6.842 Randomness and Computation October 18, 2017

Lecture 12

Lecturer: Ronitt Rubinfeld *Scribe: John Napp*

1 Undirected S-T Connectivity in Deterministic Log-Space

We have previously seen how to solve Undirected S-T Connectivity in randomized log-space. In this lecture, we will see a deterministic log-space algorithm for the problem. While the randomized algorithm was shown in 1979, the deterministic algorithm wasn't discovered until 2005 by Reingold. These notes will not contain full details and proofs. For those, consult the original paper [1] or (for example) the surveys in [2] or [3]. First, let's recall the computational problem.

Computational Problem 1 Undirected S-T Connectivity. *Given a graph G and two nodes s and t, are s and t in the same component?*

The high-level strategy for the deterministic algorithm will be to reduce the problem to an equivalent problem in which the components of the input graph are good expanders. This reduction will be done by performing operations on the original graph. These operations will be "local" in some sense, which allow them to be performed on-the-fly using only logarithmic space. It will then be easy to describe a deterministic log-space algorithm that determines the connectivity of s and t in this new, highly expanding graph.

We will first recall some facts from spectral graph theory that we will need.

1.1 Graph theory facts

Definition 1 *Call a graph G (N, D, λ) if it has N nodes, a maximal degree of D, and λ is an upper bound on the second-largest absolute value of an eigenvalue of the transition matrix of G.*

Now we recall a result relating the spectral expansion of a graph to its vertex expansion.

Theorem 2 (Tanner [4], Alon-Milman [5]) *For all $\lambda < 1$, there exists an $\epsilon > 0$ such that for every (N, D, λ)-graph G and every subset $S \subset G$ such that $|S| < N/2$, $|N(S)| \geq (1+\epsilon)|S|$.*

We will also need the fact that connected, non-bipartite graphs have spectral gap lower bounded by 1/poly(n) (the spectral gap is just defined as $1 - \lambda$). More formally,

Theorem 3 *Let G be a connected, non-bipartite graph with degree D. Then*

$$\lambda(G) \leq 1 - \frac{1}{DN^2}.$$

1.2 A deterministic algorithm for expanders

Consider the case that the input graph G comes with the promise that each of its components is a D-regular expander for some constant D, by which I mean the spectral gap of each component is lower bounded by a constant. In this case, it is not hard to show that a deterministic algorithm for Undirected S-T Connectivity exists. To see this, first note that the diameter of the components of such a graph is $O(\log N)$. This is a consequence of Theorem 2. Now, we can solve the computational problem by starting at node s and enumerating over all paths of length $O(\log N)$. Output "yes" if we ever see t, and "no" otherwise. One can check that this can be done with $O(\log N)$ space.

Theorem 3 gives us an inverse-polynomial lower bound on the spectral gap of general input graphs, but unfortunately it's not strong enough to invoke the above result. However, our strategy will be to perform operations on the input graph to transform it into a graph of the above form without changing the connectivity of s and t, at which point we can invoke the above result.

Next, we review the graph operations we will need to use.

1.3 Graph operations

Recall that given an input graph G, we'd like to boost its spectral gap to $\Omega(1)$ while keeping its degree constant and only increasing its size to at most poly(N). (We will see later that the degree of the input graph can be assumed to be O(1) wlog.) There are essentially two primitives we will use for this purpose. *Powering* will improve the expansion at the cost of increasing the degree. Taking a *Zig-Zag Product* will greatly reduce the degree without making the expansion too much worse, at the cost of making the graph larger. It turns out that by repeatedly applying these operations, we can turn any graph G into a modified graph of size poly(N) whose components are expanders of constant degree, as desired.

1.3.1 Powering

If a graph G has associated transition matrix M, then G^k is defined to be the graph whose transition matrix is M^k. So for example, squaring a graph corresponds to adding edges to next-nearest-neighbors. It's not too hard to show that if G is (N, D, λ), then G^k is (N, D^k, λ^k). Hence we see that powering can exponentially suppress λ at the cost of exponentially increasing the degree. In the setting of our problem, we see that raising the input graph to the $O(\log N)$ power will amplify the spectral gap to $O(1)$. Unfortunately, this causes the degree to become poly(N), which is unacceptable.

1.3.2 Replacement product

As a warmup for the zig-zag product that we will actually use, we first consider a related construction, the replacement product $\textcircled{r}$. In the below construction, D should be thought of as much larger than d, and H should be thought of as having good expansion.

Definition 4 *(Replacement Product). Let G be a D-regular graph with N nodes, and let H be a d-regular graph with D nodes. Then the replacement product of G and H, $G \textcircled{r} H$, is the graph constructed as follows. Replace every vertex of G with a copy of H. Let H_v denote the copy of H corresponding to $v \in G$. Place an edge from the ith node in H_v to the jth node in H_w if and only if w is the ith neighbor of v in G, and v is the jth neighbor of w in G.*

Clearly the resulting graph has ND nodes. We also see that it has degree $d+1$, since each node in $G \textcircled{r} H$ is connected to d nodes within its "cloud", as well as one additional node in a different cloud. Using this construction, the degree of the resulting graph will be $d+1$ regardless of how large D is. The point of the construction is that the degree can be greatly reduced without making the expansion of the original graph G too much worse.

As some intuition for this fact, say we are trying to find a cut in the graph which minimizes expansion across the cut. If H is a good expander, we won't want to break up the clouds H_v too much in trying to do this minimization. On the other hand, if G is a good expander, then even keeping the clouds H_v intact will lead to many edges across a given cut.

Here is another way to get intuition for this fact in terms of random walks. Say we start with some distribution on the nodes of $G \textcircled{r} H$ which is far from the stationary, uniform distribution, and we ask how long it takes for the distribution to get close to uniform. If the starting distribution is far from uniform, it must be the case that the distribution is unevenly distributed between the clouds, or the distribution is far from uniform within each cloud. But the intra-cloud distributions will quickly approach uniformity due to the expansion of H, and the inter-cloud distributions will quickly approach uniformity due to the expansion of G. So a walk on $G \textcircled{r} H$ should mix rapidly, which is associated with good expansion.

1.3.3 Zig-Zag product

The graph operation that we will actually use is a bit different than the replacement product, but similar in spirit.

Definition 5 *(Zig-Zag product). Let G be a D-regular graph with N nodes, and let H be a d-regular graph with D nodes. Then the Zig-Zag product of G and H, $G \textcircled{z} H$, is a graph whose vertices are pairs $(u, i) \in [N] \times [D]$. For $a, b \in [d]$, define the (a, b)th neighbor of (u, i) as follows.*

Let i' be the ath neighbor of i in H. Let v be the i'th neighbor of u in G. Define j' to be the integer such that u is the j'th neighbor of v in G. Let j be the bth neighbor of j' in H. Then the (a, b)th neighbor of (u, i) is defined to be (v, j).

Clearly the size of the resulting graph is ND, and the degree is d^2. To get some intuition for this construction, note that a random walk on $G \textcircled{z} H$ corresponds to first taking a random step in H, then taking a step in G corresponding to where we landed in H, then taking another random step in H. So in a sense, a random walk on $G \textcircled{z} H$ is like a modified random walk on G where the steps in G are controlled by a random walk in H. If H is a good expander, we expect this modified random walk on G to behave similarly to a random walk on G which chooses its next step uniformly at random. Hence, if G and H both have good expansion, we expect the walk on $G \textcircled{z} H$ to mix in time not much larger than the time it takes a walk on G to mix, and so we expect $G \textcircled{z} H$ to have good expansion. Note that we need the second random step in H in the definition to ensure that if (v, j) is a neighbor of (u, i), then (u, i) is a neighbor of (v, j) and hence the resulting graph can be defined as undirected.

From this intuition and the arguments in the previous section concerning the closely-related replacement product, we suspect that if G and H are good expanders, then $G \textcircled{z} H$ is still a pretty good expander despite the (potentially very large) reduction in degree. More precisely,

Theorem 6 *If $\alpha \leq 1/2$, G is (N, D, λ), and H is (D, d, α), then $G \textcircled{z} H$ is $(N \cdot D, d^2, \lambda_{G \textcircled{z} H})$ where $\lambda_{G \textcircled{z} H} \leq \frac{2}{3} + \frac{\lambda}{3}$.*

1.4 Main transformation

Now we put together the pieces. We will need the following fact.

Theorem 7 *For some constant D, there exists a $(D^{16}, D, \frac{1}{2})$-graph.*

Let H denote the $(D^{16}, D, \frac{1}{2})$-graph in the above theorem, and let G denote the input graph. We assume that G is D^{16}-regular. This is wlog, because we can do preprocessing on G to put it into this form. For example, we can increase the degree by adding self-loops. To decrease the degree, we could replace a high-degree vertex by a cycle of vertices which have the proper degree.

Let G_0 be the preprocessed input graph. Now define $G_i \leftarrow (G_{i-1} \textcircled{z} H)^8$. It is easy to check that all G_i have degree D^{16}. Recall that powering exponentially suppresses λ, while taking the Zig-Zag product doesn't change it by much. Each component of G_0 has $\lambda = 1 - 1/\text{poly}(N)$. Hence, we need to iterate this procedure $l = O(\log N)$ times to make each component of G_l have $\lambda = O(1)$. Further, one can check that this transformation preserves the connected components of G. Finally, note that the size of the final graph is $N \cdot D^{16l} = \text{poly}(N)$. Hence, we have reduced the problem to the case discussed in Section 1.2, which we know how to solve deterministically.

Of course, in order for this to work in log-space, it must be possible to perform all of the transformations "on-the-fly"! We leave it to the reader to convince themself that this is indeed possible.

References

[1] O. Reingold, "Undirected connectivity in log-space," Journal of the ACM, vol. 55, no. 4, pp. Art 17, 24, 2008.

[2] S. Arora and B. Barak, "Computational complexity: a modern approach," Cambridge University Press, 2009.

[3] S. Vadhan, “Pseudorandomness,” Now Publishers Inc., 2012.

[4] M. R. Tanner, “Explicit concentrators from generalized N-gons,” SIAM Journal on Algebraic Discrete Methods, vol. 5, no. 3, pp. 287?293, 1984.

[5] N. Alon and V. D. Milman, “Eigenvalues, expanders and superconcentrators,” in Annual Symposium on Foundations of Computer Science, pp. 320–322, Singer Island, Florida, 24–26 October 1984.