A Distributed Eigensolver for Loosely Coupled Networks

Hana Straková and Wilfried N. Gansterer
University of Vienna
Research Group Theory and Applications of Algorithms
Vienna, Austria
hana.strakova@univie.ac.at, wilfried.gansterer@univie.ac.at

***Abstract*—We introduce and analyze a new distributed eigensolver (*dOI*) for square matrices based on orthogonal iteration. In contrast to standard parallel eigensolvers, our approach performs only nearest neighbor communication and provides much more flexibility with respect to the properties of the hardware infrastructure on which the computation is performed. This is achieved by utilizing distributed summation methods with randomized communication schedules which do not require global synchronization across the nodes. Our algorithm is particularly attractive for loosely coupled distributed networks with arbitrary network topologies and potentially unreliable components.**

Based on a novel distributed matrix-matrix multiplication algorithm and on an extension of a distributed QR factorization algorithm proposed earlier, we introduce our distributed eigensolver dOI. We analyze and illustrate the advantages of dOI in terms of higher flexibility with respect to the underlying network and lower communication cost compared to a related distributed algorithm. Moreover, we investigate the inner-outer iterative structure of dOI and propose an adaptive strategy for adjusting the accuracy in the inner iteration to substantially reduce the overall communication cost of dOI.

***Keywords*-distributed eigensolver; distributed orthogonal iteration;gossip algorithm; push-sum algorithm; randomized communication schedule**

I. Introduction

We investigate the problem of computing k extreme eigenpairs of a matrix $A \in \mathbb{R}^{n \times n}$ distributed column-wise over a loosely coupled distributed system (e. g., a P2P network). We propose a localized approach which only requires nearest neighbor communication. It is based on a novel distributed matrix-matrix multiplication algorithm and on a distributed algorithm for computing a QR factorization [1], which we enhance with the ability to trade bandwidth for latency.

A. Motivation and Basic Idea

Motivating eigenvalue problems arise in various types of network analyses (e. g., identifying performance-limiting bottlenecks in computer networks or detecting communities in social networks), as important information about properties of the network structure can be obtained from the spectral analysis of the network [2].

When analyzing large-scale networks in practice, it is often—e. g., due to various technical limitations or due to security constraints—not possible to gather all data at a single node and perform the analysis in a centralized way (using a "fusion center" approach). Classical *parallel* algorithms are designed for tightly coupled parallel target systems, such as static shared memory or distributed memory systems, with static, regular topology and with reliable communication, where (global) synchronization of processes across the nodes in the system can be guaranteed. Such algorithms are not suitable for decentralized loosely coupled distributed systems, such as P2P networks or sensor networks, which are in the focus of this paper. In such networks it cannot be assumed that individual nodes have any knowledge about global system properties, such as the system's (in general arbitrary) topology. Moreover, the structure of such networks may change during the computation due to potential link and/or node failures, and (global) synchronization across nodes is very difficult and expensive or even impossible. We denote algorithms which are designed for such decentralized loosely coupled distributed systems as *distributed* algorithms in order to distinguish them from classical parallel algorithms.

The basic idea pursued in this paper for designing distributed algorithms is to utilize gossip-based summation methods with randomized communication schedules. In particular, we employ the push-sum algorithm [3], for computing, e. g, vector norms or dot products in matrix computations. These techniques greatly increase the flexibility with respect to the properties of the hardware infrastructure and reduce the synchronization requirements between the nodes. They also increase the potential for recovering from (link or node) failures, as discussed in [4].

B. Related Work and Contributions

Gossiping protocols have been studied quite extensively (see, e. g., [5], [6]). Due to their attractive properties, such as no specific requirements in terms of topology, synchronization or reliability, they have been widely utilized, e. g., for signal processing in wireless sensor networks [7]. However, most of the existing work focusses on rather simple gossip-based distributed operations, such as information dissemination [8] or network organization [9]. Most relevant for our approach are gossip-based data aggregation methods, such as the *push-sum* algorithm [3] for averaging or summing values distributed over a network. Distributed algorithms for matrix

operations based on gossiping have been explored very little so far. Recently, a gossip-based distributed QR factorization approach was investigated and compared to state-of-the-art parallel QR factorization algorithms [1].

A distributed orthogonal iteration method in the sense defined in this paper was proposed by Kempe and McSherry [10]. This method, which we denote as *KOI*, differs from our distributed eigensolver in several pivotal aspects. First, KOI is restricted to computing eigenpairs of the *adjacency matrix* of the network where the computation is performed. Second, in KOI every node always communicates with *all* its neighbors at once and therefore they actually do not exploit the full flexibility and fault tolerance potential which is offered by randomized communication schedules. Third, KOI does *not* exploit the potential of gossip-based algorithms for substantially reducing communication cost by accuracy-performance trade-offs.

Another distributed eigensolver based on *power iteration* was introduced in [11]. It differs from our work in similar aspects as KOI: the algorithm is designed for the adjacency matrix of the underlying network, nodes communicate with all neighbors and the potential of gossip-based algorithms for reducing communication cost by accuracy-performance trade-offs has not been investigated. Moreover, this method is designed for computing only one eigenvector without the corresponding eigenvalue.

Contributions of this Paper. We present and analyze a new distributed orthogonal iteration algorithm *dOI* for the in-network computation of principal eigenpairs of a square matrix. dOI only requires nearest neighbor communication, even for matrices which differ from the adjacency matrix of the network. In contrast to existing work, dOI is based exclusively on communication with randomly selected neighbors instead of using fixed communication schedules. We introduce a distributed matrix-matrix multiplication *dmmm* and we extend the distributed QR factorization *dmGS* presented in [1] in order to support trading bandwidth against latency. We utilize both distributed algorithms in the dOI algorithm.

Furthermore, we exploit the potential of gossip-based algorithms for accuracy-performance trade-offs and we suggest and evaluate an adaptive variant of dOI which adjusts the inner accuracy in each iteration of the orthogonal iteration process. We illustrate that, compared to KOI, dOI, and in particular, the adaptive dOI, have substantially lower communication cost in many scenarios and that they are more flexible with respect to the hardware infrastructure.

Synopsis. In Section II, related work is summarized. In Section III, the distributed eigensolver dOI is introduced. We describe the two main building blocks dmmm and dmGS. For the latter, we present an enhanced version of an algorithm proposed earlier. In Section IV we discuss the main strengths of dOI and illustrate them with simulation results. Section V concludes the paper and summarizes ongoing and future work.

II. Background

In this section first we review the central building block for all distributed algorithms discussed in this paper. Second, we review the structure of the standard orthogonal iteration method and third, we summarize important facts about KOI [10].

A. Gossip Protocols and Push-Sum

Gossip (or *epidemic*) protocols are communication protocols for distributed systems based on randomized information exchange which do *not* assume synchronized computation, reliable communication links or global knowledge about the topology of the underlying network. All communication is usually restricted to nearest neighbors.

All distributed matrix algorithms presented in this paper are based on the *push-sum* algorithm [3], an iterative gossip algorithm for computing a sum or an average of values distributed over N nodes. Push-sum starts with one value s_l stored in each node l and then iteratively approximates in every node the average $\sum_{l=1}^{N} \frac{s_l}{N}$ or the sum $\sum_{l=1}^{N} s_l$. The push-sum algorithm proceeds in rounds, which we call *PS-rounds*. In each PS-round, every node sends its current state to one randomly chosen neighbor, and updates its current approximation of the result based on information received from its neighbors. The final accuracy of the estimates in nodes depends on the number of PS-rounds executed.

We call the above described approach *scalar push-sum*, because the nodes exchange messages containing scalars. However, this concept can be extended to *vector push-sum* and *matrix push-sum*, when the nodes exchange messages containing vectors or matrices, which are summed up element-wise.

In [3] it was shown that the scalar push-sum converges to a final accuracy ϵ within $\mathcal{O}(\log(N/\epsilon))$ PS-rounds, under the assumption that each node can communicate with any other node in the network. Similar convergence rates are achieved also for some more general topologies [5], [4].

B. Orthogonal iteration

Classical (sequential) orthogonal iteration [12] is shown in the left part of Algorithm 2. This iterative eigensolver starts with a given matrix $A \in \mathbb{R}^{n \times n}$ and an arbitrary matrix $Q_0 \in \mathbb{R}^{n \times k}$. Each *iteration* consists of two steps – matrix-matrix multiplication (Line 3 in the left part of Algorithm 2) followed by a reduced QR factorization (Line 4 in the left part of Algorithm 2). It can be shown [12] that the columns of the matrices Q_i converge to the extreme k eigenvectors of A and the diagonal elements of the matrices R_i converge to the corresponding k eigenvalues of A under the assumption that the leading $k+1$ eigenvalues are distinct in absolute value: $|\lambda_1| > |\lambda_2| > ... > |\lambda_k| > |\lambda_{k+1}| \geq ... \geq |\lambda_n|$.

C. Existing Distributed Orthogonal Iteration (KOI)

A distributed orthogonal iteration algorithm for computing k principal eigenvectors of the *adjacency matrix* $A \in \mathbb{R}^{n\times n}$ of the underlying network with $N = n$ nodes was introduced in [10].

KOI assumes row-wise distribution of matrices $A \in \mathbb{R}^{n\times n}$ and $Q_i \in \mathbb{R}^{n\times k}$. As in Algorithm 2, the first step of KOI is a matrix-matrix multiplication. Each node l, in order to compute the row $V_i(:,l) = A(:,l) \cdot Q_i$ in iteration i, needs all rows $Q_i(:,j)$, where $A(j,l) \neq 0$. Because A is the (weighted) adjacency matrix of the underlying network, each node l needs only the rows of Q_i from its direct neighbors. As a result, V_i is computed exactly, but each node needs to communicate with all neighbors and if only nearest neighbor communication is admitted, KOI works only for the adjacency matrix of the underlying system.

The second step is reformulated in order to avoid the QR factorization, because it was not available to the authors. They substituted this step with a mathematically equivalent sequence of an outer product of vectors, distributed summing of matrices, Cholesky factorization and solving a linear system.

Although in [10] the term "push-sum" is used for the distributed summation in KOI, it is rather a *consensus* algorithm. Nodes do *not* communicate only with a single randomly chosen neighbor, but with *all* neighbors at once, which besides of availability of all connections requires also full synchronization across nodes [13].

III. Distributed Orthogonal Iteration (dOI)

In this section we introduce dOI, a distributed algorithm for computing k principal eigenpairs of a matrix $A \in \mathbb{R}^{n\times n}$ which is distributed column-wise over N nodes. The dOI algorithm corresponds to the classical orthogonal iteration (see Algorithm 2) with the difference that the two central matrix computations were replaced by their distributed versions: distributed matrix-matrix multiplication *dmmm* (see Section III-A) and distributed QR factorization *dmGS* (see Section III-B). Except of push-sum, utilized in dmmm and dmGS for distributed summation, the computations in dOI are executed locally. Thus, if each push-sum estimates the sum in double precision accuracy, which can be achieved by executing enough PS-rounds (see [3]), dOI behaves as centralized orthogonal iteration with double precision matrix computations (see also Section IV). $\hat{V}, \hat{Q}, \hat{R}$ in Algorithm 2 stand for the version of V, Q, R when computed using gossip-based building block.

A. Distributed Matrix-Matrix Multiplication (dmmm)

dmmm (see Line 3 in the right part of Algorithm 2) computes a product of two matrices in a distributed manner and it is based on a matrix push-sum algorithm (see Section II-A). It expects as input a square matrix $A \in \mathbb{R}^{n\times n}$ distributed column-wise and a matrix $Q \in \mathbb{R}^{n\times k}, k \leq n$ distributed row-wise over N nodes, $N \leq n$. If $N < n$, nodes may contain more than one column of A, resp. more than one row of Q.

In order to compute the product $V = AQ$, where $V_{ij} = \sum_{k=1}^{n} A_{ik}Q_{kj}$, each node first locally computes the outer product of the corresponding column of A and row of Q. The resulting $n \times k$ matrices in all nodes serve as initial values for the matrix push-sum. Then, matrix push-sum computes in each node l a matrix $\hat{V}^l$, which is an estimate of an element-wise sum of the initial matrices from all nodes and at the same time an estimate of the product $V = AQ$. After the matrix push-sum each node l keeps only the rows of matrix $\hat{V}^l$ which correspond to the rows of matrix Q stored in this node l. The rows of matrix $\hat{V}^l$ kept in all nodes form the matrix $\hat{V}_i$, which is then used as an input for dmGS.

Algorithm 1 Orthogonal Iteration (OI)

Input: $A \in \mathbb{R}^{n\times n}$, arbitrary $Q_0 \in \mathbb{R}^{n\times k}$, number of iterations t
Output: k eigenvectors $Q_t \in \mathbb{R}^{n\times k}$, k eigenvalues diag(R_t), $R_t \in \mathbb{R}^{k\times k}$

1: i = 0;
2: **repeat**
3: $[V_i] = A \cdot Q_i$
4: $[Q_{i+1}, R_{i+1}] = qr(V_i)$
5: $i = i + 1$;
6: **until** $i == t$

B. Distributed QR Factorization (dmGS)

dmGS ([1]) is a distributed gossip-based algorithm for computing a reduced QR factorization of an arbitrary matrix $V \in \mathbb{R}^{n\times k}, n \geq k$, distributed row-wise over a loosely coupled distributed system. It computes a matrix $Q \in R^{n\times k}$ with orthonormal columns, distributed row-wise over the nodes, and an upper-triangular matrix $R \in \mathbb{R}^{k\times k}$, such that $V = QR$. Matrix R is either distributed column-wise over the nodes or each node keeps its own local estimate of the matrix R, depending on the storage resources in each node. Simulations showed that dmGS preserves known properties of the sequential orthogonalization algorithms in terms of factorization error and orthogonality of the

Algorithm 2 Distributed Orthogonal Iteration (dOI)

Input: $A \in \mathbb{R}^{n\times n}$, arbitrary $Q_0 \in \mathbb{R}^{n\times k}$, number of iterations t
Output: k eigenvectors $Q_t \in \mathbb{R}^{n\times k}$, k eigenvalues diag(R_t), $R_t \in \mathbb{R}^{k\times k}$

1: i = 0;
2: **repeat**
3: $[\hat{V}_i] = dmmm(A, \hat{Q}_i)$
4: $[\hat{Q}_{i+1}, \hat{R}_{i+1}] = dmGS(\hat{V}_i)$
5: $i = i + 1$;
6: **until** $i == t$

computed matrix Q.

Trading Bandwidth for Latency. With the increasing number of nodes N, dmGS scales like the push-sum algorithm. As mentioned in Section II-A, for many important topologies the number of rounds required in the push-sum algorithm scales as $\mathcal{O}(\log N)$ for a fixed final accuracy ϵ. It has been shown in [1] that dmGS also scales well with increasing number of rows n of the input matrix V, but it does not scale well with increasing number of columns k. A modification of dmGS where we replace several calls of scalar push-sum by one call of a vector push-sum improves the scaling behavior of dmGS with k and reduces communication cost in terms of number of messages sent per node by a factor of k. This modification, which is well known under the term "message coalescing" for MPI codes, does not have any influence on the numerical accuracy or convergence and is suitable if bandwidth can be traded for latency.

Although distributed and parallel algorithms are designed for systems with different assumptions, it is interesting to compare the cost needed. In [1] a detailed comparison of the original dmGS to parallel mGS and to the communication-optimal CAQR algorithm [14] have been provided. The updated comparison of dmGS using vector push-sum to the parallel algorithms is summarized in Table I. Due to the improvement, dmGS is now comparable to the parallel mGS in number of messages sent and only by a factor $\sqrt{N}/log^2(N)$ worse than CAQR. But remember that in contrast to these parallel algorithms it requires only nearest neighbor communication. For the rest of this paper, we operate with the version of dmGS which is based on a vector push-sum algorithm.

	Operation count	# words	# messages
mGS	$\frac{2nk^2}{N}$	$\frac{N^2 \log N}{2}$	$2N \log N$
CAQR	$\frac{2k^3}{3} + \frac{2k^2 n}{N}$	$\frac{3N^2 \log(N)}{(4\sqrt{N})}$	$\frac{3\sqrt{(N)} \log^3(N)}{8}$
dmGS	$\mathcal{O}(k^2(\log N + n/N))$	$\mathcal{O}(k^2 \log(N))$	$\mathcal{O}(k \log(N))$

Table I
COMPARISON OF DMGS TO STATE-OF-THE-ART PARALLEL ALGORITHMS (N = K = N)

IV. THE STRENGTHS OF DOI

In this section we present simulation results showing convergence behavior and numerical accuracy of dOI. Moreover, we summarize the advantages of dOI over the KOI algorithm from [10]. We focus on test cases where the classical orthogonal iteration algorithm performs well. We used random symmetric matrices $A \in \mathbb{R}^{n \times n}$ with elements from the interval $[-100, 100]$ and for each matrix $k = 5$ eigenpairs were computed. The first six eigenvalues of each input matrix satisfy $|\lambda_1| > |\lambda_2| > |\lambda_3| > |\lambda_4| > |\lambda_5| > |\lambda_6|$ and the minimal distance between these eigenvalues is 10. The matrix Q_0 was initialized with the first k unit vectors of size n. Each dOI ran for a prescribed number t of iterations and push-sum algorithms were terminated as soon as estimates in all nodes reached a given level of accuracy ϵ. We consider the following network topologies: *fully connected* (each node is connected with every other node), *hypercube* topology, *random* topology (each (randomly deployed) node is connected to all nodes in some radius) and *random regular* topology with degree d (each node is connected to randomly selected d nodes).

A. Increased Flexibility

As summarized in Section II, KOI is designed for adjancency matrices. In principle, KOI could compute eigenpairs of a matrix which is not the adjacency matrix of the network. However, it would have to use multi-hop communication to obtain the right rows of Q_i in each iteration i and for that complex routing in the network would be needed. In contrast, fully gossip-based dOI is much more flexible in respect to the underlying hardware infrastructure. It can handle matrices with no requirements on the structure and/or larger than the number of nodes N in the underlying network when using communication *only* with direct neighbors. If $N < n$, each node stores n/N consecutive columns of the matrix A, resp. rows of the matrix Q. Before every push-sum each node computes the initial value as local sum of the corresponding elements of all stored rows, resp. columns.

Fig. 1 shows relative residuals when computing 5 dominant eigenpairs of randomly generated matrix $A \in \mathbb{R}^{n \times n}$ over a network with hypercube topology and $N = n = 2^9$. We can notice that the final accuracy of residuals corresponds to the accuracy level ϵ set in the push-sum algorithms. The interesting aspect is that the convergence speed is always the same, for sequential OI as well as for dOI with different ϵ.

Fig. 2 shows communication cost required for a prescribed final accuracy when $n \geq N$. dOI computed the first 5 dominant eigenpairs of a matrix $A \in R^{256 \times 256}$ distributed over networks with hypercube topology with sizes $N = 2^5; 2^6; 2^7$ and 2^8 and $\epsilon = 10^{-15}$. We can see that the number of messages per node stays similar regardless of the size of the underlying network.

B. Reduced Communication Cost

Total number M_k of messages sent per iteration of KOI can be expressed as $M_K = 2|E| + \text{rd}_{CO} \cdot 2|E|$, where $|E|$ is the number of edges (undirected links) in the network and rd_{CO} is the number of rounds in the distributed summing (consensus) in KOI. Total number M_d of messages sent per iteration in dOI is $M_d = N \cdot \text{rd}_{PS} + N \cdot \text{rd}_{PS} \cdot (2k - 1)$, where rd_{PS} is the number of PS-rounds and $(2k-1)$ is the number of push-sum algorithms in dmGS.

First, we investigate how the number of messages sent scales with increasing N if we compute five principal eigenpairs of the adjacency matrix of a fully connected network. We have $M_K = N(N-1) + 1 \cdot N(N-1)$, because $|E| = N(N-1)$ and in one iteration of consensus each node exchanges information with every other node, and $M_d = N \cdot \mathcal{O}(\log N) + N \cdot \mathcal{O}(\log N) \cdot (2k-1)$, because $\mathrm{rd}_{PS} = \mathcal{O}(\log N)$ (see [3]). The simulations confirm the theoretical result that in this case dOI scales better than KOI with increasing N (see Fig. 3).

For other topologies it is more difficult to compare the communication cost theoretically, since it depends in different ways on the concrete underlying topology. In particular, with increasing number of edges $|E|$ the number of messages sent per iteration in KOI increases, but the number of rounds rd_{CO} to converge decreases. Similar influence we can observe in dOI: the stronger connected the network, the smaller number of PS-rounds rd_{PS}. However, it is not easy to see, which term will be dominating. Investigation of the concrete influence of the topology is planned for the near future. In this paper we present comparison of communication cost based on simulations (see Fig. 6).

An important aspect of distributed gossip-based algorithms, e. g., dmmm and dmGS, is their ability to adjust the invested cost to the accuracy of the result. The higher accuracy ϵ is prescribed for push-sum the more PS-rounds are executed and thus, more messages are exchanged. Our goal is to use this property to minimize the overall cost of dOI. KOI cannot apply such adapting strategy because the matrix-matrix multiplication is computed exactly.

In Fig. 1 we can see that dOI with different ϵ leads to different final accuracies achieved, but the convergence speed remains the same ! Thus, it seems to be much more efficient to start the computation with low ϵ and thus with low cost and then gradually increase the accuracy of distributed matrix computations up to the desired final accuracy. Compared to the original dOI the total number of PS-rounds and thus the total cost of such *adaptive* dOI can be significantly reduced.

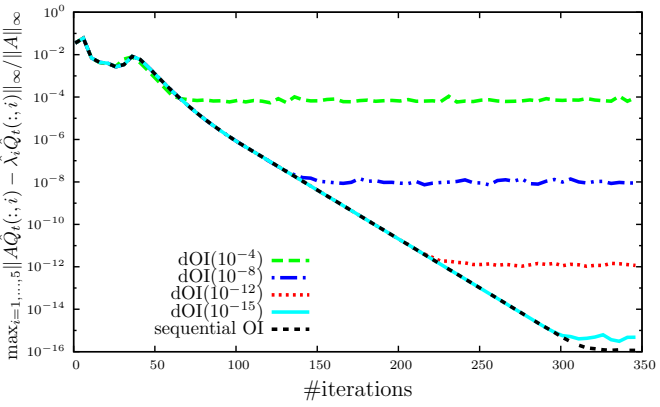

Figure 1. Convergence behavior of sequential OI and dOI with various accuracy levels ϵ on a hypercube topology with $N = 2^9$ nodes with input matrix $A \in \mathbb{R}^{512\times512}$.

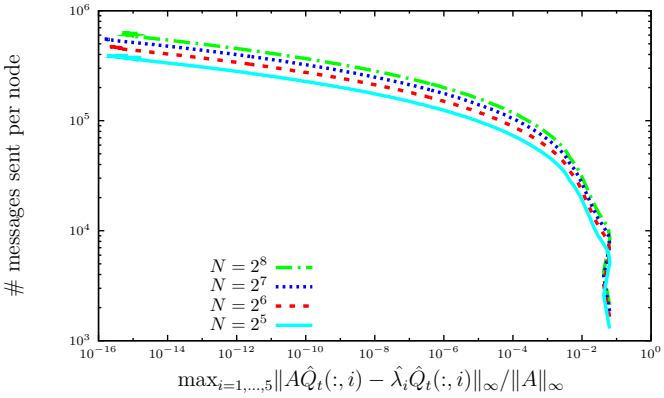

Figure 2. Communication cost of dOI(10^{-15}) per node for different sizes of hypercubes with input matrix $A \in \mathbb{R}^{256\times256}$.

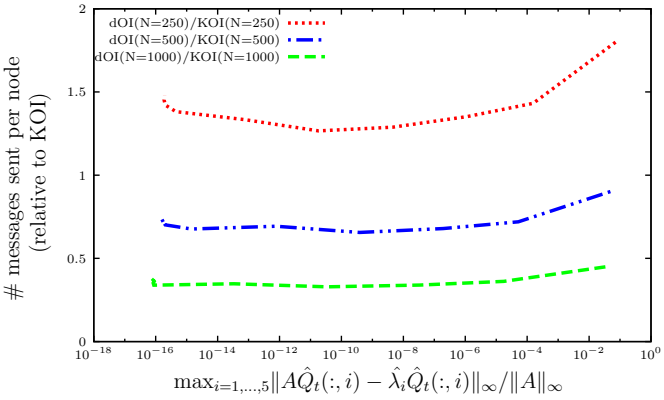

Figure 3. Communication cost per node of dOI(10^{-15}) for computing 5 eigenpairs of adjacency matrix of fully connected graphs depicted relatively to KOI.

We simulated adaptive dOI for finding 5 dominant eigenpairs of matrices $A \in \mathbb{R}^{n\times n}, n = 32; 64; 128; 256$ over hypercube topology with $N = n$ nodes. Adaptive dOI started with $\epsilon = 10^{-1}$ and the accuracy was increased by one decimal place up to $\epsilon = 10^{-15}$ every 22 iterations. For determining the step we used the knowledge of the number of iterations $t = 330$ prescribed to converge to final accuracy. Fig. 4 shows that adaptive dOI significantly reduces the number of messages sent per node compared to the original dOI with $\epsilon = 10^{-15}$. The results are averaged over simulations for 20 different random matrices. Fig. 5 illustrates that the adaptive dOI preserves the convergence speed compared to sequential OI.

The comparison of communication cost per node for topologies different from the hypercube are shown in Fig. 6. We can see the communication cost relative to KOI for computing 2 eigenpairs of an adjacency matrix for two different scenarios. First, we investigated a randomly generated

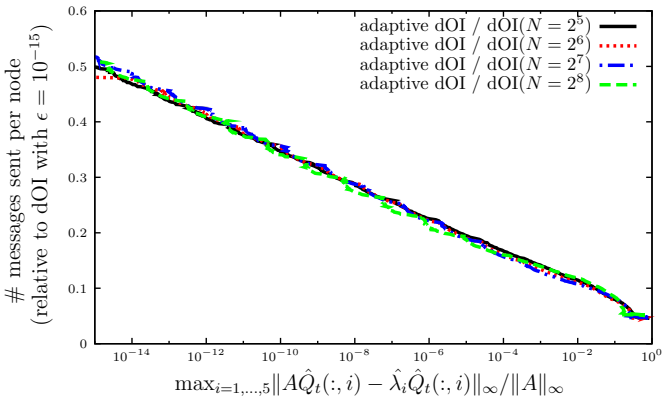

Figure 4. Communication cost of adaptive dOI with $\epsilon = 10^{-15}$ for matrices $A \in \mathbb{R}^{n\times n}, n = 32; 64; 128; 256$ over hypercubes with $N = n$ depicted relatively to dOI. (averaged over 20 simulations for different random input matrices)

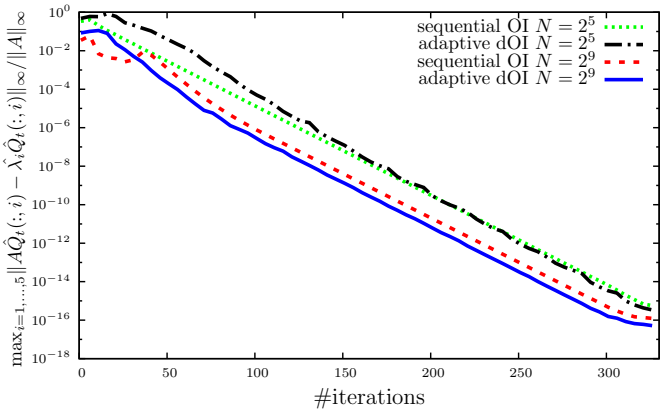

Figure 5. Convergence behavior of sequential OI and adaptive dOI for $A \in \mathbb{R}^{n\times n}, n = 32; 512$ over hypercube topology with $N = n$ nodes.

network with $N = 20$ and average node degree $\bar{d} = 7,2$, which corresponds to a small sensor network. From the relative comparison we can see that dOI needs about 3/4 of the messages sent by KOI and adaptive dOI at most 2/5. The other lines show the comparison for a random regular graph with $N = 500$ nodes, where each node has a degree $d = 40$. For this scenario dOI needs about 3/10 of the messages sent by KOI and adaptive dOI slightly more than 1/10.

C. Higher Robustness

Beyond the advantages of dOI discussed so far, it is potentially also more *robust* and *resilient* than other methods. We focus on two aspects in this direction: resilience against asynchrony between nodes and resilience against various types of failures.

With respect to asynchrony, dOI is quite robust because all nodes can proceed independently of each other. We are currently investigating how much synchronization between neighbors is required in dmGS and first results show that very efficient *local* synchronization strategies between neighbors are sufficient to guarantee accurate results. The same ideas are expected to work well for dOI. This is in contrast to KOI, which is based on a form of a consensus algorithm [13], where every node communicates with *all* its neighbors at once in each round. As a consequence, the nodes cannot proceed independently of each other.

With respect to failures, dOI also benefits from its randomized communication schedules. In [4] the potential of gossip-based algorithms to handle various types of failures, such as message loss, corruption of data or node failures has been discussed. It has been shown how dmGS can be made resilient against message loss and node failures. Similar techniques are expected to lead to fault tolerant variants of dOI. In contrast to dOI, KOI does not have this capability due to the exact computation of matrix-matrix multiplication, which cannot tolerate failures in the network. A more extensive investigation of the behavior of dOI in asynchronous and unreliable systems is work in progress.

V. Summary and Conclusions

We designed and analyzed the distributed gossip-based orthogonal iteration algorithm dOI for computing k principal eigenpairs of a square matrix A distributed column-wise over a loosely coupled decentralized network. The dOI algorithm is based on a new distributed matrix multiplication method and on the distributed QR factorization method dmGS proposed earlier. It utilizes the push-sum algorithm for summing distributed values.

To the best of our knowledge, the only previously existing (truly) distributed orthogonal iteration is the one proposed in [10], which we denote as KOI. In contrast to KOI, our solver dOI has completely randomized communication schedules, which leads to many attractive properties.

dOI is very flexible with respect to the underlying hardware infrastructure. Whereas KOI operates only on the adjacency matrix of the underlying network, dOI is not

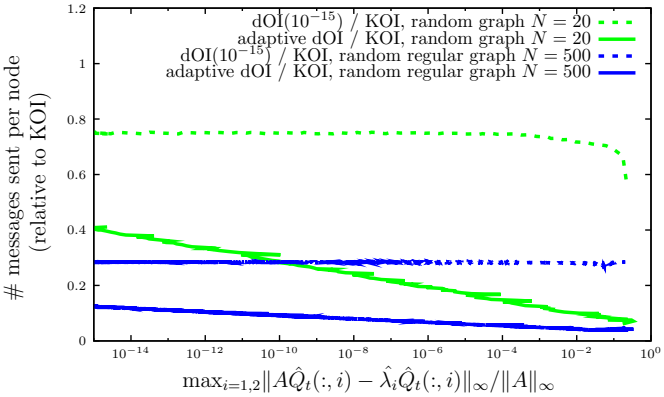

Figure 6. Communication cost per node of dOI(10^{-15}) for computing 2 eigenpairs of the adjacency matrix of a random regular graph with $N = 500$ and $d = 40$ and of a random graph with $N = 15$ and $\bar{d} = 7,2$ depicted relatively to KOI.

restricted to the adjacency matrix and decouples the matrix structure from the topology of the network. We illustrated that the accuracy achieved by dOI is of the same order of magnitude as the tolerance used for terminating the push-sum algorithm. Consequently, dOI preserves the numerical properties of the classical sequential orthogonal iteration algorithm if the distributed matrix computations deliver results to floating-point accuracy. Moreover, we also illustrated that the convergence speed of dOI remains the same regardless of the accuracy achieved by each instance of the push-sum algorithm. These insights lead to an adaptive variant of dOI, which significantly reduces the overall communication cost by dynamically adapting the termination criterion for the push-sum algorithm. The number of messages sent could be reduced by a factor of two compared to the original dOI and up to a factor of ten compared to KOI. Last, but not least, we outlined other potentially interesting properties of dOI in terms of robustness and resilience against asynchrony as well as against various types of failures.

Ongoing and Future Work.: We are currently developing on a robust variant of dOI for asynchronous and unreliable distributed networks. Moreover, we intend to refine the control of the termination criterion of the push-sum algorithm in the inner iteration such that it can be automatically adapted by the algorithm based on purely local information.

ACKNOWLEDGMENT

This work has been funded by the Austrian Science Fund (FWF) in project S10608-N13 (NFN SISE).

REFERENCES

[1] H. Straková, W. N. Gansterer, and T. Zemen, "Distributed QR factorization based on randomized algorithms," in *Proceedings of the 9th International Conference on Parallel Processing and Applied Mathematics, Part I*, ser. Lecture Notes in Computer Science, vol. 7203, 2012, pp. 235–244.

[2] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 98, pp. 298–305, 1973.

[3] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, 2003, pp. 482–491.

[4] W. N. Gansterer, G. Niederbrucker, H. Straková, and S. Schulze Grotthoff, "Robust distributed orthogonalization based on randomized aggregation," in *Proceedings of the Second Workshop on Scalable algorithms for Large-Scale Systems*, ser. ScalA '11. New York, NY, USA: ACM, 2011, pp. 7–10.

[5] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. Information Theory*, vol. 52, no. 6, pp. 2508 – 2530, 2006.

[6] A.-M. Kermarrec and M. van Steen, "Gossiping in distributed systems," *SIGOPS Oper. Syst. Rev.*, vol. 41, pp. 2–7, 2007.

[7] A. Dimakis, S. Kar, J. Moura, M. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847 –1864, 2010.

[8] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading," in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 2000, pp. 565–.

[9] D. Shah, "Gossip algorithms," *Found. Trends Netw.*, vol. 3, pp. 1–125, 2009.

[10] D. Kempe and F. McSherry, "A decentralized algorithm for spectral analysis," *Journal of Computer and System Sciences*, vol. 74, no. 1, pp. 70 – 83, 2008.

[11] M. Jelasity, G. Canright, and K. Eng-monsen, "Asynchronous distributed power iteration with gossip-based normalization," in *Euro-Par 2007, volume 4641 of Lecture Notes in Computer Science*. Springer-Verlag, 2007, pp. 514–525.

[12] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, 1997.

[13] P. Denantes, F. Benezit, P. Thiran, and M. Vetterli, "Which distributed averaging algorithm should i choose for my sensor network?" in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, 2008, pp. 986 –994.

[14] J. Demmel, L. Grigori, M. F. Hoemmen, and J. Langou, "Communication-optimal parallel and sequential QR and LU factorizations," no. UCB/EECS-2008-89, EECS Department, University of California, Berkeley, Tech. Rep., 2008.