Lecture 9 (17.05.2013) Scribe: Marcin Wrochna

Random walks, applications of expanders: SL=L

In this lecture we show an important proof that uses expanders algorithmically to derandomize random walks.

Theorem 1 (2004 Reingold)**.** *The connectivity of an undirected graph can be decided in logarithmic space.*

The theorem can be stated for two closely related problems: connectivity and STCON, the problem of finding a path from a given vertex s to t in a graph (USTCON in undirected graphs). The class of problems solvable in logarithmic space is called L or LOGSPACE, the class of problems reducible to USTCON is called SL. so the theorem is often stated as USTCON $\in$ LOGSPACE or L = SL.

Both problems in directed graphs (STCON and strong connectivity) have long been known to be NL-complete and are thus believed not to be in L. The undirected cases are also easily seen to be in NL (progressively guess any route and stop after n steps), by Savitch's theorem (or any simple algorithm that comes to mind) they can be decided in $O(\log^2 n)$ space. A randomized algorithm that performs a random walk for n^3 steps was also known, but nevertheless Reingold's theorem was a breakthrough, as many believed SL to be unequal to L.

Logarithmic space allows only to remember a fixed number of counters and pointers to vertices.

1 Modifying the graph

First, to make the graph 3-regular, we replace each vertex v with a cycle of length $\deg(v)$, each cycle vertex incident to one original edge – reachability/connectivity is preserved. To make the graph d-regular, for any d, simply add loops to all vertices – loops will also be useful later, because we can talk about paths of length 'exactly k', instead of 'at most k'. Like in all graph modifications here, we don't actually write the new graph, we only say we'll use tuples to represent new vertices (here: pairs (v, i) for $v \in V(G)$, $i \leq \deg(v)$) and notice that new edges can be easily enumerated.

We choose some d and a fixed graph H, hard-coded in the algorithm, satisfying $|H| = d^{16}$, d-regular and $\lambda(H) \leq \frac{d}{2}$ (existence was proved on exercises). We take G_0 to be G modified as described above to be d^2-regular (G_0 has $n = |G_0| = 2|E(G)|$ vertices, which is polynomial in the size of the input graph G). Define

$$G_{t+1} = G_t^8 ⓩ H$$

for $t = 1, \ldots, L$, where L is such that $\left(1 - \frac{1}{d^{16}n^2}\right)^{2^L} < \frac{1}{2}$ (so L is $\mathcal{O}(\log n)$). Inductively, G_t is d^2-regular, G_t^8 is d^{16}-regular, so the zig-zag product with H is well defined, giving a graph of degree d^2. The number of vertices is $|G_{t+1}| = |G_t^8| \cdot |H| = |G_t| \cdot d^{16}$, so $|G_L| = d^{16L}|G_0|$, which still is polynomial in n.

We'll show how to actually describe and walk the graph later (the idea is that it'll suffice to search all paths of length $\log n$, which can be done in L with some effort), for now let's focus on the expansion obtained in the construction.

Lemma 2. *If G is connected, G_L is a good expander, namely $\lambda(G_L) < (1 - \varepsilon)d^2$ for some fixed ε independent of $|G|$. Otherwise G_L is not connected either.*

Proof. G_0 is a d^2-regular, connected graph with loops in every vertex, so (from exercise 97)

$$\lambda(G_0) \leq d^2 \left(1 - \frac{1}{d^2 n^2}\right)$$

$$\lambda(G_0^8) \leq d^{16} \left(1 - \frac{1}{d^{16} n^2}\right)$$

We use the fact (cited in the previous lecture, and proved in a weaker version) that, for a D-regular graph G and d-regular graph H,

$$1 - \frac{\lambda(G ⓩ H)}{d^2} \geq \left(1 - \frac{\lambda(G)}{D}\right) \cdot \frac{1 - \left(\frac{\lambda(H)}{d}\right)^2}{2}$$

For $\lambda(H) \leq \frac{d}{2}$ the factor $\frac{1-\left(\frac{\lambda(H)}{d}\right)^2}{2}$, which describes how much we loose in the spectral gap, is at least $\frac{3}{8}$ – the main point of the zig-zag product is that it lowers the degree and doesn't lower the gap too much.

When $\lambda(G_t^8)$ is already less than half the degree, $\lambda(G_t^8) \leq \frac{1}{2} \cdot d^{16}$, so are later $G_{t'}^8$s, because $\lambda(G_t^8 ⓩ H) \leq d^2(1 - \frac{3}{8} \cdot \frac{1}{2}) = d^2 \cdot \frac{13}{16}$, and then $\lambda((G_t^8 ⓩ H)^8) \leq d^{16} \left(\frac{13}{16}\right)^8 < \frac{1}{2} \cdot d^{16}$.

On the other hand, when $\lambda(G_t^8)$ is not yet less than half the degree, let $x = 1 - \lambda(G_t^8)/d^{16} \in (0, \frac{1}{2})$. On this range it is easy to prove that $(1 - \frac{3}{8}x)^4 \leq 1 - x$, so taking the 4th power would already be enough to return to the original gap, but we square it once more (taking it to the 8th power):

$$1 - \lambda(G_t^8)/d^{16} = x$$

$$1 - \lambda(G_t^8 ⓩ H)/d^2 \geq \frac{3}{8}x$$

$$\lambda\left((G_t^8 ⓩ H)^4\right)/d^8 \leq (1 - \frac{3}{8}x)^4 \leq 1 - x = \lambda(G_t^8)/d^{16}$$

$$\lambda(G_{t+1}^8)/d^{16} = \lambda\left((G_t^8 ⓩ H)^8\right)/d^{16} \leq \left(\lambda(G_t^8)/d^{16}\right)^2$$

Inductively,

$$\lambda(G_t^8)/d^{16} \leq \max\left(\frac{1}{2}, \left(1 - \frac{1}{d^{16} n^2}\right)^{2^t}\right)$$

Thus $\lambda(G_L^8)/d^{16} \leq \frac{1}{2}$. □

2 Walking algorithm

Since we proved $\lambda(G_L)$ to be separated from d^2 by a constant factor, we know (from exercise 98) that the diameter of G_L is at most $\mathcal{O}(\log n)$ and it suffices to check all routes of that length. The description of one vertex $\tilde{s} = (s, p_1, p_2, \ldots, p_L)$ in G_L consists of the description of a vertex in G, which takes logarithmic space, and L descriptions of vertices in H, each taking constant space. To enumerate neighbors of one vertex we only need $\lg d^2 = O(1)$ bits. A naive approach would be to make a simple DFS:

Procedure 1 DFS($\tilde{v}, i$)

```
if i == 0 then
  return  ṽ == t̃
else
  result := false
  for all w̃ neighbors of ṽ in G_L do
    result |= DFS(w̃, i − 1)
  end for
  return  result
end if
```

Started with DFS($\tilde{s}, \log n$) and a global variable $\tilde{t}$ for the end vertex, this program would put $\log n$ stack frames of size $O(\log n)$ each – as we noticed earlier, a program running in $\log^2 n$ space is easy to obtain. Instead, it is enough to maintain the current vertex and depth in one global variable, and remember the returning edge index on each stack frame (every time we move to a vertex, remember which edge, as a number $1 \leq \gamma \leq d^2$, we should use to return). Let $\mathrm{GO}_i(\beta)$ in vertex $\tilde{v}$ and depth i be a procedure moving to the β-th neighbor of $\tilde{v}$ and returning the returning edge γ (described later). The following program uses only $O(1)$ memory per stack frame to remember the neighbor numbers ($\leq d^2$).

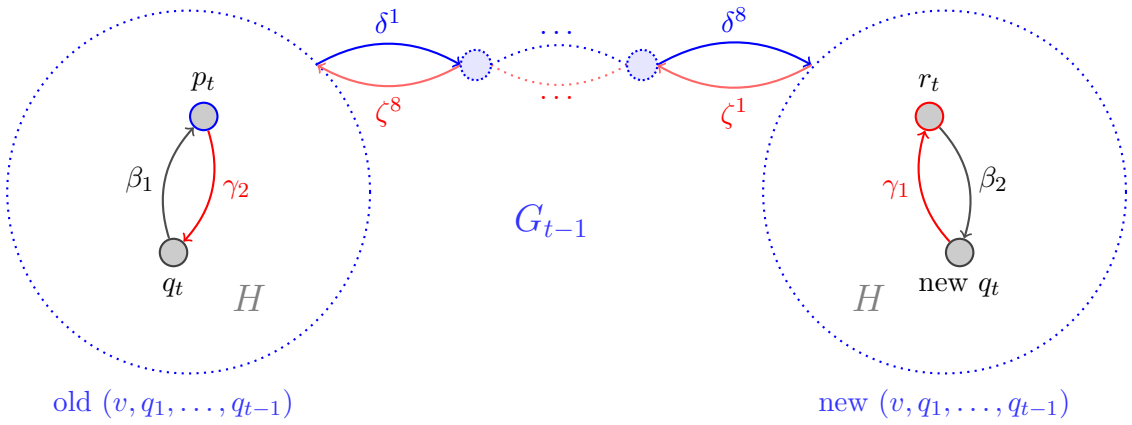
Procedure 2 DFS'

```
if i == 0 then
  return  ṽ == t̃
else
  result := false
  i := i - 1
  for all β ∈ 1 ... d^2 do
    γ := GO_L(β)
    DFS'()
    GO_L(γ)
  end for
  i := i + 1
end if
```

It remains to implement $\mathrm{GO}_t(\beta)$ in logarithmic space – this procedure walks the graph G_t by recursively calling GO_{t-1}. It allocates $O(1)$ memory one every stack frame, for a total of $O(\log n)$.

Procedure 3 GO(β)

Interpret $\tilde{v}$ as $(v, q_1, \ldots, q_t, \ldots, q_L) \in V(G) \times V(H) \times \cdots \times V(H)$
if t == 0 **then**
(simply walk in G)
$w :=$ the β'th neighbor of v in G
$\gamma :=$ returning edge index (such that v is the γ-th neighbor of w)
$v := w$
return γ
else
(walk in H, then G_{t-1}, then H, as in the definition of $G_t = G_{t-1}^8 \circledcirc H$)
Interpret $\beta \in 1 \ldots d^2$ as two numbers $\beta_1, \beta_2 \in 1 \ldots d$
$p_t :=$ the β_1-th neighbor of q_t in H
$\gamma_2 :=$ the returning edge index
Interpret p_t as 8 neighbor indices $(\delta^1, \ldots, \delta^8)$ in G_{t-1} (a neighbor index in G_{t-1}^8).
t := t - 1
for $j := 1 \ldots 8$ **do**
$\zeta^{9-j} := GO(\delta^j)$
end for
t := t + 1
Interpret the 8 returning edge indices $(\zeta^1, \ldots, \zeta^8)$ as a vertex r_t in H
(r_t is the vertex we arrived in)
$q_t :=$ the β_2-th neighbor of r_t in H
$\gamma_1 :=$ the returning edge index
return (γ_1, γ_2)
end if



3 Lower bound for expansion

We show in exercise 103 a construction of an expander graph that can be used for H in our algorithm: for any q we have a q-regular graph H_q with q^2 vertices and $\lambda(H_q) \sim \sqrt{q}$. The following theorem shows we cannot hope for much more.

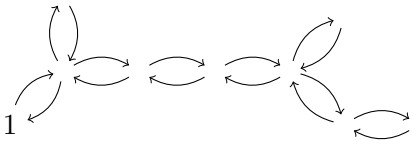
Theorem 3 (Alon,Boppana)**.** *Let $G_1, G_2, \ldots$ be a sequence of d-regular graphs with $|V(G_i)| \to \infty$. Then*

$$\liminf \lambda(G_i) \geq 2\sqrt{d-1}.$$

Proof. Consider a large k and an s such that $n_s := |V(G_s)|$ is enormous. We want to estimate G_s^{2k}'s trace (the sum of eigenvalues). G_s^{2k} is d^{2k}-regular, so $\lambda_1 = d^{2k}$. All other eigenvalues are bounded by $\lambda(G_s)^{2k}$, so

$$\operatorname{tr}(A(G_s^{2k})) = \sum_{i=1}^{n_s} \lambda_i(G_s^{2k}) \leq d^{2k} + (n_s - 1)\lambda(G_s)^{2k}$$

On the other hand, the trace can be calculated by summing the diagonal of $A(G_s^{2k})$, which contains, for each vertex v, the number of routes of length $2k$ going from v back to v. From such routes, we can count at least those that form euler walks around trees: routes constructed by going along an edge, doing a smaller route of this type, and returning along the edge (so we never use actual cycles in the graph to come back, because those would be hard to count).



To count such routes we divide the $2k$ steps of the walk into k returning steps and k non-returning steps (like in an expression with balanced parentheses), and choose, for non-returning steps, one of the remaining edges:

$$\operatorname{tr}(A(G_s^{2k})) \geq n_s \cdot \frac{\binom{2k}{k}}{k+1} \cdot (d-1)^k$$

Therefore

$$d^{2k} + (n_s - 1)\lambda(G_s)^{2k} \geq n_s \cdot \frac{\binom{2k}{k}}{k+1} \cdot (d-1)^k$$

$$\frac{n_s - 1}{n_s}\lambda(G_s) \geq \sqrt[2k]{\binom{2k}{k}\frac{1}{k+1}(d-1)^k - \frac{1}{n_s}d^{2k}}$$

$$\lambda(G_s) \geq \sqrt[2k]{\binom{2k}{k}\frac{1}{k+1}(d-1)^k - \frac{1}{n_s}d^{2k}}$$

As k increases we have

$$\sqrt[2k]{\binom{2k}{k}\frac{1}{k+1}(d-1)^k} \longrightarrow 2\sqrt{d-1},$$

so for large enough k, and s such that $\frac{1}{n_s}d^{2k} \leq \varepsilon^{2k}$ we have $\lambda(G) \geq 2\sqrt{d-1} - \varepsilon$. □