Derandomized Squaring of Graphs

Andrei Ciupan, Spencer Kwon

CS 221, Spring 2018 final project

Abstract. We will describe Rozeman and Vadhan's paper on derandomized graph squaring, and give an intuitive exposition of their logspace algorithm for $S - T$ connectivity.

1. Introduction

The paper of Rozeman and Vadhan (henceforth RV) introduces the notion of a derandomized square operation ($X \circledS G$), and applies it to give a logspace algorithm for connectivity in an undirected graph. In this section, we shall motivate their result and introduce the relevant concepts. In the next section, we shall give a detailed description of their algorithm. Finally, we shall give a sketch of the proof of some key results.

1.1. Space Complexity and the Degree of the Squared Graph. Consider an undirected graph $X(V, E)$ and two vertices s, t. We are interested in deciding if there is a path from s to t. The classic algorithm for connectivity, Savitch's algorithm, essentially involves recursive squaring of graphs: simply take the adjacency matrix of the original graph X, and compute $X^N[s, t]$, where N is the number of vertices of X. The trick is to use recursion (of depth $\log N$), where at each stage only $\log N$ space is needed, resulting in a total space complexity of $\log^2 N$.

One can consider the following alternative naive solution which clearly fails: starting from s, iterate through all the exponential number of paths of length N, and check whether t can be reached. We can frame the above approach using graph powers: let X^n be the graph in which there are k edges between v and w where k is the number of paths of length n from v to w. The weighted adjacency matrix of X^n, which accounts for the multiplicity of edges, is simply the n-th power of the adjacency matrix of X. Then, the naive solution amounts to checking all the neighbors of s in X^N, and seeing if t belongs in the neighborhood.

Here we emphasize a subtle but important point: the size of the neighborhood of s in X^N is not the unique set of vertices that s is connected to, but the degree of s in G^N. While it takes log space to verify the identity of the neighbor connected to s by its i-th edge (simply take the path of length N described by the index

i), we cannot evaluate whether s neighbors a particular vertex v without iterating through all the edges starting from s.

Hence, the reason this naive approach fails is that the degree of G^N is too large - equivalently, we need to check too many paths in the original graph. Hence, *the degree of the squared graph G^N is intimately tied with the space complexity of our algorithm.* If there was a way to contruct a graph X' "similar" to X^N, yet with sufficiently small degree - say, polynomial in N, then we can simply iterate through all the neighbors of s in X' while only using logspace.

1.2. Spectral Gap and Algorithm Correctness. How do we reduce the degree of X^N? A natural idea is to only connect edges corresponding to a subset of the paths. However, one needs to select enough paths such that given that s and t are connected in the original graph, s and t will be neighbors in the resulting graph as well, i.e. the algorithm needs to be correct. To ensure this, we need to introduce the notion of the spectral gap of a regular graph.

Definition 1.1. Let X be a directed K-regular graph on N vertices. Let M_X be the transition matrix of the random walk on X, i.e. the adjacency matrix of X divided by K. Note that $u_N = (1/N, ...1/N)$ is an eigenvector with eigenvalue 1, and as M_X is stochastic, 1 is the largest eigenvalue. Let $\lambda(X)$ be defined as follows:

$$\lambda(X) = \max_{v \perp u_N} \frac{\|M_X(v)\|}{\|v\|} \in [0,1] \tag{1.1}$$

The spectral gap of the graph X is defined to be $1 - \lambda(X)$.

Any probability vector v can be decomposed into $u_N + e$ where $e \perp u_N$. Hence, $\lambda \in [0,1]$ is the upper bound on the rate at which e decays to 0, i.e. the random walk converges to the stationary distribution: the smaller the λ, the faster the convergence. If X is undirected, M_X is symmetric, the eigenvectors are orthogonal, and hence $\lambda(X)$ can also be characterized as the second largest eigenvalue of M_X in absolute value. Hence, one can obtain for undirected graphs $\lambda(X^2) = \lambda^2(X)$, and in general for directed graphs, $\lambda(X^2) \leq \lambda(X)^2$.

$\lambda(X)$ contains information about the connectivity of the graph X. For example, if an undirected graph is not connected, then $\lambda(X) = 1$: simply set the eigenvalue to a uniform distribution on a non-trivial component. Conversely, the following hold:

Lemma 1.2. *Let X be a connected D-regular graph with N vertices, with a loop on every vertex. Then $\lambda(X) \leq 1 - 1/(2D^2N^2)$.*

The idea behind the proof is to first reduce to the undirected case: note $\lambda(X)^2 = \lambda(M^TM)$ by definition, and M^TM can be interpreted as the adjacency matrix of a D^2 regular undirected graph Y whose edges $\{v, w\}$ are vertices such that (v, z) and (w, z) are edges of X. As X contains a loop on every vertex, every directed edge is contained as an undirected edge in Y, and in particular Y has self-loops and thus cannot be bipartite. [AS] then has a result which bounds the minimal eigenvalue (and hence the second largest eigenvalue). We shall not give the full proof.

As λ grows extremely small, the graph becomes increasingly connected, until all the vertices become connected. X is an (N, D, λ) graph if it is D-regular on N vertices with $\lambda(X) \leq \lambda$.

THEOREM 1.3. *Let X be an $(N, D, 1/2N^{1.5})$ graph. Then for any pair of vertices v, w, the probability that a random neighbor of v is equal to w is at least $1/N - 1/N^2$. In particular, X contains a clique - there is an edge between any pair of vertices.*

We shall defer the proof to the third section. Hence, using the tool of spectral gap, one can indirectly prove the correctness of the repeated squaring algorithm in the following way:

(1) If s and t are connected in the graph, the subgraph corresponding to the component containing s and t will have $\lambda(X) < 1$.
(2) Repeated squaring of the graph will result in a sequence of graphs G_m, with $\lambda(X_m)$ growing exponentially smaller.
(3) We can continue this until $\lambda(X_m) < 1/2N^{1.5}$.
(4) Then, s and t must be neighbors in X_m. If they are not, this is a contradiction to Theorem 1.3, and hence s and t were not connected in the original graph.

However, the problem for naively squaring the graph lies in step 4: verifying s and t being neighbors in X^N cannot be done as X^N has exponential degree. Hence, we have reduced the problem of obtaining a logspace algorithm to s-t connectivity to solving the following problem:

PROBLEM 1.4. Given the original graph X, find a sequence of multiply connected graphs X_m on the same vertex set such that:

(1) **Basic correctness condition**: v and w are connected only if there is a path connecting v to w in the original graph.
(2) **Approximate squaring condition**: $\lambda(X_{m+1}) \approx \lambda(X_m)^2$
(3) **Bound on the degree critical for log space**: $deg(X_m)$ remains polynomial in N

RV solves this problem through a novel construction called a derandomized square $X \circledS G$.

1.3. Derandomized Squaring. We shall first quickly describe the concept of a consistent label. Given a directed K-regular graph X, one can label the K out-going edges for each vertex from 1 to K. A labelling is consistent if the K in-going edges of every vertex consist of a permutation of $[K]$. One can easily see that a K-regular graph has a consistent labelling. [1] Given a consistent labelling, one can enumerate all paths from v of length 2 as $v[x][y]$: start from v, take the out-going edge of v labelled x, arrive at vertex w, and then take the out-going edge of w labelled y to arrive at vertex u. The corresponding path is $v \to w \to u$.

Naturally, if all pairs (x, y) are allowed, we recover the original naive squaring. Intuitively, we would like to restrict the possible pairs (x, y) (equivalently, paths of length 2), while still maintaining good connectivity properties. The beautiful idea of RV is to construct a new graph, G, which describes precisely which pairs (x, y) we should include in our "pseduo" squared graphs:

[1]Here is why: Hall's Marriage lemma proves that a k-regular graph has a perfect matching. Find a perfect matching i, label the edges in the matching by i then remove those edges from the graph and repeat the algorithm, labeling the next edges by $i + 1$

Definition 1.5. Let X be a consistently labelled K-regular graph on vertex set $[N]$, G be a consistently labelled D-regular graph on vertex set $[K]$. The derandomized square graph $X \circledS G$ has vertex set $[N]$. The edges exiting v are paths $v[x][y]$ of length 2 in X such that (x, y) is an edge in G. One can easily see that $X \circledS G$ is KD-regular and is consistently labelled, where the label is done in an obvious way: $(x,(x,y))$.

If the auxiliary graph G has good mixing properties, i.e. $\lambda(G)$ is sufficiently small, then the resulting derandomized square will consider paths that have good mixing properties, while only increasing the degree by a factor of D. The following theorem makes the intuition precise:

Theorem 1.6. *If X is a consistently labelled (N, K, λ)-graph and G is a (K, D, μ)-graph, then $X \circledS G$ is an $(N, KD, f(\lambda, \mu))$ graph, where $f(\lambda, \mu) = 1 - (1 - \lambda^2) \cdot (1 - \mu)$. f is monotone increasing in λ and μ, and satisfies $f(\lambda, \mu) \leq \lambda^2 + \mu$, $1 - f(1 - \gamma, 1/100) \geq (3/2) \cdot \gamma$, when $\gamma < 1/4$.*

In other words, if G has good mixing properties, i.e. is a good *expander*, then the resulting derandomized square will have almost the same λ as the regular square of graph, while only increasing the degree by a factor of D. Hence, substituting derandomized squaring for squaring in the procedure mentioned above will lead to the desired algorithm. In the next section, we shall give a more precise account of the exact algorithm.

Remark 1.7. Why is this algorithm called a derandomized square? Here we posit an explanation. A natural alternative to derandomized squaring is to randomly add paths of length 2, up to some constant factor D. Our intuition is that this will indeed produce a randomized log space algorithm which is correct with high probability. By replacing the random mechanism with a deterministic expander graph G, one is able to obtain a deterministic version of the algorithm.

2. Algorithm

In this section, we will give a detailed description of the precise log-space algorithm for solving s-t connectivity. Consider an undirected graph X_{orig} with n vertices, consider two particular vertices indexed by s, t. RV propose the following procedure:

(1) Construct graph X defined as:
   (a) Vertices are pairs (i, e) where $0 \leq i < N$ and $0 \leq e \leq deg(i) - 1$
   (b) Every vertex (i, e) has four neighbors: $(i, e-1), (i, e), (i, e+1)$ [2] and $(j, j(i))$ where j is i-th neighbor which has index e, and $j(i)$ is i's index in the list of neighbors of j.

   This graph is 4-regular and hence it has a consistent labelling L of its edges. The number of vertices it has is twice the number of edges in X_{orig}. Call this number N. We clearly have $N < n^2$.
(2) Consider vertex $v = (s, 0)$ of X.
(3) Consider $Q = 4^q$ and consider a sequence of graphs $\{H_m\}_{m \geq 1}$ and $\{G_m\}_{m \geq 1}$ such that:
   (a) $H_m = \left(Q^m, Q, \dfrac{1}{100}\right)$ for all m

[2] The indices are taken modulo $deg(i)$

   - (b) $G_m = H_m$ for $m \leq m_0 = \lfloor \log N \rfloor$ and $G_{m_0+k} = (H_{m_0-1+2^k})^{2^k}$ for $k \geq 1$. Each graph G_m has $V(G_m)$ vertices and is $K(G_m)$-regular
- (4) Construct $X_1 = X^Q$ where Q was defined above.
- (5) Inductively construct $X_{m+1} = X_m Ⓢ G_m$ for $1 \leq m \leq m_1$, where $m_1 = m_0 + \log\log N + 10$
- (6) We will consider the following helper functions:
   - (a) $f_0 : [N] \times [Q] \mapsto [N]$ such that $f_0(u, e)$ correctly outputs vertex u's e-th neighbor in graph X_1 in constant space.
   - (b) For $m \geq 1$, $f_m : [V(G_m)] \times [K(G_m)] \mapsto [V(G_m)]$ [3] such that $f(u, e)$ correctly outputs vertex u's e-th neighbor, in space $O(m + 2^{m-m_0})$.
   - (c) For $k \geq 0$, let S_k the set of numbers represented as $[i_0][i_1]\cdots[i_k]$, such that $0 \leq i_0 < Q$ and $0 \leq i_m < K(G_m)$ for $m \geq 1$ and let $\Pi_k = Q \cdot G(K_1) \cdot G(K_2) \cdots G(K_k) - 1$. There exists a natural bijection $g_k : S_k \mapsto \{0, 1, \ldots \Pi_k\}$. Let h_k be its inverse.
   - (d) With the notations above, for all $k \geq 0$ define the function $q_k : S_k \mapsto S_{k-1}$ which does the following on input $a = [i_0][i_1]\cdots[i_k]$:
      - (i) Construct $b = g_k([i_0][i_1]\cdots[i_{k-1}])$ if $k \geq 1$. If $k = 0$ then $b = v$.
      - (ii) Construct $c = f_k(b, i_k)$
      - (iii) Output $h_{k-1}(c)$
   - (e) With the notations above, for all $k \geq 0$, define the function $p_k : S_k \mapsto [N]$ which on input s_k computes $q_0(q_1 \cdots (q_k(s_k)) \cdots)$
- (7) (Remember that we are looking for a path from s to t in X_{orig}). Consider the following procedure
   - (a) For all $i_0, i_1, \ldots, i_{m_1}$ such that $0 \leq i_0 < Q$ and $0 \leq i_k < K(G_k)$ for all $1 \leq k \leq m_1$:
      - (i) Initialize $w = [i_0][i_1]\cdots[i_{m_1}]$ and compute $u = p_{m_1}(w)$
      - (ii) By construction u will be a vertex of X_1. If it is of the form (t, j) then we are guaranteed that there exists an edge between s and t in the original graph, and print a path $v_0 \ldots v_{m_1}$ from $v = (s, 0)$ to $w = (t, j)$ in the graph X according to the procedure below:
         - (A) Initialize $k = 0$.
         - (B) While $k \leq m_1$:
            - If $k = 0$ set $u = v$. Otherwise set $u = h_{k-1}([i_0][i_1]\cdots[i_{k-1}])$
            - Print $v_k = p_k(u, i_k)$

This algorithm has the two desirable properties we are interested in :

1. Logspace: it takes space $O(\log(n))$
2. Correctness: it will output a path between s and t if and only if there exists one.

We provide a sketch of these results in the next section.

3. Proof of Key Theorems

We first prove logspace, then the correctness of the algorithm above.

For logspace, we need to show that each of the steps above can be done in logspace time. This requires proving two crucial points: the functions p_k take log

[3]Here the notation $[K]$ corresponds to the set $\{0, 1, \ldots, K-1\}$

space, and the size of all possible values of indices $i_0, \dots, i_{m_1}$ in step 7 is polynomial in N [4]

First let's look at functions p_k. Looking at its construction from point 6e above, we see that it suffices to show that q_k takes log space. In function q_k, defined in point 6d above, it suffices to show that part 6.$d.ii$ takes log space, i.e. that the function f_k takes log space. However, by definition (part 6b), function f_k takes space $O(m_1 + 2^{m_1-m_0}) = O(m_1) + O(2^{m_1-m_0}) = O(\log N) + O(\log N) = O(\log(N))$

Second, let's look at the size of the number we are looping over in step 7. This equals the regularity of graph X_{m_1+1}, i.e. the number of edges we are looping over in order to find t. We can easily show by induction that for $m \leq m_0$, X_m has regularity Q^m, while for $m > m_0$, X_m has regularity $Q^{m_0-1+2^{m_1-m_0}}$. In the previous paragraph we proved that $m_1 + 2^{m_1-m_0} = O(\log(N))$. Since Q is a constant, we trivially obtain $Q^{m_0-1+2^{m_1-m_0}} = O(poly(N))$, so we can indeed loop over the edge set in logarithmic time, and we are done.

We now outline the proofs of the main results that demonstrate the correctness of the algorithm. As explained in the introduction, suffices to prove Theorems 1.3 and 1.6. We shall closely follow the notation and the proof given in the original paper: our focus of this paper was on giving an intuitive exposition on the main results, and hence for the proofs, we shall largely replicate the proof given in the original text.

3.1. Proof of Theorem 1.3. The distribution of a random neighbor of v is simply $M_X e_v$, where e_v is the indicator of v. Let u be the vector with $1/N$ in all of its coordinates. As $M_X u = u$, suffices to show that $M_X(e_v - u)$ has absolute value at most $1/N^2$ at each coordinate. As $e_v - u$ has sum 0, $e_v - u \perp u$, and as $\|e_v - u\| \leq 2$, by the definition of $\lambda(X)$, $\|M_X(e_v - u)\|^2 \leq \left((1/2N^{1.5}) \cdot 2\right)^2 = 1/N^3$, by assumption. Then, if m is the minimal absolute value of a coordinate of $M_X(e_v - u)$, $m^2 N \leq \|M_X(e_v - u)\|^2 \leq 1/N^3$, as desired.

3.2. Proof of Theorem 1.6. By the mixing formulation of $\lambda(X \textcircled{s} G)$, it suffices to show that for every vector v orthogonal to the uniform distribution u_N, Mv is shorter than v by a factor of $f(\lambda, \mu)$. Let M be the transition matrix of the random walk on $X \textcircled{s} G$.

Let A be the transition matrix for X, and B be the transition matrix for G. Then, the transition matrix M can be represented as follows:

(1) Choose $k \in [K]$ uniformly, which corresponds to the augmented state $(v, k) \in [N] \times K$. This corresponds to multiplying by a matrix L, which corresponds to tensoring $v \otimes u_K$.
(2) Go to $(v[k], k)$, which corresponds to $A \otimes I_K = \tilde{A}$.
(3) Pick a random neighbor of k in G, k', and go to $(v[k], k')$. This corresponds to the matrix $\tilde{B} = I_N \otimes B$.
(4) Again make the move $(v[k][k'], k')$, which again corresponds to $\tilde{A}$.
(5) Finally, output $v[k][k']$, which is the projection P onto $\mathbb{R}^N$ again.

To summarize, we have expressed:

$$M = P\tilde{A}\tilde{B}\tilde{A}L \tag{3.1}$$

[4]Remember that the original graph has n vertices. However, we showed that the graph constructed in step 1 of the algorithm has $N < n^2$ vertices, so it suffices to prove these results for N

Now, we shall need a quick lemma, which intuitively states that a matrix with $\lambda(A)$ small is "close" to the perfect mixing matrix, i.e. one corresponding to the complete graph:

Lemma 3.1. *If A is a transition matrix of an (N, D, λ) graph, and J_N be the $N \times N$ matrix with entries all equal to $1/N$, then $A = (1-\lambda)J_N + \lambda C$, $\|C\| \leq 1$.*

Proof. Let $C = \lambda^{-1}(A - (1-\lambda)J)$, and as $Au_N = Ju_N = u_N$, $Cu_N = u_N$. If $v \perp u_N$, note $Av \perp u_N$, as well as $Jv \perp u_N$, and hence $Cv \perp u_N$. Thus, suffices to show $\|Cv\| \leq \|v\|$ for $v \perp u_N$, but as $\|Av\| \leq \lambda\|v\|$ for $v \perp u_N$ and $Jv = 0$, we obtain the result. □

Then, the result follows easily from the lemma: decompose $B = (1-\mu)J + \mu C$. Setting $\tilde{J} = (I_N \otimes J)$ and $\tilde{C} = (I_N \otimes C)$, we obtain:

$$M = (1-\mu)P\tilde{A}(I_N \otimes J)\tilde{A}L + \mu P\tilde{A}(I_N \otimes C)\tilde{A}L = (1-\mu)A^2 + \mu P\tilde{A}\tilde{C}\tilde{A}L \tag{3.2}$$

The latter equality comes from the fact that $P\tilde{A}\tilde{J}\tilde{A}L = P\tilde{A}LP\tilde{A}L = A^2$. There is nothing mysterious about this: $P\tilde{A}\tilde{J}\tilde{A}L$ is simply the substitute of the expander graph G with the complete graph on $[K]$, and hence the derandomized square is precisely the square.

Also, note that $\|L\| = 1/\sqrt{K}$, $\|P\| = \sqrt{K}$, hence $\|P\tilde{A}\tilde{C}\tilde{A}L\| = \|D\| \leq 1$. Hence:

$$M = (1-\mu)A^2 + \mu D \tag{3.3}$$

which implies $\lambda(M) \leq (1-\mu)\lambda^2 + \mu$, which is $f(\lambda, \mu)$, as desired.

Hence, we have completed the proofs of the two theorems, and hence have demonstrated the correctness of the logspace algorithm, as desired.

References

[1] Rozenman E., Vadhan S. (2005) Derandomized Squaring of Graphs. In: Chekuri C., Jansen K., Rolim J.D.P., Trevisan L. (eds) Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques. Lecture Notes in Computer Science, vol 3624. Springer, Berlin, Heidelberg

[2] Noga Alon and Benny Sudakov. (1998) Bipartite Subgraphs and the Smallest Eigenvalue, In: Combinatorics, Probability and Computing, vol 9 1–12

[3] https://en.wikipedia.org/wiki/Hall%27s_marriage_theorem