CMSC 858S: Randomized Algorithms
Fall 2001
Handout 4: The Lovász Local Lemma and Packet Routing

Please Note: The references at the end are given for extra reading if you are interested in exploring these ideas further. You are not required to read these references for the purposes of this course.

1 The Lovász Local Lemma

A basic problem that arises often in the design and analysis of randomized algorithms is to get a good estimate (upper bound, lower bound, or both) of $\Pr(\bigvee_{i\in[m]} E_i)$, for some given events E_i. Equivalently, a good lower bound, upper bound or both, is required for $\Pr(\bigwedge_{i\in[m]} \overline{E_i})$. As we have seen, one approach would be to use the union bound ("counting sieve"), which is unfortunately quite weak in general. Another obvious approach, which works when the E_is are independent, is to use

$$\Pr(\bigwedge_{i\in[m]} \overline{E_i}) = \prod_{i\in[m]} \Pr(\overline{E_i}),$$

the "independence sieve".

A common situation where we have to look for such bounds is when the E_is are "bad" events, *all* of which we wish to avoid simultaneously. The minimal requirement for this is that $\Pr(\bigwedge_{i\in[m]} \overline{E_i}) > 0$; however, even to prove this, the independence sieve will often not apply and the counting sieve could be quite weak. However, the independence sieve does suggest something: can we say something interesting if each E_i is independent of *most other* E_j? The powerful Lovász Local Lemma (LLL) due to Erdős and Lovász ([4]) is often very useful in such cases; see Alon & Spencer [2] for many such applications. The basic version of the LLL shows that all of a set of "bad" events E_i can be avoided under some conditions:

Lemma 1.1 ([4]) *Let $E_1, E_2, \ldots, E_m$ be any events with $\Pr(E_i) \le p\ \forall i$. If each E_i is mutually independent of all but at most $D \ge 1$ of the other events E_j and if $4pD \le 1$, then*

$$\Pr(\bigwedge_{i=1}^{m} \overline{E_i}) \ge (1-2p)^m > 0.$$

Remark. Note that if $D = 0$, then we can apply the independence sieve.

The LLL is a powerful tool. We now present a portion of a surprising result on routing due to Leighton, Maggs & Rao, which makes involved use of the LLL [5]. Suppose we have a routing problem on a network, where the paths also have been specified for each of the packets. Concretely, we are given an undirected graph G, a set of pairs of vertices (s_i, t_i), and a path P_i in G that connects s_i with t_i, for each i. Initially, we have one packet residing at each s_i; this packet has to be routed to t_i using the path P_i. We assume that a packet takes unit time to traverse each edge, and the main constraint as usual is that an edge can carry at most one packet per (synchronous) time step. Subject to these restrictions, we want to minimize the maximum time taken by any packet to reach its destination. We may further assume that the paths P_i are all *edge-simple*, *i.e.*, do not repeat any edge. Since we are also given the paths P_i, the only question is the queuing discipline we need to provide at the nodes/links (what a node must do if several packets that are currently resident at it, want to traverse the same edge on their next hop).

One motivation for assuming such a model (wherein the paths P_i are prespecified) is that many routing strategies can be divided into two phases: (a) choosing an intermediate destination for each packet (*e.g.*, the paradigm of choosing the intermediate destination randomly and independently for each packet [8, 9]) and taking each packet on some canonical path to the intermediate destination, and (b) routing each packet on a canonical path from the intermediate destination to the final destination t_i. Such a strategy can thus use two invocations of the above "prespecified paths" model.

Let us study the objective function, *i.e.*, the maximum time taken by any packet to reach its destination. Two relevant parameters are the *congestion* c, the maximum number of paths P_i that use any given edge in G, and the *dilation* d, the length of a longest path among the P_i. It is immediate that each of c and d is a lower bound on the objective function. In terms of upper bounds, the simple greedy algorithm that never lets an edge go idle if it can carry a packet at a given time step, terminates within cd steps; this is because any packet can be delayed by at most $c-1$ other packets at any edge in the network.

Can we do better than cd? Recall our assumption that the paths P_i are all *edge-simple*. Under this assumption, the work of [5] shows the amazing result that there exists a schedule of length $O(c+d)$ with *constant* queue-sizes at each edge, independent of the topology of the network or of the paths, and of the total number of packets! The proof makes heavy use of the LLL, and we just show a portion of this beautiful result here.

Henceforth, we assume without loss of generality that $c = d$, to simplify notational cluttering such as $O(c+d)$. (However, we also use both c and d in places where such a distinction would make the discussion clear.) Imagine giving each packet a random initial delay, an integer chosen uniformly at random and independently from $\{1, 2, \ldots, ac\}$ for a suitable constant $a > 1$. We think of each packet waiting out its initial delay period, and then traversing its path P_i without interruption to reach its delay. Of course this "schedule" is likely to be infeasible, since it may result in an edge having to carry more than one packet at a time step. Nevertheless, the LLL can be used to show that some such "schedules" exist with certain very useful properties, as we shall see now.

The above (infeasible) schedule clearly has a length of at most $ac + d$. Let us partition this period into *frames*, contiguous time intervals starting at time 1, with each frame having a length (number of time steps) of $\ln c$. Our basic idea is as follows. We shall prove, using the LLL, that every edge has a congestion of at most $\ln c$ in each such frame, with positive probability. Suppose indeed that this is true; fix such a choice for the initial delays. Then, we would have essentially broken up the problem into at most $(ac+d)/\ln c$ subproblems, one corresponding to each frame, wherein the congestion and the dilation are at most $\ln c$ in each subproblem. Furthermore, we can solve each subproblem recursively and independently, and "paste together" the resulting schedules in an obvious way. Finally, the facts that:

- the congestion and dilation go from d to $\ln d$, and
- a problem with constant congestion and dilation can be scheduled in constant time (*e.g.*, by the above-seen greedy protocol),

will guarantee a schedule of length $(c+d) \cdot 2^{O(\log^*(c+d))}$ for us.

We now use the LLL to prove that every edge has a congestion of at most $\ln c$ in each frame, with positive probability. For any given edge f, let E_f denote the event that there is some frame in which it has a congestion of more than $\ln c$. We want to show that $\Pr(\bigwedge_f \overline{E_f}) > 0$.

For any given edge f and any given frame F, let $E'(f, F)$ denote the event that the congestion of f in F is more than $\ln c$. It is easy to see that the expected congestion on any given edge at any given time instant, is at most $c/ac = 1/a$, in our randomized schedule. Thus, the expectation of the congestion $C(f, F)$ of f in F is at most $(\ln c)/a$. Now, crucially, using our assumption that the paths P_i are *edge-simple*, it can be deduced that $C(f, F)$ is a sum of *independent* indicator random variables. Thus, by the Chernoff-Hoeffding bounds, we see that $\Pr(E'(f, F)) \leq F^+((\ln c)/a, a-1)$. So, by the union bound,

$$\Pr(E_f) \leq \sum_F \Pr(E'(f, F)) \leq O(c+d) \cdot F^+((\ln c)/a, a-1),$$

i.e., a term that can be made the reciprocal of an arbitrarily large polynomial in c, by just choosing the constant a large enough. To apply the LLL, we also need to upper-bound the "dependency" among the E_f, which is easy: E_f "depends" only on those E_g where the edges f and g have a common packet that traverses both of them. By our definitions of c and d, we see that each E_f "depends" on at most $cd = c^2$ other events E_g; combined with our above upper-bound on each $\Pr(E_f)$, the LLL completes the result for us (by choosing a to be a suitably large positive constant). Sophisticated use of similar ideas leads to the main result of [5]–the existence of a schedule of length $O(c+d)$ with constant-sized queues.

An interesting point is that the LLL usually only guarantees an extremely small (though positive) probability for the event being shown to exist. In the notation of Lemma 1.1, $p \gg 1/m$ in many applications, and hence the probability upper bound of $(1-2p)^m$ is tiny, though positive. (For instance, in the applications of the LLL in the above routing result, p could be $(c+d)^{-\Theta(1)}$ while m could be $\Omega(N)$, where N is the total number of packets; note that c and d could be very small compared to N.) In fact, it can be proven in many applications of the LLL, that the actual probability of the desired structure occurring is very small. Thus, the LLL is often viewed as proving the existence of a *needle in a haystack*; this is in contrast to the much weaker union bound, which, when applicable, usually also gives a good probability of existence for the desired structure (thus leading to an efficient randomized or even deterministic algorithm to produce the structure). Breakthrough work of Beck [3] showed polynomial-time algorithms to produce several structures guaranteed to exist by the LLL; parallel algorithms to do this were then exhibited by Alon [1]. While these and other later results have been used to constructivize many known applications of the LLL, some applications still remain elusive (non-constructive).

For the above specific problem of routing using prespecified paths, polynomial-time algorithms have been presented in [6]. However, routing problems often need to be solved in a *distributed* fashion where each packet knows the network as well as its source and destination, and where there is no centralized controller who can guide the routing; significant progress toward such a distributed algorithmic analog of the above result of [5], has been made in [7].

References

[1] N. Alon. A parallel algorithmic version of the Local Lemma. *Random Structures & Algorithms*, 2:367–378, 1991.

[2] N. Alon and J. H. Spencer. *The Probabilistic Method, Second Edition.* Wiley–Interscience Series, John Wiley & Sons, Inc., 2000.

[3] J. Beck. An algorithmic approach to the Lovász Local Lemma. *Random Structures & Algorithms*, 2:343–365, 1991.

[4] P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and Finite Sets*, A. Hajnal et. al., editors, Colloq. Math. Soc. J. Bolyai 11, North Holland, Amsterdam, pages 609–627, 1975.

[5] F. T. Leighton, B. Maggs, and S. Rao. Packet routing and jobshop scheduling in $O(\text{congestion} + \text{dilation})$ steps. *Combinatorica*, 14:167–186, 1994.

[6] F. T. Leighton, B. Maggs, and A. Richa. Fast algorithms for finding $O(\text{congestion} + \text{dilation})$ packet routing schedules. *Combinatorica*, 19:375–401, 1999.

[7] R. Ostrovsky and Y. Rabani. Universal $O(\text{congestion} + \text{dilation} + \log^{1+\epsilon} N)$ local control packet switching algorithms. In *Proc. ACM Symposium on the Theory of Computing*, pages 644–653, 1997.

[8] L. G. Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing*, 11:350–361, 1982.

[9] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Proc. ACM Symposium on the Theory of Computing*, pages 263–277, 1981.