and apply the same transformation separately for each window to obtain schedule S_2, with makespan at most $P_2 = 3P_1$ such that for each machine, every window of size $T_2 = f(T_1)$ or more has relative congestion at most 1. Here, the function f comes from Lemma 3.4.

Hence, we have the series $T_0 := T$, $P_0 := T$, and $T_{i+1} := f(T_i)$ and $P_{i+1} := 3P_i$. This process can continue as long as $f(T_i) < T_i$.

The process stops when $f(T_k) \geq T_k$, at which point T_k is at most some constant K_f, which depends only on the function f. This means in schedule S_k, in each time step, each machine has to deal with at most K_f of jobs. Hence, it is easy to increase the makespan by a further K_f factor to make the schedule feasible. It follows that we have a feasible schedule with makespan at most $K_f \cdot P_k = O(3^k \cdot T)$.

It remains to bound the value of $k \leq \min\{i : f^{(i)}(T) \leq K_f\}$. Since $f(T) = \Theta(\log T)$, it follows that $k = O(\log^* T)$. Hence, we have a feasible schedule with makespan $2^{O(\log^* T)}T$, proving Theorem 3.1.

4 Homework Preview

1. **Calculation Involving** $\log^*$**.** In this question, you are asked to complete the details of some calculations.

   (a) **Deriving** K_f**.** Suppose $f(t) := 6\ln(36t^4)$. Derive a constant $K_f > 0$ such that $f(t) \geq t$ implies that $t \leq K_f$.

   (b) Suppose $k := \min\{i : f^{(i)}(T) \leq K_f\}$. Prove that $k = O(\log^* T)$.

   (Hint: Make use of big-O notation carefully and avoid messy calculations.)

2. **Packet Routing in a Graph.** We describe a problem that is closely related to job shop scheduling.

   Problem Instance. Suppose $G = (V, E)$ is a directed graph. We are given m source-sink pairs $\{(s_j, t_j) : j \in [m]\}$. We wish to send one data packet from each source to its corresponding sink.

   (a) For each $j \in [m]$, a packet must be sent from s_j to t_j via some specific path P_j. Each path P_j is simple: this means each (directed) edge appears at most once in P_j.

   (b) It takes one unit of time for a data packet to be sent through a directed edge. An edge can only allow at most 1 data packet to be sent at any time.

   Goal. Schedule the packets to be sent in the graph so that the *makespan*, which is the time for the last packet to arrive at its sink, is minimized.

   Show that the packet routing problem can be reduced to the job shop scheduling problem. In particular, given an instance of the packet routing problem, construct an instance of the job shop scheduling problem such that there exists a packet schedule with makespan T *iff* there exists a job shop schedule with makespan T.