CMPUT 675: Randomized Algorithms	**Fall 2005**
Lecture 10: Oct 11	
Lecturer: Mohammad R. Salavatipour	*Scribe: Tim Furtak*

10.1 Applications of the Lovász Local Lemma

Recall the simple LLL from last lecture.

Lemma 10.1 *Assume that $A_1, A_2, \ldots, A_n$ are bad events such that each A_i is mutually independent of all but at most d other events. If*

1. $\Pr[A_i] \le p \quad \forall i$,
2. $4pd \le 1$,

then $\Pr[\bigcap_i \overline{A_i}] > 0$

10.1.1 Packet routing in with low congestion

There is a surprising application of LLL in packet routing in general networks. Consider a network with source/sink pairs $(s_i, t_i), \quad 1 \le i \le N$. For every pair s_i, t_i, we are also given an (edge simple) path P_i from s_i to t_i. There is a packet that has to travel along P_i from s_i to t_i. We assume that we are in a synchronous system, i.e. each packet can move along an edge in one time step or stay in a queue. No two packets can travel the same edge at the same time step. We assume that there is a queue at the end of each edge and the edges that travel that edge wait in that queue until the next edge they want to travel in the next step is free. Once they travel that edge they enter the queue of that edge. There is also an initial queue at every s_i. The packet of s_i can wait in that queue for as long as we want. Our goal is to find a feasible schedule for the packets to minimize the longest travel time.

Let d (dilation) be the length of the longest path P_i and c (congestion) be the maximum number of paths using any edge in the network.

Clearly d and c are both lower bounds for the total time. So an obvious lower bound is $\Omega(c+d)$ and a loose upper bound is $c * d$, since each packet will wait at most for $c-1$ other packets along each step of the path which has at most d steps. The following interesting result was proved by Leighton, Mags, and Rao in 1994:

Theorem 10.2 (Leighton,Maggs,Rao) *There is a schedule of $O(c+d)$ time steps with constant queue size at each edge, independent of network topology, the paths P_i, and the number of packets.*

This result heavily relies on the LLL. It also has applications in Job shop scheduling. Suppose we are given jobs $j_1, \ldots, j_r$ and machines $m_1, \ldots, m_s$. Each job must be performed on a specified sequence of machines in that order, each of which takes a unit amount of time. We assume that no machine needs to work on any job twice. Our goal is to find a schedule for performing this jobs that will finish the earliest possible time. In this problem, it is easy to see that jobs correspond to the packets and machines correspond to the edges.

The dilation is the maximum number of machines needed to work on any single job and the congestion is the maximum number of jobs that need to work on any single machine. Theorem 10.2 implies that there is an schedule that takes $O(c+d)$ time only.

We start by proving a somewhat weaker result which uses only the first moment method.

Theorem 10.3 *Let N be the number of packets (paths). There is a schedule of length $O(c + d * log(dN))$*

Proof: To every packet assign an initial delay that is a u.r. selected number from $[1, \frac{\alpha c}{\log(Nd)}]$ for some constant α (to be specified). Every packet waits for this amount at the initial queue and then start going along its path without ever waiting in any queue. Of course, this may result in an infeasible schedule (because several packets may want to travel the same edge). We first bound the number of packets that want to travel the same edge at the same time. For edge $f \in E$, let A_f^t be the bad event that more than $\log(Nd)$ packets go through f at time t. Then

$$\begin{aligned}
\Pr[A_f^t] &= \sum_{k=\log(Nd)+1}^{c} \binom{c}{k} \left(\frac{\log(ND)}{\alpha c}\right)^k \left(1 - \frac{\log(Nd)}{\alpha c}\right)^{c-k} \\
&\leq \sum_{k=\log(Nd)+1}^{c} \left(\frac{ec}{k}\right)^k \left(\frac{\log(ND)}{\alpha c}\right)^k (1) \\
&\leq \sum_{k=\log(Nd)+1}^{c} \left(\frac{e\log(Nd)}{\alpha k}\right)^k \\
&\leq \left(\frac{e}{\alpha}\right)^{\log(Nd)+1} \sum_{k=0}^{c-\log(Nd)} \left(\frac{e}{\alpha}\right)^k \\
&\leq 2\left(\frac{e}{\alpha}\right)^{\log(Nd)+1},
\end{aligned}$$

because $\sum_{k=0}^{c-\log(Nd)} \left(\frac{e}{\alpha}\right)^k < \frac{1}{1-\frac{e}{\alpha}} \leq 2$ if $\alpha \geq 2e$. The number of choices for picking edge f is $\leq Nd$ and the number of choices for t is $\leq d + \frac{\alpha c}{\log(Nd)}$. Therefore:

$$\begin{aligned}
\Pr[\text{more than } \log(Nd) \text{ packets go through some edge}] &\leq 2\left(\frac{e}{\alpha}\right)^{\log(Nd)+1} \cdot Nd \cdot \left(d + \frac{\alpha c}{\log(Nd)}\right) \\
&\ll \frac{1}{Nd} \in o(1)
\end{aligned}$$

Thus, w.h.p. no more than $O(\log(Nd))$ packets travel any edge at any time. Each step of this schedule can be simulated by $O(\log(Nd))$ legitimate step by delaying the packets so that they go through an edge one by one. So the total time will be $O(c + d\log(Nd))$. ■

Now we prove a result similar to Theorem 10.2 that uses LLL.

Theorem 10.4 *There is a schedule with length $O\left((c+d)2^{O(\ln^*(c+d))}\right)$ and maximum queue size* $\log(c+d)2^{\log^*(c+d)}$.

Proof: *wlog* say $c = d$. Given an initial delay to each packet that is u.r. from $[1, \alpha d]$ for some constant α. Each packet then goes without interruption (waiting at a node) toward its destination. The total time is then $\leq (1+\alpha)d$.

Partition the time into frames, each frame having $\ln d$ steps. We show that with positive probability each edge has congestion $\le \ln d = \ln c$ in each frame. We can think of each subproblem (defined by a frame) independently of a problem with dilation $\ln d$. Repeating the same algorithm for each subproblem recursively $\ln^* d$ times we get to an instance where the time frame size is constant and so is the congestion. For such a small problem we can find a constant approximation; the naive $O(cd)$ solution is also $O(c+d)$ because c and d are constant. So the total delay is $2^{O(\ln^* d)}(c+d)$.

Therefore, we only need to prove the following (main) lemma.

Lemma 10.5 *With positive probability, every frame has congestion* $\le \ln d$

Proof: For every edge f let A_f be the bad event that f has congestion $> \ln d$ in some time frame. Our goal is to prove

$$Pr[\bigcap_{f \in E} \overline{A_f}] > 0.$$

Whether or not A_e happens only depends on the delays assigned to the packets that used edge e. Note that A_f and A_e are independent unless some packet goes through both edges. This implies that dependency $\le c * d = d^2$. What is the probability of a bad event A_f?

Consider some time frame F. The number of delays that will cause a frame to go through a particular edge f at time frame F is at most $\ln d$ (for larger or smaller delays the time at which the packet goes through f will be outside the time frame F). Thus the fraction of relevant delays is $\frac{\ln d}{\alpha d}$. Since there are at most d packets that go through each edge:

$$\begin{aligned} \text{E}[\#\text{ of packets that go through } f \text{ in some time frame } F] &\le \frac{\ln d}{\alpha d} \cdot d \\ &= \frac{\ln d}{\alpha} \end{aligned}$$

Since the paths for the packets are edge-simple, the number of packets going through f in time frame F is a binomial R.V. determined by d independent R.V.'s and $\mu = \frac{\ln d}{\alpha}$. So:

$$\begin{aligned} \Pr[\text{congestion of } f \text{ in } F \text{ is} > 1 + (\alpha - 1)\mu] &\le \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^\mu \\ &\le \left(\frac{e}{1+\delta}\right)^{(1+\delta)\mu} \\ &= \left(\frac{e}{\alpha}\right)^{\alpha\mu} \\ &= \left(\frac{e}{\alpha}\right)^{\ln d} \ll d^{-4} \quad \text{if } \alpha \text{ is large enough} \end{aligned}$$

We have $O(d)$ time frames, so

$$\Pr(A_f) \le O(d) \cdot O(d^{-4}) \in O\left(\frac{1}{d^3}\right).$$

Using the Local Lemma, with positive probability no bad events occur. ■

Therefore, this completes the proof of theorem. ■