Distributed QR Factorization Based on Randomized Algorithms

Hana Straková[1], Wilfried N. Gansterer[1,⋆], and Thomas Zemen[2]

[1] University of Vienna, Austria,
Research Group Theory and Applications of Algorithms
Hana.Strakova@univie.ac.at, Wilfried.Gansterer@univie.ac.at
[2] Forschungszentrum Telekommunication Wien, Austria
Thomas.Zemen@ftw.at

Abstract. Most parallel algorithms for matrix computations assume a static network with reliable communication and thus use fixed communication schedules. However, in situations where computer systems may change dynamically, in particular, when they have unreliable components, algorithms with randomized communication schedule may be an interesting alternative.
We investigate randomized algorithms based on gossiping for the distributed computation of the QR factorization. The analyses of numerical accuracy showed that known numerical properties of classical sequential and parallel QR decomposition algorithms are preserved. Moreover, we illustrate that the randomized approaches are well suited for distributed systems with arbitrary topology and potentially unreliable communication, where approaches with fixed communication schedules have major drawbacks. The communication overhead compared to the optimal parallel QR decomposition algorithm (CAQR) is analyzed. The randomized algorithms have a much higher potential for trading off numerical accuracy against performance because their accuracy is proportional to the amount of communication invested.

Keywords: distributed vs. parallel QR factorization; decentralized QR factorization; evaluation of distributed algorithms; gossip algorithms; push-sum algorithm; randomized communication schedule; fault-tolerance

1 Introduction

We consider the distributed computation of the QR factorization over a loosely coupled distributed system where existing approaches with fixed communication schedule have major drawbacks. We develop and analyze a randomized distributed QR decomposition based on the push-sum algorithm. Since the nodes proceed in an asynchronous and decentralized way, our approach is more flexible than existing parallel QR decomposition algorithms and capable of handling unreliable links as well as potential dynamic changes of the network. The main

⋆ corresponding author

goal of this paper is a comprehensive comparison of this new randomized QR decomposition approach with the existing parallel algorithms.

In this paper, we use the term *parallel* for algorithms developed primarily for tightly coupled parallel computers (comprising shared-memory systems, multi-core architectures, tightly coupled distributed memory systems, etc.) which are characterized by a *static* topology and *reliable*, relatively fast, (usually) wired communication. For such target platforms algorithms with fixed communication schedules are suitable. In contrast, we use the term *distributed* for algorithms designed for loosely coupled decentralized distributed systems. These target platforms are characterized by potentially dynamically changing topology, relatively slow and costly communication, and/or by unreliable components (e.g., communication links). This includes, but is not limited to wireless networks, peer to peer (P2P) networks, mobile ad hoc networks, etc. Most existing parallel algorithms have major drawbacks on such systems and thus are not applicable.

Motivation and Approach. Motivating problems arise, for example, in decentralized network analysis (cf. [12]) or in cooperative transceiver design in telecommunications (cf. [8]). In the latter case the objective of a method called *distributed sphere decoding* is to exploit cooperation of receiving node with other receivers in the decoding of a signal sent by a group of transmitters [14]. A distributed QR factorization of a distributed channel matrix, which describes the quality of communication channels between nodes, is needed for solving a maximum likelihood problem.

A theoretical alternative to existing parallel algorithms for the context considered would be a "fusion center" approach. The fusion center first collects the entire matrix, computes the QR factorization locally, and then returns the result to the other nodes. However, this leads to an unfair distribution of energy consumption among the cooperating nodes and may be infeasible due to memory constraints (if no single node can store the entire matrix). It has also been shown that in many situations in-network computation leads to considerable advantages over centralized approach [17, 15] (e.g., in terms of energy savings).

Consequently, we pursue a randomized decentralized QR factorization algorithm based on *gossiping*. *Gossip-based* (or *epidemic*) algorithms are characterized by asynchronous randomized information exchange, usually only within the local neighborhood of each node. They do not assume static or reliable networks, do not require any specialized routing and do not have a single point of failure. Thus, they can cope with link failures or more general dynamic network changes, and consequently achieve high fault tolerance. Due to their ideally completely decentralized structure, gossip-based algorithms tend to scale well with the number of nodes. Moreover, gossip-based algorithms allow for trading off time-to-solution against communication cost (and thus energy consumption) or fault tolerance by gradually adapting the intensity and regularity of communication with other nodes. Especially the latter property is very attractive when parallel algorithms are not applicable and fusion center approaches are too expensive, or where not only the highest achievable accuracy, but also intermediate approximate results are of interest. Existing parallel algorithms produce the full result at the full accuracy achievable for the input data only when they terminate regularly.

Related Work. Parallel algorithms for computing QR factorizations on shared memory, distributed memory or on multicore architectures have been studied extensively. State-of-the-art software libraries such as ScaLapack [3] use blocked algorithms for implementing parallel QR factorization. The latest developments are *tiled QR factorization* [5], *communication-avoiding QR factorization* (*CAQR*) [6], and the combination of the two approaches [16].

Much less work exists in the literature on distributed algorithms. Although wireless networks are mentioned as target environment in [1], the assumptions used are those typical for parallel algorithms: all data initially centralized at a single node, specific roles assigned to individual nodes (distributing node, annihilating node, etc.) and the communication schedule is fixed. Relevant for our approach are randomized algorithms, such as the push-sum algorithm [11], utilized for matrix computations. Simple gossip-based algorithms are used due to their attractive properties in many in-network computations (e.g, in distributed signal processing [7]), but to the best of our knowledge, only a few gossip-based matrix algorithms have been investigated. A gossip-based decentralized algorithm for spectral analysis was discussed in [12], and a distributed QR factorization based on push-sum has been used in distributed sphere decoding in [8]. So far, no analysis of numerical behavior or communication cost of a randomized distributed QR factorization is available. Moreover, no comparison to existing parallel approaches is available.

Contributions. We extend existing work in the following algorithmic aspects: We use *modified* Gram-Schmidt orthogonalization (mGS) as the basis for our new algorithm, which we call *dmGS*, whereas the algorithm used in [8] implements the less stable *classical* Gram-Schmidt orthogonalization (cGS). Moreover, the algorithm used in [8] assumes that after each push-sum algorithm one of the local estimates is broadcasted to all the other nodes, whereas our approach proceeds with the local estimate in each node (which is more realistic in practice).

We for the first time provide a full quantitative evaluation of a distributed QR decomposition algorithm in terms of numerical accuracy, reliability, operation count, memory requirements, and communication cost (partly based on theoretical analyses and partly based on simulations) as well as a quantitative comparison with state-of-the-art parallel QR decomposition algorithms. We also consider more general data distributions over random topologies of the underlying distributed system ([8] considered only square matrices over fully connected networks), investigate the effects of communication link failures, and we adapted termination criteria from the literature in order to be able to investigate the trade-off between numerical accuracy and performance or communication cost.

Synopsis. In Section 2, the QR factorization and the push-sum algorithm are reviewed. In Section 3, our new algorithm (dmGS) is presented. In Section 4, numerical simulations and theoretical analyses are summarized. Section 5 concludes the paper.

2 Methodology

Algorithms for QR Factorization. We are interested in computing the *thin* (reduced) QR factorization $A = QR$ (where $A \in \mathbb{C}^{n\times m}$, $n \geq m$, $Q \in \mathbb{C}^{n\times m}$ has orthogonal columns, $R \in \mathbb{C}^{m\times m}$ is upper triangular) by classical (cGS) and modified (mGS) Gram-Schmidt orthogonalization (for mGS see Algorithm 1).

Both Gram-Schmidt orthogonalization processes require $2nm^2$ flops for computing the QR factorization of an $n \times m$ matrix [10]. Denoting the machine precision with $\varepsilon_{\text{mach}}$ and the condition number of the matrix A with $\kappa_2(A)$, the matrix Q produced by mGS satisfies $\|I - Q^H Q\|_2 \approx \varepsilon_{\text{mach}}\kappa_2(A)$ [10], whereas for a square matrix A of size $n \times n$ cGS produces Q for which only $\|I - Q^H Q\|_2 \leq c(n)\varepsilon_{\text{mach}}[\kappa_2(A)]^{n-1}$ with a modest constant $c(n)$ holds [13].

The Push-Sum Algorithm. As mentioned in Section 1, the information exchange in gossiping protocols is randomized and asynchronous, and nodes iteratively update their local information according to information received from other nodes. The network topology can be arbitrary, dynamically changing and does not need to be known in individual nodes. The *push-sum algorithm* [11] is a gossip algorithm for approximating the sum or the average of values distributed over a network with N nodes. If $x(k)$ denotes a value stored in node k, the goal of the algorithm is to compute in each node k an estimate $s_k = \sum_j x(j)$ (or $a_k = \sum_j x(j)/N$). The accuracy of the estimates depends on the number of asynchronous iterations of the push-sum algorithm.

An important question is when to terminate the push-sum algorithm. In [11], an asymptotic number of iterations needed in the push-sum algorithm for reaching a certain error τ with some probability is derived (the notation is adapted to our context): *For a network with N nodes, there is a number of iterations $i_0 = O(\log N + \log \frac{1}{\tau} + \log \frac{1}{\delta})$, such that for all numbers of iterations $i \geq i_0$ with probability of at least $1 - \delta$ the estimate of the average satisfies in every node k*

$$\frac{1}{|\sum_{j=1}^{N} x(j)|} \left| \frac{s_k^{(i)}}{w^{(i)}(k)} - \frac{1}{N}\sum_{j=1}^{N} x(j) \right| \leq \tau \frac{\sum_{j=1}^{N} |x(j)|}{|\sum_{j=1}^{N} x(j)|}. \tag{1}$$

where $x(k)$ is a value stored in node k, $w^{(i)}(k)$ is a weight used in node k in iteration i, and $s_k^{(i)}$ is an estimate of the sum $\sum_{j=1}^{N} x(j)$ in node k in iteration i. In [8] the push-sum algorithm is terminated after a fixed number of iterations determined by the authors based on simulations. In [12] a blockwise extension of the push-sum algorithm is proposed and a stopping criterion with an accuracy parameter τ is suggested. We adapt this termination criterion to the scalar push-sum and use it in our simulations.

3 Distributed QR Factorization (dmGS)

In Algorithm 1 the distributed modified Gram-Schmidt orthogonalization dmGS is presented and compared to mGS. The notation used in the algorithm is explained in Section 2. We can see, that mGS and dmGS are very similar. The

Algorithm 1 Modified Gram-Schmidt orthogonalization (mGS) and distributed QR factorization (dmGS)

Input: $A \in \mathbb{C}^{n \times m}$, $n \geq m$ (matrix A distributed row-wise over N nodes; if $n > N$, each node k stores r_k consecutive rows of A)
Output: $Q \in \mathbb{C}^{n \times m}$, $R \in \mathbb{C}^{m \times m}$ (matrix Q distributed row-wise, matrix R distributed column-wise over nodes)

```
1:  for $i = 1$ to $m$ do
2:      $x(k) = A(k,i)^2$
3:
4:      $s = \sum_{k=1}^{n} x(k)$
5:      $R(i,i) = \sqrt{s}$
6:      $Q(:,i) = A(:,i)/R(i,i)$
7:
8:      for $j = i+1$ to $m$ do
9:          $x(k) = Q(k,i)A(k,j)$
10:
11:         $R(i,j) = \sum_{k=1}^{n} x(k)$
12:         $A(:,j) = A(:,j) - Q(:,i)R(i,j)$
13:
14:     end for
15: end for
```

```
1:  for $i = 1$ to $m$ do (in node $k$)
2:      $x(k) = A(k,i)^2$
3:      [ $x(k) = \sum_{t=1}^{r_k} A(k_t,i)^2$ ]
4:      $s_k$ = push-sum$(x)$
5:      $R_k(i,i) = \sqrt{s_k}$
6:      $Q(k,i) = A(k,i)/R_k(i,i)$
7:      if $k \neq i$ delete $R_k(i,i)$
8:      for $j = i+1$ to $m$ do
9:          $x(k) = Q(k,i)A(k,j)$
10:         [ $x(k) = \sum_{t=1}^{r_k} Q(k_t,i)A(k_t,j)$ ]
11:         $R_k(i,j)$ = push-sum$(x)$
12:         $A(k,j) = A(k,j) - Q(k,i)R_k(i,j)$
13:         if $k \neq j$ delete $R_k(i,j)$
14:     end for
15: end for
```

only difference is that the sums in the mGS needed for computing a 2-norm (Line 4) and a dot product (Line 11) are replaced by the push-sum algorithm. dmGS assumes as input a matrix $A \in \mathbb{C}^{n \times m}, n \geq m$ distributed row-wise over N nodes. In the special case $n = N$, each node contains one row of the input matrix $A \in \mathbb{C}^{n \times m}, n \geq m$ and the push-sum computes the sum of corresponding column elements stored in nodes. In the most common case $n > N$, each node k contains $r_k = \mathcal{O}(n/N)$ rows of the input matrix A. In this case, before each push-sum the corresponding column elements stored in one node are locally preprocessed (Lines 3 and 10) and then summed up by the push-sum algorithm. Except of the push-sum algorithms utilized for computing in all nodes k the local estimates R_k of the elements of the matrix R (Lines 4 and 11), the nodes do not communicate and can execute the computations locally. dmGS, as described in Algorithm 1, stores one column of matrix R in corresponding nodes. However, if the lines 7 and 13 are skipped, the nodes keep the computed estimates of the elements of the matrix R and thus store the full matrix R. This of course results in higher local memory requirements.

4 Evaluation of dmGS

In this section, we first investigate the numerical accuracy of dmGS and analyze the influence of the link failures and of the diameter of the network on its

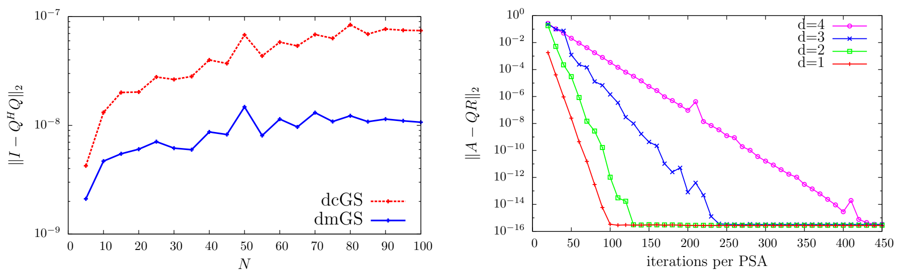

Fig. 1. Comparison of the distributed QR factorization (left) based on the classical (dcGS) resp. modified (dmGS) Gram-Schmidt method in terms of orthogonality measure and influence of random network topologies (right) with different diameter d on convergence speed of dmGS.

convergence. Then we compare dmGS to state-of-the-art parallel algorithms in terms of operation count, local memory requirements, and communication cost.

We present simulation results for dmGS applied to a random matrix $A \in \mathbb{C}^{n\times m}$ distributed over N nodes. Simulations are partly based on Matlab and partly on ns-3. In the Matlab simulations, the nodes choose their communication partners uniformly from the entire network, whereas the ns-3 simulations are based on a predetermined random network topology with a given diameter.

In Fig. 1 (left) distributed QR factorization based on cGS (dcGS) is compared to our algorithm dmGS in terms of orthogonality of Q measured as $\|I - Q^H Q\|_2$ for varying number of nodes N, when applied to $A \in \mathbb{C}^{n\times n}, n = N$. We can observe the expected difference in terms of orthogonality of Q (the factorization errors were basically identical), which illustrates that the distributed algorithm preserves the numerical properties of the sequential and parallel cases. In the experiments the overall factorization error and orthogonality was of the same order of magnitude as the accuracy parameter $\tau = 10^{-9}$ used for terminating the algorithm (see Section 2).

4.1 Reliability and Scalability

In this section we investigate the influence of link failures and random topologies on the behavior of dmGS. In Fig. 1 (right) we can see the behavior of dmGS for arbitrary networks with different diameters d. Although the increasing diameter causes slower convergence, all simulations converge to the full accuracy.

Simulations showed that loss of messages due to link failures does not influence the factorization error. However, it causes loss of orthogonality of the computed matrix Q. To achieve satisfactory orthogonality of Q we have to either avoid losing messages or recover from the loss. The simplest strategy is to resend each message several times to ensure a high probability that it will be delivered. The exact number how many times was each message sent in our experiments was chosen such that the combined probability of not delivering the message for all repeated sending attempts is smaller than 10^{-12}. Fig. 2 (left)

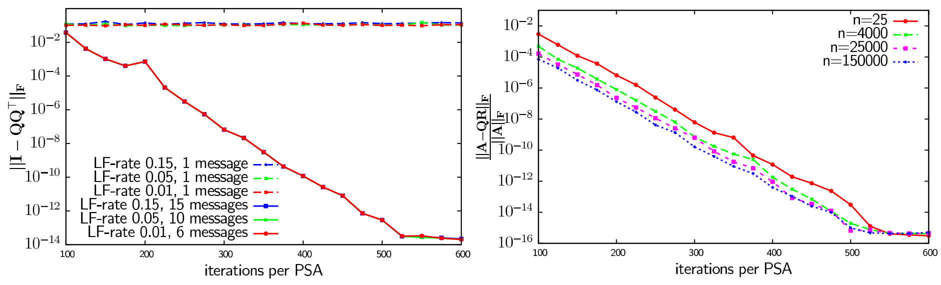

Fig. 2. Comparison of the influence of various failure rates on the orthogonality of Q for sending each message once and several times (left) and scalability of dmGS with increasing number of rows n for matrix $A \in \mathbb{R}^{n\times 25}$ (right).

shows the influence of the link failures on the orthogonality of the matrix Q for various failure rates with and without resending each message. In [9] we discuss approaches for recovering from message and data loss in more detail.

If $n > N$, nodes have to store more than one row. Consequently, for fixed N, dmGS scales very well with increasing n. As shown in Fig. 2 (right), storing several rows in one node even increases the accuracy, since more computation is done locally in nodes. Note that increasing number of rows per node does not affect the communication cost. Increasing the number of columns m is more difficult, because it causes quadratical increase of communication cost. However, a new version of dmGS which improves the scalability for growing m is currently under development.

4.2 Theoretical Analyses and Comparison with Parallel Algorithms

In the following, dmGS is compared with state-of-the-art parallel algorithms for QR factorization when applied to a rectangular matrix $n \times m$ distributed over N nodes (or P processors). In particular, we compare dmGS with parallel mGS and with parallel CAQR (which is optimal in terms of communication cost up to polylogarithmic factors) [6] along the critical path. For dmGS, the terms along the critical path are equal to the cost per node. We have to emphasize that this comparison has to be interpreted carefully, since these different types of algorithms are designed for very different target systems and are based on very different assumptions on properties of the target hardware and on the initial data distribution (see Section 1).

To simplify the analysis, we assume that push-sum algorithm proceeds in synchronous iterations and that in each iteration every node sends exactly one message. Note that this synchronization is *not* required in practice and that our simulations do not rely on this assumption. Push-sum needs $\mathcal{O}(\log N)$ iterations to converge to a fixed accuracy τ for networks where each node can communicate with any other node [11], which we use in our analyses. Similar convergence rate was also shown for more general topologies [4].

Operation count. dmGS calls $m(m+1)/2$ push-sum algorithms and before each push-sum, a local summation of $r_k = \mathcal{O}(n/N)$ elements must be performed in node k. In summary, dmGS performs $\mathcal{O}(m^2 \log N + m^2 n/N)$ flops. According to [6], parallel mGS performs $2nm^2/P$ flops and parallel CAQR performs $2m^3/3 + 2m^2 n/P$ flops along the critical path. Consequently, asymptotically parallel mGS has the lowest flop count of the three methods and the count of dmGS is a little lower than the one of parallel CAQR (for fixed $P = N$).

Local Memory Requirements. In dmGS each node k needs to store $r_k = \mathcal{O}(n/N)$ rows of matrix A, r_k rows of matrix Q and on m of the nodes one column of matrix R, each of them of length m. This leads to local memory requirements of $\Theta(r_k m) = \Theta(nm/N)$. During the push-sum algorithm each node needs a constant number of words of local memory. The local memory requirements for parallel CAQR are $\Theta(nm/P)$ [6]. This shows that asymptotically CAQR and dmGS have the same local memory requirements.

Communication Cost. We investigate three communication cost metrics: the number of messages and the number of words sent along the critical path as well as the average message size. The total number C_t of messages sent during dmGS is given as $C_t = S \cdot I \cdot M$, where S is the number of push-sum algorithms, I is the number of iterations per push-sum and M is the number of messages sent during one iteration of the push-sum algorithm. dmGS calls $S = m(m+1)/2$ push-sum algorithms (cf. Algorithm 1) and as described above, $I = \mathcal{O}(\log N)$ for a fixed final accuracy. As conclusion, the number C of messages sent per node in dmGS is $C = \mathcal{O}(m^2 \log(N))$ and that the total number C_t of messages sent is $C_t = \mathcal{O}(Nm^2 \log(N))$. The total number of words sent during dmGS equals the total number of messages C_t multiplied by their size, which in dmGS is constant (two words). Thus the number of words sent is asymptotically the same as the number of messages sent. For the special case of a square $n \times n$ matrix distributed over $n = N$ nodes, we get $C = \mathcal{O}(N^2 \log(N))$ and $C_t = \mathcal{O}(N^3 \log(N))$.

Comparison with State-of-the-Art Parallel Algorithms. According to [6], parallel mGS sends $2m \log(P)$ messages and $m^2 \log(P)/2$ words (leading terms) when factorizing a rectangular $n \times m$ matrix over P processors. Consequently, the average message size $m/4$ increases linearly with increasing m. Parallel CAQR sends $1/4 \cdot \sqrt{mP/n} \cdot \log^2(nP/m) \cdot \log(P\sqrt{nP/m})$ messages and $\sqrt{nm^3/P} \cdot \log P - 1/4 \cdot \sqrt{m^5/nP} \cdot \log(mP/n)$ words along the critical path [6]. For the special case of a square $n \times n$ matrix, CAQR needs $3\sqrt{P}\log^3(P)/8$ messages and $3n^2 \log(P)/(4\sqrt{P})$ words. Consequently, the average message size in parallel CAQR for a square matrix is $2n^2/(P \log^2(P))$.

For the special case $n = m = N = P$, which is relevant in the application problems mentioned in Section 1, we find that dmGS asymptotically sends the same number of *words* along the critical path as parallel mGS and a factor $\sqrt{N}$ more than parallel CAQR. When comparing the total number of *messages* sent along the critical path, there is a bigger difference. This is due to the fact that in dmGS the message size is constant, whereas in the parallel algorithms it grows with N, which reduces the number of messages sent. Asymptotically, parallel mGS sends a factor N fewer messages along the critical path than dmGS. The communication-optimal CAQR algorithm sends even a factor $N\sqrt{N}/\log^2(N)$ fewer messages than dmGS. This shows that dmGS would not be competitive in terms of communication cost when executed on standard target systems for parallel algorithms, because the communication overhead to be paid for reliable execution becomes quite large for large N.

5 Conclusions

We introduced dmGS, a gossip-based algorithm for distributed computation of the QR factorization of a general rectangular matrix distributed over a loosely coupled decentralized distributed system with unreliable communication. Problems of this type arise, for example, in distributed sphere decoding in telecommunications or in decentralized network analysis.

We evaluated dmGS in terms of the factorization error and orthogonality of the computed matrix Q. In terms of numerical accuracy, it preserves known properties of existing sequential and parallel QR factorization algorithms. However, dmGS is designed for completely different target platforms and its decentralized structure makes it more flexible and robust. In particular, we illustrated that although unreliable communication introduces overhead and an increase of the network diameter slows down convergence, dmGS still produces correct results.

In order to complete the picture, a quantitative comparison of dmGS with the state-of-the-art parallel algorithms in terms of operation count, local memory requirements and communication cost has been given. It shows that there is a communication overhead to be paid in dmGS over existing parallel algorithms for robust execution on dynamic and decentralized target platforms with unreliable communication. However, we currently work on an algorithmic improvement of dmGS which reduces the communication overhead. Moreover, dmGS has the additional benefit that its computational and communication cost are proportional to the target accuracy, i. e., it is possible to compute an approximative QR factorization at proportionally reduced cost.

Ongoing and Future Work. More detailed analyses and simulations of the influence of the network topology on the behavior of dmGS will be performed. We will also investigate energy efficiency aspects and the influence of dynamic networks (for example, mobile nodes) on the behavior of the algorithm. The investigation of the utilization of broadcast gossiping methods [2] is another important next task. We are also developing code suitable for runtime performance model with ScaLAPACK and PLASMA routines. Beyond that, other ideas for further reducing the communication cost in distributed QR factorization algorithms will be investigated.

Acknowledgments. This work has been supported by the Austrian Science Fund (FWF) under contracts S10608 and S10607 (NFN SISE). The Telecommunications Research Center Vienna (FTW) is supported by the Austrian Government and the City of Vienna within the competence center program COMET.

References

1. Abdelhak, S., Chaudhuri, R.S., Gurram, C.S., Ghosh, S., Bayoumi, M.: Energy-aware distributed QR decomposition on wireless sensor nodes. The Computer Journal 54(3), 373–391 (2011)
2. Aysal, T., Yildiz, M., Sarwate, A., Scaglione, A.: Broadcast gossip algorithms for consensus. IEEE Trans. Signal Processing 57(7), 2748 –2761 (2009)
3. Blackford, L., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., Whaley, R.C.: ScaLAPACK Users' Guide. SIAM, Philadelphia, PA, USA (1997)
4. Boyd, S., Ghosh, A., Prabhakar, B., Shah, D.: Randomized gossip algorithms. IEEE Trans. Information Theory 52(6), 2508 – 2530 (2006)
5. Buttari, A., Langou, J., Kurzak, J., Dongarra, J.: Parallel tiled QR factorization for multicore architectures. In: Proceedings of the 7th International Conference on Parallel Processing and Applied Mathematics. pp. 639–648. Springer-Verlag (2008)
6. Demmel, J., Grigori, L., Hoemmen, M.F., Langou, J.: Communication-optimal parallel and sequential QR and LU factorizations. Tech. rep., no. UCB/EECS-2008-89, EECS Department, University of California, Berkeley (2008)
7. Dimakis, A., Kar, S., Moura, J., Rabbat, M., Scaglione, A.: Gossip algorithms for distributed signal processing. Proceedings of the IEEE 98(11), 1847 –1864 (2010)
8. Dumard, C., Riegler, E.: Distributed sphere decoding. In: International Conference on Telecommunications ICT '09. pp. 172–177 (2009)
9. Gansterer, W.N., Niederbrucker, G., Strakova, H., Schulze Grotthoff, S.: Scalable and fault tolerant orthogonalization based on randomized aggregation. to appear in Journal of Computational Science
10. Golub, G.H., Van Loan, C.F.: Matrix Computations. The Johns Hopkins University Press, 3rd edn. (1996)
11. Kempe, D., Dobra, A., Gehrke, J.: Gossip-based computation of aggregate information. In: FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science. pp. 482–491. IEEE Computer Society (2003)
12. Kempe, D., McSherry, F.: A decentralized algorithm for spectral analysis. Journal of Computer and System Sciences 74(1), 70 – 83 (2008)
13. Kielbasinski, A., Schwetlick, H.: Numeryczna algebra liniowa. (In Polish). Second edition. Wydawnictwo Naukowo-Techniczne, Warszawa (1994)
14. Ozgur, A., Leveque, O., Tse, D.: Hierarchical cooperation achieves optimal capacity scaling in ad hoc networks. IEEE Transactions on information theory 53(10), 3549–3572 (2007)
15. Rabbat, M., Nowak, R.: Distributed optimization in sensor networks. In: Third International Symposium on Information Processing in Sensor Networks. pp. 20 – 27 (2004)
16. Song, F., Ltaief, H., Hadri, B., Dongarra, J.: Scalable tile communication-avoiding QR factorization on multicore cluster systems. In: International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 1–11 (2010)
17. Yu, Y., Krishnamachari, B., Prasanna, V.: Energy-latency tradeoffs for data gathering in wireless sensor networks. In: INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies. vol. 1 (2004)