CS 880: Advanced Complexity Theory 3/20/06

Lecture 18: Undirected ST-Connectivity in Deterministic Log-Space

Instructor: Dieter van Melkebeek Scribe: Bess Berg and Jurgen Van Gael

The next application of expanders we will see is a recent result by Omer Reingold: undirected st-connectivity (USTCON) is in deterministic logspace. Results prior to this had USTCON in as low as deterministic $O(\log^{4/3})$-space, and a very simple algorithm shows that USTCON can be done in *randomized* logspace. In 2004, however, a nice application of the zigzag product allowed Reingold to convert any input graph into an expander in a space-efficient way, such that the resulting graph can be deterministically searched for an st-path, and, as we shall see next, giving the desired result USTCON $\in L$.

1 A randomized approach

Given the adjacency matrix for an undirected graph $G = (V, E)$ and vertices s and t, we wish to determine if there is a path from s to t. One randomized approach is simply to take a random walk on G starting from s; walk for a polynomial number of steps, and if the walk ever hits t, accept, and otherwise reject. We only need to store the current vertex, the next vertex, and the number of steps taken so far, and therefore this approach requires only logarithmic space. The algorithm has one-sided error (if s and t are not connected we will always reject; if they are connected, we will accept only with some probability) which we need to argue can be made sufficiently small with only a polynomial number of steps in the walk. The idea is that when we walk from s, after a polynomial number of steps the probability distribution over the vertices in the connected component containing s will be very close to uniform. Then, if t is in this connected component, we will have probability $1/\Theta(N)$ of hitting t at this point, and a linear number of repetitions will give us a constant probability of discovering t.

More formally, let A be the normalized adjacency matrix for G, let $C_s \subseteq V$ be the vertices in the connected component of G containing s, and let u be the uniform distribution over C_s. For this analysis we assume without loss of generality that C_s is d-regular for d polynomial in n, since we can add self-loops to ensure this without affecting connectivity (recall that we allow multiple edges and so the polynomial upperbound on d is relevant). We also assume that the spectrum of the subgraph induced by C_s is non-negative, since we can square the graph to achieve this without affecting connectivity. Now, for any initial distribution π on C_s, we are concerned with upperbounding the value

$$\|A^\ell \pi - u\|_\infty,$$

which tells us how far off from $1/|C_s|$ the weight on any particular vertex in C_s can be after ℓ steps of the walk. We upper bound this quantity with

$$\begin{aligned}
\|A^\ell \pi - u\|_\infty &\leq \|A^\ell \pi - u\|_2 \\
&= \|A^\ell(\pi - u)\|_2 \\
&\leq \lambda_2^\ell \|\pi - u\|_2 \quad \text{(since } \pi - u \perp u) \\
&\leq \lambda_2^\ell \|\pi\|_2 \quad \text{(also since } \pi - u \perp u) \\
&\leq \lambda_2^\ell \|\pi\|_1 \\
&= \lambda_2^\ell \\
&\leq \left(1 - \frac{1}{d|C_2|^2}\right)^\ell \qquad \text{(by Homework 1, Problem 2b)} \\
&\leq \left(1 - \frac{1}{dN^2}\right)^\ell \\
&\leq \frac{1}{2N},
\end{aligned}$$

where the very last inequality holds provided $\ell = \Omega(dN^2 \log N)$. In particular, if we start from s then after ℓ steps of the random walk the probability that we are on t (assuming $t \in C_s$) is at least $1/N - 1/2N = 1/2N$. Running this process $O(N)$ times will be sufficient to hit t with constant probability.

¿From the above we see that it is not too hard to put USTCON in randomized logspace. But how can we get rid of this randomness without using extra space? One approach would be to derandomize the above algorithm, but currently no one knows if this can in fact be done. Instead, we will gather inspiration from Savitch's proof that USTCON is in deterministic log^2-space and see how Reingold used an idea like this to give a deterministic, logspace algorithm for USTCON.

2 A deterministic approach after Savitch

Recall Savitch's algorithm for USTCON: essentially, we space-efficiently transform the input graph so as not to affect the connectivity of s and t, but so that the components become highly connected and therefore it is easy to determine if there is a path from s to t. In particular, we can think of Savitch's algorithm as repeatedly squaring the graph (dropping multiple edges after each squaring) until each connected component is a clique. So we have

$$G_{i+1} \leftarrow G_i^2, \quad i = 1, 2, \ldots, \ell,$$

where $\ell = \log N$. The space needed for the step $G_{i+1} \leftarrow G_i^2$ is $\Theta(\log d(G_i))$, where $d(G_i)$ is the degree of graph G_i. We can do some preprocessing to ensure that $d(G_1) = 3$ by letting G_1 be the input graph G but with every vertex v replaced by a cyle of length $d(v)$ in the obvious way. The degree will at most square with each step, that is, $d(G_{i+1}) \leq d(G_i)^2$ (this would be an equality except that we are getting rid of multiple edges after each squaring). Then the log of the degree will at most double at each step, and therefore the algorithm requires a total of $O(\log^2 N)$ space.

Reingold achieves something better with the zigzag product, using the transformation

$$G_{i+1} \leftarrow G_i^k \text{ⓩ} H,$$

for some fixed k and H such that $N(H) = d(G_i^k) = d(H)^{2k}$. The idea is that after a certain number of steps every connected component will become an expander with constant degree. Notice this transformation acts independently on each connected component. Also notice that since the input graph G is not necessarily an expander, we have no guarantees on its spectral gap, unlike the case where we showed how to explicitly construct expanders. Because of this we may need to have $k > 2$, since we may need to reduce λ_2, not merely maintain it.

There are two main ingredients to the success of this transformation, as we shall see in more detail next. The first is that it preserves st-connectivity (what this even means must be specified). The second is that it maintains a constant degree. Savitch's algorithm shares this first property, but not the second, and therefore can only put USTCON in space $O(\log^2 N)$.

3 Reingold's Algorithm

Next we show that Reingold's transformation achieves the necessary goals in order to perform an exhaustive search on the transformed graph.

First of all we want to ask ourselves whether st-connectivity is maintained. This is not entirely trivial since the zig-zag product transforms one vertex into a cloud and therefore introduces some ambiguity in how we should interpret s and t. We can maintain st-connectivity though by interpreting the property that s, t are connected in G_i iff any vertex in the cloud of s and any vertex in the cloud of t are connected.

Next, we want to argue that the transformation increases the connectivity of the graph (which we measure using the spectral gap). Remember from the zig-zag product theorem that $\lambda_2(G \textcircled{z} H) \leq g(\lambda_2(G), \lambda_2(H))$ with $g(x, y) \leq x + y + y^2$. However, since we are starting from a graph G which might not even be close to an expander, $\lambda_2(G)$ could be $1 - \frac{1}{p(N)}$ (for $p(N)$ some polynomial in N, the number of vertices of G), and $\lambda_2(H)$ is some constant, this inequality does not guarantee that we can get a bound on $\lambda_2(G \textcircled{z} H) < 1$. Luckily, we can show that for $y \leq 1/2$ we have $1 - g(x, y) \geq 1/3(1 - x)$ which in terms of the recursion means that,

$$\lambda_2(G_{i+1}) \leq 1 - \frac{1}{3}(1 - \lambda_2(G_i^k)) = \frac{2}{3} + \frac{1}{3}\lambda_2(G_i)^k. \tag{1}$$

We still have the parameter k left which we can use to make sure that $\frac{2}{3} + \frac{1}{3}\lambda_2(G_i)^k \leq \lambda_2(G_i)^2$ for $\lambda_2(G_i) \geq \gamma_k$ where $\gamma_k < 1$ is some constant. In other words, we want to know for which k $x^k \leq 3x^2 - 2$ when $\gamma_k \leq x \leq 1$. From figure 1 we know that this is equivalent to asking for which k the slope of x^k is larger than the slope of $3x^2 - 2$ (being $6x$). We find that $k > 6$ suffices.

Finally, we need to figure out how many times we must run the iterative transformation. As argued before, we will assume w.l.o.g. that the graph G we are starting from has degree 3. Since the zig-zag product requires us to have a starting graph of degree $d(H)^2$, we must add self loops to the vertices. An additional advantage of doing this is that the graph cannot be bipartite and thus has a positive spectrum. Since we replaced every vertex v in G by a cycle of length d_v in G_1, we have that $N(G_1) = \sum_{v \in V} d_v = 2E(G)$. Finally, we have that for any component C of G_i,

$$\lambda_2(C_{i+1}) \leq \max(\gamma_k, \lambda_2(C_i)^2). \tag{2}$$

For the initial component C we have (again applying the results of Homework 1, Problem 2b),

$$\lambda_2(C) \leq 1 - \frac{1}{d(H)^2 \cdot N(C)^2}. \tag{3}$$

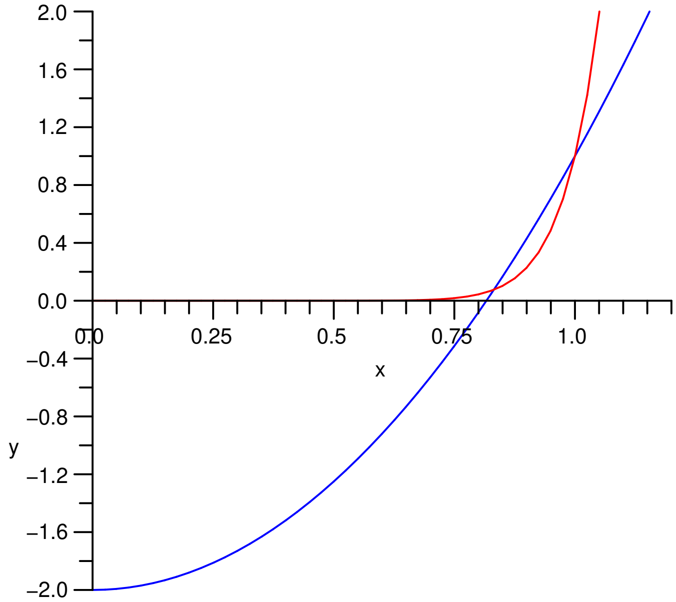

Figure 1: Plot of $3x^2 - 2$ and x^k for $k > 7$.

It follows that $O(\log(n))$ iterations suffices to get $\lambda_2(C_i) \leq \gamma_k$. What about the space requirements for the transformation? Remember our discussion about the zig-zag product that we can compute the rotation map of the squaring operation in constant space. If we reuse this space for all $O(k)$ evaluations, we only need constant space to compute the rotation map of G_i^k. Moreover, since we are taking a zig-zag product with a constant sized graph, we only need constant extra space to compute the rotation map of the zig-zag product. There is a catch however, the squaring algorithm might introduce parallel edges between two vertices. If these are not deleted from the graph, repeated squaring might increase the degree of some vertices exponentially. Since this would require a linear amount of space to describe, we should remove parallel edges after the squaring operation. We can thus conclude that since we only require a constant amount of space in each iteration, our overal space requirement is logarithmic.

We now briefly summarize Reingold's algorithm for st-connectivity in log-space:

1. Transform the input graph into G_1 (make degree constant, add self loops)
2. Transform G_1 into G_l
3. Exhaustively search all paths of length $\Theta(\log(n))$ from the cloud corresponding to s to the cloud corresponding to t.