Solving Systems of Linear Equations: Locally and Asynchronously

Christina E. Lee celee@mit.edu
Asuman Ozdaglar asuman@mit.edu
Devavrat Shah devavrat@mit.edu

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology

November 12, 2014

Abstract

We consider approximating a single component of the solution to a system of linear equations $Ax = b$, where A is an invertible real matrix and $b \in \mathbb{R}^n$. If A is either diagonally dominant or positive definite, we can equivalently solve for x_i in $x = Gx + z$ for some G and z such that spectral radius $\rho(G) < 1$. For example, if A is symmetric positive definite, there is a transformation such that $\rho(G) = (\kappa(A) - 1)/(\kappa(A) + 1)$, where $\kappa(A)$ is the condition number of A. Existing algorithms either focus on computing the full vector x or use Monte Carlo methods to estimate a component x_i under the condition $\|G\|_\infty < 1$. We consider the setting where n is large, yet G is sparse, i.e., each row has at most d nonzero entries. We present synchronous and asynchronous randomized variants of a local algorithm which relies on the Neumann series characterization of the component x_i, $e_i^T \sum_{k=0}^{\infty} G^k z$, and allows us to limit the sparsity of the vectors involved in the computation, leading to improved convergence rates. Both variants of our algorithm produce an estimate $\hat{x}_i$ such that $|\hat{x}_i - x_i| \leq \epsilon \|x\|_2$, and we provide convergence guarantees when $\|G\|_2 < 1$, thus encompassing a larger class of systems. We prove that the synchronous local algorithm uses at most $O(\min(d\epsilon^{\ln(d)/\ln(\|G\|_2)}, dn\ln(\epsilon)/\ln(\|G\|_2)))$ multiplications. The asynchronous local algorithm adaptively samples one coordinate to update among the nonzero coordinates of the current iterate in each time step. We prove with high probability that the error contracts by a time varying factor in each step, guaranteeing that the algorithm converges to the correct solution. With probability at least $1 - \delta$, the asynchronous randomized algorithm uses at most $O(\min(d(\epsilon\sqrt{\delta/5})^{-d/(1-\|G\|_2)}, -dn\ln(\epsilon\sqrt{\delta})/(1 - \|G\|_2)))$ multiplications. Thus our algorithms obtain an approximation for x_i in constant time with respect to the size of the matrix when $d = O(1)$ and $1/(1 - \|G\|_2) = O(1)$ as a function of n, which holds for sparse expanders.

1 Introduction

We consider approximating the i^{th} component of the solution to the linear system of equations

$$Ax = b, \tag{1}$$

where A is a nonsingular $n \times n$ real matrix, and b is a vector in $\mathbb{R}^n$. Under some generic conditions on A, such as either positive definite or symmetric diagonally dominant (see details in Section 2.3), there exists an appropriate choice of G and z, such that this problem is equivalent to approximating the i^{th} component of the solution to

$$x = Gx + z, \tag{2}$$

where $\rho(G) < 1$.[1] We consider the setting where n is large, yet G is sparse, i.e., the number of nonzero entries in every row of G is at most d.

Solving large systems of linear equations is a problem of great interest due to its relevance to a variety of applications across science and engineering, such as solving large scale optimization problems, approximating solutions to partial differential equations, and modeling network centralities (e.g. PageRank and Bonacich centrality). Due to the large scale of these systems, it becomes useful to have a local algorithm which can approximate only a few components of the solution without computing over the entire matrix. As solving a system of linear equations is fundamentally a global problem, obtaining a local solution is non-trivial.

1.1 Contributions

In this paper, we propose and analyze two variants of an algorithm which provides an estimate $\hat{x}_i$ for a single component of the solution vector to (2), such that $|\hat{x}_i - x_i| \leq \epsilon \|x\|_2$. One variant is synchronous and the other is asynchronous and randomized. We show that our algorithm converges when $\|G\|_2 < 1$ and provide bounds on the convergence rate as a function of $\|G\|_2$, the sparsity d, the approximation factor ϵ, and the size of the matrix n.[2] Recall that when G is symmetric, $\rho(G) = \|G\|_2$, so our algorithm converges whenever the Neumann series is well defined. When $1/(1 - \|G\|_2) = O(1)$ and $d = O(1)$ with respect to n, our algorithm converges in time which is constant with respect to n. If A is symmetric positive definite, then there exists a choice of G and z which transforms the problem from (1) to (2) such that $\rho(G) = (\kappa(A) - 1)/(\kappa(A) + 1)$.[3] Our results for solving (1) extends to solving (1) with the corresponding relation between $\rho(G)$ and $\kappa(A)$.

Our algorithm relies upon the Neumann series representation of the solution to (2)

$$x = \sum_{k=0}^{\infty} G^k z. \tag{3}$$

The Neumann series converges to the solution of (2) if and only if $\rho(G) < 1$ [2, 4, 14]. We compute[4]

$$\sum_{k=0}^{\infty} (G^T)^k e_i, \tag{4}$$

[1]Let $\rho(G)$ denote the spectral radius of G, i.e., the maximum magnitude eigenvalue of G.

[2]Let $\|G\|_2$ denote the induced 2-norm of G, i.e., $\max_{\|v\|_2=1} \|Gv\|_2 = \sqrt{\rho(G^T G)}$.

[3]$\kappa(A)$ is the condition number of matrix A, i.e., the ratio between the largest and smallest magnitude eigenvalues.

[4]Let e_i denote the standard basis vector which has as 1 at coordinate i and 0 elsewhere.

and take the inner product with the vector z to obtain an estimate. This modification allows us to guarantee that intermediate vectors involved in our algorithm are sparse as long as G. This transforms the computation from global to local updates. Our algorithm provides the estimate and residual error at each iteration, and recursively refines the estimate while contracting the error, providing clear termination conditions.

The asynchronous variant implements the local algorithm by adaptively sampling a coordinate from the current iterate, and applying a local update involving the corresponding row from the matrix G. This implementation maintains the local properties (i.e., sparsity of involved vectors) of the algorithm by sampling uniformly among coordinates which have nonzero values in the current iterate. The asynchronous variant eliminates the coordination cost in the local updates at each coordinate.

Our method is closely related to work by Andersen *et al.* which focuses on computing PageRank, and provides an algorithm and analysis which rely on the conditions that G is a nonnegative scaled stochastic matrix, z is entry-wise positive and bounded strictly away from zero, and the solution x is a probability vector (i.e., consisting of nonnegative entries that sum to 1) [1]. In contrast, our algorithm focuses on general linear systems of equations. Moreover, while their analysis proves a linear decrease in the error, we prove that the second moment of our error contracts by a time dependent factor in each iteration, and thus our algorithm converges to the correct solution with a convergence rate which is similar to the synchronous variant. Our algorithm differs from the algorithm presented in Andersen *et al.* by a different choice of termination conditions, and the use of randomization in the update of every iteration.

Throughout the paper, we will associate a graph to the matrix G, and use the concept of computation over this graph to discuss our notion of a local and asynchronous algorithm. Let $\mathcal{G}(G) = (\mathcal{V}, \mathcal{E})$ denote the graph where $\mathcal{V} = \{1, 2, \dots n\}$, and $(a, b) \in \mathcal{E}$ if and only if $G_{ab} \neq 0$. Each coordinate in the vector x corresponds to a node in $\mathcal{V}$. Our assumption that the number of nonzero entries in any row of G is at most d translates to the condition that the outdegree of any node in $\mathcal{G}(G)$ is at most d. The distance from node i to node j is the shortest path from i to j in graph $\mathcal{G}(G)$. Let $N_i(t) \subset \mathcal{V}$ denote the local neighborhood of radius t around node i, which consists of the set of nodes within distance t from node i. The algorithm is "local" when the computation only involves nodes in $N_i(t)$ such that $|N_i(t)| < n$.

1.2 Related Work

The history of solving systems of linear equations is immense, so we will only give a broad summary of the types of existing methods in order to give an understanding of where our algorithm fits within the landscape of methods. We will proceed to highlight the methods which are relevant to our setting of approximating a single component x_i locally and asynchronously. We will also give specific attention to the methods which share similar concepts or approaches to our algorithm.

Most methods for solving systems of linear equations can be classified into two broad classes: direct methods, which will compute the exact solution in a finite number of operations, and indirect (or iterative) methods, which begin with an approximation that is then improved iteratively. Conjugate gradient is a semi-iterative method, as it computes the exact solution in $O(n^3)$ time, but also produces intermediate estimates which are refined iteratively [7, 8, 3]. Direct methods include Gaussian elimination, Cholesky factorization, orthogonalization, and block decomposition [23]. These methods generally require $O(n^3)$ time, although the current best known direct algorithm for general linear systems is $O(n^q)$ where $q \approx 2.373$ [24]. Indirect methods include stationary linear methods (i.e., those that update according to a time invariant linear operator, such as Jacobi, Gauss-Seidel, and Richardson updates), gradient methods (such as steepest descent or conjugate direction methods), and Monte Carlo methods (such as Ulam von Neumann and subsequent variations) [23]. Recently, there is a class of approximation algorithms which run in nearly linear time when A is sparse and symmetric diagonally dominant [18, 13, 12]. These methods rely upon connections between the algebraic properties of graph Laplacian matrices, and combinatorial properties of its corresponding graph.

Except for the Monte Carlo methods, existing methods for solving linear systems focus on solving the full vector x. If we are only interested in x_i, we would need to compute the entire vector in the process. The Ulam von Neumann algorithm is a Monte Carlo style method which depends on the Neumann series representation of the solution x as stated in (3), which we recall converges for $\rho(G) < 1$. It characterizes this expression as a sum over weighted walks on graph $\mathcal{G}(G)$, and obtains an estimate by sampling random walks starting from node i over $\mathcal{G}(G)$ [6, 21, 5]. This requires designing a transition probability matrix over the graph, and the basic form of the algorithm requires $\|G\|_\infty < 1$. There are modifications which propose other transition probability matrices, or use correlated or importance sampling to reduce the variance of the estimator [9, 10]. However, the scope of this algorithm is still limited, as Ji, Mascagni, and Li proved that there is a class of matrices G such that $\rho(G) < 1$, $\|G\|_\infty > 1$, and there does not exist any transition probability matrix such that the Ulam von Neumann method converges [11].

Our proposed synchronous algorithm is similar to stationary linear iterative methods, which use updates of the form $x_{t+1} = Gx_t + z$ to recursively approximate leading terms of the Neumann series, as stated in (3). The error after t iterations is given by $G^t(x - x_0)$, thus the number of iterations to achieve $\|x_t - x\|_2 \leq \epsilon\|x\|_2$, is given by $\ln(\epsilon)/\ln(\|G\|_2)$. These methods do not exploit the sparsity of G and the locality of computing a single component, and thus each iteration is costly and requires nd multiplications.

The use of randomization in subsampling matrices as part of a subroutine in iterative methods has previously been used in the context of other global matrix algorithms, such as the randomized Kaczmarz method and stochastic iterative projection [19, 17, 16, 15, 20]. The randomized Kaczmarz method is used in the context of solving overdetermined systems of equations, subsampling rows to reduce the dimension of the computation matrix in each iteration. Stochastic iterative methods involve sampling a sparse approximation of matrix G to reduce the computation in each iteration while maintaining convergence.

2 Our Results

We propose two variants of an algorithm to estimate x_i given an equation of the form (2). The algorithm is local in that the computation involves only the coordinates within the local neighborhood $N_i(t)$ of graph $\mathcal{G}(G)$ for some t which is a function of ϵ, $\|G\|_2$, and d.

2.1 Synchronous

The synchronous algorithm we propose involves computing leading terms of the sum as stated in (4). In comparison, standard linear iterative methods compute leading terms of the Neumann series as stated in (3) using the update equation $x_{t+1} = Gx_t + z$. If z is not sparse, then any approximation using the first t terms of the Neumann series will be at least as dense as z, possibly involving all the coordinates. Thus $\|x_t\|_0$ can be nearly n, such that a single update step could cost nd multiplications. In contrast, observe that the sparsity of $(G^T)^t e_i$ is upper bounded by the size of the local neighborhood $N_i(t)$, which in turn is upper bounded by d^t. Therefore, an approximation which uses the first t terms of (4) will only have nonzero coordinates corresponding to the nodes in $N_i(t)$. The number of multiplications that iteration t of our algorithm requires is bounded by $d|N_i(t)|$. In fact the sparsity pattern of the iterates over time resembles that of a breadth first traversal over $\mathcal{G}(G)$ starting from node i, which corresponds to $N_i(t)$. This highlights how our algorithm keeps the computation local in the setting when we only need to compute a single component x_i, and thus saves computation.

Theorem 2.1. *If $\|G\|_2 < 1$,* `SynchronousLocalAlgorithm` *produces an estimate $\hat{x}_i$ such that $|\hat{x}_i - x_i| \leq \epsilon \|x\|_2$, and the total number of multiplications is bounded by*

$$O\left(\min\left(d\epsilon^{\ln(d)/\ln(\|G\|_2)}, \frac{dn\ln(\epsilon)}{\ln(\|G\|_2)}\right)\right).$$

We show that the number of iterations is bounded by $\ln(\epsilon)/\ln(\|G\|_2)$. Each iteration involves multiplying G with an intermediate vector which has sparsity at most $N_i(t)$. The right hand expression comes from bounding the cost of each iteration by dn, which holds even in the nonsparse setting, and obtains the same result as standard linear iterative methods. The left hand expression comes from bounding the cost of iteration k by d^{k+1}. If G is sparse, then the left hand bound may be tighter. We thus utilize sparsity to achieve an approximation for x_i in constant time with respect to the size of the matrix when n is large, and $d = O(1)$ and $-1/\ln(\|G\|_2) = O(1)$ as a function of n.

2.2 Asynchronous

The second algorithm we propose is an asynchronous randomized modification of the first algorithm. Instead of multiplying by matrix G in every iteration, the algorithm multiplies by some $\hat{G}_t$, which consists only of a single row of G, and is 0 elsewhere. This row is adaptively sampled among the nonzero coordinates of the current iterate, therefore the nodes in graph $\mathcal{G}(G)$ which correspond to nonzero coordinates of the current iterate are all connected by a path to node i. Since the coordinates can be updated in arbitrary order, the sparsity pattern of the iterates over time no longer corresponds to a breadth first traversal over $\mathcal{G}(G)$, but instead resembles a traversal which grows by adding adjacent nodes in any order beginning from node i. We prove in Theorem 2.2, that with high probability, the number of multiplications the algorithm uses takes a similar form to the bound given for the synchronous algorithm. The computation involved in each iteration corresponds to a local operation involving only the node corresponding to the chosen row, and its neighboring edges.

Theorem 2.2. *If $\|G\|_2 < 1$, with probability 1,* `AsynchronousLocalAlgorithm` *terminates and produces an estimate $\hat{x}_i$ such that $|\hat{x}_i - x_i| \leq \epsilon\|x\|_2$. With probability greater than $1-\delta$, the total number of multiplications is bounded by*

$$O\left(\min\left(d\left(\epsilon\sqrt{\delta/2}\right)^{-d/(1-\|G\|_2)}, \frac{-dn\ln(\epsilon\sqrt{\delta})}{1-\|G\|_2}\right)\right).$$

We can compare the bounds for the synchronous and asynchronous variants by considering that $1 - \|G\|_2 \approx -\ln(\|G\|_2)$ when $\|G\|_2 \approx 1$. The two right hand expressions in Theorems 2.1 and 2.2 are essentially the same, and provide a comparison of our algorithm to standard linear iterative methods. The left hand bound from Theorem 2.2, which provides a local analysis, is constant with respect to n as long as $d = O(1)$ and $1/(1-\|G\|_2) = O(1)$. However, the bound which applies in the sparse setting for the asynchronous variant grows exponentially in d, while the corresponding bound for the synchronous variant grows only polynomially in d. We will provide a brief discussion of the gap between the analyses of both algorithms in Section 4.3. To summarize, the rate of convergence of the asynchronous variant is slower by a factor of $d/\ln(d)$ because the proven contraction of the error in each iteration is now spread out among the nonzero coordinates of the current iterate. The different rates come from comparing the convergence of the error along with the number of nonzero entires in the iterates over time.

2.3 Transforming $Ax = b$ to $x = Gx + z$

We discuss conditions under which a system of linear equations of the form (1) can be transformed into the form given by (2) with $\rho(G) < 1$. In this setting, the solution x is given by the Neumann series as stated in (3). There are many existing methods of choosing G and z which satisfy these conditions [23, 22]. We particularly highlight the Richardson and Jacobi methods, which specify G such that G is as sparse as A, $\rho(G) < 1$, and G_{ij} can be computed as a simple function of A_{ij} and A_{ii}. For any γ such that $0 \leq \gamma \leq \min_{\|x\|_2=1}(2x^TAx)/(x^TA^TAx)$, the Richardson method chooses $G = I - \gamma A$ and $z = \gamma b$. If A is positive definite,[5] it is guaranteed that $\|G\|_2 \leq \rho(G) < 1$. Let D be a diagonal matrix such that $D_{uu} = A_{uu}$. The Jacobi method chooses $G = -D^{-1}(A - D)$ and $z = D^{-1}b$. If A is strictly or irreducibly diagonally dominant, it can be shown that $\rho(G) < 1$. If G is symmetric, then in addition $\rho(G) = \|G\|_2 < 1$.

Corollary 2.3. *If A is positive definite ($A \succ 0$) or diagonally dominant, then we can use standard methods (e.g. Jacobi or Richardson), to choose a matrix G and vector z such that $\rho(G) < 1$, and the solution x which satisfies $x = Gx + z$ will also satisfy $Ax = b$.*

Corollary 2.3 implies that when A is either positive definite or symmetric diagonally dominant, we can use either Jacobi or Richardson to choose G and z as a function of A and b so that $\|G\|_2 < 1$. If A is symmetric positive definite, then using the Richardson method with an optimal choice of γ results in a choice of G such that $\rho(G) = \|G\|_2 = (\kappa(A)-1)/(\kappa(A)+1)$. We can apply our proposed algorithm on the transformed problem to obtain an estimate for $\hat{x}_i$, achieving the same accuracy and convergence rate as specified in Theorems 2.1 and 2.2, with the corresponding corresponding relation between $\rho(G)$ and $\kappa(A)$. Given (A, b), there are potentially infinitely many ways to choose (G, z) to satisfy the condition that $\|G\|_2 < 1$. Finding the optimal choice of (G, z) given (A, b) is beyond the scope of this paper.

3 Synchronous Local Algorithm

The key observation for the local algorithm is that since we only want to compute the i^{th} component, we can compute $\sum_{k=0}^{\infty}(G^T)^k e_i$ and take the inner product with z at the end, instead of computing $\sum_{k=0}^{\infty} G^k z$, where z could be a dense vector. Even though this is a small modification, it controls the sparsity of intermediate vectors involved in the algorithm. We proceed to describe our algorithm. First, observe that x_i can be expressed as

$$x_i = e_i^T \sum_{k=0}^{\infty} G^k z = \sum_{k=0}^{t-1} e_i^T G^k z + e_i^T \sum_{k=t}^{\infty} G^k z = \sum_{k=0}^{t-1} e_i^T G^k z + e_i^T G^t x$$

The algorithm keeps track of two intermediate vectors such that at iteration t, $p^{(t)} = (\sum_{k=0}^{t-1} e_i^T G^k)^T$ and $r^{(t)} = (e_i^T G^t)^T$. Then it follows that for all iterations t, $x_i = p^{(t)T} z + r^{(t)T} x$. To achieve this,

[5]Matrix A is positive definite, denoted by $A \succ 0$, if for all nonzero x, $x^TAx > 0$.

we initialize the vectors with $p^{(0)} = 0$ and $r^{(0)} = e_i$, and we use the following updates:

$$p^{(t+1)} = p^{(t)} + r^{(t)}, \tag{5}$$

$$r^{(t+1)} = G^T r^{(t)}. \tag{6}$$

The algorithm computes the estimate according to $\hat{x}_i = p^{(t)T} z$. Thus the error will be exactly $x_i - \hat{x}_i = r^{(t)T} x$. We refer to $r^{(t)}$ as the residual vector.

3.1 Algorithm and Results

Algorithm: `SynchronousLocalAlgorithm`(G, z, i, ϵ)

Input: G, i, ϵ

```
1: $p^{(0)} = 0, r^{(0)} = e_i, t = 0$
2: while $\|r^{(t)}\|_2 > \epsilon$ do
3:     $p^{(t+1)} = p^{(t)} + r^{(t)}$
4:     $r^{(t+1)} = G^T r^{(t)}$
5:     $t = t + 1$
6: end while
7: return $\hat{x}_i = p^{(t)T} z$
```

Theorem 2.1. *If $\|G\|_2 < 1$,* `SynchronousLocalAlgorithm` *produces an estimate $\hat{x}_i$ such that $|\hat{x}_i - x_i| \leq \epsilon \|x\|_2$, and the total number of multiplications is bounded by*

$$O\left(\min\left(d\epsilon^{\ln(d)/\ln(\|G\|_2)}, \frac{dn\ln(\epsilon)}{\ln(\|G\|_2)}\right)\right).$$

Proof. The initial vectors and update rules are chosen such that $p^{(t)} = (\sum_{k=0}^{t-1} G^k)^T e_i$, and $r^{(t)} = (G^t)^T e_i$. Therefore for all t, $x_i = z^T p^{(t)} + x^T r^{(t)}$, and the error in the estimate is given by $x^T r^{(t)}$. Since the algorithm terminates when $\|r^{(t)}\|_2 \leq \epsilon$, then it follows that

$$|\hat{x}_i - x_i| = |r^{(t)T} x| \leq \|r^{(t)}\|_2 \|x\|_2 \leq \epsilon \|x\|_2. \tag{7}$$

In order to upper bound the computation, we observe that

$$\|r^{(t)}\|_2 = \|(G^T)^t e_i\|_2 \leq \|G\|_2^t. \tag{8}$$

Therefore the algorithm terminates with $\|r^{(t)}\|_2 < \epsilon$ within at most $\ln(\epsilon)/\ln(\|G\|_2)$ iterations. Since each row of G has at most d nonzero entries, $\|r^{(t)}\|_0 \leq d^t$. The number of multiplications in each iteration is at most $d\|r^{(t)}\|_0$. Therefore, we can upper bound the total number of multiplications by

$$\begin{aligned} d\left(\sum_{t=0}^{\lfloor \ln(\epsilon)/\ln(\|G\|_2)\rfloor} d^t\right) &= \left(\frac{d}{d-1}\right)\left(d^{\lfloor \ln(\epsilon)/\ln(\|G\|_2)\rfloor + 1} - 1\right) \\ &= O\left(d \cdot d^{\ln(\epsilon)/\ln(\|G\|_2)}\right) = O\left(d\epsilon^{\ln(d)/\ln(\|G\|_2)}\right). \end{aligned} \tag{9}$$

Alternatively, we can naively bound the number of multiplications in each iteration by dn. Then it follows that the total number of multiplications is also bounded by $O(dn \ln(\epsilon)/\ln(\|G\|_2))$. $\square$

The right hand expression in Theorem 2.1 does not require sparsity, and applies when the computation reaches all the coordinates and the algorithm behaves similarly to standard linear iterative methods. The left hand expression in Theorem 2.1 proves that when G is sparse and $-1/\ln(\|G\|_2)$ is small, the algorithm utilizes the sparsity to save computation over standard methods.

When G is symmetric, then $\|G\|_2 = \rho(G)$, proving that the algorithm converges whenever the infinite series converges. When G is nonsymmetric, this condition may be stricter since $\|G\|_2$ could be larger than $\rho(G)$. However, we suspect that even for nonsymmetric G, the spectral radius $\rho(G)$ governs the performance of the algorithm. This intuition comes from Gelfand's spectral radius formula, which states that $\rho(M) = \lim_{k\to\infty} \|M^k\|_2^{1/k}$. The precise result would depend on the convergence rate of this limit.

3.2 Local Convergence Properties

We can also interpret the algorithm in terms of computation over the graph $\mathcal{G}(G)$. Multiplying $r^{(t)}$ by G^T then corresponds to a message passing operation from each of the nonzero coordinates of $r^{(t)}$ along their corresponding adjacent edges in the graph. The sparsity of $r^{(t)}$ thus grows as a breadth first traversal over the graph starting from node i. Therefore, $\|r^{(t)}\|_0 \le N_i(t)$, which we consider to be the set of nodes involved in the computation up to time t.

The algorithm only involves nodes that are within distance $\ln(\epsilon)/\ln(\|G\|_2)$ from the initial node i. We define the matrix $G_{N_i(t)}$ such that $G_{N_i(t)}(a,b) = G(a,b)$ if and only if $(a,b) \in N_i(t) \times N_i(t)$. It follows that

$$\|r^{(t)}\|_2 = \|e_i^T G^t\|_2 = \|e_i^T \prod_{k=1}^{t} G_{N_k(i)}\|_2 = \|e_i^T G_{N_t(i)}^t\|_2 \le \|G_{N_t(i)}\|_2^t. \tag{10}$$

It is possible that for some choices of i and t, $\|G_{N_t(i)}\|_2 < \|G\|_2$, in which case the algorithm would converge more quickly as a function of the local neighborhood. If G corresponds to a scaled adjacency matrix of an unweighted undirected graph, then it is known that

$$\max\left(d_{\text{average}}, \sqrt{d_{\max}}\right) \le \rho(G) \le d_{\max}. \tag{11}$$

In this case, we would only expect $\|G_{N_t(i)}\|_2$ to be smaller than $\|G\|_2$ if the local degree distribution of the neighborhood around node i is different from the global degree distribution.

4 Asynchronous Randomized Local Algorithm

The previous solution requires the multiplications in each iteration to be synchronous. In this section, we modify the algorithm to only apply the update to a single coordinate at a time, which corresponds to multiplying by only a single row of G, rather than the full matrix. The asynchronous algorithm directly follows from a randomized coordinate-based implementation of the synchronous algorithm. We rewrite the update equations of the synchronous algorithm to show the incremental updates that each coordinate of $r^{(t)}$ is involved in:

$$p^{(t+1)} = p^{(t)} + \sum_u e_u e_u^T r^{(t)}, \tag{12}$$

$$r^{(t+1)} = r^{(t)} + \sum_u (G^T - I) e_u e_u^T r^{(t)} \tag{13}$$

The asynchronous algorithm chooses a single coordinate u, and applies the updates corresponding to the calculations involving the uth coordinate of $r^{(t)}$:

$$p^{(t+1)} = p^{(t)} + e_u e_u^T r^{(t)}, \tag{14}$$

$$r^{(t+1)} = r^{(t)} + (G^T - I) e_u e_u^T r^{(t)}. \tag{15}$$

The algorithm samples the coordinate u according to the following distribution:[6]

$$\text{For all } u, \quad \mathcal{P}(u) = \frac{\mathbf{1}\left(r_u^{(t)} \neq 0\right)}{\|r^{(t)}\|_0}. \tag{16}$$

To prove that the estimate converges to the correct solution, we establish an invariant that for all t, $x_i = p^{(t)T} z + r^{(t)T} x$. This holds for all possible selections of update coordinates. Since the distribution $\mathcal{P}$ chooses uniformly among the nonzero coordinates of $r^{(t)}$, in expectation the update step corresponds to multiplying a scaled version of matrix G to vector $r^{(t)}$. We proceed to show that in addition $\|r^{(t)}\|_2$ contracts with high probability due to the choice of distribution $\mathcal{P}$.

4.1 Algorithm and Results

```
Algorithm: AsynchronousLocalAlgorithm($G, z, i, \epsilon$)
Input: $G, i, \epsilon$
1: $p^{(0)} = 0, r^{(0)} = e_i, t = 0$
2: while $\|r^{(t)}\|_2 > \epsilon$ do
3:     Pick a coordinate $u$ with probability $\mathcal{P}(u)$
4:     $p^{(t+1)} = p^{(t)} + e_u e_u^T r^{(t)}$
5:     $r^{(t+1)} = r^{(t)} - (I - G^T) e_u e_u^T r^{(t)}$
6:     $t = t + 1$
7: end while
8: return $\hat{x}_i = p^{(t)T} z$
```

Theorem 2.2. *If $\|G\|_2 < 1$, with probability 1,* `AsynchronousLocalAlgorithm` *terminates and produces an estimate $\hat{x}_i$ such that $|\hat{x}_i - x_i| \leq \epsilon \|x\|_2$. With probability greater than $1 - \delta$, the total number of multiplications is bounded by*

$$O\left(\min\left(d\left(\epsilon\sqrt{\delta/2}\right)^{-d/(1-\|G\|_2)}, \frac{-dn\ln(\epsilon\sqrt{\delta})}{1-\|G\|_2}\right)\right).$$

[6]Let $\mathbf{1}(\cdot)$ denote the indicator function. Let $r_u^{(t)}$ denote the uth coordinate of vector $r^{(t)}$. For simpler notation, we suppress the dependence of $\mathcal{P}$ on $r^{(t)}$.

Lemma 4.1 (Invariant). *The variables in the* `AsynchronousLocalAlgorithm`(G, z, i, ϵ) *satisfy the invariant that for all t, $x_i = p^{(t)T} z + r^{(t)T} x$.*

Proof. Recall that $x = z + Gx$. We prove that the invariant holds by using induction. First verify that for $t = 0$, $p^{(0)T} z + r^{(0)T} x = 0 + e_i^T x = x_i$. Then assuming that $x_i = p^{(t)T} z + r^{(t)T} x$, we show that

$$\begin{aligned} p^{(t+1)T} z + r^{(t+1)T} x &= p^{(t)T} z + (e_u e_u^T r^{(t)})^T z + r^{(t)T} x - (e_u e_u^T r^{(t)})^T x + (G^T e_u e_u^T r^{(t)})^T x \\ &= p^{(t)T} z + (e_u e_u^T r^{(t)})^T z + r^{(t)T} x - (e_u e_u^T r^{(t)})^T (z + Gx) + (G^T e_u e_u^T r^{(t)})^T x \\ &= p^{(t)T} z + r^{(t)T} x = x_i. \end{aligned}$$

□

The left hand expression of Theorem 2.2 applies when G is sparse, and $1/(1 - \|G\|_2)$ is not too large, such that the algorithm terminates while the vector $r^{(t)}$ is still sparse, i.e., the computation remains local to a neighborhood of i. The right hand expression of Theorem 2.2 is comparable to the cost of standard linear iterative methods, and applies when $\|G\|_2$ is close to 1 and n is not too large, such that the algorithm does not terminate before the computation reaches the entire graph.

4.2 Proof Sketch of Theorem 2.2

Since the algorithm terminates when $\|r^{(t)}\|_2 \leq \epsilon$, it follows from Lemma 4.1 that $|\hat{x}_i - x_i| = |r^{(t)T} x| \leq \epsilon \|x\|_2$. Recall that the algorithm chooses a coordinate in each iteration according to the distribution $\mathcal{P}$, as specified in (16). To simplify the analysis, we introduce another probability distribution $\tilde{\mathcal{P}}$, which has a fixed size support of $\min(td, n)$ rather than $\|r^{(t)}\|_0$. We first analyze the convergence of a modified algorithm which samples coordinates according to $\tilde{\mathcal{P}}$. Then we translate the results back to the original algorithm.

Observe that for any $t \in \mathbb{Z}_+$, there exists a function $S_t : \mathbb{R}^n \to \{0, 1\}^n$, which satisfies the properties that for any $v \in \mathbb{R}^n$ and $u = S_t(v)$, if $v_i \neq 0$, then $u_i = 1$, and if $\|v\|_0 \leq td$, then $\|u\|_0 = \min(td, n)$. In words, $S_t(v)$ is a function which takes a vector of sparsity at most td, and maps it to a binary valued vector which preserves the sparsity pattern of v, yet adds extra entries of 1 in order that the sparsity of the output is exactly $\min(td, n)$. We define the distribution $\tilde{\mathcal{P}}$ to choose uniformly at random among the nonzero coordinates of $S_t\left(r^{(t)}\right)$, as follows:[7]

$$\tilde{\mathcal{P}}(u) = \frac{e_u^T S_t\left(r^{(t)}\right)}{\min(td, n)}. \tag{17}$$

This is a valid probability distribution since for all t, $\|r^{(t)}\|_0 \leq td$. We can show that for the variation of the asynchronous algorithm which samples coordinates accoridng to $\tilde{\mathcal{P}}$,

$$\mathbb{E}_{\tilde{\mathcal{P}}}\left[r^{(t+1)} \middle| r^{(t)} \right] = \left(I - \left(\frac{I - G^T}{\min(td, n)} \right) \right) r^{(t)}. \tag{18}$$

This shows that in expectation, the error contracts in each iteration by $(1 - (1 - \|G\|_2)/\min(td, n))$. We prove an upper bound on the expected L2-norm of the residual vector $r^{(t)}$, stated in Lemma 4.2. Then we apply Markov's inequality to prove that the algorithm terminates with high probability within a certain number of multiplications.

[7]For simpler notation, we suppress the dependence of $\tilde{\mathcal{P}}$ on t and $r^{(t)}$.

Lemma 4.2. *If $\|G\|_2 < 1$, $d \geq 4$, and $n \geq 8$,*

$$\mathbb{E}_{\tilde{\mathcal{P}}}\left[\|r^t\|_2^2\right] \leq \min\left(2t^{-2(1-\|G\|_2)/d}, 4e^{-2(t-1)(1-\|G\|_2)/n}\right).$$

In order to extend the proofs from $\tilde{\mathcal{P}}$ to $\mathcal{P}$, we define a coupling between two implementations of the algorithm, one which sample coordinates according to $\tilde{\mathcal{P}}$, and the other which samples coordinates according to $\mathcal{P}$. We prove that in this joint probability space, the implementation which uses distribution $\mathcal{P}$ always terminates in number of iterations less than or equal to the corresponding termination time of the implementation using $\tilde{\mathcal{P}}$. Therefore, computing an upper bound on the number of multiplications required under $\tilde{\mathcal{P}}$ is also an upper bound for the algorithm which uses $\mathcal{P}$. Further details and formalization of the coupling argument and the proof of Lemma 4.2 are included in the Appendix.

4.3 Comparison between Synchronous and Asynchronous Algorithms

The main difference between Theorem 2.1 and Theorem 2.2 is in the dependence on d. There is an intuitive reasoning which shows why our analyses result in such a gap. Let $C_S(t) = d^t$ denote the upper bound on the number of multiplications performed by the synchronous algorithm in the first t iterations. Let $C_A(t) = td$ denote the upper bound on the number of multiplications performed by the asynchronous algorithm in the first t iterations. For some $t_S > 0$, let $t_A = C_A^{-1}(C_S(t_S)) = d^{t_S - 1}$, such that $C_A(t_A) = C_S(t_S)$. This allows us to compare the error in the two algorithms given the same number of multiplications. From (8), the residual vector of the synchronous algorithm after t_S iterations is bounded by $\|r^{(t_S)}\|_2^2 \leq \|G\|_2^{2t_S}$. From Lemma 4.2, the expected residual vector of the asynchronous algorithm after t_A iterations is bounded by

$$\mathbb{E}[\|r^{(t_A)}\|_2^2]\| \leq 2t_A^{-2(1-\|G\|_2)/d} \approx 2t_A^{2\ln(\|G\|_2)/d} = 2\|G\|_2^{2\ln(t_A)/d} = 2\|G\|_2^{2(t_S-1)\ln(d)/d}. \tag{19}$$

Therefore the ratio of the convergence rates of the two algorithms obtained by our analysis is given by $\ln(\|r^{(t_S)}\|_2)/\ln(\|\mathbb{E}[r^{(t_A)}]\|_2) \approx d/\ln(d)$.

5 Discussion

In this paper, we presented two variants of a local algorithm which estimates a single component x_i of the solution to a system of linear equations. The synchronous algorithm is a modification of standard stationary linear iterative methods using an observation which allows the vectors involved in the computation to remain sparse when G is sparse, $1/(1 - \|G\|_2)$ is small, and n is large. The asynchronous algorithm is a randomized modification of the synchronous algorithm, allowing the coordinates to update themselves in arbitrary order according to a given distribution. We show that the asynchronous algorithm converges to the correct solution with similar converges guarantees.

Compared to the Ulam von Neumann algorithm, which is the standard approach to solving for single components of the solution, our algorithm had broader conditions for convergence, encompassing a larger set of systems of linear equations. Our method is similar to the algorithm by Andersen *et al.* for computing PageRank, but we use randomization techniques to improve the analysis and extend it past limiting properties specific to the PageRank application.

It is open problem to design the optimal distribution for choosing coordinates within the asynchronous randomized algorithm. Our analysis only applies when the algorithm samples uniformly among nonzero coordinates of the residual vector $r^{(t)}$. However, simulations indicate that choosing the coordinate proportional to $r_u^{(t)}$ seems to improve the convergence of the algorithm. Establishing theoretical analysis of this observation remains an open problem.

Acknowledgements

This work is supported in parts by ARO under MURI award W911NF-11-1-0036, by AFOSR under MURI award FA9550-09-1-0538, by ONR under the Basic Research Challenge No. N000141210997, and by NSF under grant CNS-1161964 and a Graduate Fellowship.

References

[1] Reid Andersen, Christian Borgs, Jennifer Chayes, John Hopcraft, Vahab S. Mirrokni, and Shang-Hua Teng. Local computation of PageRank contributions. In *Algorithms and Models for the Web-Graph*, pages 150–165. Springer, 2007.

[2] Abraham Berman and Robert J. Plemmons. *Nonnegative matrices in the mathematical sciences.* Classics in applied mathematics: 9. Philadelphia : Society for Industrial and Applied Mathematics, 1994., 1994.

[3] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and distributed computation: numerical methods.* Prentice-Hall, Inc., 1989.

[4] Richard Cottle, Jong-Shi Pang, and Richard E. Stone. *The linear complementarity problem.* Computer science and scientific computing. Boston : Academic Press, c1992., 1992.

[5] J.H. Curtiss. *A theoretical comparison of the efficiencies of two classical methods and a monte carlo method for computing one component of the solution of a set of linear algebraic equations.* Courant Institute of Mathematical Sciences, New York University, 1954.

[6] George E. Forsythe and Richard A. Leibler. Matrix inversion by a monte carlo method. *Mathematical Tables and Other Aids to Computation*, pages 127–129, 1950.

[7] J.E. Gentle. Matrix algebra: Theory, computations, and applications in statistics. *AStA Advances in Statistical Analysis*, 92(3):343–344, 2008.

[8] Gene H. Golub and Charles F. Van Loan. *Matrix computations / Gene H. Golub, Charles F. Van Loan.* Johns Hopkins studies in the mathematical sciences. Baltimore : The Johns Hopkins University Press, 2013., 2013.

[9] John H. Halton. A retrospective and prospective survey of the monte carlo method. *Siam review*, 12(1):1–63, 1970.

[10] John H. Halton. Sequential monte carlo techniques for the solution of linear systems. *Journal of Scientific Computing*, 9(2):213–257, 1994.

[11] Hao Ji, Michael Mascagni, and Yaohang Li. Convergence analysis of markov chain monte carlo linear solvers using ulam–von neumann algorithm. *SIAM Journal on Numerical Analysis*, 51(4):2107–2122, 2013.

[12] Jonathan A. Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. A simple, combinatorial algorithm for solving sdd systems in nearly-linear time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 911–920. ACM, 2013.

[13] Ioannis Koutis, Gary L. Miller, and Richard Peng. A nearly-m log n time solver for sdd linear systems. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 590–598. IEEE, 2011.

[14] Zhi-Quan Luo and Paul Tseng. On the convergence of a matrix splitting algorithm for the symmetric monotone linear complementarity problem. *SIAM Journal on Control and Optimization*, 29(5):1037–1060, 1991.

[15] K. Sabelfeld and N. Mozartova. Sparsified randomization algorithms for large systems of linear equations and a new version of the random walk on boundary method. *Monte Carlo Methods and Applications*, 15(3):257–284, 2009.

[16] Karl Sabelfeld. Stochastic algorithms in linear algebra-beyond the markov chains and von neumann-ulam scheme. In *Numerical Methods and Applications*, pages 14–28. Springer, 2011.

[17] Karl Sabelfeld and Nadja Loshchina. Stochastic iterative projection methods for large linear systems. *Monte Carlo Methods and Applications*, 16(3-4):343–359, 2010.

[18] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *arXiv preprint cs/0607105*, 2006.

[19] Thomas Strohmer and Roman Vershynin. A randomized kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15(2):262–278, 2009.

[20] Mengdi Wang and Dimitri P. Bertsekas. Stabilization of stochastic iterative methods for singular and nearly singular linear systems. *Mathematics of Operations Research*, 39(1):1–30, 2013.

[21] W.R. Wasow. A note on the inversion of matrices by random walks. *Mathematical Tables and Other Aids to Computation*, pages 78–81, 1952.

[22] Ermin Wei, Asuman Ozdaglar, and Ali Jadbabaie. A distributed newton method for network utility maximization-part ii: Convergence. *Automatic Control, IEEE Transactions on*, 58(9):2176–2188, Sept 2013.

[23] Joan R. Westlake. *A handbook of numerical matrix inversion and solution of linear equations*, volume 767. Wiley New York, 1968.

[24] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 887–898. ACM, 2012.

A Proof of Theorem 2.2

Lemma A.1. *If $\|G\|_2 < 1$, for all t,*

(a) $\mathbb{E}_{\tilde{\mathcal{P}}}\left[r^{(t+1)}\middle|r^{(t)}\right] = \left(I - \left(\frac{I-G^T}{\min(td,n)}\right)\right)r^{(t)}$,

(b) $\left\|\mathbb{E}_{\tilde{\mathcal{P}}}\left[r^{(t)}\right]\right\|_2 \le \min\left(t^{-(1-\|G\|_2)/d}, e^{-(t-1)(1-\|G\|_2)/n}\right)$.

Proof. We will use induction to get an expression for $\mathbb{E}_{\tilde{\mathcal{P}}}\left[r^{(t)}\right]$. Recall that $r^{(0)} = e_i$. Since there is only a single coordinate to choose from, $r^{(1)}$ is also predetermined, and is given by $r^{(1)} = G^T e_i$.

$$\mathbb{E}_{\tilde{\mathcal{P}}}\left[r^{(t+1)}\middle|r^{(t)}\right] = r^{(t)} - (I-G^T)\sum_u \tilde{\mathcal{P}}(u)e_u r_u^{(t)}. \tag{20}$$

By design of $\tilde{\mathcal{P}}$, we know that $\tilde{\mathcal{P}}(u) = 1/\min(td, n)$ for all u such that $r_u^{(t)} \neq 0$. Therefore,

$$\sum_u \tilde{\mathcal{P}}(u) e_u r_u^{(t)} = \frac{r_u^{(t)}}{\min(td, n)}. \tag{21}$$

We substitute this into (20) to show that

$$\mathbb{E}_{\tilde{\mathcal{P}}}\left[r^{(t+1)} \middle| r^{(t)} \right] = \left(I - \left(\frac{I - G^T}{\min(td, n)} \right) \right) r^{(t)}. \tag{22}$$

Using the initial conditons $r^{(1)} = G^T e_i$ and the law of iterated expectation, it follows that

$$\mathbb{E}_{\tilde{\mathcal{P}}}\left[r^{(t)T} \right] = e_i^T G \prod_{k=1}^{t-1} \left(I - \left(\frac{I - G}{\min(kd, n)} \right) \right). \tag{23}$$

Therefore,

$$\begin{aligned} \left\| \mathbb{E}_{\tilde{\mathcal{P}}}\left[r^{(t)} \right] \right\|_2 &\leq \|G\|_2 \prod_{k=1}^{t-1} \left(1 - \left(\frac{1 - \|G\|_2}{\min(kd, n)} \right) \right), \\ &\leq \|G\|_2 \exp\left(- \sum_{k=1}^{t-1} \frac{1 - \|G\|_2}{\min(kd, n)} \right), \\ &\leq \|G\|_2 \min\left(\exp\left(- \sum_{k=1}^{t-1} \frac{1 - \|G\|_2}{kd} \right), \|G\|_2 \exp\left(- \sum_{k=1}^{t-1} \frac{1 - \|G\|_2}{n} \right) \right). \end{aligned} \tag{24}$$

Since $\|G\|_2 < 1$ by assumption, and using the property that $\sum_{k=1}^{t-1} \frac{1}{k} > \ln(t)$, it follows that

$$\left\| \mathbb{E}_{\tilde{\mathcal{P}}}\left[r^{(t)} \right] \right\|_2 \leq \min\left(t^{-(1 - \|G\|_2)/d}, e^{-(t-1)(1 - \|G\|_2)/n} \right). \tag{25}$$

□

Lemma 4.2. *If* $\|G\|_2 < 1$, $d \geq 4$, *and* $n \geq 8$,

$$\mathbb{E}_{\tilde{\mathcal{P}}}\left[\left\| r^t \right\|_2^2 \right] \leq \min\left(2t^{-2(1 - \|G\|_2)/d}, 4e^{-2(t-1)(1 - \|G\|_2)/n} \right).$$

Proof. Observe that

$$\mathbb{E}_{\tilde{\mathcal{P}}}\left[\left\| r^{(t+1)} \right\|_2^2 \right] = \mathbb{E}_{\tilde{\mathcal{P}}}\left[\left\| r^{(t+1)} - \mathbb{E}_{\tilde{\mathcal{P}}}\left[r^{(t+1)} \right] \right\|_2^2 \right] + \left\| \mathbb{E}_{\tilde{\mathcal{P}}}\left[r^{(t+1)} \right] \right\|_2^2, \tag{26}$$

and

$$\mathbb{E}_{\tilde{\mathcal{P}}}\left[\left\| r^{(t+1)} - \mathbb{E}_{\tilde{\mathcal{P}}}\left[r^{(t+1)} \right] \right\|_2^2 \middle| r^{(t)} \right] = \mathbb{E}_{\tilde{\mathcal{P}}}\left[\left\| r^{(t+1)} \right\|_2^2 \middle| r^{(t)} \right] - \left\| \mathbb{E}_{\tilde{\mathcal{P}}}\left[r^{(t+1)} \middle| r^{(t)} \right] \right\|_2^2. \tag{27}$$

Based on the update equation $r^{(t+1)} = r^{(t)} - (I - G^T)e_u r_u^{(t)}$, we can compute that

$$\begin{aligned}\mathbb{E}_{\tilde{\mathcal{P}}}\left[\left\|r^{(t+1)}\right\|_2^2 \middle| r^{(t)}\right] &= \sum_u \tilde{\mathcal{P}}(u)(r^{(t)} - (I-G^T)e_u r_u^{(t)})^T(r^{(t)} - (I-G^T)e_u r_u^{(t)}), \\ &= r^{(t)T}r^{(t)} - \left(\sum_u \tilde{\mathcal{P}}(u) r_u^{(t)} e_u^T\right)(I-G)r^{(t)} - r^{(t)T}(I-G^T)\left(\sum_u \tilde{\mathcal{P}}(u) e_u r_u^{(t)}\right) \\ &\quad + \sum_u \tilde{\mathcal{P}}(u) r_u^{(t)2}\left[(I-G)(I-G^T)\right]_{uu}. \end{aligned} \tag{28}$$

By the design, $\tilde{\mathcal{P}}(u) r_u^{(t)} = r_u^{(t)}/\min(td, n)$, so that

$$\sum_u \tilde{\mathcal{P}}(u) e_u r_u^{(t)} = \frac{r_u^{(t)}}{\min(td, n)}. \tag{29}$$

Similarly, since $\tilde{\mathcal{P}}(u) r_u^{(t)2} = r_u^{(t)2}/\min(td, n)$,

$$\sum_u \tilde{\mathcal{P}}(u) r_u^{(t)2}\left[(I-G)(I-G^T)\right]_{uu} = \frac{r^{(t)T} D r^{(t)}}{\min(td, n)}, \tag{30}$$

where D is defined to be a diagonal matrix such that

$$D_{uu} = \left[(I-G)(I-G^T)\right]_{uu} = 1 - 2G_{uu} + \sum_k G_{uk}^2. \tag{31}$$

Therefore, we substitute (29) and (30) into (28) to show that

$$\mathbb{E}_{\tilde{\mathcal{P}}}\left[\left\|r^{(t+1)}\right\|_2^2 \middle| r^{(t)}\right] = r^{(t)T}\left(I - \frac{2I - G - G^T - D}{\min(td, n)}\right) r^{(t)}. \tag{32}$$

We substitute (32) and Lemma A.1a into (27) to show that

$$\begin{aligned}&\mathbb{E}_{\tilde{\mathcal{P}}}\left[\left\|r^{(t+1)} - \mathbb{E}_{\tilde{\mathcal{P}}}\left[r^{(t+1)}\right]\right\|_2^2 \middle| r^{(t)}\right], \\ &= r^{(t)T}\left(I - \frac{2I - G - G^T - D}{\min(td, n)}\right) r^{(t)} - r^{(t)T}\left(I - \left(\frac{I-G}{\min(td, n)}\right)\right)\left(I - \left(\frac{I-G^T}{\min(td, n)}\right)\right) r^{(t)}, \\ &= r^{(t)T}\left(\frac{D}{\min(td, n)} - \frac{(I-G)(I-G^T)}{\min(td, n)^2}\right) r^{(t)}, \\ &\le \left\|\frac{D}{\min(td, n)} - \frac{(I-G)(I-G^T)}{\min(td, n)^2}\right\|_2 \left\|r^{(t)}\right\|_2^2. \end{aligned} \tag{33}$$

By definition, for all u,

$$\|G\|_2 = \left\|G^T\right\|_2 = \max_{\|x\|_2 = 1}\left\|G^T x\right\|_2 \ge \sqrt{e_u^T G G^T e_u} = \sqrt{\sum_k G_{uk}^2}. \tag{34}$$

Therefore, $G_{uu}^2 \leq \sum_k G_{uk}^2 \leq \|G\|_2^2$, and

$$D_{uu} = 1 - 2G_{uu} + \sum_k G_{uk}^2 \leq 1 + 2\|G\|_2 + \|G\|_2^2 = (1 + \|G\|_2)^2. \tag{35}$$

Substitute (35) into (33) to show that

$$\mathbb{E}_{\tilde{\mathcal{P}}}\left[\left\|r^{(t+1)} - \mathbb{E}_{\tilde{\mathcal{P}}}\left[r^{(t+1)}\right]\right\|_2^2 \middle| r^{(t)}\right] \leq \frac{(1 + \|G\|_2)^2}{\min(td, n)}\left(1 + \frac{1}{\min(td, n)}\right)\left\|r^{(t)}\right\|_2^2. \tag{36}$$

We will use the two expressions given in Lemma A.1b to get different upper bounds on $\mathbb{E}_{\tilde{\mathcal{P}}}[\|r^{(t+1)}\|_2^2]$, and then take the minimum. The first bound is most relevant in the sparse setting when n is large and d and $\|G\|_2$ are small. We substitute (36) and the first expression in Lemma A.1b into (26) to show that

$$\mathbb{E}_{\tilde{\mathcal{P}}}[\|r^{(t+1)}\|_2^2] \leq a_t \mathbb{E}_{\tilde{\mathcal{P}}}[\|r^{(t)}\|_2^2] + b_t, \tag{37}$$

for

$$a_t = \frac{(1 + \|G\|_2)^2}{\min(td, n)}\left(1 + \frac{1}{\min(td, n)}\right), \tag{38}$$

and

$$b_t = (t+1)^{-2(1-\|G\|_2)/d}. \tag{39}$$

Therefore, $\mathbb{E}_{\tilde{\mathcal{P}}}[\|r^{(t+1)}\|_2^2] \leq \sum_{k=1}^t Q_k$ for

$$Q_k = \left(\prod_{m=k+1}^t a_m\right) b_k = \left(\prod_{m=k+1}^t \frac{(1 + \|G\|_2)^2}{\min(md, n)}\left(1 + \frac{1}{\min(md, n)}\right)\right)(k+1)^{-2(1-\|G\|_2)/d}. \tag{40}$$

The ratio between subsequent terms can be upper bounded by

$$\frac{Q_k}{Q_{k+1}} \leq \frac{(1 + \|G\|_2)^2}{\min((k+1)d, n)}\left(1 + \frac{1}{\min((k+1)d, n)}\right)\left(\frac{k+1}{k+2}\right)^{-2(1-\|G\|_2)/d}. \tag{41}$$

For $k \geq 1$, $d \geq 4$, and $n \geq 8$,

$$\frac{Q_k}{Q_{k+1}} \leq \frac{4}{8}\left(1 + \frac{1}{8}\right)\left(\frac{2}{3}\right)^{2/4} < \frac{1}{2}. \tag{42}$$

It follows that

$$\mathbb{E}_{\tilde{\mathcal{P}}}\left[\left\|r^{(t+1)}\right\|_2^2\right] \leq Q_t \sum_{k=1}^t \left(\frac{1}{2}\right)^{t-k} \leq 2(t+1)^{-2(1-\|G\|_2)/d}. \tag{43}$$

We similarly obtain another bound by using the second expression of Lemma A.1b. This bound applies in settings when the residual vector $r^{(t)}$ is no longer sparse. By Lemma A.1b,

$\mathbb{E}_{\tilde{\mathcal{P}}}[\|r^{(t+1)}\|_2^2] \leq a_t \mathbb{E}_{\tilde{\mathcal{P}}}[\|r^{(t)}\|_2^2] + b'_t$, for $b'_t = e^{-2t(1-\|G\|_2)/n}$. Therefore, $\mathbb{E}_{\tilde{\mathcal{P}}}[\|r^{(t+1)}\|_2^2] \leq \sum_{k=1}^{t} Q'_k$ for

$$Q'_k = \left(\prod_{m=k+1}^{t} \frac{(1+\|G\|_2)^2}{\min(md,n)} \left(1 + \frac{1}{\min(md,n)}\right)\right) e^{-2k(1-\|G\|_2)/n}. \tag{44}$$

The ratio between subsequent terms can be upper bounded by

$$\frac{Q'_k}{Q'_{k+1}} \leq \left(\frac{(1+\|G\|_2)^2}{\min((k+1)d,n)} \left(1 + \frac{1}{\min((k+1)d,n)}\right)\right) e^{2(1-\|G\|_2))/n}. \tag{45}$$

For $k \geq 1$, $d \geq 4$, and $n \geq 8$,

$$\frac{Q'_k}{Q'_{k+1}} \leq \frac{9e^{2(1-\|G\|_2)/n}}{16} < \frac{3}{4}. \tag{46}$$

It follows that

$$\mathbb{E}_{\tilde{\mathcal{P}}}[\|r^{(t+1)}\|_2^2] \leq Q'_t \sum_{k=1}^{t} \left(\frac{3}{4}\right)^{t-k} \leq 4e^{-2t(1-\|G\|_2)/n}. \tag{47}$$

□

By Markov's inequality, $\mathbb{P}(\|r^{(t)}\|_2 \geq \epsilon) \leq \delta$ for $\mathbb{E}_{\tilde{\mathcal{P}}}[\|r^{(t)}\|_2^2] \leq \delta\epsilon^2$. Therefore, we can directly apply Lemma 4.2 to show that if $\|G\|_2 < 1$, $d \geq 4$, and $n \geq 8$, the algorithm terminates with probability at least $1-\delta$ for

$$t \geq \min\left(\left(\frac{2}{\delta\epsilon^2}\right)^{d/2(1-\|G\|_2)}, 1 + \frac{n}{2(1-\|G\|_2)} \ln\left(\frac{4}{\delta\epsilon^2}\right)\right). \tag{48}$$

Since we are concerned with asymptotic performance, the conditions $d \geq 4$ and $n \geq 8$ are insignificant. To bound the total number of multiplications, we multiply the number of iterations by the maximum degree d.

Translating analysis for $\tilde{\mathcal{P}}$ to $\mathcal{P}$

Let us consider implementation A, which samples coordinates from $\tilde{\mathcal{P}}$, and implementation B, which samples coordinates from $\mathcal{P}$. Let R_A denote the sequence of residual vectors $r^{(t)}$ derived from implementation A, and let R_B denote the sequence of residual vectors $r^{(t)}$ derived from implementation B. The length of the sequence is the number of iterations until the algorithm terminates. We define a joint distribution such that $\mathbb{P}(R_A, R_B) = \mathbb{P}(R_A)\mathbb{P}(R_B|R_A)$.

Let $\mathbb{P}(R_A)$ be described by the algorithm sampling coordinates from $\tilde{\mathcal{P}}$. The sequence R_A can be sampled by separately considering the transitions when non-zero valued coordinates are chosen, and the length of the repeat in between each of these transitions. Given the current iteration t and the sparsity of vector $r^{(t)}$, we can specify the distribution for the number of iterations until the next transition. If we denote $\tau_t = \min\{s : s > t \text{ and } r^{(s)} \neq r^{(t)}\}$, then

$$\mathbb{P}(\tau_t > k | r^{(t)}) = \prod_{q=1}^{k} \left(1 - \frac{\|r^{(t)}\|_0}{\min((t+q)d, n)}\right). \tag{49}$$

Conditioned on the event that a non-zero valued coordinate is chosen at a particular iteration t, the distribution over the chosen coordinate is the same as $\mathcal{P}$.

For all t, $r^{(t+1)} \neq r^{(t)}$ if and only if the algorithm chooses a non-zero valued coordinate of $r^{(t)}$ at iteration t, which according to $\tilde{\mathcal{P}}$, occurs with probability $1 - \|r^{(t)}\|_0 / \min(td, n)$. Therefore, given the sequence R_A, we can identify in which iterations coordinates with non-zero values were chosen. Let $\mathbb{P}(R_B|R_A)$ be the indicator function which is one only if R_B is the subsequence of R_A corresponding to the iterations in which a non-zero valued coordinate was chosen.

We can verify that this joint distribution is constructed such that the marginals correctly correspond to the probability of the sequence of residual vectors derived from the respective implementations. For every (R_A, R_B) such that $\mathbb{P}(R_B|R_A) = 1$, it also follows that $|R_A| \geq |R_B|$, since R_B is a subsequence. For every q,

$$\{(R_A, R_B) : |R_A| \leq q\} \subset \{(R_A, R_B) : |R_B| \leq q\}, \implies \mathbb{P}(|R_A| \leq q) \leq \mathbb{P}(|R_B| \leq q). \tag{50}$$

Therefore, we can conclude that since the probability of the set of realizations such that implementation A terminates within the specified bound is larger than $1-\delta$, it also follows that implementation B terminates within the specified bound with probability larger than $1 - \delta$. Therefore, since we have proved Theorem 2.2 for implementation A, the result also extends to implementation B, i.e., our original algorithm.