Exact Local Pagerank on Cartesian Products of Graphs

Tian Luo, advised by Gary Miller

Abstract—We investigate the effect of Local PageRank on Cartesian products of two graphs; specifically, we found a way to determine how we can simulate and easily compute the exact result of Local PageRank on a Cartesian product graph. The motivation behind this is to be able to replicate the limitations of Local PageRank on Cartesian product graphs even when the size of the Cartesian product is large. In this project, we give an $O(m^n + n^m + mn^2 + m^2n)$ algorithm, where $\omega$ is the matrix multiplication exponent, to compute the PageRank vector of the Cartesian product of connected graphs $G$ and $H$ where $|G| = n$ and $|H| = m$.

I. INTRODUCTION

Graph clustering algorithms are an important subroutine in many machine learning applications and in areas of applied mathematics. In recent years, local algorithms that solve graph clustering have become a topic of interest; due to the sheer size of the graphs that are used in practice, local clustering algorithms present many advantages over global graph clustering algorithms. The advantage of only performing computation on vertices that are near the cluster provides a significant speedup over global algorithms since most local algorithms have a run-time dependent on the size of the cluster and not the size of the entire graph.

In recent years, advances have been made to solve local graph clustering via diffusion methods, such as the Capacity Releasing Diffusion (CRD) algorithm [7], and more classically, via spectral methods, such as personalized/local PageRank [1][2][6]. While spectral methods seem to perform well in practice, they are not guaranteed to find an optimal cluster because of it’s inherent ties to random walks and the Cheeger Inequality [3][7]. Theoretically, a well-known class of graphs that spectral methods sometimes fail on is Cartesian product graphs.

Our problem arises from the need to experimentally test the performance of the CRD algorithm, which claims to be the first algorithm to escape the pitfalls of spectral methods and break the Cheeger barrier, and local PageRank on the Cartesian product of a tree and a path. Since the Cartesian product of two graphs is rather large, a natural question to ask is if it’s possible to simulate local PageRank on the Cartesian product by performing local PageRank on the two graphs that compose the Cartesian product separately. There has been work that defined random walks on the Cartesian product of two graphs [4]; however, this definition is distinct with how random walks should behave if the Cartesian product graph was treated as a general graph.

Our main contribution is a $O(m^n + n^m + mn^2 + m^2n)$ algorithm, where $\omega$ is the matrix multiplication exponent, to compute the exact random walk vector for local PageRank on Cartesian product of graphs of size $n$ and $m$, an improvement over the naive way of $O((mn)^\omega)$.

II. PRELIMINARIES

For a weighted undirected graph $G$, let the weight of the edge $\{v_i, v_j\}$ be $w(v_i, v_j) = w(v_j, v_i)$. Define the degree of $v_i$ to be $\text{deg}(v_i) = \sum_{v_j \in N(v_i)} w(v_i, v_j)$. Let the adjacency matrix of $G$ be

$$A_{Gi,j} = \begin{cases} 
0 & \text{if } i = j \\
\ell(v_i, v_j) & \text{if } \{v_i, v_j\} \in E_G \\
0 & \text{o.w.}
\end{cases}$$

Let the degree matrix $D_G$ be a diagonal matrix with $D_{Gi,i} = \text{deg}(v_i)$.

We define the Laplacian of a weighted undirected graph $G$ to be $L_G = D_G - A_G$. It is positive semi-definite which means $\forall x \neq 0$, $x^T L_G x \geq 0$; this also implies that all its eigenvalues are non-negative.

If $G$ is connected, then the Laplacian with row and column $i$ removed, denoted as $L_{Gi}$, has full rank (i.e. it has rank $n - 1$). Notice that $L_{Gi}$ could be written as $L(G') + D$ where $G' = G - v_i$ and $D$ is some diagonal matrix with positive entries. Immediately then, we realize that $L_{Gi}$ is also positive semi-definite.

Next, the Cartesian of graphs $G$ and $H$, $G \square H$, is defined as:

- $V(G \square H) = V(G) \times V(H)$
- $\{(u_1, v_1), (u_2, v_2)\} \in E(G \square H)$ if $(u_1 = u_2$ and $(v_1, v_2) \in E(H))$ or $(v_1 = v_2$ and $(u_1, u_2) \in E(G))$
- $w((u_1, v_1), (u_2, v_2)) = w(v_1, v_2)$ and $w((u_1, v), (u_2, v)) = w(u_1, u_2)$

Intuitively, we can visualize $G \square H$ as having a graph $H$, turning all it’s vertices (labeled $v_i$)’s into copies of the graph $G$ (labeled $G_i$’s) and connecting two isomorphic equivalent vertices in different copies of $G$, say $G_i$ and $G_j$, together if $\{v_i, v_j\} \in E(H)$.

It turns out that the Laplacian of $G \square H$ can be expressed in a particularly nice form. $L(G \square H) = L_G \oplus L_H$ where $\oplus$ is the Kronecker sum defined as $A \oplus B = (A \otimes I_m) + (I_n \otimes B)$ when $A$ is a $n \times n$ matrix and $B$ is a $m \times m$ matrix. $\otimes$ denotes the Kronecker product. Given a $m \times n$ matrix $A$ and $p \times q$ matrix $B$, $A \otimes B$ is a $mp \times nq$ matrix defined as

$$A \otimes B = \begin{bmatrix}
a_{1,1}B & \cdots & a_{1,n}B \\
\vdots & \ddots & \vdots \\
a_{m,1}B & \cdots & a_{m,n}B
\end{bmatrix}$$

Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of $A$ and $v_1, \ldots, v_n$ be the corresponding eigenvectors. Similarly define $\mu_1, \ldots, \mu_n$
A balanced binary tree is the second smallest eigenvalue of the Laplacian of a fully susceptible to this error.

θ(∆0) and smallest positive eigenvalue since the smallest eigenvalue is edges.

Furthermore, the eigenvector corresponding to λ1 is vi ⊗ wij.

A. Cartesian products as counterexamples to spectral methods

Before we get into why spectral methods perform poorly on Cartesian products of two graphs, we need to understand the limitations of spectral methods. First, we define the quotient cut of a graph.

\[ q(G) = \min_{S \subseteq V(G)} \frac{\text{Cap}(S, V \setminus S)}{\min(|S|, |V \setminus S|)} \]

where Cap(S, V \setminus S) = \sum_{(i,j) \in E(G), i \in V \setminus S} w(i, j).

Another notion is the conductance of a partition S, defined similarly as

\[ C(S) = \frac{\text{Cap}(S, V \setminus S)}{\min(|S|, |V \setminus S|)} \]

Notice that the quotient cut is just the best (lowest) conductance among all subsets of the vertices. Intuitively, q(G) defines a “best” cut on a graph, a cut that partitions the graph into two approximately equal sizes and cuts minimal edges.

Now, Cheeger’s Inequality [3] states that

\[ \frac{\lambda_2}{2} \leq q(G) \leq \sqrt{2\Delta \lambda_2} \]

where λ2 is the second smallest eigenvalue of LG (or smallest positive eigenvalue since the smallest eigenvalue is 0) and Δ is the maximum degree of the graph. We could then approximate the best cut via the second eigenvalue with some error; however, it is well known that all spectral methods are susceptible to this error.

We now examine a case in which this error becomes problematic. We first state two more well known facts, the second smallest eigenvalue of the Laplacian of a fully balanced binary tree is \( Θ\left(\frac{1}{n}\right) \) and the second smallest eigenvalue of the Laplacian of a path is \( Θ\left(\frac{1}{\sqrt{n}}\right) \). Suppose we have a balanced binary tree \( T \) of size \( Θ(n^\frac{3}{2}) \) and a path \( P \) of size \( Θ(n^\frac{1}{2}) \). Now, \( \lambda_2(T \square P) = \min(\lambda_2(T), \lambda_2(P)) = Θ\left(\frac{1}{\sqrt{n}}\right) \).

By adjusting constants, we can either make the cut correspond to the second eigenvalue of the path or the tree. Thus, we can make it so that the spectral method will produce a cut corresponding to a cut on the path (in red Fig. 1) instead of the optimal cut of a cut along the trees (in blue Fig. 2). Notice that the cut in red cuts \( Θ(n^\frac{1}{2}) \) edges while the cut in blue cuts \( Θ(n^\frac{3}{2}) \) edges.

B. Local PageRank

First, we define the a very basic random walk. We begin the random walk on a vertex of the graph G. At each step, say that we are on vertex \( v \), we travel to a neighbour, \( u \), of \( v \) with probability \( \frac{w(u,v)}{\text{deg}_G(v)} \). The natural way to view this is in terms of a Markov Chain; here, the transition matrix is given as \( AD^{-1} \) where \( D \) is the diagonal degree matrix and \( A \) is the adjacency matrix.

Now, one variant of this is the lazy random walk, where at each step, with some probability, we stay at vertex \( v \) instead of moving. The modification in terms of the transition matrix is simple; our transition matrix is now \( W = \frac{1}{2}(cI + AD^{-1}) \).

Local PageRank takes this idea one step further [1][6]; at each step, with probability \( \alpha \), we jump back to our starting vertex. Without loss of generality, suppose we start on \( v_1 \); let \( s \) be the all-zero vector except the first index where there is a 1. Let \( p^t \) be the vector where \( p^t_i \) is the probability that the random walk is at vertex \( v_1 \) at time \( t \) (note here that \( p^0 = s \)). Then, we have \( p^{t+1} = \alpha s + (1 - \alpha)Wp^t \). Local PageRank is interested in the stationary vector, or PageRank vector, \( p = \alpha s + (1 - \alpha)Wp \). Given this vector \( p \), we then compute a vector \( q \) where \( q_i = \frac{p_i}{\text{deg}_G(v_i)} \); after we let \( q \) be vertices sorted by their value in \( q \), as in \( q_1 \geq q_2 \geq ... \). We then find the \( k \) such that the set of vertices \( S_k = \{ q_1, ..., q_k \} \) gives the lowest conductance.

In practice, we can approximate \( p \), or more specifically, indices of \( p \) without having to solve the entire stationary [1][6]. However, for this report, we are interested in the complete stationary.

The following is due to [5]. The solution to the stationary is exactly \( p = (I - (1 - \alpha)W)^{-1}\alpha s \) or equivalently, we solve for \( p \) in

\[ (I - (1 - \alpha)W)p = \alpha s \]

We know that since \( q_i = \frac{p_i}{\text{deg}_G(v_i)} \), \( q = D^{-1}p \) and \( (D - (1 - \alpha)A)q = \alpha s \). We can further rewrite this as \( (\alpha D + (1 - \alpha)L)q = \alpha s \) where \( L \) is the Laplacian. To clean up
constants, we can solve for $q$ in the following linear system

$$(\beta D + L)q = \beta s$$

where $\beta = \frac{\omega}{1-\omega}$.

Here, to compute local PageRank exactly, we just need to compute this vector $q$ in this linear system.

### III. Origins

The motivation behind trying to solve for the exact local PageRank vector for Cartesian products of graphs stemmed from experimentation on the CRD and local PageRank algorithm. Since the CRD algorithm claimed to be the first algorithm not subjected to the Cheeger barrier, we thought testing it on the Cartesian product of a tree and a path (or tree-cross-line) would be a good direction for experimenting with the algorithm and a solid first step in understanding the algorithm.

As a benchmark for CRD, we used local PageRank; however, it was quickly realized that it would be extremely helpful if we were able to find a way to compute the rank vector ($q$) for the tree-cross-line from the rank vector of the tree and the line since computing the exact PageRank vector would be computationally expensive given the size of the tree-cross-line examples we were working with. Naturally, a more general question to solve for is if we can compute the rank vector for any Cartesian product given the rank vector of the two smaller graphs that compose the Cartesian product.

### IV. Approach

To solve for $\beta D_G \square H + L_G \square H$, one of the first ideas that we had was to model $\beta D_G + L_G$ as $L(G)_{n+1}$ for a graph $G$ of size $n+1$ and similarly model $\beta D_H + L_H$ as $L(H)_{m+1}$. Notice here that $G$ is just $G$ with super vertex. Then, perhaps we could say something about $L(G \square H)$ compared to $\beta D_G \square H + L_G \square H$; however, notice that $|G \square H| = nm + n + m + 1$; we would have to remove $n + m + 1$ rows and columns of this matrix due to the extra vertices created from the two super vertices. While this resulting matrix would still be full rank because $\beta D_G \square H + L_G \square H$ is full rank, it wasn’t clear how to proceed from there.

Another idea was to consider $d$-regular graphs because then $\beta D_G + L_G$ would be $\beta \cdot \text{deg}(v)I + L_G$ for some $v \in V(G)$.

As evidence of why we believed that there was a way to express the PageRank vector Cartesian product graph as some formulation of the PageRank vectors of the two graphs composing the Cartesian product, there were also experiments conducted on the Cartesian product of two paths (a grid graph) which showed that the PageRank vector for each horizontal path (row on the grid graph) resembled “scaled” versions of the previous row. Obviously, it was not as simple as directly scaling the previous row.

The eventual idea was then to write $s$ as a linear combination of the eigenvectors of $\beta D_G \square H + L_G \square H$. As shown in the next section, this direction turned out to be fruitful not only for $d$-regular graphs but for general connected graphs as well.

### V. Results

In this section, we provide a $O(m^2 + n^2 + mn^2 + m^3n)$ algorithm that can compute the exact PageRank vector for the Cartesian product of $G$ and $H$ where $|G| = n$ and $|H| = m$. We assume that $G$ and $H$ are connected.

First note that by relabeling vertices, for any graph, our local PageRank can always start on $v_1$. Thus, our vector $s$ can always be defined as

$$s_i = \begin{cases} 1 & \text{if } i = 1 \\ 0 & \text{o.w.} \end{cases}$$

Let $s^n$ denote the $n$-length vector that has 1 at the first index and 0 elsewhere.

For any connected graph $G$, $\beta D_G + L_G$ can always be interpreted as the Laplacian of $\tilde{G}$ ($\tilde{G}$ is $G$ with a super vertex) with 1 row and column removed. Therefore, we know that $\beta D_G + L_G$ has full rank and since $\beta D_G + L_G$ is Hermitian, we know that the eigenvalues form an orthogonal basis for $R^n$ where $|G| = n$.

Now, given connected graphs $G$ of size $n$ and $H$ of size $m$, let matrices $A = \beta D_G + L_G$ and $B = \beta D_H + L_H$. We will show how to compute $q_{G \square H}$ in $\beta D_{G \square H} + L_{G \square H} \cdot q_{G \square H} = s^{nm}$ given decompositions of $q_G$ and $q_H$ in $Aq_G = s^n$ and $Bq_H = s^m$.

Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of $A$ and let $v_1, \ldots, v_n$ be the corresponding eigenvectors. Similarly, let $\mu_1, \ldots, \mu_m$ be the eigenvalues of $B$ and let $w_1, \ldots, w_m$ be the corresponding eigenvectors.

Since the eigenvectors of $A$ span $R^n$, there exists some linear combination of the eigenvectors of $A$ such that

$$c_1 \lambda_1 v_1 + \ldots + c_n \lambda_n v_n = s^n$$

Similarly, for $B$, there exists a linear combination of eigenvectors such that

$$k_1 \mu_1 w_1 + \ldots + k_m \mu_m w_m = s^m$$

Because of this, we know that $q_G = c_1 v_1 + \ldots + c_n v_n$ and $q_H = k_1 w_1 + \ldots + k_m w_m$.

We make the following 2 crucial observations:

1) $s^{nm} = s^n \otimes s^m$.

This is easy to verify given the definition of Kronecker products.

2) $\beta D_G \square H + L_G \square H = A \oplus B$.

First notice that $D_G \square H = (D_G \otimes I_m) + (I_n \otimes D_H)$.

Then it’s clear that $\beta D_G \square H + L_G \square H = ((\beta D_G + L_G) \otimes I_m) + (I_n \otimes (\beta D_H + L_H)) = A \oplus B$.

Given that $s^{nm} = s^n \otimes s^m$, we then know that

$$s^{nm} = (c_1 \lambda_1 v_1 + \ldots + c_n \lambda_n v_n) \otimes (k_1 \mu_1 w_1 + \ldots + k_m \mu_m w_m)$$

Then, we can write $s^{nm}$ as

$$s^{nm} = \sum_i \sum_j c_i k_j \lambda_i \mu_j (v_i \otimes w_j)$$

Notice that because $\beta D_G \square H + L_G \square H = A \oplus B$, the eigenvalues of $A \oplus B$ are exactly $\{v_i \otimes w_j | 1 \leq i \leq n \text{ and } 1 \leq$
of a Cartesian product in $O(nm)$ time to sum these vectors up.

to compute.

sum will have $nm$ for all pairs $v_i \otimes w_j$. To compute for the coefficients $c_i \ldots c_n$ also takes $O(n^m)$ time (coefficients $k_1 \ldots k_m$ takes $O(m^ω)$).

Now, to compute the sum, we need to compute $(v_i \otimes w_j)$ for all pairs $i,j$. Naively, this takes $O(nm^2)$ time and the sum will have $nm$ terms that need $O(nm)$ time to add; therefore, naively, $q_{G□H}$ takes $O(n^m + m^ω + (nm)^2)$ time to compute.

However, we can be a bit clever with our computation of the sum. Notice that

$$\sum_i \sum_j c_i k_j \lambda_i \mu_j \frac{(v_i \otimes w_j)}{\lambda_i + \mu_j}$$

The inner sum takes $O(m^2)$ time to compute; for each $i$, computing $v_i \otimes \sum_j c_i k_j \lambda_i \mu_j w_j$ takes would take $O(nm^2)$ time. The total time to compute all $n$ of such vectors $v_i \otimes \sum_j c_i k_j \lambda_i \mu_j w_j$ takes $O(n^m^2 + m^2n)$ time. It takes $O(n^m^2 m^ω)$ time to sum these vectors up.

Thus, we can compute the exact rank vector for PageRank of a Cartesian product in $O(n^m^2 + m^ω + mn^2 + m^ω n)$ time.

VI. SURPRISES AND TAKEAWAYS

This project has been one filled with twists and turns. The direction of focus would shift regularly because we often realized that we needed to show more preliminary results in order to achieve our original goal. The result that is presented in this report, for example, answered one of questions that stemmed from our attempts to do experiments on Local PageRank.

The biggest surprise of this project was the amount of work that it took to convert an idea that was almost certainly correct to a full-fledged proof and analysis. When generating ideas or new approaches to solve problems, properties are taken for granted and constants are ignored; however, these turned out to be the holes in proof and required a surprising amount of time to resolve later on. In retrospect, seemingly easy to prove properties should’ve been given more thought because what seems trivially true is often not.

At first, although intuitively it seemed as if there should be a clean and relatively simple way to compute the PageRank vector for Cartesian product graphs, it turned out to be quite difficult to decipher how the PageRank vectors related to each other. In particular, there isn’t a particularly good way of expressing multiplication and Kronecker products together; getting around this was a roadblock took up a large portion of the effort. At one point, it was thought that there might even not be a way to compute the PageRank vector of $G□H$ from the PageRank vectors of $G$ and $H$; however, everything is messy until someone finds a way to make it clean. The takeaway from this project is that working towards a solution often involves multiple trials and errors; however, with each error, we gain a tiny bit of information and creep $ε$ > 0 closer to the solution.

VII. CONCLUSION AND FUTURE WORK

We presented an $O(m^ω + n^ω + mn^2 + m^2n)$ algorithm to compute the PageRank vector of the Cartesian product of $G$ and $H$ where $|G| = n$ and $|H| = m$. This result is an improvement over computing the PageRank vector by solving the linear system by at least a linear factor. While this result doesn’t have immediate practical applications, and will likely not ever be of practical use since approximating local PageRank is able to approximate up to an $ε$ error of the exact PageRank vector in time $O(\frac{1}{ε})$ time [1].

However, for the rare applications that require exact PageRank vectors, the algorithm presented might be of use. However, the main contribution from this algorithm lies in it’s analytical value. Cartesian products graphs are a heavily studies topic in spectral graph theory because of their particularly interesting spectral properties. This algorithm adds the to already existing pool of work on understanding how random walks operate on Cartesian products. Furthermore, realize that since we generalized the Laplacian to a linear system of Hermitian positive semi-definite matrices, this result also extends into the niche field of study on Kronecker products.

One immediate question if there exists some sort of Cartesian product decomposition where we recursively decompose a graph into Cartesian products and recursively compute the PageRank vector. With this, it may be possible to get rid of the $O(n^ω + m^ω)$ term and replace it with terms dominated by $O(n^m^2 + m^2n)$. However, the key issue with this idea is that inherently Cartesian product graphs are structured in a particular way and graphs in general will not fall under such a structure. Then, the question arises as to whether given graph $G$, we can find two graphs $H$ and $I$ of roughly equal sizes whose Cartesian product is a spectral approximate of $G$.

Another question is if there exists an efficient way of approximating the PageRank vector of $G□H$ given either exact or approximate PageRank vectors of $G$ and $H$. This result would be much more useful for practical purposes.

Lastly, as a more distant and unrelated research topic, understanding CRD’s behaviour on Cartesian product graphs is also of importance since it claims to be the first graph clustering method to break the Cheeger barrier.

VIII. ACKNOWLEDGEMENTS

I would like to express my immense gratitude and appreciation to my mentor, Prof. Gary Miller for his invaluable guidance and continued support throughout this project; he has not only showed me what theory research is like but also instilled in me a solid appreciation of the wonders of linear algebra.
Thank you to Andy Yang and Lichen Zhang for being wonderful collaborators and for always being there to hear my ideas out. Without them, this project would not have come to fruition.

REFERENCES


