Journal of Parallel and Distributed Computing 136 (2020) 52–62

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Fault-tolerant least squares solvers for wireless sensor networks based on gossiping

Karl E. Prikopa, Wilfried N. Gansterer [*]

University of Vienna, Faculty of Computer Science, Vienna, Austria

ARTICLE INFO

Article history:
Received 28 October 2017
Received in revised form 18 March 2019
Accepted 10 September 2019
Available online 11 October 2019

Keywords:
Distributed least squares solver
Algorithmic fault tolerance
Message loss
Mixed precision iterative refinement
Gossip-based aggregation

ABSTRACT

Many applications in large loosely connected distributed networks (such as wireless sensor networks) require the distributed solution of linear least squares (dLLS) problems. Ideally, a *truly distributed* algorithm should require very little coordination between the nodes. This favours algorithms which do not require a fusion centre, cluster heads or any multi-hop communication.

We present the novel dLLS solver GLS-IR for overdetermined linear systems. We investigate two variants of our novel solver, one of them based on the semi-normal equations, the other based on the normal equations. Both are combined with iterative refinement in mixed precision, which not only stabilises the methods but also decreases the communication cost. In GLS-IR, all communication between nodes is contained within a gossip-based algorithm for distributed aggregation, which limits the communication of each node to its immediate neighbourhood. Therefore, GLS-IR benefits directly from efficient and fault-tolerant algorithms for distributed aggregation. We use a fault-tolerant alternative to the push-sum method, the push-flow algorithm, which is able to recover from silent message loss and temporary or permanent link failures.

We analytically compare the communication cost of GLS-IR to existing truly distributed algorithms. Since the theoretical analysis contains problem-dependent parameters, numerical experiments are needed in order to get a complete picture. Our simulation experiments illustrate a significantly reduced communication cost of GLS-IR compared to other existing truly distributed least squares solvers. We also illustrate that due to the properties of iterative refinement and push-flow, GLS-IR can achieve a result accurate to machine precision even if a high amount of message loss occurs.

1. Introduction

Several applications require the distributed solution of a linear least squares (LLS) problem in loosely connected, decentralised sensor networks, e. g. target tracking [21], the reconstruction of physical fields [19], localisation [18] or monitoring volcanic activity and solving the seismic tomography inversion problem [24]. In a fully decentralised environment, the sensors themselves have to be able to make decisions and can be combined with actuators to interact autonomously with the physical world. Wireless Sensor Networks (WSNs) typically consist of a large number of inexpensive sensor nodes which act autonomously but cooperate with each other to achieve a common goal. The resources on typical sensor nodes are normally very restricted, especially their power supply and computational capabilities. Communication is one of the main sources of high power consumption. The energy required for communication is directly proportional to the communication range [25]. By restricting the communication to the immediate neighbourhood of a node, the power requirements can be reduced significantly, which is not only beneficial to the lifespan of the nodes, but also to the entire network.

Most distributed linear least squares (dLLS) solvers found in the literature require centralised fusion centres, cluster heads or multi-hop communication, all of which cannot be considered *truly distributed* (see Section 2.1). Multi-hop communication requires routing tables, and setting those up requires additional communication. The overhead is particularly large if the routing tables have to be updated frequently to handle mobile nodes or nodes joining or leaving the network.

1.1. Problem statement

In this paper, we propose a *truly distributed* approach for solving the dLLS problem

$$\min_x \|b - Ax\|_2$$

for $x \in \mathbb{R}^m$, where $A \in \mathbb{R}^{n\times m}$ with $n \geq m$ and $b \in \mathbb{R}^n$. The input data A and b is scattered over all participating nodes. In particular,

[*] Corresponding author.
E-mail addresses: karl.prikopa@univie.ac.at (K.E. Prikopa), wilfried.gansterer@univie.ac.at (W.N. Gansterer).

https://doi.org/10.1016/j.jpdc.2019.09.006
0743-7315/© 2019 Elsevier Inc. All rights reserved.

we focus on situations where A is distributed row-wise over the N nodes of the network and the element $b(i)$ resides on the same node as the ith row of A. For $n > N$, each node contains a block of consecutive rows of A. This distribution also corresponds to the before mentioned applications and many other big data problems, which naturally exhibit this data structure by having significantly more data points (n) than descriptors (m).

1.2. Approach

We introduce two variants of our novel distributed solver GLS-IR (gossip-based least squares solver) for solving the dLLS problem, which is either based on the method of semi-normal equations or on the method of normal equations. One of the innovations of GLS-IR is the adaptation of mixed precision iterative refinement to a truly distributed setup. In our algorithm, the communication is limited to the immediate neighbourhood and no fusion centre or multi-hop communication is required. By design, all internode communication in GLS-IR is contained in gossip-based reduction operations across the participating nodes and the solution $x \in \mathbb{R}^m$ is replicated across the nodes.

Gossip algorithms, also known as epidemic algorithms, spread their information by only communicating with the immediate neighbourhood and in each step the nodes randomly choose their communication partners. A prime example of a gossip aggregation algorithm is the push-sum algorithm [9], which can calculate the sum or the average of values distributed over a set of nodes. At each point in time, each node has an estimate of the target solution which will converge to the correct result. Rumour spreading [15] is another application of the gossip protocol where a node distributes information to all the other nodes in the network. It is also based on randomised communication and many variations exist which differ on the approach of distributing the information, either by push or pull operations. Both algorithms, push-sum and rumour spreading, are employed by our novel solver GLS-IR.

A very important factor in the design of a distributed algorithm for WSNs is distributed fault tolerance. Gossip-based algorithms already exist to handle some types of faults at the aggregation level. The push-flow algorithm [6] is a fault-tolerant alternative to the push-sum method and is able to recover from silent message loss and temporary or permanent link failures. Furthermore, the use of IR itself provides resilience against faults in the initial solution (see Section 4.4). We will demonstrate experimentally the fault tolerance of GLS-IR based on push-flow against message loss.

1.3. Paper outline

Section 2 reviews the related work and discusses the differences between centralised, clustered and truly distributed approaches, providing further insight into the advantages of a truly distributed approach. In Section 3 the mathematical basis for our novel dLLS solver is described, followed by the introduction of the GLS-IR algorithm and its components in Section 4. Section 5 provides an analysis of the communication cost and a comparison with existing dLLS solvers. In Section 6 we present numerical experiments simulated in MPI to compare the performance of GLS-IR and existing dLLS solvers in terms of number of messages. Special experiments investigate the fault tolerance properties of our algorithms and also analyse the benefits of using lower working precisions for the majority of the gossip-based reduction operations. Finally, Section 7 concludes our paper.

2. Related work

Most distributed least squares solvers found in the literature require centralised fusion centres, cluster heads or multi-hop communication. All these approaches cannot be considered truly distributed, as we will discuss in Section 2.1. In Section 2.2, we will limit our summary of the related work to *truly distributed* LLS solvers. For a more extensive discussion of dLLS solvers found in the literature, including centralised and clustered approaches, please refer to [17].

2.1. Centralised vs. decentralised approaches

An extensively studied strategy for distributed computations is the **fusion centre approach**, where a central unit performs the computation for the entire network. The fusion centre approach first collects the data from all nodes in the network, then solves a problem at the fusion centre and finally distributes the result to all nodes. Both steps, collecting and distributing the data, require global communication for each node to reach the fusion centre. Multi-hop communication and setting up routing tables incur additional overhead. Challenges also arise with the positioning of the fusion centre which directly affects the communication cost and the scalability of the method (cf. [24]). Potential congestion effects (particularly around the fusion centre [10]) can lead to delays and in the worst case to data loss. Last, but not least, the fusion centre is a single point of failure.

A first step towards a more decentralised setting than the fusion centre approach is based on **clustering**. The network is divided into clusters. In each cluster, one node acts as the cluster head, which often is a more powerful node than the other nodes in the cluster to handle the higher volume of messages received. Many techniques exist to form clusters, e. g. using the geographical location or setting a communication radius for the cluster head. The cluster heads act as intermediate fusion centres for the clusters. The nodes of a cluster only communicate with their cluster head and with nodes within the same cluster. Compared to the fusion centre approaches, a multi-tier model is used where only the cluster heads communicate with the fusion centre, reducing the communication cost and also the risk of congestion. Although clustering reduces the risk of a single point of failure affecting the entire network, it does not eliminate that risk completely. If a cluster head fails, the complete area covered by the cluster and its data are lost until a new cluster head takes over.

The most decentralised approach, the **truly distributed approach**, is to limit the communication of the nodes to their immediate neighbourhood (defined by their communication range). Each communication partner has to be reachable in a single hop as multi-hop communication would incur additional overhead through routing and thus increase the energy consumption of the resource restricted nodes. A truly distributed approach does not have the limitations of scalability seen in the fusion centre approach. There is no need for more powerful nodes to handle massive amounts of messages and the risk of congestion is limited by the low number of communication partners. A single node failure will not cause the failure of a part or even the entire network. Naturally, a node failure in any scenario will affect the computational results, but methods can be put into place to mitigate or eliminate these effects, as presented in this paper with the use of fault-tolerant gossip algorithms and iterative refinement to recover from faults.

2.2. Truly distributed LLS solvers

Zhou et al. [31] propose a distributed least squares solver which they claim is robust against reported node failures. The algorithm is designed for $m = 1$ and higher dimensions are not considered in [31]. The distributed iterative algorithm exchanges the values of A and b with the neighbours and updates them using a Metropolis weight based on the degree of the node's neighbours, which are determined before the iterative algorithm initialises. In the event of a node failure, convergence is still guaranteed, but the result will no longer be correct. Therefore, the authors extend their algorithm, trying to reduce the magnitude of the occurring error. A disadvantage is that node failures have to be detectable. Once detected, the weights used in the computation have to be updated throughout the network, which poses a global updating problem requiring communication across the entire network. In the event of a node failure, the magnitude of the error depends on the network topology. Although the algorithm presented in [31] is truly distributed, we do not consider it in our analysis and in our simulations because it is restricted to the special case $m = 1$.

Sayed et al. [5,21,29] propose a diffusion-based least mean square estimator (diffLMS) using steepest-descent iterations for solving the normal equations. Diffusion strategies are seen as an alternative to consensus strategies for distributed optimisation problems, both limiting the communication to the neighbourhood. The data A and b are both distributed row-wise. In each iteration, diffLMS consists of two main steps, an adaption step and a combination step, and delivers an estimate of the solution x on each node. The authors provide two variants of their algorithm, adapt-then-combine (ATC) and combine-then-adapt (CTA), which differ in the order of these computation steps and mathematically and numerically do not result in the same solution. diffLMS can also exchange the local parts of the observations b and the local matrix rows from A with the neighbouring nodes to improve the estimate of the solution. Further research into diffLMS has been conducted in [1], where the authors investigate the tradeoff between the amount of communication and the accuracy of the result. They reduced the amount of communication by only transmitting the estimates to a subset of the neighbours in each iteration while reportedly achieving only slightly less accuracy. Different selection processes were presented to keep the additional overhead of tracking which estimates are being received to a minimum.

Another fully distributed approach only using neighbourhood communication is the distributed least mean squares method (D-LMS) by Schizas, Mateos and Giannakis [12,22,23]. D-LMS is based on Lagrange multipliers and uses the least squares residual and the difference between the estimates of x from the neighbourhood in a correction step to compute the least squares solution iteratively. The data distribution of A and b is again row-wise. At each step an estimate for the solution x is available on each node. The convergence of D-LMS also depends on a problem dependent step-size parameter μ. D-LMS communicates twice in each iteration, once to broadcast the current estimate to all neighbours and a second time to send individual correction vectors to each neighbour (single-hop unicast).

Linear least squares problems are convex optimisation problems. Algorithmic ideas which are very similar to D-LMS and diffLMS also appear in the distributed optimisation literature [13,27,28]. In [13], the authors provide a general framework how to solve convex optimisation problems in a distributed environment. The goal of the research is to cooperatively optimise a global objective function while the local objective functions are only known to the nodes themselves. The research builds on the work by Tsitsiklis et al. [28], who developed a framework for the analysis of asynchronous distributed iterative optimisation algorithms. Tsitsiklis et al. considered algorithms that are gradient-like and each update minimises a cost function in a descent direction. Nedic and Ozdaglar [13] combine first-order methods, in this case the subgradient method, with the consensus algorithm to achieve distributed optimisation methods. The local objective function is minimised using the subgradient method, while the consensus step aligns its decision with the decisions of its neighbours, leading to a decentralised solver.

In [17], we presented the push-sum distributed least squares solver (PSDLS), a truly distributed LLS solver using the gossip algorithm push-sum [9] as its aggregation function, limiting its communication to only the neighbouring nodes. Our solver is based on the distributed modified Gram–Schmidt method to compute the QR factorisation of A, followed by a distributed matrix–vector multiplication and a local back substitution which only requires the locally available data and no further communication. The algorithm has the same row-wise distribution of the input data A and b used by the previously described methods. An approximation of the solution x is available on each node. Our numerical simulations in [17] have shown, that PSDLS requires significantly fewer messages per node than the previously existing methods to reach a predefined solution accuracy. The simulation results have also shown that the accuracy achieved by diffLMS is usually very low. In this paper, we will provide further experimental results comparing our novel distributed gossip-based linear least squares solver GLS-IR to PSDLS and D-LMS.

3. Iterative refinement for least squares problems

Mathematically, our approach for solving the dLLS problem is either based on semi-normal equations (SNE) or normal equations (NE), both in combination with iterative refinement (IR).

The method of normal equations (NE) solves the LLS problem by forming and solving the normal equations:

$$A^\top Ax = A^\top b\ .$$

Assuming A has full rank, the LLS problem has a unique solution. The matrix $C := A^\top A \in \mathbb{R}^{m\times m}$ is symmetric and positive definite. Therefore, the NE can be solved using the Cholesky factorisation $C = LL^\top$. The main drawback of the approach based on the NE is that it is only guaranteed to be backward stable if A is well-conditioned [8], i. e. if $\kappa(A) \leq \varepsilon^{-1/2}$ with ε being the machine epsilon.

The method of semi-normal equations (SNE) is derived from the normal equations using a QR factorisation of A. If $A = QR$, then the solution can be computed by solving $R^\top Rx = A^\top b$, which does not need the factor Q from the factorisation due to $Q^\top Q = I$. Having the original matrix A, it is still possible to solve multiple right-hand sides b with SNE without the need for Q. The stability of the SNE method for the LLS problem has been analysed extensively in [3]. It has been shown that SNE are not backward stable and that the error in x is similar to the error arising when using the NE. The dominating error arises from the rounding errors in the computation of the right hand-side $A^\top b$. However, it has been shown in [2,7] that an iterative refinement correction step leads to much more satisfactory results and under certain conditions makes the method of SNE backward stable as long as A does not have widely differing row norms. Furthermore, the SNE are applicable to worse conditioned matrices that cannot be solved by the NE.

Iterative refinement (IR) [30] is a strategy for improving the accuracy of a computed solution of a linear system by trying to reduce round-off errors. The method iteratively computes a correction term to an approximate solution by solving a system using the residual of the approximate solution. The cost of IR is very low compared to the cost of the factorisation of the matrix, but results in a solution which can be accurate to machine precision. A wide range of variations of IR exist which mainly differ in the precisions used for computing the different steps in the process. Standard IR performs all computations in the same floating-point precision. Mixed precision iterative refinement (MPIR) [11] is a special performance-oriented case of IR which has been studied for solving linear systems of equations. The majority of operations, the matrix factorisation and solving the linear systems, is computed in single precision (SP) and only the critical parts, computing the residual and updating the solution, are performed in double precision (DP), operations of low complexity compared to the factorisation. MPIR using SP and DP achieves the same and often higher accuracy than a DP direct solver, as long as the system is not too ill-conditioned. The number of iterations required for convergence directly relates to the condition number of A [20].

Using a lower working precision has many benefits. Processors can perform more lower precision operations per cycle and the data uses less storage, reducing the number of cache misses. In the context of distributed algorithms, as considered in this paper, the communication cost is reduced by using mixed precision IR instead of standard IR. The gossip algorithms require fewer rounds and therefore fewer messages to converge to the lower working precision. The amount of data sent per message is also reduced.

For LLS problems, the IR method [7] is defined by

$$\min_{\Delta x} \|r - A\Delta x\|_2$$

for a given A and the residual vector of the LLS problem $r := b - A\hat{x}$, which satisfies $A^\top r = 0$ [4], with $\hat{x}$ being the initial approximate solution. In [2], the rate of convergence is shown to be roughly linearly dependent on the condition number $\kappa(A)$.

To the best of our knowledge, the application of *mixed precision* IR has only been considered in the context of *parallel* algorithms [16], but not in the context of *truly distributed algorithms*, which are the focus of this paper. However, formulating the LLS problem as an augmented linear system [2] makes all IR methods developed for linear systems applicable to LLS problems. Therefore the same proofs of convergence and numerical improvement apply to MPIR for dLLS problems.

For more details on the mathematical background and some numerical properties of a parallel algorithm based on the methods described in this section, please refer to [16].

4. A truly distributed LLS solver

In this section, we discuss a *truly distributed* variant of the LLS solver based on SNE or NE and IR (as summarised in Section 3). All communication of the resulting algorithm is contained in reduction operations across the participating nodes. These reduction operations are realised using gossip algorithms, and we therefore call our algorithm *gossip-based distributed least squares solver with iterative refinement* (*GLS-IR*). Through the use of fault tolerant reduction methods, such as the push-flow algorithm [6], GLS-IR becomes resilient against silent communication failures. GLS-IR can directly benefit from any future improvements to the algorithms of the reduction operations, either in performance (e. g. by reducing the number of messages) or in fault tolerance.

GLS-IR consists of three main components: (*i*) a distributed QR factorisation of A in the case of GLS-IR-SNE or distributed construction of the normal equations in the case of GLS-IR-NE, (*ii*) a distributed matrix–vector multiplication followed by two local triangular solves to compute an initial solution to the dLLS problem, and (*iii*) IR to stabilise and improve the initial solution computed in the previous step, requiring a distributed matrix–vector multiplication and rumour spreading. Each of these components will be discussed in the following.

4.1. Improving the efficiency of distributed QR factorisation

The first component required for the SNE is a distributed QR factorisation of the matrix A. We use a variation of the distributed modified Gram–Schmidt orthogonalisation (dmGS) from [26] which used push-sum for the reduction operations. dmGS only differs from sequential mGS in the distributed computation of two sums, one for the 2-norm of a vector and one for a dot product. No additional communication is necessary and the rest of the computations are performed locally. dmGS assumes that the matrix A is distributed row-wise across the computing nodes. Therefore, the part of A available locally at node u will be denoted by $A^{(u)}$. dmGS returns the factor Q distributed row-wise, the same distribution as A, and an approximation R_u of the upper-triangular matrix R, which is fully available on every node ($R_u \in \mathbb{R}^{m\times m}$).

We had to make some adjustments for the use with the SNE and could also reduce the communication cost by combining the communication steps in each iteration. For the SNE, the factor Q is not required to solve the LLS problem. The Q-less dmGS approach (denoted as dmGSR in the following) therefore only returns the full factor R of the QR factorisation and discards the computed columns of Q. This reduces the memory requirements of dmGS by $n(m-1)$ floating-point numbers as only one vector of length n instead of a full matrix Q is needed for the computation of R. Both methods, dmGS and dmGSR, can further be improved in terms of communication by postponing the scaling of the column of A by the diagonal element of R *after* the computation of the second distributed summation. The first summation of a scalar can then be combined with the second distributed reduction operation by appending a single value to the vector. This reduces the communication cost from $2m-1$ to m messages and eliminates the overhead caused by the communication of a scalar value.

4.2. Distributed LLS solver with IR

One of the standard methods for solving the LLS problem is the use of the QR factorisation to solve the linear system $Rx = Q^\top b$. The push-sum based PSDLS algorithm which we introduced in [17] computes the QR factorisation of A using dmGS, followed by a distributed matrix–vector multiplication dmv of $Q^\top$ and b. dmv first computes the product of the locally available factors $Q^{(u)\top}$ and $b^{(u)}$ and then forms the sum of the local results using a distributed reduction operation. The final step in PSDLS is the local back substitution using the full locally available factor R_u and the result z_u of dmv. Each node u then holds an approximation x_u of the solution x.

The steps for GLS-IR-SNE are shown in Algorithm 1 on lines 2–4. The Q-less mGS method, dmGSR, requires the same amount of computation and communication as dmGS, but only returns the factor R_u. The full matrix Q is never stored and the computed columns of Q are discarded during the computation, which reduces the memory requirements compared to the dmGS algorithm by $n(m-1)$ scalars. dmv is used to compute $A^\top b$, with the local parts $A^{(u)\top}$ and $b^{(u)}$ having the same row-wise distribution as $Q^{(u)\top}$ in PSDLS. Finally, the system is solved by a forward and back substitution using R_u, both operations being performed locally.

For the NE, the algorithm is very similar, the only changes being the factorisation in line 2 and using the Cholesky factor L instead of the QR factor R when solving the systems in lines 4 and 12. dmGSR is replaced with a distributed sum, dsum, to form the normal equations, followed by a local Cholesky factorisation of the distributed result. dmGSR requires m reduction operations (dsum), whereas the NE variant requires only a single reduction operation in line 2. Therefore, GLS-IR-NE requires less communication than PSDLS and GLS-IR-SNE, the number of reduction operations being independent of m.

Table 1
Fault tolerance of IR: example of message loss during the initial factorisation.

Initial solution		IR
$p_\beta = 10^{-8}$, $p_\alpha = 10^{-15}$	$\rightarrow$	IR reaches p_α in 2 iterations
$p_\beta = 10^{-4}$, $p_\alpha = 10^{-15}$	$\rightarrow$	IR reaches p_α in 3–4 iterations
$p_\beta = 10^{-8}$, $p_\alpha = 10^{-15} \xrightarrow{\text{Message loss}} p_\beta = 10^{-4}$, $p_\alpha = 10^{-15}$	$\rightarrow$	IR reaches p_α in 3–4 iterations

After computing an initial solution x_u, it is improved using IR. The residual is computed locally (line 7 in Algorithm 1) and then $A^{(u)\top}$ is applied using the distributed method dmv. Subsequently, a correction term Δx_u is computed in line 12 by solving the system using the already computed factor R_u or L_u. Finally, the solution is updated by the correction term in line 13. The process continues until a convergence criterion is met.

4.3. Reducing communication cost with mixed precision IR

GLS-IR uses two different precisions throughout the process, a higher target precision p_α (e.g. 10^{-15} for DP accuracy) and a lower working precision p_β (e.g. 10^{-8} for SP accuracy), leading to a *mixed precision* approach. The choice of the precisions directly affects the number of messages required by the gossip-based reduction operations to reach the requested accuracy. Most gossip-based reductions are performed during the factorisation of A in line 2 and therefore the majority of the communication benefits from the lower working precision p_β. For well-conditioned problems, IR requires very few iterations to converge (on average 2–3 iterations suffice, depending on p_β). Each IR step only requires one gossip-based reduction in line 8 which has to be accurate to the higher target precision p_α. In GLS-IR, the initial solution and the correction term can be computed completely in the lower working precision p_β, whereas computing the residual, applying $A^{(u)\top}$ to r_u and updating the solution requires the higher target precision p_α. Performing the majority of the computations and communication in the lower precision p_β leads to fewer messages during the reduction operations and improved performance for the local computations.

Gossip-based reduction algorithms produce different approximations of the aggregation result at each node. In order for iterative refinement to work in a distributed setting, it is important that all nodes use the *same* approximation for x_u, at least to the accuracy p_α targeted for the solution, when computing the residual in the first step of each IR iteration (line 7 in Algorithm 1). Otherwise the computation of the correction term Δx will fail. This can either be achieved by computing an average of the approximate solution vectors x_u over all nodes, accurate to the targeted accuracy p_α, or by selecting one node to distribute its value of x_u to all other nodes in the network. In GLS-IR, a rumour spreading method is employed to achieve this condition (line 6 in Algorithm 1).

4.4. Improving fault tolerance with IR

IR has many responsibilities within the algorithm: it stabilises the solution of the semi-normal equations, improves an initial solution computed in a lower working precision and reduces the communication cost through the use of mixed precision. Aside from these tasks, IR is also one of the strategies employed in the GLS-IR algorithms to provide fault tolerance IR is a naturally self-healing algorithm. In each iteration, a correction term Δx_u is computed, allowing for an improvement of a solution by at most $\log_{10} p_\beta$ digits. Before applying IR, the solution can either be of low quality due a low precision initial factorisation, as is the case in the mixed precision approach discussed and applied in this paper, or due to a fault, e.g. a silent message loss or silent data corruption (i.e. bit-flips in the memory). Either way, the recovery process from such faults is identical to the situation where the initial solution is already being computed in a lower precision corresponding to the accuracy of the result after a fault.

Table 2
Analytical comparison of the communication cost per node. k and l denote the number of iterations required by IR and D-LMS, respectively. $|D_u|$ denotes the number of neighbours of node u and $A \in \mathbb{R}^{n\times m}$.

dLLS method	Number of messages sent per node	Total amount of data sent per node
D-LMS [23]	$(B(\|D_u\|) + \|D_u\|)\, l$	$(B(\|D_u\|) + \|D_u\|)\, ml$
PSDLS [17]	$(m+1)R_\alpha$	$(\frac{m(m+1)-2}{2} + m)R_\alpha$
GLS-IR-SNE	$(m+1)R_\beta + R_{RS} + kR_\alpha$	$(\frac{m(m+1)-2}{2} + m)R_\beta + 2mkR_\alpha$
GLS-IR-NE	$2R_\beta + R_{RS} + kR_\alpha$	$(\frac{m(m+1)}{2} + m)R_\beta + 2mkR_\alpha$

For example, consider the case of $p_\beta = 10^{-8}$ and $p_\alpha = 10^{-15}$. Let us assume that silent message loss prevents push-sum to achieve the requested precision of p_β and it only reaches 10^{-4}. This corresponds to the fault-free case of $p_\beta = 10^{-4}$. The message loss only affects the number of IR iterations required to reach the target precision p_α, which slightly increases with the reduction of the working precision. The number of iterations is the same as if the initial problem had chosen $p_\beta = 10^{-4}$ as its input parameter. Therefore, IR can still improve the initial solution starting from 10^{-4} as if p_β was set to this precision initially. The steps of this example are shown in Table 1.

In Section 6.2 we demonstrate the healing capabilities of IR if faults occurred during the initial factorisation of A and in Section 6.3 we investigate the effects of different working precisions on the communication cost and the number of IR iterations required to reach the target precision $p_\alpha = 10^{-15}$.

5. Communication cost analysis

In this section, we analyse the communication cost for GLS-IR and compare the cost to existing distributed least squares solvers (see Table 2). For reasons of simplicity, we assume that the number of required messages is independent of the size of the data being transmitted.

PSDLS and GLS-IR-SNE have to compute a QR factorisation. dmGS or dmGSR both require $2m-1$ sum reduction operations and send the same amount of data: $\frac{m(m+1)-2}{2} + 2m = O(m^2)$ scalars per node. In the distributed mGS methods, by postponing the scaling of the column of A by the diagonal element of R and combining the two reduction operations (as discussed in Section 4.1), the number of reduction operations can be further reduced to m. Solving the LLS problem requires one additional reduction operation for the matrix–vector product, resulting in a total of $m+1$ reduction operations for PSDLS and for computing the initial solution in GLS-IR-SNE. GLS-IR-NE only requires a single reduction operation to form the normal equations and one reduction operation for the matrix–vector product to solve the LLS problem. Each reduction operation is performed using a gossip-based aggregation algorithm, which communicates randomly and requires R rounds to reach a requested accuracy. We denote R_α and R_β as the number of rounds required to reach precisions p_α and p_β, respectively. Note that in practice R may vary slightly

Algorithm 1: GLS-IR-SNE and GLS-IR-NE.

```
Input: $A \in \mathbb{R}^{n \times m}$ with $n > m$, $b \in \mathbb{R}^n$, both distributed row-wise over $N$ nodes
Output: $x \in \mathbb{R}^m$ on every node
 1: in each node $u$ do
 2:     $R_u \leftarrow$ dmGSR($A^{(u)}$) (SNE) or                              ▷ distributed, $p_\beta$
        $L_u \leftarrow$ cholesky(dsum($A^{(u)\top} A^{(u)}$)) (NE)    ▷ distributed sum, local Cholesky, $p_\beta$
 3:     $z_u \leftarrow$ dmv($A^{(u)\top}$, $b^{(u)}$)                            ▷ distributed, $p_\beta$
 4:     $x_u \leftarrow$ solve $R_u^\top R_u x_u = z_u$ (SNE) or $L_u L_u^\top x_u = z_u$ (NE)   ▷ local, $p_\beta$
 5:     for $i = 1$ : maxiter do
 6:         $x_u \leftarrow$ rumour_spreading($x_u$)                      ▷ distributed, $p_\alpha$
 7:         $r_u \leftarrow b^{(u)} - A^{(u)} x_u$                                ▷ local, $p_\alpha$
 8:         $s_u \leftarrow$ dmv($A^{(u)\top}$, $r_u$)                          ▷ distributed, $p_\alpha$
 9:         if $\|s_u\| <$ tolerance then
10:             break $\rightarrow$ converged
11:         end if
12:         $\Delta x_u \leftarrow$ solve $R_u^\top R_u \Delta x_u = s_u$ (SNE) or $L_u L_u^\top \Delta x_u = s_u$ (NE)   ▷ local, $p_\beta$
13:         $x_u \leftarrow x_u + \Delta x_u$                                     ▷ local, $p_\alpha$
14:     end for
```

for different push-sum calls even for reaching the same precision due to the randomised communication schedule. In each push-sum or push-flow call, a vector of values and a weight have to be transmitted [9].

Iterative refinement slightly increases the communication cost due to the additional sum reduction operation for each iteration in line 8 of Algorithm 1 and due to the rumour spreading in line 6. However, due to the use of the lower precision p_β for the communication dominant QR factorisation, the total communication cost decreases compared to computing all steps in the higher precision p_α. For the rumour spreading of x, we denote the number of rounds as R_{RS}. Compared to the dmGS QR factorisation, the communication cost of IR is negligible, since the number of iterations k is normally very small. Usually, 2–4 iterations suffice, even in the case of faults affecting the result, as we will show in Section 6.2. Each reduction operation sums the elements of a vector of length m. Using a lower working precision p_β, GLS-IR requires fewer rounds to reach p_β and GLS-IR-SNE also sends less data for the bulk of the communication performed in the QR factorisation. The effects of different choices for p_β in relation to p_α will be illustrated in Section 6.

The D-LMS method [23] communicates twice in each iteration. First, a local broadcast to the neighbourhood D_u of a node u is required, distributing the vector x_u of size m to the neighbours. Then D-LMS sends $|D_u|$ individual messages of size m to distribute a different correction term to each node in the neighbourhood (one-hop unicast). This results in $(B(|D_u|) + |D_u|)l$ messages and $(B(|D_u|)+|D_u|)ml$ data values sent per node, where l is the number of iterations required for D-LMS to converge and $B(d)$ denotes the number of messages required for broadcasting to d neighbours. In a WSN, $B(d) = 1$. As already mentioned in Section 2.2, we omit diffLMS [21] from this comparison due to its low accuracy demonstrated in [17].

Comparing PSDLS and the GLS-IR variants analytically, GLS-IR-NE has the lowest communication cost due to the single reduction operation to form the NE instead of m operations for the QR factorisation. Considering that the number of rounds required for gossip-based reduction grows linearly with the logarithm of the accuracy [9], for $p_\beta = 10^{-8}$ only about half the number of messages are required compared to $p_\beta = 10^{-15}$. In this case, as long as $m \geq 5$, the lower working precision leads to a lower communication cost for GLS-IR-SNE than for PSDLS because the number of IR iterations k to reach p_α is very low (usually about 2–4). Moreover, the mixed precision approach can also benefit from transmitting smaller floating-point representations for the majority of the communication, leading to a lower communication cost even for (some) $m < 5$.

For a direct comparison of the concrete communication cost, information about the number of iterations l required by D-LMS and the number of rounds R required by the distributed reduction operations in PSDLS and GLS-IR is necessary, where R further depends on the precisions p_α and p_β. For this purpose, we also conducted experimental comparisons. Their summary in Section 6 illustrates that these quantities differ significantly across the methods and also depend on the network topology.

6. Experiments

In this section, we present experimental performance results for our GLS-IR solvers. The simulation experiments are based on MPI implementations of the gossip-based aggregation algorithms push-sum (PS) and push-flow (PF) and were run on the Vienna Scientific Cluster VSC-2.[1] Using MPI implementations allows us to simulate large WSNs without the need of setting up and maintaining hundreds or thousands of physical sensor nodes. The gossip-based aggregation algorithms use asynchronous communication and in each round a node communicates with a single, randomly chosen neighbour.

A remaining open question is how to terminate gossip-based algorithms efficiently in a distributed environment. However, this question is beyond the scope of this paper. In our experiments, we terminate each gossip-based aggregation once the local approximation reaches a predefined accuracy compared to the exact value. The results presented in this section are averages over five runs on different random geometric topologies with similar average node degree (see Appendix for details on the random geometric topologies). The transmission radius r was chosen as $\sqrt{\log N/N}$, which leads to the vertex degree growing logarithmically in the number of nodes N [14]. The maximum number of iterations for IR was set to 10, but this upper limit was never reached in the fault-free experiments. In all experiments, well conditioned matrices were used and $\|A^\top r\|_2 \leq p_\alpha = 10^{-15}$ was achieved at termination. The convergence of D-LMS [23] depends on a problem dependent step-size parameter μ which has a vast search space. The parameter μ is highly dependent on the input data, the network size and topology. Even for the same input size and a matrix with the same condition number the step-size varies greatly. In order to achieve a fair comparison of the methods,

1 http://vsc.ac.at/systems/vsc-2/.

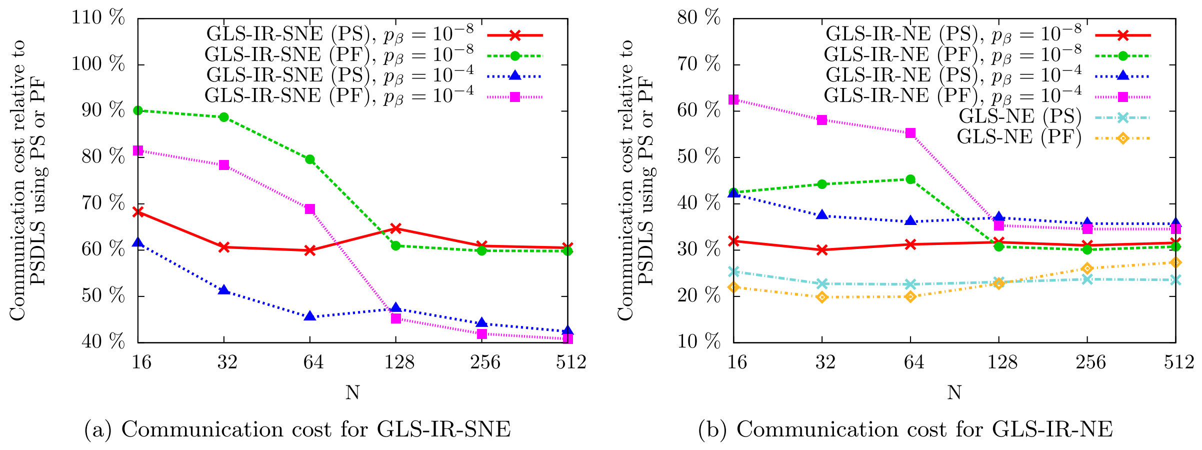


Fig. 1. Communication cost for GLS-IR relative to PSDLS using PS or PF, respectively, on random geometric topologies for $n = N$, $m = 8$ and $p_\alpha = 10^{-15}$.

we determined the best value for μ for each set of input data A and b, for each number of processors and for each topology. In the following figures, we report the average minimum number of messages required by D-LMS to reach the targeted accuracy p_α for each combination of the input parameters n, m, N and the topology.

6.1. Communication cost

Fig. 1 shows the communication cost, i.e. the average total number of messages sent per node, relative to PSDLS for different aggregation methods and different working precisions p_β to reach an accuracy of $p_\alpha = 10^{-15}$. Each node holds one row of A ($n = N$) and solves the dLLS problem for a skinny matrix with $m = 8$. For a larger choice of m the GLS-IR methods achieve even higher improvements compared to PSDLS. `dmGS` has to compute m columns, which corresponds to m gossip-based aggregations. Reducing the communication cost for the QR factorisation through the use of lower working precisions becomes even more significant for larger m, while the IR costs are independent of m. However, due to the limited CPU resources on the VSC2, we limit our experiments to skinny matrices. On average, 2–3 iterations were necessary for IR to converge to the desired accuracy. All variants of GLS-IR need significantly fewer messages than PSDLS. Two different working precisions p_β are used, 10^{-8} and 10^{-4}. In all cases, IR achieves a final accuracy of $p_\alpha = 10^{-15}$, but GLS-IR-SNE with $p_\beta = 10^{-4}$ requires fewer rounds than PSDLS for both PS and PF. Using $p_\beta = 10^{-8}$, for large N GLS-IR-SNE requires about 60% of the messages of PSDLS for PS and PF. For $p_\beta = 10^{-4}$, the message count is further reduced to only 42%. This shows the significant advantage of IR and lower working precisions while still achieving the same accuracy as a solver without IR. GLS-NE without IR is the fastest method requiring about 20% of the messages of PSDLS for large N. For GLS-IR-NE, the communication cost of IR cannot be compensated by using lower working precisions to form the NE, but for growing N the overhead is less than 7% higher than GLS-NE (PF). This is a very low overhead while providing fault tolerance through PF *and* IR. In the case of message loss, the accuracy of the initial factorisation is reduced because PS is then unable to reach the required working precision p_β. This corresponds to a fault-free situation where the initial factorisation was already computed in a working precision lower than p_β (e.g. 10^{-4} instead of 10^{-8}). IR can then improve the initial solution to the target precision p_α in exactly the same way as if the initial factorisation had already been computed in a lower p_β.

In Fig. 2, the communication cost for PSDLS and all GLS-IR methods is shown relative to D-LMS. PSDLS using PS requires on average 75% of the messages used by D-LMS, but PSDLS using PF already comes close to GLS-IR-SNE using PS and $p_\beta = 10^{-8}$ with less than 62% and 52% of the messages used by D-LMS, respectively. GLS-IR-SNE using PF and $p_\beta = 10^{-8}$ and GLS-IR-SNE using PS and $p_\beta = 10^{-4}$ almost require the same amount of messages to converge, averaging on a third (34%) of the messages used by D-LMS. The best SNE method in Fig. 2a is GLS-IR-SNE using PF and $p_\beta = 10^{-4}$ requiring only a quarter of the messages used by D-LMS to reach the target accuracy p_α. Fig. 2b shows the communication cost of the GLS-IR-NE methods relative to D-LMS, almost all of which require fewer messages than the GLS-IR-SNE methods. Again, the algorithms using PF as their aggregation method are always lower than their PS-based counterparts. GLS-NE using PF achieves the best performance compared to D-LMS, only requiring on average 15% of the messages to reach the target accuracy, being more than 6 times faster than D-LMS.

6.2. Fault tolerance

To examine the fault tolerance properties of the GLS-IR solvers, a message loss probability was introduced. In these experiments, we focus on lost messages, but temporary and permanent link failures could also be modelled as a continuous message loss. In our simulation, for each received message it is randomly decided if it is processed or discarded. We tested various message loss probabilities between 10^{-5} and 0.25 for $N = 128$, again on 5 different random geometric topologies. m was fixed at 8 and the target precision p_α was again set to 10^{-15}. The maximum number of rounds per gossip-based aggregation was limited to 10 000 to ensure termination even in the event of failures, which could cause the aggregation method to not be able to converge to the prescribed accuracy. The maximum number of IR iterations was increased to 100 to tolerate slow improvements due to large errors in the initial computation. The results for GLS-IR-SNE are shown in Fig. 3a and for GLS-IR-NE in Fig. 3b.

PSDLS using the non-fault-tolerant PS is not able to handle any message loss, leading to an accuracy of less than 10^{-2} for the lowest message loss probability of 10^{-5} and decreasing for higher loss probabilities. GLS-IR-SNE and GLS-IR-NE using PS as their aggregation method also cannot achieve any meaningful results. However, restricting the message loss only to the initial solution (lines 2–4 in Algorithm 1) and running IR without dropping any messages allows GLS-IR-SNE and GLS-IR-NE with PS to achieve an accurate result. The number of IR iterations required to reach the desired target accuracy increases with the message loss probability, but only for the most extreme loss probability of 0.25 IR failed to improve the solution.

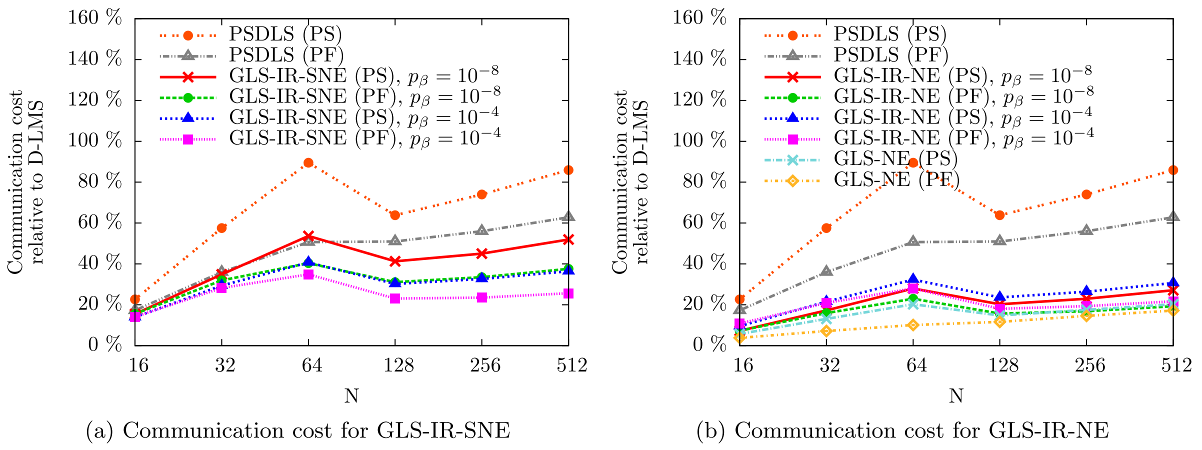


Fig. 2. Communication cost for PSDLS and GLS-IR relative to D-LMS on random geometric topologies for $n = N$, $m = 8$ and $p_\alpha = 10^{-15}$.

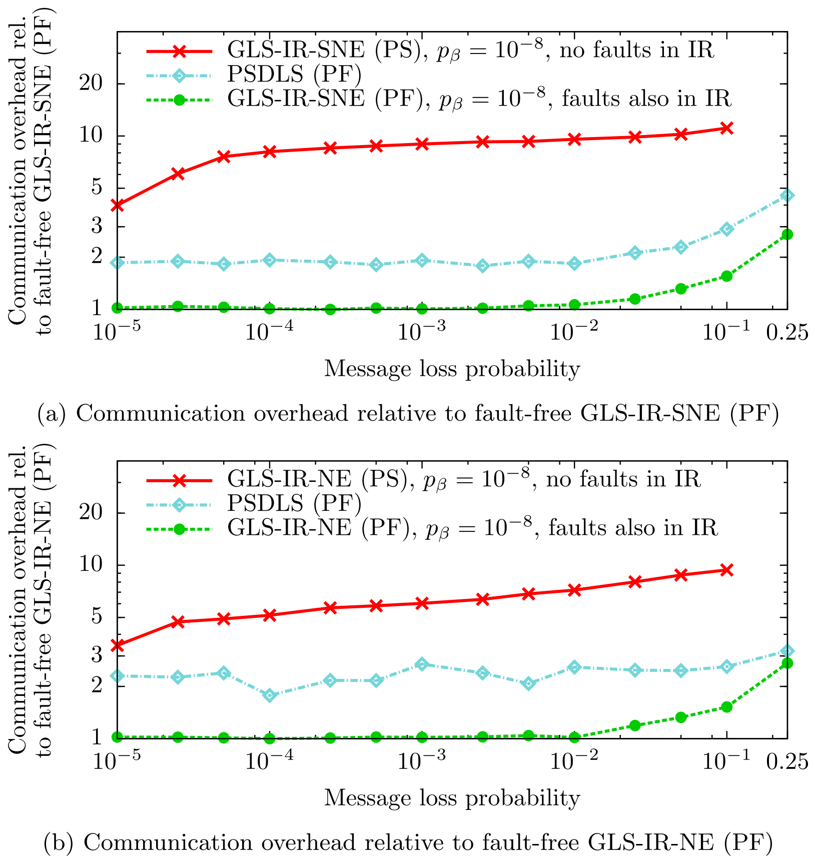


Fig. 3. Communication overhead with $p_\beta = 10^{-8}$ for increasing message loss probability for $n = N = 128$, $m = 8$ and $p_\alpha = 10^{-15}$ on random geometric topologies.

The fault-tolerant PF in combination with PSDLS or the GLS-IR methods can handle any message loss probabilities within the tested range. For the GLS-IR methods using PF, the number of IR iterations does not increase and the method converges to a solution accurate to 10^{-15} within only 2 IR iterations. The communication cost remained almost the same for all three methods using PF for message loss probabilities up to 10^{-2} in comparison to their fault-free runs. Only for very high message loss probabilities, PF required more messages to converge and for a message loss probability of 0.25, which on average corresponds to loosing every fourth message, the number of messages required to reach an accuracy of 10^{-15} tripled.

6.3. Working precisions

In Section 6.1 we compared the communication cost for two specific working precisions p_β, 10^{-8} and 10^{-4}. In this section, we analyse the range of working precisions $p_\beta = 10^i$ with $i = -14, \ldots, -2$ and examine the reduction of the communication cost depending on the working precision used. The target precision p_α was set to 10^{-15} and was reached in all cases. The experiments were again run on 5 different random geometric topologies with $N = 128$ and fixed $m = 8$ (Fig. 4a) or $m = 32$ (Fig. 4b).

In both cases, the total number of messages sent by GLS-IR-NE remains almost the same up until $p_\beta = 10^{-5}$. Only for the

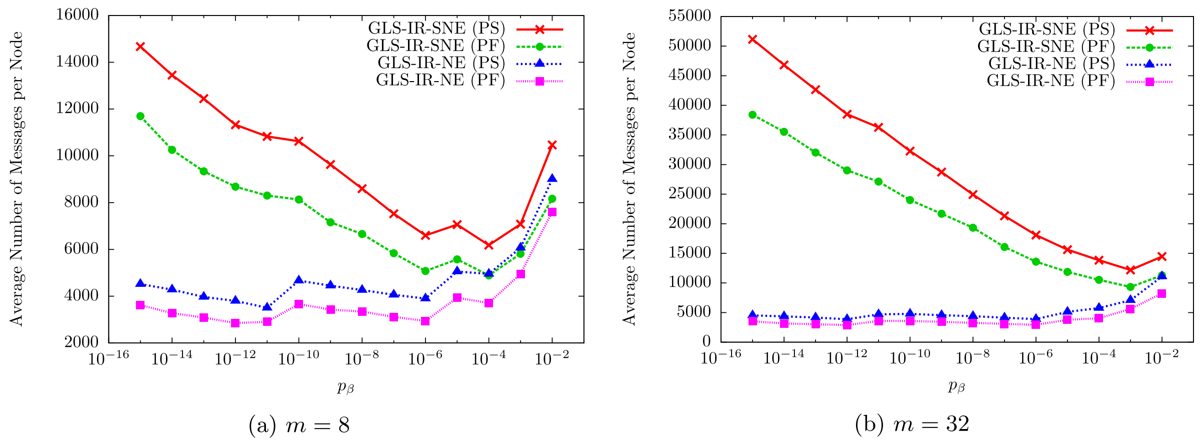


Fig. 4. Average number of messages per node for GLS-IR for various working precisions p_β and different m with $n = N = 128$ and $p_\alpha = 10^{-15}$ on random geometric topologies.

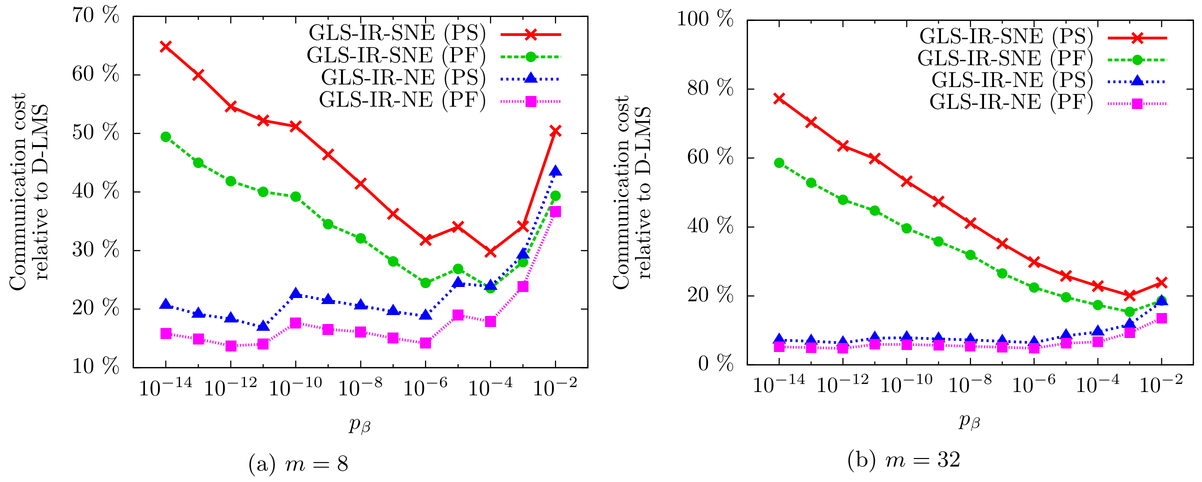


Fig. 5. Communication cost for GLS-IR relative to D-LMS for various working precisions p_β and different m with $n = N = 128$ and $p_\alpha = 10^{-15}$ on random geometric topologies.

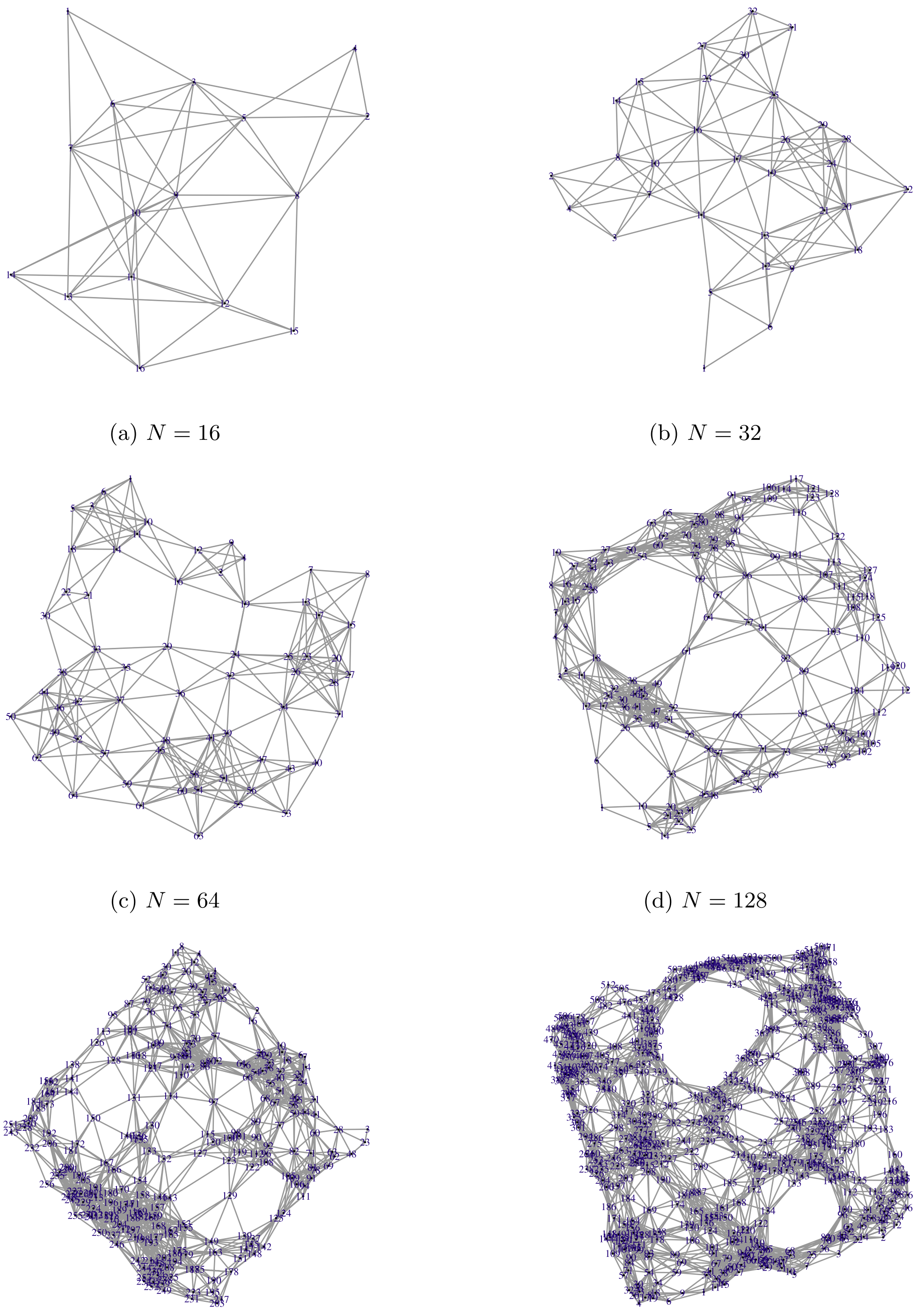
lowest working precisions, the number of messages increases up to a factor of 2 in the case of $m = 8$ and 2.5 for $m = 32$ compared to GLS-IR-NE using the same precision throughout the calculation ($p_\beta = p_\alpha$). For GLS-IR-SNE using PS or PF, decreasing the working precision p_β continuously decreases the number of messages required to reach the target precision p_α. For $m = 8$, GLS-IR-SNE requires the least number of messages to reach p_α for $p_\beta = 10^{-4}$ and only uses 40% of the messages compared to the same algorithm using $p_\beta = p_\alpha = 10^{-15}$. In the second case $m = 32$, an even further reduction can be observed for GLS-IR-SNE. To reach p_α using $p_\beta = 10^{-3}$, GLS-IR-SNE using PS or PF requires less than 25% of the messages compared to the case of $p_\beta = p_\alpha$. In all cases shown in Fig. 4, IR required 1–2 iterations for $p_\beta \in [10^{-14}, 10^{-6}]$, 3–5 iterations for $p_\beta \in [10^{-5}, 10^{-3}]$ and 7–8 iterations for $p_\beta = 10^{-2}$.

The communication cost of GLS-IR using various working precisions p_β relative to D-LMS is shown in Fig. 5. For $m = 8$ (Fig. 5a), GLS-IR-NE requires either 15% (PF) or 20% (PS) of the number of messages used by D-LMS for the majority of working precisions analysed in this section. GLS-IR-SNE starts off with 64% (PS) and 49% (PF) of the number of messages for $p_\beta = 10^{-14}$ and reaches 29% (PS) and 23% (PF) for $p_\beta = 10^{-4}$, the working precision requiring the least number of messages to reach p_α. For wider matrices, as shown for $m = 32$ in Fig. 5b, even for the lowest working precision $p_\beta = 10^{-2}$ GLS-IR-SNE does not reach 20% of the messages used by D-LMS, making GLS-IR-SNE more than 5 times faster than D-LMS. Up until $p_\beta = 10^{-5}$, GLS-IR-SNE using PF requires only about 6% and using PS about 7% of the messages used by D-LMS to reach the target accuracy of $p_\alpha = 10^{-15}$. For GLS-IR-SNE, the communication cost steadily declines from 77% for PS and 58% for PF to about 20% and 15% using $p_\beta = 10^{-3}$ for PS and PF, respectively.

These results demonstrate the benefits of using lower working precisions for the majority of the aggregation operations, while still being able to achieve the high target accuracy through the use of IR. Furthermore, the recovery capabilities of IR can be seen for various potential accuracies of the initial factorisation. As already mentioned in Section 4.4, the recovery from a fault only increases the number of iterations required by IR slightly. Due to the low computational complexity of the IR iterations, the effect of a fault on the total computation time will be very low compared to the factorisation of the matrix.

7. Conclusion

We presented two variants of the distributed gossip-based linear least squares solver GLS-IR, one of them based on the semi-normal equations, the other based on the normal equations, both combined with mixed precision iterative refinement. In these solvers, all communication operations between participating nodes are contained in gossip-based reduction operations. Consequently, GLS-IR directly benefits from all improvements


(a) $N = 16$ (b) $N = 32$

(c) $N = 64$ (d) $N = 128$

(e) $N = 256$ (f) $N = 512$

Fig. A.6. Examples for random geometric topologies from $N = 16$ to $N = 512$ as defined in Table A.3.

in such reduction operations. The fault-tolerance of the GLS-IR algorithms is achieved through the use of the fault-tolerant gossip algorithm push-flow and employing the self-correcting properties of iterative refinement. Thus, the algorithms become fault-tolerant against silent message loss and temporary or permanent node failures.

The experiments demonstrated that GLS-IR significantly reduces the number of messages compared to existing dLLS solvers. The use of lower working precisions has been shown to further reduce the communication cost without loss of accuracy. IR not only stabilises the method of SNE, but itself provides resilience against faults that occurred during the QR factorisation or the formation of the normal equations. The resilience of GLS-IR is further improved through the use of push-flow, which has been illustrated to handle high message loss probabilities in the context of our dLLS solver at a very low communication overhead.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have

Table A.3
Properties of the random geometric topologies.

N	Diameter *d*	Average node degree
16	0.411497	7.07
32	0.325317	7.87
64	0.251989	10.18
128	0.192460	12.88
256	0.145486	15.33
512	0.109114	17.98

impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.jpdc.2019.09.006.

Acknowledgments

This work has been partially supported by the Vienna Science and Technology Fund (WWTF) through project ICT15-113. The computational results presented have been achieved using the Vienna Scientific Cluster (VSC).

Appendix. Random geometric topologies

All experiments are averaged over five random geometric topologies, which were generated using the `igraph` R package (http://igraph.org/r/). The diameter used to generate the topologies and the resulting average node degrees are shown in Table A.3 and examples of the generated topologies are shown in Fig. A.6.

References

[1] R. Arablouei, S. Werner, Y. Huang, K. Dogancay, Distributed least mean-square estimation with partial diffusion, IEEE Trans. Signal Process. 62 (2) (2014) 472–484.
[2] A. Björck, Iterative refinement of linear least squares solutions I, BIT 7 (4) (1967) 257–278.
[3] A. Björck, Stability analysis of the method of semi-normal equations for linear least squares problems, Linear Algebra Appl. 48 (1987) 31–48.
[4] A. Björck, Numerical Methods for Least Squares Problems, SIAM, 1996.
[5] F. Cattivelli, A. Sayed, Diffusion LMS strategies for distributed estimation, IEEE Trans. Signal Process. 58 (3) (2010) 1035–1048.
[6] W.N. Gansterer, G. Niederbrucker, H. Straková, S. Schulze-Grotthoff, Scalable and fault tolerant orthogonalization based on randomized distributed data aggregation, J. Comput. Sci. 4 (6) (2013) 480–488.
[7] G. Golub, Numerical methods for solving linear least squares problems, Numer. Math. 7 (3) (1965) 206–216.
[8] N. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, 2002.
[9] D. Kempe, A. Dobra, J. Gehrke, Gossip-based computation of aggregate information, in: 44th Ann. IEEE Symp. Found. 2003, pp. 482–491.
[10] M.I. Khan, W.N. Gansterer, G. Haring, Static vs. mobile sink: The influence of basic parameters on energy efficiency in wireless sensor networks, Comput. Commun. 36 (9) (2013) 965–978.
[11] J. Langou, J. Langou, P. Luszczek, J. Kurzak, A. Buttari, J. Dongarra, Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy, in: Proc. Int. Conf. SC, 2006, pp. 50–50.
[12] G. Mateos, I.D. Schizas, G.B. Giannakis, Performance analysis of the consensus-based distributed LMS algorithm, EURASIP J. Adv. Signal Process. 2009 (1) (2009) 981030.
[13] A. Nedic, A. Ozdaglar, Cooperative distributed multi-agent optimization, in: Y. Eldar, D. Palomar (Eds.), Convex Optimization in Signal Processing and Communications, Cambridge University Press, 2010, pp. 340–386.
[14] M. Penrose, Random Geometric Graphs, Oxford University Press, 2003.
[15] B. Pittel, On spreading a rumor, SIAM J. Appl. Math. 47 (1) (1987) 213–223.
[16] K.E. Prikopa, W.N. Gansterer, E. Wimmer, Parallel iterative refinement linear least squares solvers based on all-reduce operations, J. Parallel Comput. 57 (2016) 167–184.
[17] K.E. Prikopa, H. Straková, W.N. Gansterer, Analysis and comparison of truly distributed solvers for linear least squares problems on wireless sensor networks, in: Euro-Par., Vol. 8632, Springer, 2014, pp. 403–414.
[18] F. Reichenbach, A. Born, D. Timmermann, R. Bill, A distributed linear least squares method for precise localization with low complexity in wireless sensor networks, in: IEEE DCOSS, Vol. 4026, Springer, 2006, pp. 514–528.
[19] G. Reise, G. Matz, K. Gröchenig, Distributed field reconstruction in wireless sensor networks based on hybrid shift-invariant spaces, IEEE Trans. Signal Process. 60 (10) (2012) 5426–5439.
[20] J. Rice, Matrix Computations and Mathematical Software, McGraw-Hill, 1981.
[21] A.H. Sayed, Diffusion Adaptation over Networks, first ed., in: ser. Academic Press Library in Signal Processing, Elsevier, 2013, ch. 9.
[22] I. Schizas, Consensus in ad hoc WSNs with noisy links - part II: Distributed estimation and smoothing of random signals, IEEE Trans. Signal Process. 56 (4) (2008) 1650–1666.
[23] I. Schizas, G. Mateos, G.B. Giannakis, Distributed LMS for consensus-based in-network adaptive processing, IEEE Trans. Signal Process. 57 (6) (2009) 2365–2382.
[24] L. Shi, W.-Z. Song, M. Xu, Q. Xiao, G. Kamath, J. Lees, G. Xing, Imaging seismic tomography in sensor network, in: IEEE DCOSS, 2013, pp. 304–306.
[25] H. Sivasankari, R. Leelavathi, K. Shaila, K. Venugopal, S.S. Iyengar, L.M. Patnaik, Dynamic cooperative routing (DCR) in wireless sensor networks, in: CNC 2012, Springer, 2012, pp. 87–92.
[26] H. Straková, W.N. Gansterer, T. Zemen, Distributed QR factorization based on randomized algorithms, in: PPAM, Vol. 7203, Springer, 2012, pp. 235–244.
[27] S. Sundhar Ram, A. Nedić, V.V. Veeravalli, Distributed stochastic subgradient projection algorithms for convex optimization, J. Optim. Theroy Appl. 147 (3) (2010) 516–545.
[28] J. Tsitsiklis, D. Bertsekas, M. Athans, Distributed asynchronous deterministic and stochastic gradient optimization algorithms, IEEE Trans. Automat. Control 31 (9) (1986) 803–812.
[29] S.-Y. Tu, A. Sayed, Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks, IEEE Trans. Signal Process. 60 (12) (2012) 6217–6234.
[30] J.H. Wilkinson, Rounding Errors in Algebraic Processes, Her Majesty's Stationery Office, 1963.
[31] Q. Zhou, S. Kar, L. Huie, H.V. Poor, Robust distributed least-squares estimation in sensor networks with node failures, in: IEEE Global Telecommunications Conference, 2011, pp. 1–6.

Karl E. Prikopa is a Ph.D. student in Computer Science at the University of Vienna, Austria. He received his B.Sc. and M.Sc. degree in Scientific Computing with distinction from the University of Vienna in 2009 and 2011, respectively, and is currently also completing a M.Sc. in Computational Science at the Institute of Physics. His research interests include high-performance computing, parallel and distributed algorithms and fault-tolerance.

Wilfried N. Gansterer is full professor and deputy head of the research group Theory and Applications of Algorithms at the Faculty of Computer Science of the University of Vienna. He received M.Sc. degrees in Mathematics from TU Vienna and in Scientific Computing & Computational Mathematics from Stanford University, and a Ph.D. degree in Scientific Computing from TU Vienna. His research interests cover various aspects of parallel computing, distributed computing and high performance computing, including resilient and fault tolerant algorithms, as well as applications in machine learning and Internet security.