Secure Fast Evaluation of Iterative Methods: With an Application to Secure PageRank

Daniele Cozzo[1], Nigel P. Smart[1,2], and Younes Talibi Alaoui[1]

[1] imec-COSIC, KU Leuven, Leuven, Belgium.
[2] University of Bristol, Bristol, UK.
daniele.cozzo@kuleuven.be, nigel.smart@kuleuven.be,
younes.talibialaoui@kuleuven.be

Abstract. Iterative methods are a standard technique in many areas of scientific computing. The key idea is that a function is applied repeatedly until the resulting sequence converges to the correct answer. When applying such methods in a secure computation methodology (for example using MPC, FHE, or SGX) one either needs to perform enough steps to ensure convergence irrespective of the input data, or one needs to perform a convergence test within the algorithm, and this itself leads to a leakage of data. Using the Banach Fixed Point theorem, and its extensions, we show that this data-leakage can be quantified. We then apply this to a secure (via MPC) implementation of the PageRank methodology. For PageRank we show that allowing this small amount of data-leakage produces a much more efficient secure implementation, and that for many underlying graphs this 'leakage' is already known to any attacker.

1 Introduction

Iterative methods are a standard technique in scientific computing; indeed a vast array of the problems have traditionally been mapped to iterative methods. Examples include finding roots of systems of equations (e.g. the Newton-Raphson method for polynomials in a single variable), finding eigenvalues and eigenvectors of matrices (and hence performing tasks such as Principal Component Analysis), or of finding solutions to ordinary and partial differential equations. Indeed the solution of many real world problems involve mapping the problem into a mathematical formulation in which an iterative method can be applied.

Leakage From Iterative Methods: At its heart an iterative method involves a map $F : \mathcal{M} \longrightarrow \mathcal{M}$ on a metric space $\mathcal{M}$ for which we want to compute a stationary point, i.e. a value $x \in \mathcal{M}$ s.t. $F(x) = x$. That $\mathcal{M}$ is a metric space implies we have a well defined distance metric $d(x, y)$, and thus a well defined notion of convergence. If $\mathcal{M}$ is a normed vector space with norm $\|\cdot\|$ then this induces the distance $d(x, y) = \|x - y\|$. The iterative method requires one to determine a subset $X \subset A$ containing the desired fixed point x, s.t. if we pick any starting value $x_0 \in X$, then the sequence $x_{i+1} \leftarrow F(x_i)$ will converge to x. In applying

the iterative method often the most difficult task is the initial choice of the set X.

However, when examined from the point of view of secure computation one has an additional problem. Suppose the function F is itself secret, for example F could be a polynomial of bounded degree whose coefficients are unknown for which it is desired to compute a root, or F could be a matrix operator of a given dimension for which an eigenvector is desired. Apart from the actual computation of the iteration, in the secure domain we need to determine how many iterations we need to perform. This is a problem irrespective of whether we try to perform the secure computation with Multi-Party Computation (MPC), Fully-Homomorphic Encryption (FHE) or using a form of Trusted Execution Environment (TEE) such as Intel's SGX platform.

When operating in the clear the iteration is performed until the difference satisfies $d(x_{i+1}, x_i) \leq \epsilon_{\mathsf{abs}}$, for some fixed tolerance ϵ_{abs}. We could perform such a termination condition in the secure domain, but that would leak information. Thus a tempting solution to this problem is simply to compute the sequence $(x_i)_{i=0}^{N}$ for a large enough N so that we do not need to leak any information. Clearly, the latter solution is more expensive.

In this paper we examine this 'when to terminate' problem in generality, and relate the information leakage to the classical Banach Fixed Point Theorem (a.k.a. the Contraction Mapping Theorem) which was proved by Banach in 1922. This restricts the function $F : \mathcal{M} \longrightarrow \mathcal{M}$ to a function $F : X \longrightarrow X$ for a subset $X \subset \mathcal{M}$ for which the resulting function is a contraction (see below for the definition). In particular the speed of convergence is related to the Lipschitz constant of the underlying contraction mapping $F : X \longrightarrow X$. Thus the information leakage is the single value giving the number of iterations until convergence. This value itself encodes information about the function F and the set X, as well as the starting position x_0. We answer the question as to what this information actually encodes, specifically with respect to the map F. By quantifying the precise information leakage, the user of the secure computation environment can determine whether this leakage is acceptable or not. In many examples this leakage is indeed totally acceptable.

We focus on what information is leaked about the function F from the number of iterations. Clearly the number of iterations taken also leaks something about the initial starting vector x_0, in particular how close it is to the final solution. For some applications of iterated methods this could itself leak information about F, for example when using a Newton iteration to find a root of a polynomial. Thus the number of iterations leaks information about F and the initial starting point. For the power method to find the dominant eigenvalue one can select x_0 to be a random vector (as long as it has a non-zero component in the direction of the corresponding eigenvector). Hence, for our application to the power method the parties can select x_0 at random, meaning even less information is revealed about F via a single iteration. Of course if one applies the method repeatedly with different random x_0 values, then the contribution from the starting value x_0 can be averaged out. Hence, in what follows one needs

to bear in mind that we are considering the best possible case for an attacker. In practice, for a single execution of an iterative method, the ability to extract meaningful information about either x_0 or F is limited.

The Power Method for Matrices: To illustrate this we then go on to discuss one of the most famous applications of iterative methods; namely the power method for matrices used to compute eigenvalues/eigenvectors. In this problem, which forms the basis of many numerical computation problems, one is given an $m \times m$ complex valued matrix $A \in \mathcal{C}^{m \times m}$, and one is asked to compute an eigenvalue/eigenvector, i.e. a solution to the equation $A \cdot \mathbf{x} = \lambda \cdot \mathbf{x}$. If we order the eigenvalues of A as $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_m|$ then the iteration

$$\mathbf{x}_{i+1} \leftarrow \frac{A \cdot \mathbf{x}_i}{\|A \cdot \mathbf{x}_i\|}$$

will converge to an eigenvector $\mathbf{x}$ of A corresponding to the *dominant eigenvalue* λ_1, assuming the starting vector $\mathbf{x}_0$ is has a non-zero component in the direction of eigenvector $\mathbf{x}$.

The advantage of iterative methods for solving the eigenvector problem is that they respect any sparseness of the underlying matrix A. In other words memory is constant. In addition they scale well with the dimension, which the traditional methods used for low dimension matrices do not. Iterative methods are also data oblivious, i.e. apart from the termination test the algorithm requires no data dependent branching or memory access operations.

It is a classical result, found in almost all undergraduate mathematics courses, that the speed of convergence of the power method iteration is *related* to the quotient λ_2/λ_1, called the spectral gap. Thus, revealing the number of iterations required in order to satisfy $\|\mathbf{x}_{i+1} - \mathbf{x}_i\| \leq \epsilon_{\mathsf{abs}}$ reveals something about the quantity $|\lambda_2|/|\lambda_1|$. For large dimensional matrices A revealing this quantity is a small price to pay for the performance improvement of the computation. Note, the number of iterations does not leak the exact value of λ_2, as the exact number of iterations depends on all eigenvalues and the initial starting position. However, the dominant term is dependent on λ_2. Whether revealing something about λ_2 reveals something adversarially interesting about the original input matrix A depends upon the problem one is solving, i.e. from where A originates.

PageRank: PageRank is originally an algorithm devised by Brin and Page, in order to give 'importance' rankings to web-pages [28], which formed the basis of the original Google search algorithm. The algorithm models the Internet as a directed graph G in which the m nodes are web-pages. A node i has an edge towards node j, if web-page i contains a hyper-link to web-page j. The PageRank algorithm aims to simulate a random surfer performing a random walk on the graph of such web-pages. The output of the algorithm is a probability distribution $\boldsymbol{\pi}$ over the set of all web pages (i.e. all elements are in the range $[0, 1]$ and we have $\|\boldsymbol{\pi}\|_1 = 1$). The vector $\boldsymbol{\pi}$ is known as the PageRank vector. The PageRank vector represents the probabilities of our random surfer ending up on

a specific web-page. Those probabilities serve as a ranking to web-pages, that is further used to sort the results that will be displayed for a user after a search request. The idea being that a page which the random surfer is more likely to land on is more likely to be important/useful than one which they are less likely to land on.

To run PageRank we first compute the adjacency matrix A of the graph G of the m web-pages. This matrix contains zeros except in entry $a_{i,j}$ when page i has a link to page j, in which case we place a one. We then transform A into a row-stochastic matrix P. To do this we replace each value of one in P by the value of one over the Hamming weight of the corresponding row in A. Thus $p_{i,j}$ equals the probability of a surfer on page i clicking one of the outgoing links with uniform probability.

However, the matrix P has some issues. In particular there are rows which are all zero, which correspond to webpages which have no outgoing links. In the graph such pages are called 'dangling nodes'. The fix for this problem is to add to P a matrix $D = \mathbf{d} \cdot \mathbf{v}^T$, where $d^{(i)} = 1$ if node i is a dangling node, and $\mathbf{v}$ represents a probability distribution, called the *personalization vector*. In 'traditional' PageRank the personalization vector is set as $\mathbf{v}$ with $v^{(i)} = 1/m$. This models the idea that once the random surfer reaches a dangling web-page, he then jumps to a completely random page on the Internet. However, this is purely a choice and other vectors could be selected. For example users might be more likely to jump to Facebook or Google in an Internet application. The resulting matrix $Q = P + D$ is then a row-stochastic matrix.

However, this is yet to model a realistic random surfer, as it supposes that the surfer will be restricted to the links contained in non-dangling web-pages he passes through during the random walk, but we know this not to be true. To take this into account, PageRank has a parameter $\lambda \in [0, 1]$ called the damping factor, which (in classical PageRank) represents the probability that a user does not click on a link on a web-page while moving to the next one, but instead jumps to a random one following the probabilities contained in the personalization vector $\mathbf{v}$. From this we define the matrix $E = \mathbf{e} \cdot \mathbf{v}^\mathsf{T}$, where $\mathbf{e}$ denotes the all one m-dimensional column vector, and then compute the final PageRank matrix as

$$M = \big(\, (1 - \lambda) \cdot Q \; + \; \lambda \cdot E \big)^\mathsf{T} . \tag{1}$$

The value λ is usually selected to be between 0.1 and 0.2, for reasons we will explain later.

Clearly, by construction, M is a column stochastic matrix, and since it is stochastic the right eigenvalues of M satisfy $\lambda_1 = 1 \geq |\lambda_2| \geq \ldots \geq |\lambda_m| \geq 0$. The solution to the PageRank problem is the eigenvector corresponding to the dominant eigenvalue; i.e. the solution to the problem $M \cdot \boldsymbol{\pi} = \boldsymbol{\pi}$. Thus PageRank is an example of a problem which can be solved via iterative methods. Indeed, iterative methods are preferred since the matrix M is very large.

Note, there is a whole line of work within graph theory which is dedicated to the relationship between the spectra of adjacency matrices and the structure of the underlying graphs [1, 2, 32]. However, this is only applicable in the case

of *undirected* graphs. For directed graphs underlying PageRank, it is known [12, 21, 22] that if the eigenvalues of Q are given by $\{1, \lambda'_2, \ldots, \lambda'_n\}$ then the eigenvalues of M are given by $\{1, (1-\lambda)\cdot\lambda'_2, \ldots, (1-\lambda)\cdot\lambda'_n\}$. This generalized a result in [16] which showed that if the Markov chain underlying the stochastic matrix Q has at least two irreducible closed subsets then $\lambda'_2 = 1$. Thus assuming the underlying directed graph has two irreducible closed subsets, then we already know that the second eigenvalue of the PageRank matrix is equal to $1-\lambda$. Thus for this *specific* application of the power method to the PageRank matrix leaking a function of the second eigenvalue λ_2 reveals no more information than one already knows.

Why a Secure PageRank?: Graph analysis techniques are nowadays a crucial tool for financial institutions, serving as a means to identify fraudulent bank accounts, or fraudulent or suspicious transactions (such as transactions related to money laundering). See [24] for a discussion on secure computation technologies applied to financial intelligence sharing, or [15] for a specific example of secure graph analysis for financial stress testing. These techniques help to extract features from a large amount of data, consisting of, for example, money movements between bank accounts. Leading to accounts being investigated if anomalies are detected. One of the techniques proposed is the PageRank algorithm, which turns out to be well suited for the financial context.

PageRank can be used in different ways in this direction. One way [27] consists of modeling the bank accounts and transactions taking place between them as a directed graph, where nodes consist of bank accounts, and an edge is set from bank account i to bank account j if i has sent to j a transaction, and then running a random *biased* walk to extract the PageRank vector, where the random walker bias his walk towards acknowledged fraudulent accounts. The bias is introduced by making the personalisation vector zero on all accounts which were not known to be fraudulent, with the same for the starting vector.

Doing this will help determine the bank accounts that are important from the point of view of fraud. See also [35] for similar ideas in the context of social security fraud. In another method explained, in [26], the use of PageRank is to reduce false positive rates in traditional fraud scoring

In order to obtain reliable results with PageRank, we need many financial institutions to collaborate. In fact, each financial institution I_k can locally only build its own transaction graph G_k, modeling transactions in which the sender or the receiver is a bank account from the set B_k of the bank accounts that I_k manages. While this will cover the full activity of transfers within B_k, data will be missing (for example) from regarding bank accounts that receive transactions from accounts in B_k but which are not in B_k. Such extended transaction graphs are crucial in applications in which one is trying to locate money laundering. Thus the more financial institutions that combine their graphs, the more accurate the results of PageRank will be in detecting fraud; this is explained in more detail in [11].

However, financial institutions are not willing, or able for regulatory reasons, to simply share this information among each other. Therefore, they need to

engage in a protocol, where they can perform the computation (the PageRank algorithm) on their respective inputs (G_k), and the only information revealed out of this computation is the output (the rankings of the bank accounts).

Another use case where one would take use of a secure implementation of PageRank is the analysis of social networks. That is, similarly to the Web and to transactions among institutions, social networks such as YouTube, Twitter and Facebook etc, can be modeled as graphs [17] where nodes are users' accounts. Running PageRank within one network on the publicly available data is already deemed to be useful, where PageRank can be used to evaluate the reputation of users [14]. However, addressing a more complicated problem such as understanding the flow of photos or news stories etc across interconnected networks may require the owners of these social networks to engage in a secure computation of PageRank in order to provide the relevant data.

PageRank and MPC: As already remarked the convergence of fixed point methods is an issue in any secure computation paradigm in relation to the number of steps needed to determine convergence. There has already been some work in looking at MPC as a means of securely evaluating the PageRank algorithm [30], due to the above mentioned applications in fraud detection.

MPC protocols can be divided into two big families with respect to the level of security they can offer; protocols providing passive security, and protocols providing active security. In a passively secure MPC protocol, the adversary is assumed to be honest, but curious. That is, the parties are assumed to follow the exact description of the protocol, however they can use the information they see in order to infer information about inputs of the other parties. In an actively secure MPC protocol, the adversary can deviate from the protocol, and yet the privacy of the parties' inputs must be still maintained.

In [30] the authors presented a *passively* secure MPC protocol for PageRank in the context of fraud detection for bank account transactions. The authors are able to obtain a very fast and scalable protocol due to the fact that they assume all banks with relevant accounts *participate* in the protocol. In particular each bank locally holds their view of the transaction graph, i.e. the movements between accounts which they have sight of. This enables the protocol to be efficiently implemented using partially homomorphic encryption. Note, the work in [30] could be improved using our analysis of termination conditions for the power method.

Roughly speaking in [30], party I_k is in charge of updating the rankings of the nodes of B_k in iteration i, using the encryptions it received of the rankings of the other nodes after iterations $i-1$, by taking use of the fact that the encryption scheme is partially homomorphic and the needed data to update the ranking is held in clear by I_k. Thus only linear operations are needed to be performed.

Apart from being passively secure, the protocol crucially relied on each part of the graph being held in the clear to at least one party. In many situations this may not be feasible. Indeed the protocol is unsuitable for use in a secure outsourcing scenario for precisely this reason. For an outsourcing based approach to

be secure, the financial institutions need to provide all their data encrypted, and therefore a partially homomorphic encryption scheme will no longer be sufficient.

To cope with these limitations we provide a 'pure' MPC implementation of the PageRank power iteration. In other words a method in which parts of the graph are hidden from all parties. The protocol will benefit from our early termination procedure. Furthermore, since we are employing general MPC techniques that guarantee active security, the resulting implementation will be inherently actively secure.

Our solution does not require from all the institutions to participate in the computation, but only a subset of them or third parties, which the institutions agree upon. Thus we distinguish here between three different entities, the financial institutions $\{I_1, \ldots, I_u\}$, the computing servers $S = \{S_1, \ldots, S_v\}$ with $v = 2$ or $v = 3$, and the bank accounts $\{b_1, \ldots, b_m\}$ that the institutions manage. The institutions do not need to trust all the servers in S, but they do need to trust at least one party for the case where $v = 2$, and at least two parties not to collude for the case of $v = 3$.

Even though we are unable to cope with the size of graphs reported on in [30], we feel our solution can be applied in other situations where one has a different trade off. In the course of which we provide an optimization to the secure dot-product computation over fixed point numbers at the heart of PageRank which is similar to that introduced in [25]; this results in a number of $N \cdot m^2$ secure additions, $N \cdot (m^2 + m)$ secure multiplications and $N \cdot m$ truncations, where N is the number of the iterations performed. From this it is clear that the computations are mainly algebraic therefore motivating our using of arithmetic modulo p to represent secret shared numbers. Indeed linear secret sharing seems to have an advantage over garbling techniques even for the case of two servers, due to that adding and multiplying large numbers many times is expensive with such techniques.

2 Preliminaries

In many applications, such as ours of securely evaluating the PageRank algorithm, we require to work with approximations to real numbers. One could try to emulate precise IEEE floating point arithmetic, but this is rather expensive. Thus we need to somehow 'encode' the real numbers within the arithmetic of $\mathbb{F}_p$. A common way of doing this is to use a form of fixed point arithmetic, introduced in [6].

To represent a real number $\mathfrak{e}$ we approximate it as a fixed point number $\overline{e}$, where

$$\overline{e} = e \cdot 2^{-f}.$$

The value f is a fixed public value that defines the precise position of the fixed point, i.e. 2^{-f} is the smallest unit we can represent and the increment between two successive values. The value e is an integer in the range $[-2^{k-1}, \ldots, 2^{k-1}]$, for some public parameter k. The value k determines the total number of binary points of data we can represent, thus the biggest value we can represent is 2^{k-1-f}.

To hold a secret version of this approximation $\overline{e}$ to the real number $\mathfrak{e}$ we simply secret share the integer e above as an element in $\mathbb{F}_p$. In particular this means that we must have $p > 2^k$; and indeed to perform arithmetic we will require an even larger value of p as we shall later see. We write $\langle\ \overline{e}\ \rangle$ to represent the sharing of a fixed point value $\overline{e}$, and the mod-p value we actually store we shall denote as $\langle e \rangle$.

If $\langle h \rangle$ is a shared integer in the range $[-2^{k-1-f}, \ldots, 2^{k-1-f}]$ then we can obtain the shared fixed point representation $\langle\ \overline{g}\ \rangle$ of the same value h by computing $\langle g \rangle = \langle h \rangle \cdot 2^f$ (and therefore $\overline{g} = g \cdot 2^{-f} = h$), which is a linear operation. Fixed point shared values can also be added by simply adding the underlying shared integer representation.

Multiplying fixed point numbers $\langle\ \overline{e}\ \rangle$ and $\langle\ \overline{g}\ \rangle$ is however a little more complex. We first multiply the underlying shared integer representations $\langle e \rangle$ and $\langle g \rangle$, to obtain the value $\langle h \rangle \leftarrow \langle e \rangle \cdot \langle g \rangle$ and then we shift and truncate $\langle h \rangle$ by f bits, see Figure 1. The value h will be an integer in the range $[-2^{2 \cdot k-2}, \ldots, 2^{2 \cdot k-2}]$ thus to avoid wrap-around modulo p we will require $2^{2 \cdot k-2} < p$.

Fixed Point Multiply

Input: $\langle\ \overline{e}\ \rangle, \langle\ \overline{g}\ \rangle$, **Output:** $\langle\ \overline{h}\ \rangle = \langle\ \overline{e}\ \rangle \cdot \langle\ \overline{g}\ \rangle$.

- $\langle v \rangle \leftarrow \langle e \rangle \cdot \langle g \rangle$.
- $\langle h \rangle \leftarrow \mathsf{TruncPr}(\langle v \rangle, 2 \cdot k, f)$.

Figure 1. Algorithm to Multiply Two Shared Fixed Point Values

The truncation by f-bits is done using a technique from [6]. The algorithm $\mathsf{TruncPr}(\langle x \rangle, t, o)$ takes a shared integer value $\langle x \rangle$ where $x \in [-2^{t-1}, \ldots, 2^{t-1}]$, a value $o \in [1, \ldots, t-1]$ and outputs the value $\langle y \rangle$ where $y = \lfloor x/2^o \rfloor + u$ for an (essentially random) unknown bit $u \in \{0, 1\}$. This probabilistic truncation algorithm turns out to be more efficient in MPC than a method which avoids the random bit u. The method works by opening the value $\langle a + r \rangle$ for some blinding value $r \in [0, \ldots, 2^{t+\kappa}]$ where κ is a statistical security parameter, and then computing the result from the clear value $c = a + r$. In particular we require, to avoid overflows modulo p, that $2^{t+\kappa} < p$. To ensure correctness of the entire procedure we hence require that $2 \cdot k + \kappa < \log_2 p$.

Another computation we will be performing on secret shared fixed-point values is comparison. The protocols implementing comparison that we will be using are also taken from [6], where it is also performed using truncation. That is, comparison is based on the observation that if $a < 0$, then $\lfloor a/2^{k-1} \rfloor = -1$ and if $a \geq 0$ then $\lfloor a/2^{k-1} \rfloor = 0$. Therefore, we can compute the sign of a secret shared value $\langle\ \overline{a}\ \rangle$ by truncating it by $k-1$ bits.

However, truncation here uses a deterministic sub-routine Trunc unlike the sub-routine $\mathsf{TruncPr}$, as we need to round to the correct integer here. While Trunc does not add any extra conditions to the requirements on the parameters for correctness, it is relatively expensive compared to $\mathsf{TruncPr}$. However, we will

be performing orders of magnitude more TruncPr's than Trunc operations and so, whilst it is more expensive, the effect of Trunc on the final runtime of the overall algorithm can be ignored.

3 Banach Fixed Point Theorem and the Power Method

Suppose we have a function $F : \mathcal{M} \longrightarrow \mathcal{M}$ on a complete metric space $\mathcal{M}$ with distance $d(x, y)$. The function F we will compute securely, and then repeat the application, thus producing a sequence of values

$$x_{i+1} \leftarrow F(x_i),$$

for some starting value x_0. We assume, for the moment, that this sequence tends to a value $x_i \longrightarrow x$ which is a fixed point of the function F, i.e. $F(x) = x$. Our goal is to securely compute a value x_N such that $d(x_N, x) \leq \epsilon_{\mathsf{abs}}$; indeed it may be that we keep the final value x_N secure and do not release it to the computing parties. But as we are computing this in the secure domain, we have two options:

1. We pick a large value of N, irrespective of the specific value of F and x_0.
2. At each iteration we reveal to the parties the value $d(x_i, x_{i-1})$, or whether $d(x_i, x_{i-1}) \leq \epsilon_{\mathsf{abs}}$ and terminate the iteration if this is less than some given tolerance ϵ_{abs}. Let $N = i$ denote the first instance when this happens.

Clearly the second methodology, irrespective of the underlying secure computation technology (be it MPC, FHE or TEE) potentially reveals more information than the first. The question is; how much? The answer to this question is provided by the Banach Fixed Point Theorem.

To ensure the sequence (x_i) converges we need to make some assumptions about the starting point x_0 and the map F. We first need to restrict the domain/codomain $\mathcal{M}$ to a complete subset $X \subset \mathcal{M}$ on which F is a *contraction mapping*.

Definition 3.1. *A map $F : X \longrightarrow X$ is said to be a contraction mapping on X if there exists a constant $q \in [0, 1)$ such that, for all $x, y \in X$,*

$$d(\ F(x),\ F(y)\ ) \leq q \cdot d(\ x,\ y\ ).$$

The constant q is called the Lipschitz constant for F and X, it depends on both F and (sometimes) the set X. The following theorem is classical, and proved by Banach in 1922,

Theorem 3.1 (Banach Fixed Point Theorem). *Let (X, d) be a non-empty complete metric space and let $F : X \longrightarrow X$ be a function. If F is a contraction mapping, with constant q, then there is a* unique *fixed point $x \in X$, i.e. $F(x) = x$. If we pick $x_0 \in X$ and define the sequence $x_{i+1} = F(x_i)$ for $i \geq 0$, then $x_i \longrightarrow x$ as $i \longrightarrow \infty$.*

The speed of convergence is itself controlled by the value q; in particular we have

$$d(x, x_i) \leq \frac{q^i}{1-q} \cdot d(x_1, x_0), \qquad d(x, x_{i+1}) \leq \frac{q}{1-q} \cdot d(x_{i+1}, x_i),$$
$$d(x, x_{i+1}) \leq q \cdot d(x, x_i), \qquad d(x, x_{i+1}) \leq q^i \cdot d(x, x_0).$$

Thus in our second secure computation strategy above if we terminated at step N when $d(x_N, x_{N-1}) \leq \epsilon_{\mathsf{abs}}$ then we have that

$$d(x, x_N) \leq \frac{q}{1-q} \cdot \epsilon_{\mathsf{abs}}.$$

We examine this within the context of the power method, which is the standard example application of the above theorem. Here we aim to find the eigenvector corresponding to the dominant eigenvalue λ_1 for $A \in \mathcal{C}^{m \times m}$, i.e. the solution to $A \cdot \mathbf{x} = \lambda_1 \cdot \mathbf{x}$. The iteration is given by

$$\mathbf{x}_{i+1} \leftarrow \frac{A \cdot \mathbf{x}_i}{\|A \cdot \mathbf{x}_i\|} = \frac{A^i \cdot \mathbf{x}_0}{\|A^i \cdot \mathbf{x}_0\|}$$

for some vector norm $\|\cdot\|$. The norm ensures that $\mathbf{x}_{i+1}$ has norm one. We select $\mathbf{x}_0$ to also have norm one at random, thus we have a mapping F from vectors of norm one to vectors of norm one.

We make the simplifying assumption (for exposition) that A has distinct eigenvalues $\lambda_1, \ldots, \lambda_m$ with corresponding eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$, with $|\lambda_i| > |\lambda_j|$ for $j > i$, and define F via

$$F(\mathbf{x}) = \frac{A \cdot \mathbf{x}}{\|A \cdot \mathbf{x}\|_1}. \tag{2}$$

For concreteness in the above we chose the 1-norm, as in the following we will be treating stochastic matrices and vectors. Normalizing the eigenvectors so that $\|\mathbf{v}_i\| = 1$, we can write

$$\mathbf{x}_0 = c_1 \cdot \mathbf{v}_1 + \cdots + c_m \cdot \mathbf{v}_m.$$

Assuming $c_1 \neq 0$ the iteration $\mathbf{x}_i = F(\mathbf{x}_{i-1})$ will converge to the eigenvector corresponding to the dominant eigenvalue since

$$A^i \cdot \mathbf{x}_0 = c_1 \cdot \lambda_1^i \cdot \left(\mathbf{v}_1 + \frac{c_2}{c_1} \left(\frac{\lambda_2}{\lambda_1} \right)^i \cdot \mathbf{v}_2 + \cdots + \frac{c_m}{c_1} \left(\frac{\lambda_m}{\lambda_1} \right)^i \cdot \mathbf{v}_m \right). \tag{3}$$

Thus the speed of convergence is predominantly determined by the value $|\lambda_2|/|\lambda_1|$, with the precise number of iterations depending on *all* of the eigenvalues and the distance between the starting position $\mathbf{x}_0$ and the final solution.

We would like to iterate until the difference between two successive iterations $\mathbf{x}_{N-1}$ and $\mathbf{x}_N$ is sufficiently close. In other words we will terminate when

$$\|\mathbf{x}_N - \mathbf{x}_{N-1}\|_2^2 \leq \epsilon_{\mathsf{abs}}^2$$

holds for the first time, for some constant ϵ_{abs}. Note, in our experiments we later also use termination using the relative error

$$\|\mathbf{x}_N - \mathbf{x}_{N-1}\|_2^2 \leq \epsilon_{\mathsf{rel}}^2 \cdot \|\mathbf{x}_N\|_2^2,$$

for some other constant ϵ_{rel}, as (for large dimension m) the entries in the solution to PageRank behave like $1/m$ for a graph with many links, and thus the relative error allows us to deal with increasing m, without needing to adjust ϵ_{abs} accordingly. Note, for our termination condition we would prefer to use, for computational reasons, the square of the 2-norm, (as performing square roots is expensive in an MPC system).

We make the simplifying assumption that we select our starting set X so that the map *is* indeed a contraction mapping for the 2-norm with Lipschitz constant $q = |\lambda_2|/|\lambda_1|$[3]. With the above simplifying assumption, and for this value of N, we have

$$\|\mathbf{v}_1 - \mathbf{x}_N\|_2 \leq \frac{q}{1-q} \cdot \|\mathbf{x}_N - \mathbf{x}_{N-1}\|_2 \leq \frac{q}{1-q} \cdot \epsilon_{\mathsf{abs}} = \frac{\lambda_2 \cdot \epsilon_{\mathsf{abs}}}{\lambda_1 - \lambda_2}.$$

So the bigger the difference between λ_2 and λ_1, the smaller the error between where we terminate and the correct solution.

For the iteration before we reach this value of N we have

$$\begin{aligned}
\epsilon_{\mathsf{abs}} &< \|\mathbf{x}_{N-1} - \mathbf{x}_{N-2}\|_2 \\
&\leq \|\mathbf{v}_1 - \mathbf{x}_{N-1}\|_2 + \|\mathbf{v}_1 - \mathbf{x}_{N-2}\|_2 \\
&\leq \left(\frac{q^{N-1}}{1-q} + \frac{q^{N-2}}{1-q}\right) \cdot \|\mathbf{x}_1 - \mathbf{x}_0\|_2 \\
&\leq \frac{q^{N-2} \cdot (q+1)}{1-q} \cdot (\|\mathbf{x}_1\|_2 + \|\mathbf{x}_0\|_2) \\
&= \frac{2 \cdot q^{N-2} \cdot (q+1)}{1-q}.
\end{aligned}$$

Thus revealing N reveals information about q, and thus information about the spectral gap $|\lambda_2|/|\lambda_1|$. In the context of a given application some information about the eigenvalues may have already leaked as part of the problem statement. For example in the PageRank algorithm we already know that $\lambda_1 = 1$ and that λ_2 is $(1-\lambda)$ times the second eigenvalue of the original stochastic matrix Q. As already remarked, when the underlying graph contains at least two irreducible closed subsets (which holds in practice for the internet graph) we already know that λ_2 is *exactly* equal to $1 - \lambda$.

In conclusion if the mapping is indeed a contraction mapping for the metric used to determine when to terminate an iterative method, then the number of

[3] This is not quite true, but is true if we modify the metric used to measure convergence. But then the metric depends on the final answer, which is perfect for theoretical considerations, but useless in practice. See Appendix A for more details.

iterations leaked by performing this test leaks information about the Lipschitz constant and the distance between the starting value and the fixed point. For all mappings for which the power method converges one can find a metric for which it is a contraction mapping, but this metric may not be applicable for use in an algorithm as a convergence test.

For the power method applied to matrices it is known that the speed of convergence is related to the value $q = |\lambda_2|/|\lambda_1|$. For some matrices and sets X this does indeed define a contraction mapping, but not for all. Making the simplifying assumption, for exposition, that the power method is defined from a contraction mapping with this Lipschitz constant, one can from N derive information about q.

Note that whether this is an acceptable leakage or not depends on the application in hand, i.e., from where the map F originates. It is worth recalling that the speed of convergence of an iterative method does not leak the value of q, but only provides *some information* about its distribution, which may be considered as a minor leakage in many cases, or even no leakage at all, such as the case of PageRank over the internet graph.

4 Stability of PageRank

Being a numerical algorithm we need to worry about the stability of the PageRank algorithm. However, as we are computing in the secure domain we not only need to worry about the traditional stability of the algorithm, but we also need to worry about stability caused by the representation of the floating point numbers within the secure computation system. In this section we address these two issues (normal numerical stability and stability within the MPC system).

4.1 Traditional Stability of PageRank

As the value of m is large in applications of PageRank solving for $\boldsymbol{\pi}$ analytically or via high-school linear algebra is not feasible. Thus in practice the *only* method one can use to solve the PageRank problem is to apply the power method.

Applying the power method on our matrix M requires that we compute in iteration i

$$\mathbf{x}_i = (1 - \lambda) \cdot Q^{\mathsf{T}} \cdot \mathbf{x}_{i-1} + \lambda \cdot \mathbf{v}$$

Note that here we used the fact that M (from equation (1)) is stochastic, and the initial vector $\mathbf{x}_0$ is a probability distribution. Thus all $\mathbf{x}_i$ will be probability distributions, which implies that $\|\mathbf{x}_i\|_1 = 1$ (this explains why $\mathbf{x}_{i-1}$ is dropped from the term $\lambda \cdot \mathbf{v}$ above; since $\lambda \cdot E^{\mathsf{T}} \cdot \mathbf{x}_{i-1} = \lambda \mathbf{v} \cdot \mathbf{e}^{\mathsf{T}} \cdot \mathbf{x}_{i-1} = \lambda \cdot \mathbf{v}$). Besides, it also implies that we do not need to normalize $\mathbf{x}_i$ throughout the computation. See Method 1 in Figure 2 for PageRank in the clear when we iterate over a fixed number N of iterations.

The problem though with the power method is that it could be unstable, i.e. floating point errors in the computation could affect the final outcome. In [7] the

Three Variants of PageRank

Input: λ, $\mathbf{v}$, $\mathbf{x}_0$, $\epsilon_{\mathsf{abs}}/\epsilon_{\mathsf{rel}}$, Q, N. For the secure variants Q and $\mathbf{v}$ may be represented in secret shared form.
Output: The PageRank vector $\boldsymbol{\pi}$ of M from equation (1)

Method 1: Using Standard Floating Point (in clear)

1. $C \leftarrow (1-\lambda) \cdot Q^{\mathsf{T}}$.
2. For i in $1, \ldots, N$
   (a) $\mathbf{x}_i \leftarrow C \cdot \mathbf{x}_{i-1} + \lambda \cdot \mathbf{v}$
3. $\boldsymbol{\pi} \leftarrow \mathbf{x}_N$

Method 2: Using Fixed Point Arithmetic (in MPC or in the clear)

1. $\langle\ \overline{C}\ \rangle \leftarrow (1-\lambda) \cdot \langle\ \overline{Q}\ \rangle^{\mathsf{T}}$.
2. For i in $1, \ldots, N$
   (a) For l in $1, \ldots, m$
      i. $\langle\ \overline{y_i^{(l)}}\ \rangle \leftarrow \mathsf{dot\text{-}product}(\langle\ \overline{\mathbf{c}_l}\ \rangle, \langle\ \overline{\mathbf{x}_{i-1}}\ \rangle)$
      ii. $\langle\ \overline{x_i^{(l)}}\ \rangle \leftarrow \langle\ \overline{y_i^{(l)}}\ \rangle + \lambda \cdot \langle\ \overline{v^{(l)}}\ \rangle$
3. $\langle\ \overline{\boldsymbol{\pi}}\ \rangle \leftarrow (\langle\ \overline{\mathbf{x}^N}\ \rangle)$

Method 3: Using Fixed Point Arithmetic and Loop Truncation (in MPC or in the clear)

1. $\langle\ \overline{C}\ \rangle \leftarrow (1-\lambda) \cdot \langle\ \overline{Q}\ \rangle^{\mathsf{T}}$.
2. For l in $1, \ldots, N$
   (a) For l in $1, \ldots, m$
      i. Parties compute $\langle\ \overline{y_i^{(l)}}\ \rangle \leftarrow \mathsf{dot\text{-}product}(\langle\ \overline{\mathbf{c}_l}\ \rangle, \langle\ \overline{\mathbf{x}_{i-1}}\ \rangle)$
      ii. $\langle\ \overline{x_i^{(l)}}\ \rangle \leftarrow \langle\ \overline{y_i^{(l)}}\ \rangle + \lambda \cdot \langle\ \overline{v^{(l)}}\ \rangle$.
   (b) $\langle g \rangle \leftarrow (\|\langle\ \overline{\mathbf{x}_{i-1}}\ \rangle - \langle\ \overline{\mathbf{x}_i}\ \rangle\|_2^2 < \epsilon_{\mathsf{rel}}^2 \cdot \|\langle\ \overline{\mathbf{x}_{i-1}}\ \rangle\|_2^2$ if using relative error) or $\langle g \rangle \leftarrow (\|\langle\ \overline{\mathbf{x}_{i-1}}\ \rangle - \langle\ \overline{\mathbf{x}_i}\ \rangle\|_2^2 < \epsilon_{\mathsf{abs}}^2$ if using absolute error).
   (c) Open $\langle g \rangle$
   (d) If $g = 1$ then break.
3. $\langle\ \overline{\boldsymbol{\pi}}\ \rangle \leftarrow (\langle\ \overline{\mathbf{x}_i}\ \rangle)$ and Open $\boldsymbol{\pi}$ to all parties.

Figure 2. Our Various PageRank Algorithms

authors proved that solving the PageRank problem is equivalent to solving the matrix equation $R \cdot \mathbf{y} = \mathbf{v}$, where $R = I - (1-\lambda) \cdot P^{\mathsf{T}}$ and then normalizing y to obtain $\boldsymbol{\pi}$ via $\boldsymbol{\pi} = \mathbf{y}/\|\mathbf{y}\|_1$. The authors of [7] also bound the condition number of R (with respect to the 1-norm)

$$\kappa(R) \leq \|R^{-1}\|_1 \cdot \|R\|_1 \leq \frac{2-\lambda}{\lambda},$$

where $\|R\|_1 = \mathsf{max}_{1 \leq j \leq m} \sum_{i=1}^{m} |r_{i,j}|$. A similar result is given in [19] for the case when the diagonal entries of Q are all null.

The condition number explains how numerical errors in data can propagate after doing computation on it to errors in the result of this computation. Thus as λ approaches zero any algorithm to solve the PageRank problem will likely become unstable; this explains the traditional choice of $0.1 \leq \lambda \leq 0.2$.

4.2 Stability due to Approximate Computations

MPC systems based on linear secret sharing usually work on values defined in a finite field $\mathbb{F}_p$ of large prime characteristic, e.g. a prime p of size 128 bits. A data item x secret shared among the parties is denoted as $\langle x \rangle$. Calculation is then performed by expressing the computation in terms of additions, multiplications, and openings over $\mathbb{F}_p$. Linear operations on shared values is essentially for free, whereas multiplications typically require some pre-processed data and communication. We extend the notation of secret shared values to vectors of shared values $\langle \mathbf{x} \rangle$ and matrices of shared values $\langle A \rangle$.

In our application to PageRank we will need to compute mainly dot-products between vectors of fixed point values, i.e.

$$\langle\, \overline{\mathbf{x}} \,\rangle^{\mathsf{T}} \cdot \langle\, \overline{\mathbf{y}} \,\rangle = \sum_{j=1}^{m} \langle\, \overline{x^{(j)}} \,\rangle \cdot \langle\, \overline{y^{(i)}} \,\rangle.$$

The dot-product is one of the problems which have been previously studied in the MPC literature; see [4] for over the integers, [29] and [25] for over fixed point values, and [18] for over floating point values. For fixed point values recall that addition is cheap, but multiplication is expensive. Indeed the most expensive part of multiplication is the truncation step. In [29], authors used the two-party ABY framework [10], which allows to do conversions between linear secret sharing and garbled circuit. Their strategy consisted of converting to garbled circuit after each multiplication between fixed point numbers, in order to perform the truncation as garbled circuits are well suited for this latter operation. As well as being focused on the two-party passively secure case, this method introduces to the computation the cost of converting to-and-from the secret shared form. In [25], authors proposed a passively secure protocol for fixed point multiplications, for a setting of three-parties assuming an honest majority, and showed how we can use an optimization to perform the dot-product. The core idea of this optimization consists of performing the truncation step only after that all

the necessary additions have been calculated. Authors also proposed an actively secure protocol under the same setting for matrix by matrix products. The protocol works by pre-processing matrix triples (U, V, W), s.t. $W = U \cdot V$, using the fixed point multiplication protocol of [25], and use a generalization of [13] to perform matrix by matrix product using the pre-processed matrix triples.

In contrast our work is focused on the many party, actively secure. We perform all computation with linear secret sharing, however, we introduce a similar optimization to [25] into the procedure for executing a dot-product. This optimization is given in Figure 3 and we refer to the algorithm as $\mathsf{dot\text{-}product}(\langle\, \overline{\mathbf{x}} \,\rangle, \langle\, \overline{\mathbf{y}} \,\rangle)$.

Optimized Dot-Product

Input: $\langle\, \overline{\mathbf{x}} \,\rangle, \langle\, \overline{\mathbf{y}} \,\rangle$, **Output:** $\langle\, \overline{z} \,\rangle = \sum_{j=1}^{m} \langle\, \overline{x^{(j)}} \,\rangle \cdot \langle\, \overline{y^{(j)}} \,\rangle = \mathsf{dot\text{-}product}(\langle\, \overline{\mathbf{x}} \,\rangle, \langle\, \overline{\mathbf{y}} \,\rangle)$.

- $\langle s \rangle \leftarrow 0$.
- For $j \in [1, \ldots, m]$ do
  - $\langle s \rangle \leftarrow \langle s \rangle + \langle x^{(j)} \rangle \cdot \langle y^{(j)} \rangle$.
- $\langle z \rangle \leftarrow \mathsf{TruncPr}(\langle s \rangle, 2 \cdot k, f)$.

Figure 3. Optimized Dot-Product of Vectors of Shared Fixed Point Values

The method works as [25] by delaying the necessary truncation until all additions have been performed. Since truncation is the expensive part of the procedure, this basically produces a $1/m$ performance improvement. Ignoring the fact that our truncation procedure can be incorrect by a single bit (which occurs for normal fixed point multiplications in any case), we need to ensure that the output of this procedure is correct. Indeed, as we apply TruncPr less, we will introduce less errors in the truncation procedure overall.

The correctness depends to some extent on our precise application. In the PageRank algorithm using fixed point approximations (Method 2 in Figure 2) we apply dot-product on vectors $\langle\, \overline{\mathbf{c}} \,\rangle$ and $\langle\, \overline{\mathbf{x}} \,\rangle$ such that each entry in $\langle\, \overline{\mathbf{c}} \,\rangle$ is a positive value bounded by $1 - \lambda$, and each entry in $\langle\, \overline{\mathbf{x}} \,\rangle$ is a positive value bounded by one. Thus we can assume that the integers representing the fixed point values satisfy $0 \leq c^{(j)}, x^{(j)} \leq 2^f$, and in fact we have $\|\mathbf{c}\|_\infty \leq 2^f$ and $\|\mathbf{x}\|_1 \leq 2^f$. We then find a bound on s as

$$s = \sum_{j=1}^{m} c^{(j)} \cdot x^{(j)} \leq \|\mathbf{c}\|_\infty \cdot \|\mathbf{x}\|_1 \leq 2^{2 \cdot f}.$$

This means that the intermediate sum value s, as an integer, will be at most $2^{2 \cdot f}$, thus (since $k > f$) the application of TruncPr, using second parameter of $2 \cdot k$, will be correct assuming we can deal with the expansion needed in TruncPr due to the statistical security parameter κ. Thus, we additionally require $2 \cdot k + \kappa < \log_2 p$, Hence, we require exactly the same correctness requirement as we have for standard multiplication.

The only remaining parameter which needs to be set for our fixed point representation is the value k. It is easily seen that all vectors in the algorithm

consist of positive values less than one. Indeed the only place we utilize values outside the range $[0, \ldots, 1]$ is in computing the 2-norms (which we will do in Method 3 to be discussed later), where we utilize values in the range $[-1, \ldots, 1]$. To cope with these negative values we set $k = f + 1$.

Using fixed point representation, instead of floating point representation, hence does not affect the stability of PageRank, as long as the precision chosen is big enough to handle the computation taking place. However, for an expander graph we would expect the final PageRank vector $\boldsymbol{\pi}$ to be uniform, and thus have entries of the form $1/m$ in each coordinate. Thus we need to cope with an output which might have entries all close to $1/m$. To cope with this possibility in fixed point representation we need to make f a function of m. In particular we set $f = 30 + \log_2 m$ so that entries which are around $1/m$ can have around nine decimal digits of 'interesting' data. We can show that this strategy of setting f works through the following experiment, which compares the two first methodologies presented in Figure 2. We think of operations on values $\langle\ \overline{x}\ \rangle$ as being (for the time being) not on secret shared values but on fixed point values with the above representation, i.e. on $\overline{x}$ alone. Take a random graph $G^{m,l}$ of size m and number of links l, and run one hundred iterations of PageRank using floating point representation to obtain $\mathbf{x}_{100}$, and then run one hundred iterations on the same graph using the fixed point representation (in the clear) to obtain $\overline{\mathbf{z}_{100}}$.

We generated graphs for various values of with m between 100 and 10000, and for number of links $l = i \cdot m$ for $i = 2, \ldots, 40$. For each (m, l) considered we generated a set of $T = 100$ graphs $S^{m,l} = \{G_k^{m,l} \text{ for } k \in \{1, \ldots, T\}\}$ using the `NetworkX` package of `python3`. We then computed for each set the maximum error observed

$$e_{\max}^m = \max_{G \in \{S^{m,2} \cup \ldots \cup S^{m,40}\}} e_G,$$

where

$$e_G = \|\overline{\mathbf{z}_{100}} - \mathbf{x}_{100}\|_\infty = \max_{j \in \{1,\ldots,m\}} \left| \overline{\mathbf{z}_{100}^{(j)}} - \mathbf{x}_{100}^{(j)} \right|.$$

with the results given in Table 1. As expected, the error induced by using fixed point representation is negligible, thanks to how we chose f for the experiments.

m	100	500	1000	5000	10000
$e_{\max}^m$	1.1e-10	4.2e-11	1.5e-11	5.7e-12	2.3e-12

Table 1. Max error observed for the PageRank vector between fixed point and floating point representations (i.e. comparing Method 1 to Method 2 in Figure 2).

5 Effect of Early Termination of PageRank

If we examine Method 3 of Figure 2 we now terminate the main loop when the error meets a given condition. We use the square of the 2-norm for the

terminating conditions as this will be easier to implement securely in our MPC system; since there is no need for costly absolute values as with the 1-norm and no need for costly square roots as with the non-squared 2-norm.

We define two conditions, one defined by an absolute error

$$\|\langle\ \overline{\mathbf{x}_{i-1}}\ \rangle - \langle\ \overline{\mathbf{x}_i}\ \rangle\|_2^2 < \epsilon_{\mathsf{abs}}^2.$$

and one defined by a relative error

$$\|\langle\ \overline{\mathbf{x}_{i-1}}\ \rangle - \langle\ \overline{\mathbf{x}_i}\ \rangle\|_2^2 < \epsilon_{\mathsf{rel}}^2 \cdot \|\langle\ \overline{\mathbf{x}_{i-1}}\ \rangle\|_2^2$$

Note, that since $\|\mathbf{x}_i\|_1 = 1$ the relative error implies the absolute error bound in the case when $\epsilon_{\mathsf{rel}} = \epsilon_{\mathsf{abs}}$. However, we will be choosing $\epsilon_{\mathsf{rel}} \neq \epsilon_{\mathsf{abs}}$ in such a way that

$$\epsilon_{\mathsf{abs}}^2 < \epsilon_{\mathsf{rel}}^2 \cdot \|\langle\ \overline{\mathbf{x}_{i-1}}\ \rangle\|_2^2.$$

This should enable using the relative error to produce almost as accurate a solution, but with fewer iterations.[4]

Recall the speed of the convergence of the power method follows a geometric distribution with ratio $|\lambda_2/\lambda_1|$. Namely if λ_2 is close to λ_1 then the method converges slowly. As we are dealing with stochastic matrices we already know that $\lambda_1 = 1$, i.e. for a given dimension m, thus the number of iterations needed is proportional to $|\lambda_2|$. As remarked earlier, for graphs with at least two irreducible closed subsets one has $|\lambda_2| = 1-\lambda$ [16]. Recall a closed subset S is one in which if $x \in S$ and y can be reached from x, then y is also in S. A closed set is irreducible if it contains no proper closed subset, i.e. there is a path between each pair of elements in S.

For the traditional PageRank case, i.e. the application to the internet graph, the underlying graph does indeed have at least two irreducible closed subsets [5]. In addition experimentally it has been shown that the power method produces the correct value $\boldsymbol{\pi}$ (assuming no floating point errors accumulate) for iteration values between 50 and 100 [23].

For graphs generated uniformly at random (called random graphs from now on) $|\lambda_2|$ is not necessarily as big as $1 - \lambda$ (recall λ is between 0.1 and 0.2 and in the case of the original PageRank algorithm $\lambda = 0.15$). Therefore one may need fewer iterations to ensure convergence. By inserting the abort-test into Method 3 we potentially improve the performance of PageRank, but at the same time we leak the number of iterations needed to achieve our level of convergence. See below for experimental validation of this for random graphs. As explained in Section 3 the number of iterations N leaked, for the absolute error variant, implies information is leaked about the second eigenvalue λ_2 and $\mathbf{x}_0$.

However, for transaction graphs for bank accounts, one cannot tell whether there are at least two irreducible subsets. In [33], authors studied the topology

[4] In practice it is easy to choose ϵ_{abs} and ϵ_{rel} satisfying the above. Observe that the $\|\bullet\|_2$ norm on the hypercube given by equation $\|\mathbf{x}\|_1 = 1$ attains its minimum $\frac{1}{m}$ at the points $\left(\pm\frac{1}{m}, \ldots, \pm\frac{1}{m}\right)$. Therefore it suffices to choose any $\epsilon_{\mathsf{abs}} < \epsilon_{\mathsf{rel}}$ such that $\epsilon_{\mathsf{abs}}^2 < \epsilon_{\mathsf{rel}}^2 \cdot \frac{1}{m}$. This way the choice of the errors will be independent of the sequence.

of the daily graphs, of the interbanks payments transferred between a set of participants (commercial banks) of the Fedwire Funds Service, corresponding to the first quarter of 2004. The structure of the underlying graph observed shows that the degree distribution corresponds to a scale free graph, and from the results they obtained, one can conclude that over at least 50% of the days, there exist at least two irreducible subsets.

For our second set of experiments we generated graphs according to the simulator of transaction graphs from [34]; in what follows we call these banking graphs. This simulator generates scale free graphs, in particular, following the Barabasi-Albert model, with a tweak over the strength of the preferential attachment. The resulting graphs as the experiments will show, do not contain two closed subsets.

To choose the tolerance $\epsilon_{\mathsf{abs}}/\epsilon_{\mathsf{rel}}$, one needs to consider how small the components of $\mathbf{x}$ are, as it may occur that $\|\mathbf{x}_i - \mathbf{x}_{i-1}\|_2^2$ triggers the abort due to the fact that the components of $\mathbf{x}_i$ and $\mathbf{x}_{i-1}$ are small (if the tolerance was not chosen to be small enough) while there could be still room for convergence. As explained earlier if the vector $\boldsymbol{\pi}$ was uniform then we would expect each coordinate to be $1/m$, thus for absolute errors it makes sense to have the tolerance depend on m; just as we made f depend on m in the previous section. An alternative approach, which we examine, is to instead look at relative errors instead of the absolute errors, in which case the effect of small values in $\boldsymbol{\pi}$ is already accounted for by taking relative errors. Thus we always set $\epsilon_{\mathsf{rel}} = 2^{-10}$ irrespective of m, i.e. we want our two final iterations to be within 0.1 percent of each other. For the absolute error we set $\epsilon_{\mathsf{abs}} = 2^{-f/2}$, and thus we terminate when the $\|\langle\, \overline{\mathbf{x}_{i-1}} \,\rangle - \langle\, \overline{\mathbf{x}_i} \,\rangle\|_2^2$ is identically equal to zero in our fixed point representation.

To verify that the early termination indeed provides an efficiency improvement, and does not affect the overall accuracy of the output compared to non-termination, we compared Method 1 against Method 3 using the same type of experiments, in the clear, as performed in Section 4; for both random and our simulated banking graphs. We computed the average number of iterations N needed to obtain the required termination condition (when we set ϵ_{abs} and ϵ_{rel} as above), and compared the result with the values which would have been obtained in the clear. The accuracy was measured according to the metric $e^m_{\max}$ from the previous section, the results being given in Table 2.

From the results of these experiments, it is clear that one does not need to run many iterations before obtaining convergence, for both the cases of random graphs and banking graphs. In particular, for a randomly generated graph, we can see that the more links within the graph, the fewer iterations are needed to reach convergence. We can also see that the banking graphs take longer time to process compared to the random graphs the more links we have. Besides, the experiments show that using the absolute error to test convergence, requires running more iterations to obtain convergence than using the relative error. This is due to the fact that the epsilon chosen ϵ_{abs}, implies that convergence is only obtained when $\|\langle\, \overline{\mathbf{x}_{i-1}} \,\rangle - \langle\, \overline{\mathbf{x}_i} \,\rangle\|_2^2$ is equal to zero as explained earlier, while for the case of the relative error, convergence can happen without necessarily having

			Number of Links									
			Random Graphs					Banking Graphs				
m	Error		$2 \cdot m$	$5 \cdot m$	$10 \cdot m$	$20 \cdot m$	$40 \cdot m$	$2 \cdot m$	$5 \cdot m$	$10 \cdot m$	$20 \cdot m$	$40 \cdot m$
100	abs	N	21	13	9	7	5	16	12	11	9	11
		$e^m_{\max}$	3.1e-6	6.3e-7	1.8e-7	7.3e-8	6.5e-8	6.8e-6	7.2e-6	4.2e-6	5.4e-7	3.3e-7
100	rel	N	15	9	6	5	4	11	8	7	6	7
		$e^m_{\max}$	3.2e-4	1.5e-5	7.6e-6	5.3e-6	5.1e-7	4.5e-4	4.1e-4	2.8e-4	3.6e-5	2.0e-5
500	abs	N	21	13	9	7	5	15	11	9	9	13
		$e^m_{\max}$	3.3e-6	4.3e-7	2.9e-7	5.1e-8	4.8e-8	1.5e-6	9.9e-7	8.3e-7	1.0e-6	2.8e-7
500	rel	N	15	9	6	5	4	10	8	7	7	9
		$e^m_{\max}$	2.9e-4	1.6e-5	3.0e-6	4.9e-7	2.1e-7	3.9e-5	2.8e-5	1.5e-5	1.7e-5	1.2e-5
1000	abs	N	21	13	9	7	6	17	12	10	10	15
		$e^m_{\max}$	4.5e-6	3.6e-7	5.2e-8	8.6e-9	3.4e-9	2.6e-6	2.4e-6	3.3e-7	6.7e-8	8.3e-8
1000	rel	N	15	9	6	5	4	11	8	7	7	10
		$e^m_{\max}$	1.5e-4	9.7e-6	1.0e-6	2.3e-7	1.4e-7	1.8e-4	1.6e-4	1.4e-5	7.2e-6	1.0e-5
5000	abs	N	21	12	8	7	5	15	11	10	10	15
		$e^m_{\max}$	3.9e-6	9.8e-8	3.3e-8	5.1e-9	3.4e-9	2.7e-6	2.7e-7	1.5e-7	3.8e-7	1.0e-7
5000	rel	N	14	9	6	5	4	10	8	7	7	10
		$e^m_{\max}$	9.1e-5	6.2e-7	1.9e-7	8.9e-8	7.8e-8	7.0e-5	6.9e-6	4.5e-6	6.8e-6	6.9e-6
10000	abs	N	21	12	8	6	5	16	12	10	10	16
		$e^m_{\max}$	3.8e-6	8.5e-8	1.2e-8	5.3e-9	3.1e-9	8.1e-7	1.5e-7	5.0e-8	8.5e-8	2.4e-8
10000	rel	N	14	9	6	5	4	11	8	7	7	10
		$e^m_{\max}$	4.5e-5	4.9e-7	1.8e-7	3.7e-8	2.1e-8	5.6e-5	5.0e-6	2.9e-6	2.7e-6	6.4e-6

Table 2. Iterations needed for convergence and accuracy of the result for the PageRank algorithm using Method 3 on random and banking graphs with absolute and relative errors, with respect to the number of nodes and links in the graphs tested.

it. As for the difference between the PageRank vector obtained after convergence, and the PageRank vector obtained after one hundred iterations, we can see that for both cases of using the absolute error and the relative error, the difference is very small. Besides, we can also see that this difference is slightly bigger for the case of the relative error. This would imply that considering a termination condition for the PageRank algorithm, either with an absolute error or a relative error set as specified in this section, would produce a fairly close PageRank vector to the one produced after running one hundred iteration.

6 A multiparty actively-secure protocol for the PageRank algorithm

Recall our motivating application of using PageRank for financial fraud detection. We discussed earlier how the more institutions which are involved the better the analysis will be. This means that the methodology of [30] which requires all financial institutions which contribute data to be involved in the protocol may not scale to the large number of institutions required in a cross-border analysis of money laundering. Thus we look at a methodology in which a large number of financial institutions $\{I_1, \ldots, I_u\}$ wish to apply PageRank over the bank accounts they maintain. They will do this by securely distributing their data to a smaller set of computing servers $S = \{S_1, \ldots, S_v\}$, with $v = 2$ or $v = 3$. Our solution will allow us to arbitrarily scale u, but it results in us not being

able to deal with such a large matrix, i.e. number of accounts, as the prior work did [30], due to the fact that the whole matrix for our case is being secret shared, for reasons described in the introduction. It is interesting none-the-less to see the difference in performance between the two approaches.

The first stage of our algorithm is for each institution I_k to secret share its component of the PageRank matrix to the servers in S. This is simply a data-entry phase which we ignore in our analysis. Thus we start by assuming that the parties in S hold a secret sharing of the initial matrix $\langle\ \overline{Q}\ \rangle$ and the personalisation vector $\langle\ \overline{\mathbf{v}}\ \rangle$.

Our experiments are performed using Scale-Mamba [3], which is an MPC framework that utilizes the above secret sharing based methodology and in addition already has a number of built in routines for dealing with the above fixed point representation. Scale-Mamba is an actively secure framework with abort, which means that it offers the strong security guarantee that if an adversary deviates from the protocol then it is detected with overwhelming probability.

The system works in an offline-online manner. In particular in the function independent offline phase pre-processed data is generated, such as so-called Beaver triples (random triples shared values $\langle a\rangle$, $\langle b\rangle$, and $\langle c\rangle$ s.t. $c = a \cdot b$) and random bits (shared values $\langle b\rangle$ s.t. $b \in \{0, 1\}$). In the online phase the actual computation takes place, during this phase the pre-processed random data is consumed. The main metric for measuring cost in the online phase is the number of rounds of communication required by an operation. Whilst the online phase is relatively fast, the offline phase can be an order of 10 to 100 times slower.

To summarize the cost of the different operations we need in terms of pre-processed data, we present Table 3; where $h(k)$ is the function $\sum_{i=1}^{\lceil log_2(k)\rceil} g(i)$, for $g(i) = f(i) - 2 \cdot \left(\frac{f(i-1)}{2} \mod 2\right) - 1$ and $f(i+1) = \frac{f(i)}{2} + \left(\frac{f(i)}{2} \mod 2\right)$ and $f(0) = 2 \cdot k$ (see [3] for details about the function h). Note that while we provided the rounds of communication required by each operation we need, these can be merged if many operations can be performed in parallel.

Operation	Open	$\langle a\rangle \cdot \langle b\rangle$	$\langle\ \overline{a}\ \rangle \cdot \langle\ \overline{b}\ \rangle$	$\langle\ \overline{a}\ \rangle < \langle\ \overline{b}\ \rangle$	TruncPr$(\langle a\rangle, k, f)$	Trunc$(\langle a\rangle, k, f)$
No. Triples	0	1	1	$h(k)$	0	$h(k)$
No. Bits	0	0	$2 \cdot k + \kappa$	$k + \kappa$	$k + \kappa$	$k + \kappa$
Rounds	1	1	2	$\lceil \log_2 k \rceil + 1$	1	$\lceil \log_2(k) \rceil + 1$

Table 3. Costs of Basic Scale-Mamba Operations over Integers

A key parameter of an MPC system is the number of parties in the system, and the number of 'bad' players which can be tolerated. Scale-Mamba supports various options for these access structures. Each one coming with different advantages and disadvantages. To illustrate our protocol we focus on two cases:

1. Two party protocol with one active corruption. Here the Scale-Mamba system makes use of the SPDZ protocol [8]. The offline phase is roughly 180 times slower than the online phase, but the online phase is very fast.
2. Three party protocol with one active corruption. Here we utilize a secret sharing scheme based on Shamir sharing [31]. In this case Scale-Mamba implements the protocol of [20] to obtain a fast online phase, at the expense of having an offline phase which is roughly 4 times slower.

We implemented PageRank within the Scale-Mamba system and then run various experiments, with different transaction graphs to see how performance behaved. We varied the value of m to range from around $m = 100$ to $m = 10000$. We fixed the initialization vector $\mathbf{x}_0$ and the personalization vector $\mathbf{v}$ to be the vectors with $1/m$ in each entry, although modifying our code to deal with secret shared value $\mathbf{v}$ is trivial. Finally we fixed λ to be 0.15 in the PageRank algorithm itself, modeling the damping factor that the institutions would use, but of course the institutions can choose any value they wish prior to executing PageRank. For our approximation of floating point by fixed point numbers considered earlier we used $f = 30 + \log_2 m$, $k = f + 1$, $\kappa = 40$, and a modulus of $2 \cdot k + \kappa$ bits.

The most expensive part (by a large margin) of the entire procedure is the execution of Step 2a in Method 3 of Figure 2. For single execution of this step we present our runtimes in Table 4, in addition to the amount of data sent per party. Our experiments were run on Intel i-9900 CPU based machines with 128GB of RAM, connected by a local network with a ping time around 0.048 milliseconds, connected with a switch of bandwidth 1 Gb. We notice that the case of the two parties is slightly faster than the case of three parties.

m	100	500	1000	5000	10000
Two parties	0.03	0.40	2.11	55.23	231.61
Three parties	0.03	0.61	2.42	60.82	245.34
Data sent	1.56	14.50	45.19	1305.93	5014.23

Table 4. Average online runtimes in seconds for Step 2a in Method 3 of Figure 2 for one iteration of the PageRank algorithm for two players and three players with respect to the number of nodes m, as well as the size of data sent by each player in MB.

Thus we have the expected time for execution of a single iteration of the PageRank algorithm. To obtain the final runtime we need to multiply this by the expected number of iterations, from Table 2, to obtain Table 5. Here we can see the effect of the terminating condition on the runtime. Without our terminating condition we would need to run for a large number (say 100) of iterations, which is costly. By terminating after a suitable convergence has been reached we save a lot of time, *but* we leak some information. However, as we have explained the information leaked is essentially only information about the spectral gap; which may be considered a minor leakage depending on the application.

m	Setting	Runtimes 100 iteration	Random Graphs abs	Random Graphs rel	Banking Graphs abs	Banking Graphs rel
100	Two parties	3	0.6	0.4	0.5	0.3
100	Three parties	3	0.6	0.4	0.5	0.3
500	Two parties	40	8.4	6.0	6.0	4.0
500	Three parties	61	12.8	9.1	9.1	6.1
1000	Two parties	210	44.1	31.5	35.7	23.1
1000	Three parties	242	50.8	36.3	41.1	26.6
5000	Two parties	5500	1161.3	774.2	829.5	553.0
5000	Three parties	6070	1274.7	849.8	910.5	607.0
10000	Two parties	23100	4855.2	3236.8	3699.2	2543.2
10000	Three parties	24530	5151.3	3434.2	3924.8	2698.3

Table 5. Average runtimes in seconds for the PageRank algorithm of Figure 2 for Method 2, as well as Method 3 with absolute and relative errors on random and banking graphs, for two parties and three parties with respect to the number of nodes m.

Acknowledgments

The authors would like to thank Dragoş Rotaru and Titouan Tanguy, for suggestions in relation to this paper, and Frederik Vercauteren for the helpul discussions in the early stages of this work. This work was supported in part by CyberSecurity Research Flanders with reference number VR20192203, by ERC Advanced Grant ERC-2015-AdG-IMPaCT, by the Defense Advanced Research Projects Agency (DARPA) and Space and Naval Warfare Systems Center, Pacific (SSC Pacific) under contract No. FA8750-19-C-0502, and by the FWO under an Odysseus project GOH9718N. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the ERC, DARPA, the US Government or the FWO. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

1. Alon, N., Milman, V.: Lambda1, isoperimetric inequalities for graphs, and superconcentrators. Journal of Combinatorial Theory, Series B 38(1), 73 – 88 (1985), http://www.sciencedirect.com/science/article/pii/0095895685900929
2. Alon, N.: Eigenvalues and expanders. Combinatorica 6(2), 83–96 (Jun 1986)
3. Aly, A., Cong, K., Keller, M., Orsini, E., Rotaru, D., Scherer, O., Scholl, P., Smart, N.P., Tanguy, T., Wood, T.: SCALE and MAMBA v1.9: Documentation (2020), https://homes.esat.kuleuven.be/~nsmart/SCALE/Documentation.pdf
4. Bogdanov, D., Laur, S., Willemson, J.: Sharemind: a framework for fast privacy-preserving computations. Cryptology ePrint Archive, Report 2008/289 (2008), http://eprint.iacr.org/2008/289
5. Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., Wiener, J.: Graph structure in the web (2000)
6. Catrina, O., Saxena, A.: Secure computation with fixed-point numbers. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 35–50. Springer, Heidelberg (Jan 2010)

7. Corso, G.M.D., Gulli, A., Romani, F.: Fast PageRank computation via a sparse linear system (extended abstract). In: Leonardi, S. (ed.) Algorithms and Models for the Web-Graph: Third International Workshop, WAW 2004, Rome, Italy, October 16, 2004, Proceeedings. Lecture Notes in Computer Science, vol. 3243, pp. 118–130. Springer (2004), https://doi.org/10.1007/978-3-540-30216-2_10
8. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 643–662. Springer, Heidelberg (Aug 2012)
9. Daskalakis, C., Tzamos, C., Zampetakis, M.: A converse to Banach's Fixed Point Theorem and its CLS completeness. CoRR abs/1702.07339 (2017), http://arxiv.org/abs/1702.07339
10. Demmler, D., Schneider, T., Zohner, M.: ABY - A framework for efficient mixed-protocol secure two-party computation. In: NDSS 2015. The Internet Society (Feb 2015)
11. Dikland, T.: Added value of combining transaction graphs on fraud detection using the PageRank algorithm. Internship Report, TNO and TU Delft (2018)
12. Elden, L.: A note on the eigenvalues of the Google matrix (2004), http://arxiv.org/abs/math/0401177
13. Furukawa, J., Lindell, Y., Nof, A., Weinstein, O.: High-throughput secure three-party computation for malicious adversaries and an honest majority. In: Coron, J., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part II. LNCS, vol. 10211, pp. 225–255. Springer, Heidelberg (Apr / May 2017)
14. Han, Y., Kim, L., Cha, J.: Evaluation of user reputation on youtube. In: Ozok, A.A., Zaphiris, P. (eds.) Online Communities and Social Computing, Third International Conference, OCSC 2009, Held as Part of HCI International 2009, San Diego, CA, USA, July 19-24, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5621, pp. 346–353. Springer (2009), https://doi.org/10.1007/978-3-642-02774-1_38
15. Hastings, M., Falk, B.H., Tsoukalas, G.: Privacing preserving network analytics (2020), https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3680000
16. Haveliwala, T., Kamvar, S.: The second eigenvalue of the Google matrix. Technical Report 2003-20, Stanford InfoLab (2003), http://ilpubs.stanford.edu:8090/582/
17. Huynh, T.D.: Extension of PageRank and application to social networks. (Extension de PageRank et application aux réseaux sociaux). Ph.D. thesis, Pierre and Marie Curie University, Paris, France (2015), https://tel.archives-ouvertes.fr/tel-01187929
18. Kamm, L., Willemson, J.: Secure floating-point arithmetic and private satellite collision analysis. Cryptology ePrint Archive, Report 2013/850 (2013), http://eprint.iacr.org/2013/850
19. Kamvar, S., Haveliwala, T.: The condition number of the PageRank problem. Technical Report 2003-36, Stanford InfoLab (June 2003), http://ilpubs.stanford.edu:8090/597/
20. Keller, M., Rotaru, D., Smart, N.P., Wood, T.: Reducing communication channels in MPC. In: Catalano, D., De Prisco, R. (eds.) SCN 18. LNCS, vol. 11035, pp. 181–199. Springer, Heidelberg (Sep 2018)
21. Langville, A.N., Meyer, C.D.: Deeper inside PageRank. Technical report, Department of Mathematics, N. Carolina State University (2003)
22. Langville, A.N., Meyer, C.D.: Fiddling with PageRank. Technical report, Department of Mathematics, N. Carolina State University (2003)

23. Langville, A.N., Meyer, C.D.: Survey: Deeper inside PageRank. Internet Math. 1(3), 335–380 (2003), https://doi.org/10.1080/15427951.2004.10129091
24. Maxwell, N.: Innovation and discussion paper: Case studies of the use of privacy preserving analysis to tackle financial crime. Tech. rep., Royal United Services Institute (2020), https://www.future-fis.com/uploads/3/7/9/4/3794525/ffis_innovation_and_discussion_paper_-_case_studies_of_the_use_of_privacy_preserving_analysis.pdf
25. Mohassel, P., Rindal, P.: ABY^3: A mixed protocol framework for machine learning. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 35–52. ACM Press (Oct 2018)
26. Molloy, I., Chari, S., Finkler, U., Wiggerman, M., Jonker, C., Habeck, T., Park, Y., Jordens, F., van Schaik, R.: Graph analytics for real-time scoring of cross-channel transactional fraud. In: Grossklags, J., Preneel, B. (eds.) Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9603, pp. 22–40. Springer (2016), https://doi.org/10.1007/978-3-662-54970-4_2
27. Moreau, A.: How to perform fraud detection with personalized PageRank. Blog: https://www.sicara.ai/blog/2019-01-09-fraud-detection-personalized-page-rank (9 January 2019)
28. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: Bringing order to the Web. In: Proceedings of the 7th International World Wide Web Conference. pp. 161–172 (1998)
29. Riazi, M.S., Weinert, C., Tkachenko, O., Songhori, E.M., Schneider, T., Koushanfar, F.: Chameleon: A hybrid secure computation framework for machine learning applications. In: Kim, J., Ahn, G.J., Kim, S., Kim, Y., López, J., Kim, T. (eds.) ASIACCS 18. pp. 707–721. ACM Press (Apr 2018)
30. Sangers, A., van Heesch, M., Attema, T., Veugen, T., Wiggerman, M., Veldsink, J., Bloemen, O., Worm, D.: Secure multiparty PageRank algorithm for collaborative fraud detection. In: Goldberg, I., Moore, T. (eds.) Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers. Lecture Notes in Computer Science, vol. 11598, pp. 605–623. Springer (2019), https://doi.org/10.1007/978-3-030-32101-7_35
31. Shamir, A.: How to share a secret. Communications of the Association for Computing Machinery 22(11), 612–613 (Nov 1979)
32. Simić, S.K., Andelić, M., da Fonseca, C.M., Živković, D.: Notes on the second largest eigenvalue of a graph. Linear Algebra and its Applications 465, 262 – 274 (2015), http://www.sciencedirect.com/science/article/pii/S002437951400617X
33. Soramaki, K., Bech, M.L., Arnold, J., Glass, R.J., Beyeler, W.E.: The topology of interbank payment flows. Physica A: Statistical Mechanics and its Applications 379(1), 317 – 333 (2007), http://www.sciencedirect.com/science/article/pii/S0378437106013124
34. Soramaki, K., Cook, S.: Sinkrank: An algorithm for identifying systemically important banks in payment systems. Economics: The Open-Access, Open-Assessment E-Journal 7 (06 2013)
35. Vlasselaer, V.V., Eliassi-Rad, T., Akoglu, L., Snoeck, M., Baesens, B.: GOTCHA! Network-based fraud detection for social security fraud. Manag. Sci. 63(9), 3090–3110 (2017), https://doi.org/10.1287/mnsc.2016.2489

A Converses to Banach's Fixed Point Theorem

To understand the convergence of fixed point iterations of contraction mappings more precisely, we need to appeal to so-called converse to Banach's Theorem. These are results which show that if an iterative method converges for some set X on a metric space with metric d, then there is a (potentially different) metric d' for which the map is a contraction mapping for any Lipschitz constant q. In particular in [9] the following converse theorem is proved

Theorem A.1 (Theorem 1 of [9]). *Let (X, d) be a complete, proper metric space and $F : X \longrightarrow X$ be continuous with respect to d such that F has a unique fixed point x^*, and the iteration $x_i \leftarrow F(x_{i-1})$ converges to x^* with respect to d, and there exists an open neighbourhood U of x^* such that $F^{(n)}(U) \longrightarrow \{x^*\}$ as $n \longrightarrow \infty$.*

Then, for all $q \in (0, 1)$ and $\epsilon > 0$, there is a metric $d_{q,\epsilon}$ which is topologically equivalent to d, such that $(X, d_{c,\epsilon})$ is a complete metric space and

1. $\forall x, y \in X : d_{q,\epsilon}(\ f(x), f(y)\ ) \leq q \cdot d_{q,\epsilon}(\ x.y\ )$.
2. $\forall x, y \in X : d_{q,\epsilon}(\ x, y\ ) \leq \epsilon$ *implies that*

$$\min\{\ d_{q,\epsilon}(\ x^*, x\ ), d_{q,\epsilon}(\ x^*, y\ ), d_{q,\epsilon}(\ x, y\ )\ \} \leq 2 \cdot \epsilon.$$

The second property here is used to bound the number of iterations needed in terms of the constants q, ϵ and the distance $d(\ x_0, x^*\ )$.

The authors of [9] illustate this in terms of the power method for matrices. Indeed in [9][Proposition 1] it is shown that if we restrict to real matrices A (with eigenvalues $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_m|$) then there is a metric $d(\mathbf{x}, \mathbf{y})$ on $\mathcal{R}^m$ such that the mapping, for any vector norm $\|\cdot\|$,

$$F(\mathbf{x}) = \frac{A \cdot \mathbf{x}}{\|A \cdot \mathbf{x}\|}$$

is a contraction mapping with

$$d(F(\mathbf{x}), F(\mathbf{y})) \leq \frac{|\lambda_2|}{|\lambda_1|} \cdot d(\mathbf{x}, \mathbf{y})$$

for all $\mathbf{x}, \mathbf{y} \in \mathcal{R}^m$. The metric being

$$d(\mathbf{x}, \mathbf{y}) = \left\| \frac{\mathbf{x}}{\mathbf{x}^\mathsf{T} \cdot \mathbf{v}_1} - \frac{\mathbf{y}}{\mathbf{y}^\mathsf{T} \cdot \mathbf{v}_1} \right\|_2.$$

Thus *there is* a metric for which the Lipschitz constant *is* $q = |\lambda_2|/|\lambda_1|$. In addition for any $\mathbf{x}_0 \in \mathcal{R}^n$ which has a nonzero component in the direction of $\mathbf{v}_1$, we have that after

$$N^* = \frac{\log(d(\ \mathbf{x}_0, \mathbf{v}_1\ )/\epsilon)}{\log(|\lambda_1|/|\lambda_2|)}$$

steps we have $\|\mathbf{x}_{N^*} - \mathbf{v}_1\|_2 \leq d(\ \mathbf{x}_{N^*}, \mathbf{v}_1\ ) \leq \epsilon$. This allows us to upperbound the number of iterations needed for a given level of convergence. However, the metic is not suitable to use within the algorithm to determine the first eigenvalue/eigenvector as it depends on the value of the first eigenvector itself.