Nerve-Skeleton Message Passing for Federated Optimization with Overlapping Parameters

Wei Zhang
Harbin Institute of Technology Shenzhen
zhangwei.sz@hit.edu.cn

Abstract

Existing federated learning setups assume either a single global model or a fixed decomposition with one globally shared parameter block and one client-specific private block. In this paper, we focus on federated optimization with overlapping parameters, where each client optimizes a local parameter block that may overlap with those of other clients. We encode these coordinate overlaps by a graph on clients, called the nerve skeleton. Based on this structure, we propose Nerve-Skeleton Message Passing (NSMP), a two-phase protocol on a spanning tree: a leaf-to-root pass composes reduced objectives by partial minimization, and a root-to-leaf pass reconstructs a joint assignment. We further show that, under the tree-elimination schedule in the paper, NSMP is equivalent to minimizing the joint objective, where the leaf-to-root phase evaluates the optimal objective value, and the root-to-leaf phase recovers a globally optimal assignment. Experiments show that NSMP solves federated optimization with overlapping parameters using only neighbor-to-neighbor message passing.

1 Introduction

Training machine learning models at the network edge reduces privacy exposure and the cost of uploading raw data to a remote cloud [1, 2]. Edge deployments also incur end-to-end latency from propagation, computation, and radio resources [3–5], which encourages local computation and infrequent aggregation. Federated learning follows this paradigm: clients update local models and exchange compressed updates with a server or peers [6, 7]. However, existing federated learning and decentralized optimization literature does not explicitly treat *overlapping local parameter coordinates* as the modeling primitive, and therefore lacks a unified and principled treatment of partially shared yet non-global variables.

The dominant federated learning template is a *star* topology: clients run local optimization on their data and upload parameter updates to a central server, which aggregates them—most visibly by averaging in FedAvg—to coordinate a *single* global model W shared by all clients [2, 6]:

$$\min_{W\in\mathbb{R}^d} \sum_{i=1}^{M} f_i(W). \tag{1}$$

Proximal local losses, limited data sharing, and client scheduling [8–10] refine (1) under non-IID data but still train one W. Personalization methods typically keep a *fixed* decomposition into shared and private parameters. FedRep [11] is representative: the *feature extractor* g_θ is shared by all clients, while each client keeps a *private* classification head h_{ϕ_i}. Writing ℓ for the loss and $\mathcal{D}_i$ for client i's data distribution, one may express the federated objective as

$$\min_{\theta,\{\phi_i\}_{i=1}^{M}} \sum_{i=1}^{M} f_i(\theta,\phi_i). \tag{2}$$

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

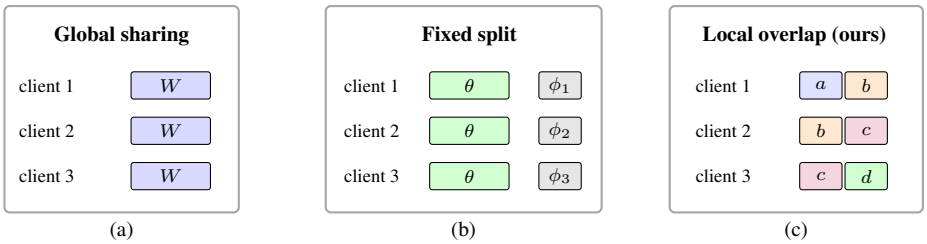


Figure 1: Three parameter-sharing regimes. (a) Global sharing, where every client holds a single full block W. (b) Fixed split, where all clients share a common block θ and each client has its own private block ϕ_i. (c) Local overlap, where client i holds a block W_i over the coordinates it shares with its neighbors (shown by the shared coordinates a, b, c).

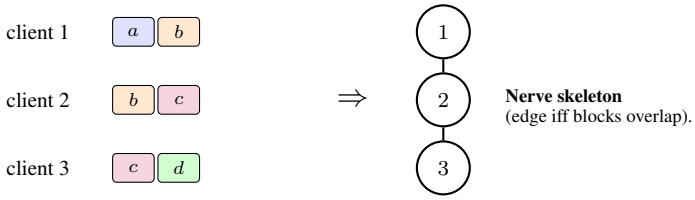


Figure 2: Illustration of the nerve skeleton. Each row lists one client's local parameter block; identical labels denote the same coordinate shared across clients. Clients are connected by an undirected edge if and only if their blocks share at least one coordinate, yielding the three-node path on the right.

Other fixed-split schemes (regularization toward a global model, heterogeneous blocks on the server, etc.) follow the same modeling pattern [12, 13].

Another relevant line of work performs federated learning over a graph [14–16]. In these works, the objective is still a *single* global model W as in (1), aggregated at the server or locally. For example, cross-silo federated learning [14] aggregates information along graph neighbors rather than only with a central server, which can improve throughput and reduce training delay. Federated graph learning, including FedGCN and related systems [15, 16], partitions a graph across clients so each keeps a subgraph $\mathcal{G}_i$, and the objective is

$$\min_{W} \sum_{i=1}^{M} f_i\big(W; \mathcal{G}_i\big). \tag{3}$$

Here f_i is client i's objective with global model W and local graph $\mathcal{G}_i$. Since evaluating f_i requires neighbors' features, FedGCN [15] uses preliminary communication to obtain those features, so later local training does not require frequent interaction with neighbors every round.

For the related work discussed above, the dominant pattern is still global-level parameter sharing, as summarized in Figure 1. Specifically, (1) and (3) correspond to Figure 1a, where all clients optimize one common model, while (2) corresponds to Figure 1b, where a globally shared block is coupled with client-specific private blocks. By contrast, in many applications each client shares trainable parameters only with neighbors in the client network, rather than with all clients, as illustrated in Figure 1c. This pattern is common: for example, in a regional hospital network each site may align model parameters only with nearby partners, not with every institution. Parameter overlap is therefore local and conditional on who collaborates with whom. Thus the setting differs from the existing federated works above, where shared parameters are strictly global.

To address the local parameter-sharing regime described above, this paper studies generalized federated optimization in which the optimized parameters overlap only between collaborating clients, as shown in Figure 1c. This is a more general framework; its special cases reduce to Figure 1a and Figure 1b. Whenever two clients have overlapping coordinates in their local parameter blocks, we place an edge between them; the resulting graph on clients is the *nerve skeleton* (see Figure 2). Based on this, we develop Nerve-Skeleton Message Passing (NSMP), a two-phase tree protocol where the leaf-to-root phase sends reduced objectives upward and the root-to-leaf phase recovers optimal pa-

rameters. We prove that with exact reductions this procedure is exactly equivalent to solving the joint federated objective centrally. The main contributions of this paper are as follows.

- We formulate a generalized federated optimization problem in which each client optimizes over a local parameter set, and different clients may share overlapping coordinates. We encode these overlaps by a graph on clients as the nerve skeleton. This unifies classical global-parameter federated learning and partially personalized models.
- To solve the generalized federated optimization, we propose a NSMP procedure on a spanning tree of the nerve skeleton. The NSMP has two phases, where the leaf-to-root phase partially minimizes local objectives into messages toward the parent, and a root-to-leaf phase recovers the optimal parameter values along the tree. This message passing only appears among neighbors, and will save communication time and reduce delay.
- We prove that NSMP is exactly equivalent to minimizing the generalized federated optimization globally. The leaf-to-root phase evaluates the optimal value, and the root-to-leaf phase recovers a globally optimal solution. Numerical experiments corroborate our analysis and validate the practical applicability of the proposed NSMP.

The paper is organized as follows. Section 2 describes the problem formulation for overlapping blocks in federated learning; Section 3 presents the procedures of NSMP for the leaf-to-root phase and the root-to-leaf phase; Section 4 analyzes exactness of the proposed NSMP under that schedule. Section 5 presents the experiments and brief application sketches. Proofs and supplementary details are in the appendices. Section 6 concludes the paper.

2 Problem formulation

In the standard federated learning formulation, all M clients optimize a *shared* parameter vector W, leading to the aggregate objective

$$\min_{W} \sum_{i=1}^{M} f_i(W), \tag{4}$$

where f_i denotes the local objective of client i. In this classical setting, every client updates the same coordinates of W, so (4) ties the entire network to one common parameter vector.

In this paper, we instead let client i optimize only over a local parameter block $W_i = \{w_{i,1}, \ldots, w_{i,|W_i|}\}$ and study the generalized federated optimization problem

$$\min_{W_1,\ldots,W_M} \sum_{i=1}^{M} f_i(W_i). \tag{5}$$

Compared with (4), each local objective in (5) depends on its own block W_i rather than on a single global argument W. In particular, two clients i and j may share some coordinates whenever $W_i \cap W_j \neq \emptyset$. We treat the problem of the form in (5) as generalized federated optimization. When every W_i coincides with the full coordinate set, the problem (5) reduces to (4).

To describe the structure of M clients in (5), let $\mathcal{V} = \{1, \ldots, M\}$ index the clients and let $\mathcal{W} = \bigcup_{i\in\mathcal{V}} W_i$ denote the union of all coordinates. As noted above, the local objectives are coupled whenever clients share coordinates, i.e.,

$$W_i \cap W_j \neq \emptyset \quad \text{for some } i \neq j.$$

As illustrated in Figure 2, we encode this coupling by an undirected overlap graph on $\mathcal{V}$, called the *nerve skeleton*,

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}), \qquad (i,j) \in \mathcal{E} \iff W_i \cap W_j \neq \emptyset.$$

To obtain a tractable distributed procedure, we assume $\mathcal{G}$ is connected[1] and select a spanning tree

$$\mathcal{T} = (\mathcal{V}, \mathcal{E}_{\mathcal{T}}), \qquad \mathcal{E}_{\mathcal{T}} \subseteq \mathcal{E},$$

of $\mathcal{G}$ and fix a root $r \in \mathcal{V}$. The following sections describe how to solve the generalized federated optimization problem (5) via message passing on $\mathcal{T}$.[2]

[1] If $\mathcal{G}$ is disconnected, clients in different connected components share no coordinates, so (5) decomposes into independent subproblems and each component can be optimized separately.

[2] The choice of tree and root does not affect the optimal value of the proposed NSMP. The precise conditions and analysis are given in Proposition 1 in Section 4 and Appendix C.1.

Algorithm 1 Tree-based two-phase optimization for NSMP

Input: Node set $\mathcal{V}$; local blocks $\{W_i\}_{i\in\mathcal{V}}$; local objectives $\{f_i(W_i)\}_{i\in\mathcal{V}}$
Output: Global variable assignment on $\mathcal{W}$

Construct the overlap graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $(i, j) \in \mathcal{E}$ iff $W_i \cap W_j \neq \emptyset$
Extract a spanning tree $\mathcal{T} = (\mathcal{V}, \mathcal{E}_\mathcal{T})$ from $\mathcal{G}$
Select a root r and orient $\mathcal{T}$ toward r
Define separator variables Z_i and eliminated coordinates U_i using (7)
// Leaf-to-root phase (leaves $\rightarrow$ root)
for each non-root node i in post-order traversal of $\mathcal{T}$ **do**
    Form the aggregated local objective $\tilde{f}_i(U_i, Z_i)$ using (8)
    Compute reduced message $\Phi_i(Z_i)$ using (9)
    Transmit $\Phi_i(Z_i)$ to its parent $p(i)$
end for
Compute root value: Φ_r using (10)
// Root-to-leaf phase (root $\rightarrow$ leaves)
Initialize $Z_r = \emptyset$
Compute $U_r^\star$ using (11)
Recover $W_r^\star,\ \{Z_k^\star\}_{k\in\mathcal{C}(r)}$ using (12)
for each node i in pre-order traversal of $\mathcal{T}$ starting from r **do**
    Compute $U_i^\star$ using (13)
    Recover $W_i^\star,\ \{Z_k^\star\}_{k\in\mathcal{C}(i)}$ using (14)
    Pass $Z_k^\star$ to child k
end for
return the recovered assignment on $\mathcal{W}$

3 Tree-based two-phase optimization

This section develops a two-phase protocol on the spanning tree $\mathcal{T}$ for the generalized federated problem (5). The *leaf-to-root* phase propagates reduced objectives upward and evaluates the optimal value of the joint objective. The *root-to-leaf* phase propagates messages downward and recovers an optimal solution at each node. Throughout both phases, messages traverse only tree edges, so each client communicates only with neighbors in $\mathcal{T}$.

3.1 Leaf-to-root phase

In the leaf-to-root phase, computation moves from the leaves to the root. This phase decomposes (5) into distributed subproblems over a spanning tree of the nerve skeleton. Before detailing the procedure, we introduce the involved notations. For each node i, let $\mathcal{C}(i)$ denote its children, $p(i)$ be its parent, and $\mathcal{S}_i$ denote the node set of the subtree rooted at i.

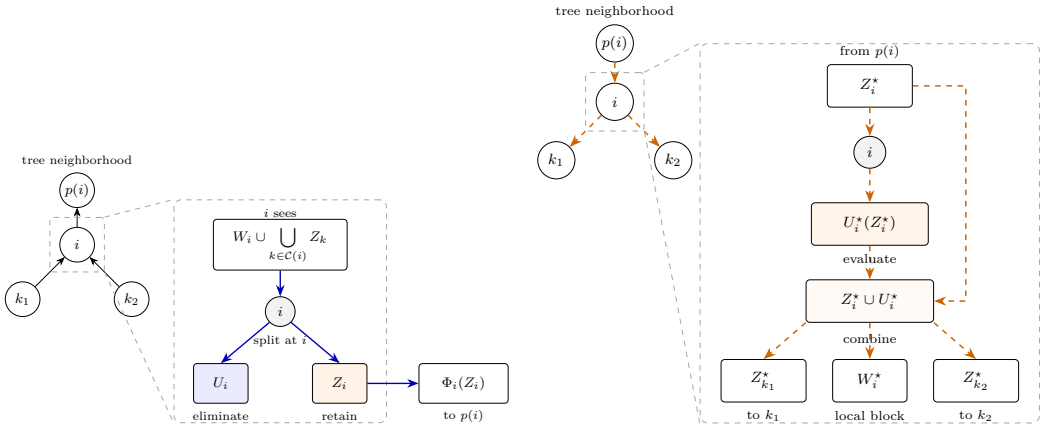
As illustrated in Figure 3a, node i receives from each child $k \in \mathcal{C}(i)$ a reduced objective whose argument is Z_k. Hence, the coordinates processed at node i are $W_i \cup (\cup_{k\in\mathcal{C}(i)} Z_k)$. Node i then splits these coordinates into eliminable and retained parts. Specifically, the local block W_i together with the incoming child variables $\{Z_k\}_{k\in\mathcal{C}(i)}$ is repartitioned as

$$W_i \cup \bigcup_{k\in\mathcal{C}(i)} Z_k \;=\; U_i \cup Z_i, \tag{6}$$

where U_i denotes the coordinates eliminated at node i, and Z_i denotes those retained as arguments of the reduced objective sent to the parent $p(i)$. More precisely, Z_i consists of the coordinates shared by the subtree rooted at i and the rest of the tree:

$$Z_i = \left(\bigcup_{j\in\mathcal{S}_i} W_j\right) \cap \left(\bigcup_{j\notin\mathcal{S}_i} W_j\right), \qquad U_i = \left(W_i \cup \bigcup_{k\in\mathcal{C}(i)} Z_k\right) \setminus Z_i. \tag{7}$$

We refer to Z_i as the separator variables. Thus, if a client in subtree $\mathcal{S}_i$ shares a parameter with a client outside it, that parameter is included in Z_i. This holds even when the two clients are not



(a) Leaf-to-root phase at node i.

(b) Root-to-leaf phase at node i.

Figure 3: (a) Leaf-to-root phase at node i. The coordinates visible at node i consist of its local block W_i together with the separator variables received from its children. These coordinates are split into U_i, which is eliminated locally, and Z_i, which remains as the argument of the reduced objective passed to the parent in the leaf-to-root phase. (b) Schematic view of the root-to-leaf phase. Once the separator values inherited from the parent are fixed, node i evaluates its stored minimizer map $U_i^\star(Z_i^\star)$, recovers its local coordinates, and propagates the induced separator values to its children.

directly connected in the selected tree. For the root node of the spanning tree $\mathcal{T}$, there is no parent, so $Z_r = \emptyset$.

Under the partition (6), in the leaf-to-root phase, node i adds the child's reduced objectives to the local term f_i to obtain

$$\tilde{f}_i(U_i, Z_i) = f_i(W_i) + \sum_{k \in \mathcal{C}(i)} \Phi_k(Z_k). \tag{8}$$

Then, node i eliminates the coordinates in U_i in (6). We define $\Phi_i(Z_i)$ as a reduced objective for the subtree rooted at i,

$$\Phi_i(Z_i) = \min_{U_i} \tilde{f}_i(U_i, Z_i). \tag{9}$$

As shown in (9), node i aggregates reduced objectives from its children, adds its local term f_i, and minimizes out U_i. The resulting reduced objective $\Phi_i(Z_i)$ is then transmitted to its parent $p(i)$.

In particular, if node i is a leaf node (i.e., $\mathcal{C}(i) = \emptyset$), the recursion in (9) reduces to

$$\Phi_i(Z_i) = \min_{U_i} f_i(W_i).$$

If node i is the root node r, we have $Z_r = \emptyset$, so Φ_r is a scalar:

$$\Phi_r = \min_{U_r} \left[f_r(W_r) + \sum_{k \in \mathcal{C}(r)} \Phi_k(Z_k) \right]. \tag{10}$$

This scalar is the final objective value produced by the leaf-to-root phase. That is, after each node solves the local problem in (9) and passes the reduced objective to its parent, the root obtains (10). We will show in Section 4 that Φ_r is equivalent to the minimization of the objective in (5).

3.2 Root-to-leaf phase

The leaf-to-root phase produces reduced objectives but does not yet recover the optimal local assignment for any client i. The root-to-leaf phase recovers these values by evaluating the stored minimizer maps while fixing separator variables from the root toward the leaves; see Figure 3b.

From the objective at the root in (10), the solution $U_r^\star$ is obtained through the following,

$$U_r^\star = \arg\min_{U_r} \left[f_r(W_r) + \sum_{k \in \mathcal{C}(r)} \Phi_k(Z_k) \right]. \tag{11}$$

Then, at the root r, since we have $Z_r^\star = \emptyset$, so the relation in (6) implies

$$U_r^\star = W_r^\star \cup \bigcup_{k \in \mathcal{C}(r)} Z_k^\star. \tag{12}$$

From $U_r^\star$ in (11) and based on the relation in (12), we recover the optimal local block $W_r^\star$ and the child separator values $\{Z_k^\star\}_{k \in \mathcal{C}(r)}$. These separator values are then propagated to the children of r.

As illustrated in Figure 3b, for a non-root node i, once it receives $Z_i^\star$ from its parent, it computes $U_i^\star$ according to the reduced objective in (9),

$$U_i^\star = \arg\min_{U_i} \tilde{f}_i(U_i, Z_i^\star). \tag{13}$$

Given $U_i^\star$ and $Z_i^\star$, specializing (6) to the values of $U_i^\star$ and $Z_i^\star$ yields the local block $W_i^\star$ and the child separator values $\{Z_k^\star\}_{k \in \mathcal{C}(i)}$ via

$$W_i^\star \cup \bigcup_{k \in \mathcal{C}(i)} Z_k^\star = U_i^\star \cup Z_i^\star. \tag{14}$$

After that, node i fixes its local assignment $W_i^\star$ and passes the induced separator values $\{Z_k^\star\}_{k \in \mathcal{C}(i)}$ to its children. Repeating this procedure along a preorder traversal of $\mathcal{T}$ recovers the values on every local block, i.e., $\{W_i^\star\}_{i=1}^M$. In summary, the root-to-leaf phase assigns the solution $W_i^\star$ to each node i, and Algorithm 1 summarizes the complete NSMP procedure.

Overall, one can find that the proposed NSMP uses a *compress-then-recover* schedule on a spanning tree of the nerve skeleton. In the leaf-to-root phase, clients pass messages *up* the tree: each node combines its local objective with incoming child messages into a reduced objective that depends only on the separator variables Z_i defined in (7), and sends this objective to its parent. In the root-to-leaf phase, messages move *down* the tree: each client receives separator values from its parent, evaluates its stored minimizer map, and recovers its local block W_i. It is worth noting that only neighboring clients on the spanning tree exchange these updates, rather than broadcasting to the entire network. Section 4 proves that the procedure solves (5) exactly.

4 Theoretical analysis and optimization consistency

In this section, we show that NSMP can recover the optimal solution of the original problem in (5). Throughout the exactness analysis, we assume that the partial minima in (9) and (10) and the joint minimum in (5) are finite and attained. When the exact reduced objectives $\Phi_i(Z_i)$ in (9) have no tractable representation, NSMP can instead transmit fitted approximations. The construction procedure and analysis are provided in Appendix D.2.

4.1 Exact reduced objectives in the leaf-to-root phase

We first show that, in the leaf-to-root phase, each node computes the exact reduced objective of its subtree. The following lemma establishes this property for $\Phi_i(Z_i)$ defined in (9).

Lemma 1 (Exact subtree reduction)**.** *For any node $i \in \mathcal{V}$, let $\mathcal{S}_i$ denote the node set of the subtree rooted at i, then the reduced objective $\Phi_i(Z_i)$ in* (9)*, which is computed by NSMP in the leaf-to-root phase of Algorithm 1, satisfies*

$$\Phi_i(Z_i) = \min_{\left(\bigcup_{j \in \mathcal{S}_i} W_j\right) \setminus Z_i} \sum_{j \in \mathcal{S}_i} f_j(W_j). \tag{15}$$

Lemma 1 shows that the leaf-to-root phase performs exact elimination rather than heuristic aggregation: at every node, $\Phi_i(Z_i)$ is exactly the reduced objective of the corresponding subtree. As a straightforward extension, the following lemma holds at the root node.

Lemma 2 (Root exactness)**.** *Suppose Φ_r defined in* (10) *is returned at the root by the leaf-to-root phase of NSMP in Algorithm 1. Then Φ_r equals the optimal value of* (5)*, i.e.,*

$$\Phi_r = \min_{\mathcal{W}} \sum_{i \in \mathcal{V}} f_i(W_i). \tag{16}$$

Therefore, Lemma 2 implies that the root value computed by the leaf-to-root phase equals the global optimum of (5). In other words, the root objective in (10) is an exact reduction of the original generalized federated optimization problem in (5).

4.2 Exact recovery and optimization consistency

We next show that the root-to-leaf phase reconstructs client-wise solutions $W_i^\star$ that are globally optimal for the joint problem (5). In the root-to-leaf phase, node i receives $Z_i^\star$ from its parent and recovers $W_i^\star$ from the minimizer map (13) together with (14). The following lemma shows that the resulting assignment $\{W_i^\star\}_{i \in \mathcal{V}}$ is globally optimal for the original problem in (5).

Lemma 3 (Recoverability)**.** *Starting from the root boundary condition $Z_r = \emptyset$, the root-to-leaf phase of NSMP in Algorithm 1 recursively reconstructs a complete assignment $\{W_i^\star\}_{i \in \mathcal{V}}$ over $\mathcal{W}$. Moreover, this assignment is globally optimal for* (5)*.*

Therefore, combining the results in Lemma 2 and Lemma 3, we obtain the exactness of NSMP in Algorithm 1 in the following theorem.

Theorem 1 (Exact equivalence)**.** *For the generalized federated optimization problem in* (5)*, NSMP in Algorithm 1 is exactly equivalent to the original problem in* (5)*, in the sense that the assignment recovered by NSMP is globally optimal and attains the same objective value as the original problem.*

Remark 1 (Optimization consistency)**.** *By Theorem 1, NSMP in Algorithm 1 is an exact solver for* (5)*: the leaf-to-root phase computes exact reduced objectives, and the root-to-leaf phase reconstructs a globally optimal assignment. This implies optimization consistency directly.* Value consistency *follows from $\Phi_r = \min_{\mathcal{W}} \sum_{i \in \mathcal{V}} f_i(W_i)$, and* solution consistency *follows because the root-to-leaf phase reconstructs a complete assignment that attains this value.*

Remark 2 (Uniqueness)**.** *If the original problem in* (5) *admits a unique global minimizer, then the assignment reconstructed by the root-to-leaf phase must coincide with that unique optimizer and is therefore itself unique. If, instead, the original problem has multiple global minimizers, a local argmin in $U_i^\star(Z_i)$ of* (9) *can have more than one solution. The root-to-leaf phase then returns one of the globally optimal full assignments, depending on which minimizer is picked.*

It is worth noting that the proposed NSMP relies on the spanning tree from the nerve skeleton of the clients, and the preceding conclusions and discussion hold for this fixed spanning tree. The following proposition establishes that cycles in the nerve skeleton and the choice of spanning tree do not affect the exactness of NSMP.

Proposition 1 (Exactness under spanning tree selection)**.** *Cycles in the nerve skeleton and the choice of spanning tree do not affect the exactness of NSMP for the fixed problem in* (5)*. In other words, NSMP attains the same optimal value and recovers a global minimizer of the original problem for any spanning tree of the nerve skeleton.*

Thus, tree selection does not affect the exactness or the optimal value of NSMP. It may change the associated computation and communication costs, which we discuss in Appendix C.2.

4.3 Communication complexity

Since NSMP operates on a tree, every non-root node i has a unique parent $p(i)$. In the leaf-to-root phase, each non-root node sends its reduced objective $\Phi_i(Z_i)$ in (9) to its parent exactly once. In the root-to-leaf phase, it receives the separator values $Z_i^\star$ from its parent exactly once and uses them to recover its local assignment according to (13) and (14).

The communication overhead of the upward message depends on the functional form and representation of $\Phi_i(Z_i)$ in (9). Let s_i denote the number of real scalars required to encode this message, and let $z_i = |Z_i|$ be the separator dimension. Since the downward message $Z_i^\star$ contains z_i values,

the total numerical payload is

$$C_{\mathrm{NSMP}} = \sum_{i \neq r} (s_i + z_i). \tag{17}$$

Thus, each of the $M-1$ tree edges carries one message in each direction, giving $2(M-1)$ directed messages in total. For fixed message representations, the communication cost in (17) is determined by the chosen rooted tree. In Appendix C.2, we discuss how to optimize the choice of tree for NSMP by accounting for both communication and computation costs.

As a special case, suppose each local objective $f_i(W_i)$ in (5) is quadratic in W_i. Then, the reduced objective $\Phi_i(Z_i)$ in (9) remains quadratic and requires $s_i = z_i(z_i+1)/2 + z_i + 1$ real scalars for transmission. Substituting into (17) gives

$$C_{\mathrm{NSMP}} = \sum_{i \neq r} \left[\frac{z_i(z_i+1)}{2} + 2z_i + 1 \right]. \tag{18}$$

Let $z_{\max} = \max_{i \neq r} z_i$ denote the largest separator dimension, then the total payload is $O(M z_{\max}^2)$. For bounded separator dimensions, the communication cost grows at most linearly with the number of clients. A detailed analysis of communication complexity is provided in Appendix C.

5 Numerical experiments

In this part, we first verify that NSMP recovers the same solution as centralized joint optimization. We then evaluate its predictive performance and communication cost on real-world datasets.

5.1 NSMP for least-squares regression

First of all, we evaluate NSMP on a least-squares regression task with six clients $\mathcal{V} = \{1, \ldots, 6\}$. Global coordinates are $\mathcal{W} = \{a_1, \ldots, a_{11}\}$; the overlapping local blocks W_i are given in Table 1.

For each $i \in \mathcal{V}$, let $\mathbf{w}_i \in \mathbb{R}^{|W_i|}$ stack the coordinates in W_i. For a local sample index $n \in \{1, \ldots, N_i\}$, the scalar $y_{i,n}$ is the response, $\mathbf{x}_{i,n} \in \mathbb{R}^{|W_i|}$ is the associated feature vector, and $\varepsilon_{i,n}$ is additive noise. The local observation model and the corresponding squared-error loss are

$$y_{i,n} = \mathbf{x}_{i,n}^{\mathsf{T}} \mathbf{w}_i + \varepsilon_{i,n}, \tag{19}$$

$$f_i(\mathbf{w}_i) = \frac{1}{2} \sum_{n=1}^{N_i} \left(y_{i,n} - \mathbf{x}_{i,n}^{\mathsf{T}} \mathbf{w}_i \right)^2, \tag{20}$$

where N_i is the number of local samples and f_i is the ordinary least squares objective of client i. The joint least-squares objective aggregates the local terms as

$$\min_{\mathcal{W}} \sum_{i \in \mathcal{V}} f_i(\mathbf{w}_i). \tag{21}$$

When NSMP is applied to solving (21), each leaf-to-root message $\Phi_i(\cdot)$ in (9) is quadratic in the separator variables Z_i. See Appendix E for the detailed derivation and procedure.

For effectiveness verification, we compare the NSMP solution with direct centralized least-squares computation for (21). As shown in Figure 4, the two solutions coincide coordinate-wise up to numerical precision, confirming that NSMP attains the same global optimum as centralized computation, which is fully consistent with the exact-equivalence result in Theorem 1.

From a communication and privacy perspective, the proposed NSMP transmits only reduced quadratic messages $\Phi_i(\cdot)$, rather than raw samples, i.e., $\{x_{i,n}, y_{i,n}\}_{i=1}^{N_i}$. This avoids direct data sharing; its payload is quantified in (18). Combined with the exactness results and the empirical agreement in Figure 4, this experiment verifies optimization consistency. Appendix G.1 additionally compares communication with objective-matched iterative solvers.

5.2 Traffic prediction on METR-LA

We evaluate NSMP on METR-LA [17] with 206 traffic-sensor clients. Each client predicts the next speed value using 12 lag values and a bias, giving 13 model parameters. For each client, we

Client i	Local block W_i
1	$\{a_1, a_2\}$
2	$\{a_3, a_4\}$
3	$\{a_2, a_4, a_5, a_6\}$
4	$\{a_7, a_8\}$
5	$\{a_8, a_9, a_{10}\}$
6	$\{a_6, a_{10}, a_{11}\}$

Table 1: Local parameter blocks in the six-client regression example.

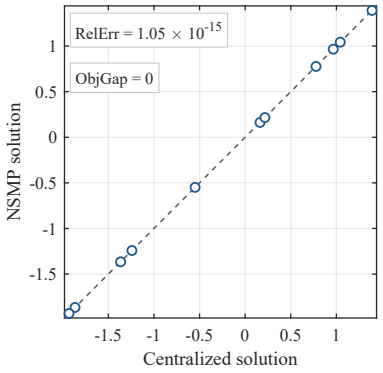


Figure 4: NSMP and centralized solutions agree coordinate-wise.

Table 2: METR-LA test RMSE (lower is better) and communication in bytes. Percentages indicate retained training data. Byte counts are identical in both settings and use Float64, excluding protocol overhead and hyperparameter searches.

Method	RMSE (1%)	RMSE (0.5%)	Communication bytes
Local	4.2548	4.5508	0
FedAvg	4.1512	4.1627	12,854,400
FedRep	4.1595	4.1805	11,865,600
NSMP	**4.0742**	**4.1106**	193,520

chronologically split its data into 70% for training, 10% for validation, and 20% for testing. To evaluate federated optimization with limited local training data, we consider two settings in which each client has access to only 1% or 0.5% of its own 70% training split. Each client's validation and test splits remain unchanged. All methods use the same data splits and retained training samples.

We compare NSMP with Local, FedAvg [6], and FedRep [11]. Local fits each client independently; FedAvg shares all parameters; FedRep shares lag coefficients selected by validation while keeping the remaining parameters private. NSMP solves a graph-regularized personalized objective by passing reduced quadratic messages on a maximum-similarity spanning tree. The comparison evaluates predictive performance under these different parameter-sharing rules. Detailed settings and the NSMP objective are provided in Appendix G.2.

As shown in Table 2, NSMP achieves the lowest test RMSE in both settings, outperforming Local, FedAvg, and FedRep even when each client uses only 0.5% of its training split. NSMP transmits 193,520 bytes through one upward and one downward tree pass, whereas FedAvg and FedRep each transmit over 60 times as many bytes. These results show that NSMP preserves parameter sharing across clients while achieving lower prediction error with low communication overhead.

5.3 Potential applications of NSMP

In general, the proposed NSMP is potentially useful in forecasting tasks where each client has both a shared trend and a local effect. Intuitively, neighboring clients (e.g., nearby regions or connected sites) should have similar shared components, while each client still keeps its own private component. For example, retail demand forecasting can combine regional trends with store-specific effects. Hospital risk modeling can account for both regional factors and factors specific to each hospital. Similarly, cell-tower traffic prediction can capture shared spatial trends alongside site-specific variation. Full model definitions are provided in Appendix F.

In Appendices G.1 and G.3, we compare NSMP with iterative distributed optimization methods, such as ADMM and Gossip. The results show that NSMP matches the centralized solution with substantially lower communication cost than these methods. In Appendix G.4, we evaluate NSMP with approximate messages. The results show that NSMP achieves better predictive performance than the evaluated federated baselines while transmitting a much smaller payload.

6 Conclusion

We studied federated optimization with overlapping local coordinates and proposed NSMP on a spanning tree of the nerve skeleton graph. Under the tree-elimination schedule presented in this paper, the leaf-to-root phase computes the global optimal value of the original problem, and the root-to-leaf phase reconstructs a globally optimal assignment. We established exact equivalence to the joint optimization problem, with consistency in both objective value and recovered solution. Experiments on synthetic and real-world datasets demonstrate the feasibility of NSMP and show that its message-passing mechanism substantially reduces communication overhead. More broadly, NSMP reaches the exact global optimum through efficient message passing only between neighboring clients, which is attractive under communication and privacy constraints.

Acknowledgments and Disclosure of Funding

This work was supported in part by the National Natural Science Foundation of China under Grant No. 62401180, the Guangdong Basic and Applied Basic Research Foundation under Grant No. 2025A1515011733, and the Shenzhen Science and Technology R&D Funds under Grant No. GXWD20231130104830003.

References

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy H Katz, Andrew Konwinski, Gunho Lee, David A Patterson, Ariel Rabkin, Ion Stoica, et al. Above the clouds: A Berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.

[2] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2):1–210, 2021.

[3] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 49–62, 2010.

[4] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 8(4):14–23, 2009.

[5] Q. Chen, L. Zhang, Y. Gao, and W. Zhang. A cloud-edge collaborative framework for object detection using KubeEdge. In *2025 10th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, pages 412–417, 2025.

[6] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282. PMLR, 2017.

[7] Solmaz Niknam, Harpreet S Dhillon, and Jeffrey H Reed. Federated learning for wireless communications: Motivation, opportunities, and challenges. *IEEE Communications Magazine*, 58 (6):46–51, 2020.

[8] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems*, volume 2, pages 429–450, 2020.

[9] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[10] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 1698–1707. IEEE, 2020.

[11] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *International Conference on Machine Learning*, volume 139, pages 2089–2099. PMLR, 2021.

[12] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, volume 139, pages 6357–6368. PMLR, 2021.

[13] Samiul Alam, Luyang Liu, Ming Yan, and Mi Zhang. FedRolex: Model-heterogeneous federated learning with rolling sub-model extraction. In *Advances in Neural Information Processing Systems*, volume 35, pages 29677–29690, 2022.

[14] Othmane Marfoq, Chuan Xu, Giovanni Neglia, and Richard Vidal. Throughput-optimal topology design for cross-silo federated learning. *Advances in Neural Information Processing Systems*, 33:19478–19487, 2020.

[15] Yuhang Yao, Weizhao Jin, Srivatsan Ravi, and Carlee Joe-Wong. FedGCN: Convergence-communication tradeoffs in federated training of graph convolutional networks. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[16] Chaoyang He, Emir Ceyani, Keshav Balasubramanian, Murali Annavaram, and Salman Avestimehr. SpreadGNN: Decentralized multi-task federated learning for graph neural networks on molecular data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6865–6873, 2022.

[17] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

[18] Paul Erdős, A. W. Goodman, and Lajos Pósa. The representation of a graph by set intersections. *Canadian Journal of Mathematics*, 18:106–112, 1966.

[19] Umberto Bertelè and Francesco Brioschi. *Nonserial Dynamic Programming*, volume 91 of *Mathematics in Science and Engineering*. Academic Press, 1972.

[20] Rina Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1–2):41–85, 1999.

[21] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194, 1988.

[22] Srinivas M. Aji and Robert J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

[23] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[24] Yair Weiss and William T. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, 2001.

[25] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.

[26] Angelia Nedić and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[27] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[28] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[29] Yiheng Xie, Lucien Werner, Kaibo Chen, Thuy-Linh Le, Christine Ortega, and Steven Low. Synchrophasors and synchrowaveforms for the distribution grid: The SoCal 28-bus dataset. arXiv:2504.06588, 2025.

A Related Work

In this part, we review related work on graph-structured modeling, inference, and distributed optimization, and summarize how our algorithm, NSMP, differs from existing methods.

The set-intersection representation studied by Erdős et al. [18] provides a classical basis for describing overlap among variable sets. Given scopes $\{\mathcal{B}_i\}_{i=1}^M$, the associated graph has edge set

$$\mathcal{E} = \big\{\{i,j\} : i \neq j,\ \mathcal{B}_i \cap \mathcal{B}_j \neq \emptyset\big\}. \tag{22}$$

This representation identifies which local terms may interact, but does not by itself specify an inference or optimization procedure. In particular, pairwise adjacency alone leaves the identities and dimensions of shared coordinates unspecified. This distinction between representing overlap and organizing computation is central to the subsequent literature on elimination and junction trees.

Nonserial dynamic programming [19] and Dechter's bucket-elimination framework [20] exploit factorization by successively eliminating variables and retaining functions over the remaining interfaces. Lauritzen and Spiegelhalter [21] developed local probabilistic computation on graphical structures, while the generalized distributive law [22] and factor-graph framework [23] clarify the algebraic and representational connections among message-passing algorithms. In the min-sum setting, their common elimination principle is expressed by

$$m_{i\to j}(x_{S_{ij}}) = \min_{x_{\mathcal{B}_i \setminus S_{ij}}} \left[\psi_i(x_{\mathcal{B}_i}) + \sum_{k\in\mathcal{N}_T(i)\setminus\{j\}} m_{k\to i}(x_{S_{ki}})\right], \tag{23}$$

where ψ_i is a cluster cost, $\mathcal{N}_T(i)$ denotes its tree neighbors, and $S_{ij} = \mathcal{B}_i \cap \mathcal{B}_j$ is a separator. Exact tree computation requires running intersection: all clusters containing a coordinate must form a connected subtree. Thus, an arbitrary spanning tree of an intersection graph need not be a valid junction tree for the original scopes. These works establish both the power and the principal cost of exact elimination: intermediate scopes depend on the elimination order, and message representations can grow rapidly with separator size.

Gaussian belief propagation is a particularly relevant specialization because quadratic information-form messages admit finite-dimensional representations. Weiss and Freeman [24] analyze Gaussian BP beyond trees, showing that converged messages yield correct means even though the inferred variances need not be exact. The connection to quadratic optimization follows from the Schur complement. For a quadratic with block Hessian $\left[\begin{smallmatrix} A & B \\ B^\top & D \end{smallmatrix}\right]$, linear coefficients (a, b), and eliminated block $A \succ 0$, the retained quadratic and linear coefficients are

$$H = D - B^\top A^{-1} B, \qquad h = b - B^\top A^{-1} a. \tag{24}$$

Gaussian integration and partial minimization give the same coefficients, up to an additive normalization term in the negative log message. This algebraic connection does not make exact tree elimination and iterative loopy BP interchangeable: the former relies on valid separators, whereas the latter raises a separate convergence question.

A different line of research studies repeated exchange of local estimates. Boyd et al. [25] analyze randomized gossip for distributed averaging and relate its convergence speed to the spectral properties of the mixing process. Their results characterize how network connectivity affects the communication needed to reach agreement. To optimize heterogeneous local objectives, Nedić and Ozdaglar [26] combine neighbor averaging with subgradient steps, with a representative update

$$x_i^{t+1} = \sum_j P_{ij}^t x_j^t - \alpha_t g_i^t, \qquad g_i^t \in \partial f_i(x_i^t), \tag{25}$$

where P^t is a neighbor-mixing matrix and α_t is a stepsize. This literature makes explicit the interaction between optimization error and disagreement across agents. Compared with function-valued elimination messages, these methods use vector exchanges and inexpensive local first-order operations, with convergence governed by objective assumptions, connectivity, mixing weights, and stepsize choices.

Beyond first-order methods based on neighbor averaging, the alternating direction method of multipliers (ADMM) offers a complementary approach through augmented-Lagrangian splitting. Boyd

et al. [27] develop consensus and sharing formulations for distributed optimization, including problems with local variable copies coupled by linear constraints. For $\min_{x,z} F(x) + G(z)$ subject to $Ax + Bz = c$, the scaled updates are

$$\begin{aligned} x^{t+1} &\in \arg\min_x \left\{ F(x) + \tfrac{\rho}{2}\|Ax + Bz^t - c + u^t\|_2^2 \right\}, \\ z^{t+1} &\in \arg\min_z \left\{ G(z) + \tfrac{\rho}{2}\|Ax^{t+1} + Bz - c + u^t\|_2^2 \right\}, \\ u^{t+1} &= u^t + Ax^{t+1} + Bz^{t+1} - c, \end{aligned} \tag{26}$$

with scaled dual variable u and penalty $\rho > 0$. Unlike pure averaging, the dual updates explicitly enforce the coupling constraints. Partial sharing can be expressed through coordinate-selection constraints $R_{ij}x_i = R_{ji}x_j$, so consensus formulations need not impose a single fully shared parameter block. Appropriate splitting supports separable local solves and neighbor communication on cyclic graphs. The established convex theory gives convergence under suitable assumptions, while practical performance depends on subproblem cost, penalty selection, and residual tolerances; general non-convex settings require separate analysis.

Within this literature, NSMP is a federated specialization of min-sum elimination for client-owned objectives with partially overlapping parameters. Its contribution lies in defining the messages exchanged between clients with overlapping parameters and recovering consistent local solutions through a two-pass procedure. With exact reductions, the proposed NSMP solves the optimization problem through leaf-to-root and root-to-leaf message passing. While gossip-based optimization and ADMM repeatedly exchange local iterates, NSMP completes the solution process in two tree passes. Our experiments show that this two-pass scheme substantially reduces communication overhead compared with gossip-based optimization and ADMM.

B Proofs

B.1 Proof of Lemma 1

We show the proof by induction for the subtree rooted at i. First of all, if node i is a leaf, then $\mathcal{C}(i) = \emptyset$, and by definition in (9)

$$\Phi_i(Z_i) = \min_{U_i} f_i(W_i). \tag{27}$$

Since the leaf node has no child node, the decomposition in (6) reduces to $W_i = U_i \cup Z_i$. Hence, $\mathcal{S}_i = \{i\}$ and $\left(\bigcup_{j\in\mathcal{S}_i} W_j\right) \setminus Z_i = U_i$. Thus, (15) holds trivially for the leaf node.

Now assume the claim holds for every child $k \in \mathcal{C}(i)$, that is,

$$\Phi_k(Z_k) = \min_{\left(\bigcup_{j\in\mathcal{S}_k} W_j\right)\setminus Z_k} \sum_{j\in\mathcal{S}_k} f_j(W_j). \tag{28}$$

Using the recursive definition of $\Phi_i(Z_i)$ in (9),

$$\Phi_i(Z_i) = \min_{U_i} \left[f_i(W_i) + \sum_{k\in\mathcal{C}(i)} \Phi_k(Z_k) \right]. \tag{29}$$

Substituting the induction hypothesis in (28) yields

$$\Phi_i(Z_i) = \min_{U_i} \left[f_i(W_i) + \sum_{k\in\mathcal{C}(i)} \min_{\left(\bigcup_{j\in\mathcal{S}_k} W_j\right)\setminus Z_k} \sum_{j\in\mathcal{S}_k} f_j(W_j) \right]. \tag{30}$$

Then, the nested minimizations can be merged into one joint minimization:

$$\Phi_i(Z_i) = \min_{U_i,\, \left\{\left(\bigcup_{j\in\mathcal{S}_k} W_j\right)\setminus Z_k\right\}_{k\in\mathcal{C}(i)}} \left[f_i(W_i) + \sum_{k\in\mathcal{C}(i)} \sum_{j\in\mathcal{S}_k} f_j(W_j) \right]. \tag{31}$$

Because $\mathcal{S}_i = \{i\} \cup \bigcup_{k\in\mathcal{C}(i)} \mathcal{S}_k$, we may combine the child sums into one sum over $\mathcal{S}_i$:

$$\Phi_i(Z_i) = \min_{U_i,\ \left\{\left(\bigcup_{j\in\mathcal{S}_k} W_j\right)\setminus Z_k\right\}_{k\in\mathcal{C}(i)}} \sum_{j\in\mathcal{S}_i} f_j(W_j). \tag{32}$$

It remains only to prove that the optimization-variable sets in (15) and (32) are identical:

$$\begin{aligned}
\left(\bigcup_{j\in\mathcal{S}_i} W_j\right) \setminus Z_i &= \left(W_i \cup \bigcup_{k\in\mathcal{C}(i)} \bigcup_{j\in\mathcal{S}_k} W_j\right) \setminus Z_i \\
&= \left(\left(W_i \cup \bigcup_{k\in\mathcal{C}(i)} Z_k\right) \setminus Z_i\right) \cup \bigcup_{k\in\mathcal{C}(i)} \left(\left(\bigcup_{j\in\mathcal{S}_k} W_j\right) \setminus Z_k\right) \\
&= ((U_i \cup Z_i) \setminus Z_i) \cup \bigcup_{k\in\mathcal{C}(i)} \left(\left(\bigcup_{j\in\mathcal{S}_k} W_j\right) \setminus Z_k\right) \quad \text{(by (6))} \\
&= U_i \cup \bigcup_{k\in\mathcal{C}(i)} \left(\left(\bigcup_{j\in\mathcal{S}_k} W_j\right) \setminus Z_k\right).
\end{aligned} \tag{33}$$

Thus, this completes the induction that (15) holds for each node i. □

B.2 Proof of Lemma 2

By Lemma 1, the following holds

$$\Phi_r = \min_{\left(\bigcup_{j\in\mathcal{S}_r} W_j\right)\setminus Z_r} \sum_{j\in\mathcal{S}_r} f_j(W_j). \tag{34}$$

Because the subtree rooted at r is the whole tree, we have $\mathcal{S}_r = \mathcal{V}$, and because r is the root, $Z_r = \emptyset$. Hence

$$\Phi_r = \min_{\left(\bigcup_{j\in\mathcal{V}} W_j\right)} \sum_{j\in\mathcal{V}} f_j(W_j) = \min_{\mathcal{W}} \sum_{i\in\mathcal{V}} f_i(W_i). \tag{35}$$

Thus, this completes the proof of Lemma 2. □

B.3 Proof of Lemma 3

At the root, recall the definition of $U_r^\star$ in (11) and the definition of Φ_r in (10), Lemma 2, imply that $U_r^\star$ is a minimizer in the root elimination step. Since $Z_r = \emptyset$ and based on (6), we have

$$W_r^\star \cup \bigcup_{k\in\mathcal{C}(r)} Z_k^\star = U_r^\star \cup Z_r = U_r^\star. \tag{36}$$

This recovers the root local block $W_r^\star$ and the child separator values $\{Z_k^\star\}_{k\in\mathcal{C}(r)}$ from $U_r^\star$. By optimality of $U_r^\star$, the resulting $W_r^\star$ and $\{Z_k^\star\}_{k\in\mathcal{C}(r)}$ are optimal for the joint problem (5), and the child separator values are passed to the children.

The root case is the base of a recursion along a root-to-leaf phase. Let i be a non-root node, and assume that its parent has already provided the separator values $Z_i^\star$, which are optimal for the joint problem (5). Node i then evaluates $U_i^\star$ in (13). By definition, $U_i^\star$ is the optimal for (5) once $Z_i^\star$ is optimal. In the same way as for the root, (6) at the optimum yields (14), namely

$$W_i^\star \cup \bigcup_{k\in\mathcal{C}(i)} Z_k^\star = U_i^\star \cup Z_i^\star. \tag{37}$$

This recovers the local block $W_i^\star$ and the child separator values $\{Z_k^\star\}_{k\in\mathcal{C}(i)}$ from $U_i^\star$ and $Z_i^\star$. Similarly, by optimality of $U_i^\star$ and $Z_i^\star$, the resulting local block $W_i^\star$ and child separator values $\{Z_k^\star\}_{k\in\mathcal{C}(i)}$ are consistent with a global minimizer of in (5), and the child separator values are passed to $\mathcal{C}(i)$.

By induction along the root-to-leaf order, the base and recursive steps together determine a local block $W_i^\star$ for every $i \in \mathcal{V}$ and hence reconstruct a complete and optimal assignment $\{W_i^\star\}_{i\in\mathcal{V}}$ over $\mathcal{W}$. Thus, this completes the proof of Lemma 3. □

B.4 Proof of Theorem 1

By Lemma 2, the leaf-to-root phase produces the exact root value

$$\Phi_r = \min_{\mathcal{W}} \sum_{i \in \mathcal{V}} f_i(W_i). \tag{38}$$

Thus the reduced problem at root has exactly the same optimal value as the original problem in (5).

By Lemma 3, the root-to-leaf phase reconstructs a complete and optimal assignment over $\mathcal{W}$. Equivalently, the reconstructed assignment $\{W_i^\star\}_{i \in \mathcal{V}}$ attains the root value Φ_r in (38). By Lemma 2, this root value equals the minimum objective value of (5). Therefore, NSMP is exact in both value and solution: it recovers a feasible global assignment and certifies its global optimality for the original problem. Thus, this completes the proof of Theorem 1. □

C Tree invariance, tree selection, and communication costs

This appendix provides further analysis of tree selection and communication in NSMP. We prove Proposition 1, develop a tree-selection criterion, and extend Section 4.3 with a communication comparison against representative distributed optimization algorithms.

C.1 Cyclic overlap graphs and separator variables

Proof of Proposition 1. Fix any spanning tree $\mathcal{T}$ and root r. Selecting $\mathcal{T}$ changes no term in (5), since overlap edges are not loss terms. By (7), Z_i retains every coordinate shared by $\mathcal{S}_i$ and its complement. Thus, no coordinate needed outside a subtree is eliminated within it, and the argument of Lemma 1 applies to this tree:

$$\Phi_i(Z_i) = \min_{\left(\bigcup_{j \in \mathcal{S}_i} W_j\right) \setminus Z_i} \sum_{j \in \mathcal{S}_i} f_j(W_j). \tag{39}$$

At the root, $\mathcal{S}_r = \mathcal{V}$ and $Z_r = \emptyset$. Applying the recovery rule (13) and Lemma 3 therefore gives

$$\sum_{i \in \mathcal{V}} f_i(W_i^\star) = \Phi_r = \min_{\mathcal{W}} \sum_{i \in \mathcal{V}} f_i(W_i). \tag{40}$$

The right-hand side is independent of $\mathcal{T}$ and r. Hence, every spanning tree yields the same optimal value, and each recovered assignment is globally optimal.

Finally, cycles in the nerve skeleton $\mathcal{G}$ do not affect this conclusion: NSMP operates on a spanning tree of $\mathcal{G}$, rather than directly on the cyclic graph, and the argument above applies to any such tree. □

C.2 Tree-selection criterion

In this part, we formulate the choice of spanning tree as an optimization problem, following the communication analysis in Section 4.3. For a fixed spanning tree, we first obtain the total computation and transmission time. We then minimize this time over candidate spanning trees and roots.

Below, we take a quadratic objective as an example and give the computation time and the transmission time of the message for each client. Here, the quadrative objective means that each local objective $f_i(W_i)$ in (5) is quadratic in W_i. The computational time for each client i is the reduction operation of the partial minimization in (9). For a candidate rooted tree, let $u_i = |U_i|$ and $z_i = |Z_i|$, where U_i and Z_i are defined in (7). Then , performing the minimization in (9) at client i requires $O(u_i^3 + u_i^2 z_i + u_i z_i^2)$ arithmetic operations. If P_i denotes its effective processing rate in operations per unit time, the elimination time scales as

$$t_i^{\mathrm{comp}} = O\left(\frac{u_i^3 + u_i^2 z_i + u_i z_i^2}{P_i}\right). \tag{41}$$

Before computing its own reduced objective, client i must receive $\Phi_k(Z_k)$ from each child $k \in \mathcal{C}(i)$ to form the objective in (8). We next estimate the time required to transmit $\Phi_k(Z_k)$ from client k to

client i. As shown in Section 4.3, this quadratic message contains $s_k = z_k(z_k+1)/2 + z_k + 1$ real scalars. With an effective link rate R_{ki} in scalars per unit time, its transmission time is estimated by

$$t_{k\to i}^{\mathrm{comm}} \approx \frac{s_k}{R_{ki}} = \frac{z_k(z_k+1)/2 + z_k + 1}{R_{ki}}. \tag{42}$$

We next combine the computation and transmission times in (41) and (42) to estimate the total completion time of the leaf-to-root phase. Since client i performs its minimization after receiving all child messages, its upward completion time for client i is given by

$$T_i^{\uparrow} = \max_{k\in\mathcal{C}(i)} \left(T_k^{\uparrow} + t_{k\to i}^{\mathrm{comm}}\right) + t_i^{\mathrm{comp}}, \tag{43}$$

with $T_i^{\uparrow} = t_i^{\mathrm{comp}}$ at a leaf. Compard to the upward time in (43), the root-to-leaf phase transmits only the variable values $Z_i^{\star}$; here, we treat its transmission and computation times as negligible, so that the total execution time is dominated by the leaf-to-root phase. We can then select the tree and root by minimizing the estimated completion time at the root:

$$(\mathcal{T}^{\star}, r^{\star}) \in \arg\min_{\mathcal{T}\in\mathfrak{T}(\mathcal{G}),\, r\in\mathcal{V}} T_r^{\uparrow}(\mathcal{T}, r), \tag{44}$$

where $\mathfrak{T}(\mathcal{G})$ is the set of spanning trees of the nerve skeleton. The objective in (44) accounts for tree depth, separator and eliminated-variable dimensions, client computation, and link rates. In practical, for small graphs, we can exhaustively evaluate all spanning trees. For larger graphs, we can initialize a latency-aware shallow tree and improve it through rerooting, edge swaps, or local reparenting.

C.3 Communication comparison with ADMM

In this part, we compare the communication costs of NSMP and ADMM, a representative distributed optimization algorithm, using quadratic objectives as a concrete example.

By (18), the total communication cost of the proposed NSMP is

$$C_{\mathrm{NSMP}} = O(Mz_{\max}^2), \tag{45}$$

where $z_{\max} = \max_{i\neq r} |Z_i|$. NSMP sends $2(M-1)$ directed messages in one leaf-to-root and one root-to-leaf pass; a tree of height h requires $2h$ synchronous hop-level rounds.

For peer-to-peer edge-consensus ADMM [27], let d_e denote the dimension of the shared parameter on edge e. In each iteration, each endpoint sends one d_e-dimensional shifted local copy to the other endpoint, while the consensus and dual updates are computed locally. Thus, the communication costs per iteration and over K iterations are

$$C_{\mathrm{ADMM/iter}} = 2\sum_e d_e, \qquad C_{\mathrm{ADMM,total}} = 2K\sum_e d_e. \tag{46}$$

On a tree or a communication graph with $O(M)$ edges, this gives

$$C_{\mathrm{ADMM,total}} = O(KMd_{\max}), \tag{47}$$

where $d_{\max} = \max_e d_e$. ADMM therefore exchanges messages linear in the shared-parameter dimension repeatedly until convergence, whereas NSMP sends each quadratic reduced objective only once and then returns the recovered variable values.

Experiments on the six-client least-squares problem in Section 5.1, as well as on the EPFL and SoCal state-estimation datasets, show that NSMP achieves substantially lower communication overhead than ADMM and Gossip under the reported settings; see Appendices G.1 and G.3.

D Discussion and extensions

This section discusses two natural extensions of NSMP. The first extends the framework beyond minimization to a broader class of operators. The second addresses practical settings where reduced messages are unavailable in closed form and must be approximated numerically.

D.1 Extension to more general operators

In the basic setting, the operator of interest in our work is the minimization map

$$\Gamma[f(\cdot)] = \min f(\cdot), \tag{48}$$

so that the leaf-to-root phase of NSMP computes $\Gamma[f(\cdot)]$ by recursively eliminating local variables and evaluating the resulting reduced objective at the root. This viewpoint naturally raises the question of whether the same recursive architecture can be used for operators other than minimization.

Let $\Gamma(\cdot)$ be a general operator mapping a function to a real number. To extend NSMP beyond the optimization setting, we require Γ to be compatible with local elimination. Specifically, in parallel with (9), suppose that at node i an intermediate objective $\tilde{f}_i(U_i, Z_i)$ is reduced to the message $\Phi_i(Z_i)$ by eliminating the local variables U_i while retaining the separator variables Z_i, namely,

$$\Phi_i(Z_i) = \Gamma_{U_i}[\tilde{f}_i(U_i, Z_i)]. \tag{49}$$

We assume that this local reduction preserves the value of the global operator in the sense that

$$\Gamma[\tilde{f}_i(U_i, Z_i) + g(Z_i)] = \Gamma_{Z_i}[\Phi_i(Z_i) + g(Z_i)]. \tag{50}$$

where $g(Z_i)$ is any function of the separator variables Z_i that does not depend on U_i, provided the operator evaluations are well-defined. Condition (50) states that Γ is invariant under the local reduction step induced by message passing. Several operators satisfy this property, including

- $\Gamma[f(W)] = \max(f(W))$
- $\Gamma[f(W)] = \dfrac{1}{\mathrm{vol}(\mathcal{D}_W)} \int_{\mathcal{D}_W} f(W)\, dW,$

where, in the integration example, f is integrable and $\mathcal{D}_W$ is a Cartesian product of coordinate domains with finite, positive volume. Each partial operator Γ_{U_i} uses the same normalization over its corresponding domain $\mathcal{D}_{U_i}$. Condition (50) provides a sufficient criterion for extending the leaf-to-root computation beyond minimization. The following proposition formalizes this extension.

Proposition 2 (Operator compatibility)**.** *If the condition in* (50) *holds for* $\Gamma(\cdot)$*, then the proposed message-passing framework of NSMP extends directly to the computation of* $\Gamma[f(\cdot)]$ *by replacing each local minimization step with the corresponding local application of* $\Gamma(\cdot)$*.*

Proof. We consider the same leaf-to-root elimination order as in the minimization case. Specifically, at node i, the local objective $\tilde{f}_i(U_i, Z_i)$ is reduced to $\Phi_i(Z_i) = \Gamma_{U_i}[\tilde{f}_i(U_i, Z_i)]$, in parallel with (9). By the assumption in (50), replacing $\tilde{f}_i(U_i, Z_i)$ with the message $\Phi_i(Z_i)$ leaves the value of $\Gamma(\cdot)$ on the subtree objective rooted at node i unchanged. Applying this argument inductively from the leaves to the root shows that the final aggregate equals $\Gamma[f(\cdot)]$ for the original global objective. The operator compatibility result thus establishes the validity of the leaf-to-root computation. □

D.2 Approximate message representations

Recall that NSMP in Algorithm 1 transmits the reduced objective $\Phi_i(Z_i)$ defined in (9). When this function is non-convex and has no tractable representation, the leaf-to-root phase can send a fitted approximation instead. In this subsection, we describe how to construct and use these approximate messages, then bound the error in the root optimal value.

D.2.1 Message approximation

For non-convex objectives without closed-form reduced messages, we approximate the partial minimum $\Phi_i(Z_i)$ in (9) and fit its representation in two stages. First, client i fixes Z_i and approximately minimizes the aggregated objective over U_i:

$$\widetilde{\Phi}_i(Z_i) = \mathrm{ApproxMin}_{U_i}\left[f_i(W_i) + \sum_{k \in \mathcal{C}(i)} \widehat{\Phi}_k(Z_k) \right], \tag{51}$$

where $\widehat{\Phi}_k(Z_k)$ denotes the approximation of the reduced objective $\Phi_k(Z_k)$ received from child k. This minimization can be performed using gradient-based methods and does not require finding a

global minimum. Repeating it at different sampled values $Z_i^{(s)}$ produces a local training dataset $\{(Z_i^{(s)}, \widetilde{\Phi}_i(Z_i^{(s)}))\}_s$.

Since MLPs provide a flexible model class for approximating continuous reduced messages on compact domains [28], then client i uses this dataset to train an MLP $\widehat{\Phi}_i(Z_i;\theta_i)$ to approximate the function $\widetilde{\Phi}_i(Z_i)$ in (51). The parameters are fitted by minimizing

$$\min_{\theta_i} \sum_s \left| \widehat{\Phi}_i(Z_i^{(s)};\theta_i) - \widetilde{\Phi}_i(Z_i^{(s)}) \right|^2. \tag{52}$$

After training, we write $\widehat{\Phi}_i(Z_i;\theta_i)$ as $\widehat{\Phi}_i(Z_i)$ for brevity. With the MLP architecture agreed in advance, this function is determined by θ_i. Thus, client i only needs to send θ_i to its parent in the existing leaf-to-root pass. The sampled pairs and raw data remain local, and no additional communication rounds are required.

It is worth noting that the fitting procedure in (52) relies on samples obtained by evaluating (51). When sampling a high-dimensional separator domain is impractical, we can instead construct a local quadratic message by approximating the aggregated objective in (8) before eliminating U_i:

$$F_i(U_i, Z_i) \xrightarrow{\text{quadratic approximation}} Q_i(U_i, Z_i) \xrightarrow{\text{eliminate } U_i} \widehat{\Phi}_i(Z_i) = \min_{U_i} Q_i(U_i, Z_i). \tag{53}$$

Here, $F_i(U_i, Z_i) = f_i(W_i) + \sum_{k\in\mathcal{C}(i)} \widehat{\Phi}_k(Z_k)$ follows the aggregation in (8), with the child messages replaced by their approximations. We first use gradient-based optimization over (U_i, Z_i) to obtain a local reference point (U_i^0, Z_i^0). Following the procedure in (53), we construct a local quadratic approximation $Q_i(U_i, Z_i)$ of $F_i(U_i, Z_i)$ around this reference point, and analytically minimize it over U_i to obtain the quadratic message

$$\widehat{\Phi}_i(Z_i) = c_i + g_i^\top Z_i + \tfrac{1}{2} Z_i^\top S_i Z_i, \tag{54}$$

Equation (54) is a quadratic function of Z_i, where c_i is the constant term, g_i is the linear coefficient vector, and S_i is the quadratic coefficient matrix. Client i sends these coefficients to its parent to represent the approximate message. This construction avoids explicit sampling of the separator domain and fits directly into the existing leaf-to-root message-passing procedure.

Thus, when exact minimization of the aggregated objective in (8) over U_i is impractical, client i can instead transmit to its parent an approximate message $\widehat{\Phi}_i(Z_i)$ obtained either by MLP fitting in (52) or by the quadratic construction in (54), which makes NSMP applicable without requiring the exact reduced message $\Phi_i(Z_i)$ defined in (9).

D.2.2 Error bounds

We now bound the difference between the root objective value computed using approximate messages $\widehat{\Phi}_i$ and exact messages Φ_i. The message error is defined as

$$E_i = \sup_{Z_i} \left| \widehat{\Phi}_i(Z_i) - \Phi_i(Z_i) \right|, \tag{55}$$

where $\widehat{\Phi}_i(Z_i)$ is the approximate message obtained through MLP fitting in (52) or quadratic approximation in (54), and $\Phi_i(Z_i)$ is the exact reduced objective.

Assume exact and approximate reductions use the same compatible domains and have finite attained minima. Define the local approximation error as

$$\epsilon_i := \sup_{Z_i} \left| \widehat{\Phi}_i(Z_i) - \min_{U_i} \left[f_i(W_i) + \sum_{k\in\mathcal{C}(i)} \widehat{\Phi}_k(Z_k) \right] \right|. \tag{56}$$

Here, the received child messages $\widehat{\Phi}_k$ are held fixed, so ϵ_i measures only the error introduced at client i. In contrast, E_i in (55) also includes errors inherited from its descendants. We assume ϵ_i is

finite; at the root, it is the numerical minimization error. Using (9) and (56), we obtain

$$\begin{aligned} E_i &= \sup_{Z_i} \left| \widehat{\Phi}_i(Z_i) - \Phi_i(Z_i) \right| \\ &\leq \epsilon_i + \sup_{Z_i} \left| \min_{U_i} \left[f_i(W_i) + \sum_{k \in \mathcal{C}(i)} \widehat{\Phi}_k(Z_k) \right] - \Phi_i(Z_i) \right| \\ &\leq \epsilon_i + \sup_{U_i, Z_i} \left| \sum_{k \in \mathcal{C}(i)} \left(\widehat{\Phi}_k(Z_k) - \Phi_k(Z_k) \right) \right| \\ &\leq \epsilon_i + \sum_{k \in \mathcal{C}(i)} \sup_{Z_k} \left| \widehat{\Phi}_k(Z_k) - \Phi_k(Z_k) \right| \\ &= \epsilon_i + \sum_{k \in \mathcal{C}(i)} E_k. \end{aligned} \tag{57}$$

Applying this inequality recursively from the leaves to the root yields

$$E_r = |\widehat{\Phi}_r - \Phi_r| \leq \sum_{i \in \mathcal{V}} \epsilon_i. \tag{58}$$

Thus, the root optimal-value error is bounded by the sum of the local errors. When the local approximations in (56) are sufficiently accurate to keep this sum small, NSMP still provides an accurate estimate of the global optimal value. In Appendix G.4, we evaluate the proposed NSMP with approximate messages on PEMS-BAY. The results show competitive predictive performance with low communication overhead, supporting its practical feasibility with approximate messages.

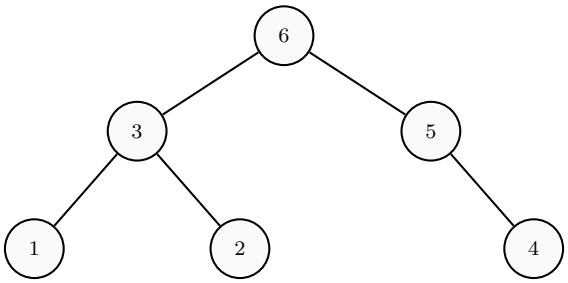


Figure 5: Spanning-tree structure used in the simulation.

E Synthetic six-client least-squares regression

In this part, we illustrate the details of the synthetic experiment in Section 5. In particular, Figure 5 depicts the spanning tree generated from the coordinate overlaps in Table 1. All experiments were conducted on a workstation with an Intel i9-14900K CPU, 32 GB RAM, and an NVIDIA GeForce RTX 4090 GPU. To illustrate how NSMP in Algorithm 1 behaves, we trace nodes 1, 2, and 3 explicitly.

For **node** 1, the local objective of (20) is written as

$$f_1(\mathbf{w}_1) = \frac{1}{2} \left\| \mathbf{y}_1 - \mathbf{X}_1 \mathbf{w}_1 \right\|_2^2, \tag{59}$$

where $\mathbf{y}_1 \in \mathbb{R}^{N_1}$, $\mathbf{X}_1 \in \mathbb{R}^{N_1 \times 2}$. With the partition $U_1 = \{a_1\}$ and $Z_1 = \{a_2\}$, the expression in (9) at node 1 becomes

$$\Phi_1(a_2) = \min_{a_1} f_1(a_1, a_2). \tag{60}$$

For convenience, let $\mathbf{X}_1^{(a_1)}$ and $\mathbf{X}_1^{(a_2)}$ denote the columns of $\mathbf{X}_1$ for a_1 and a_2. To calculate the formulation in (60), we rewrite the local objective in (59) by separating the eliminated variable a_1 from the separator variable a_2,

$$f_1(a_1, a_2) = \frac{1}{2} \left\| \mathbf{y}_1 - \mathbf{X}_1^{(a_1)} a_1 - \mathbf{X}_1^{(a_2)} a_2 \right\|_2^2, \tag{61}$$

and then project out the a_1 to obtain the explicit formulation of $\Phi_1(a_2)$ in (60)

$$\Phi_1(a_2) = \min_{a_1} \frac{1}{2} \left\| \mathbf{y}_1 - \mathbf{X}_1^{(a_1)} a_1 - \mathbf{X}_1^{(a_2)} a_2 \right\|_2^2 \tag{62}$$

$$= \frac{1}{2} \left\| \mathbf{P}_1^{\perp} \left(\mathbf{y}_1 - \mathbf{X}_1^{(a_2)} a_2 \right) \right\|_2^2, \quad \mathbf{P}_1^{\perp} := \mathbf{I} - \mathbf{X}_1^{(a_1)} \left((\mathbf{X}_1^{(a_1)})^{\mathsf{T}} \mathbf{X}_1^{(a_1)} \right)^{-1} (\mathbf{X}_1^{(a_1)})^{\mathsf{T}}. \tag{63}$$

Therefore, $\Phi_1(a_2)$ is a scalar quadratic function:

$$\Phi_1(a_2) = \frac{1}{2} \left(\alpha_1 a_2^2 - 2\beta_1 a_2 + \gamma_1 \right), \tag{64}$$

where $\alpha_1 = (\mathbf{X}_1^{(a_2)})^{\mathsf{T}} \mathbf{P}_1^{\perp} \mathbf{X}_1^{(a_2)}$, $\beta_1 = (\mathbf{X}_1^{(a_2)})^{\mathsf{T}} \mathbf{P}_1^{\perp} \mathbf{y}_1$, $\gamma_1 = \mathbf{y}_1^{\mathsf{T}} \mathbf{P}_1^{\perp} \mathbf{y}_1$. This quadratic message $\Phi_1(a_2)$ is then sent to node 1's parent, namely node 3. Since the message is fully represented by the three coefficients $(\alpha_1, \beta_1, \gamma_1)$, the communication overhead is very low.

As for **node** 2 in Figure 5, the local objective is

$$f_2(a_3, a_4) = \frac{1}{2} \left\| \mathbf{y}_2 - \mathbf{X}_2^{(a_3)} a_3 - \mathbf{X}_2^{(a_4)} a_4 \right\|_2^2. \tag{65}$$

With the partition $U_2 = \{a_3\}$ and $Z_2 = \{a_4\}$, the expression in (9) at node 2 becomes

$$\Phi_2(a_4) = \min_{a_3} f_2(a_3, a_4). \tag{66}$$

Following the same projection step as for node 1, we obtain

$$\Phi_2(a_4) = \min_{a_3} \frac{1}{2} \left\| \mathbf{y}_2 - \mathbf{X}_2^{(a_3)} a_3 - \mathbf{X}_2^{(a_4)} a_4 \right\|_2^2 \tag{67}$$

$$= \frac{1}{2} \left\| \mathbf{P}_2^{\perp} \left(\mathbf{y}_2 - \mathbf{X}_2^{(a_4)} a_4 \right) \right\|_2^2, \quad \mathbf{P}_2^{\perp} := \mathbf{I} - \mathbf{X}_2^{(a_3)} \left((\mathbf{X}_2^{(a_3)})^{\mathsf{T}} \mathbf{X}_2^{(a_3)} \right)^{-1} (\mathbf{X}_2^{(a_3)})^{\mathsf{T}}. \tag{68}$$

Therefore, $\Phi_2(a_4)$ is also a scalar quadratic function:

$$\Phi_2(a_4) = \frac{1}{2} \left(\alpha_2 a_4^2 - 2\beta_2 a_4 + \gamma_2 \right), \tag{69}$$

where $\alpha_2 = (\mathbf{X}_2^{(a_4)})^{\mathsf{T}} \mathbf{P}_2^{\perp} \mathbf{X}_2^{(a_4)}$, $\beta_2 = (\mathbf{X}_2^{(a_4)})^{\mathsf{T}} \mathbf{P}_2^{\perp} \mathbf{y}_2$, $\gamma_2 = \mathbf{y}_2^{\mathsf{T}} \mathbf{P}_2^{\perp} \mathbf{y}_2$. This quadratic message $\Phi_2(a_4)$ is then sent to node 2's parent, namely node 3, and is again represented by three coefficient.

At **node** 3, after reciving the messges from node 1 and node 2, i.e., $\Phi_1(a_2)$ and $\Phi_2(a_4)$, we aggregate $f_3 + \Phi_1(a_2) + \Phi_2(a_4)$. Following the decomposition rule in (6), the coordinates visible at node 3 are split into an eliminable block and a separator block:

$$W_3 \cup Z_1 \cup Z_2 = U_3 \cup Z_3, \qquad U_3 = \{a_2, a_4, a_5\},\ Z_3 = \{a_6\}. \tag{70}$$

That is, (a_2, a_4, a_5) are eliminated locally at node 3, while a_6 is retained as the argument passed upward to its parent:

$$\Phi_3(a_6) = \min_{a_2, a_4, a_5} \Big[f_3(a_2, a_4, a_5, a_6) + \Phi_1(a_2) + \Phi_2(a_4) \Big]. \tag{71}$$

One can find that $\Phi_3(a_6)$ is also a scalar quadratic function of a_6, and its coefficients can be obtained by applying the same procedure used for $\Phi_1(a_2)$ and $\Phi_2(a_4)$; we omit the explicit expansion for brevity. This message $\Phi_3(a_6)$ is then sent to node 3's parent, namely node 6.

In parallel, nodes 4 and 5 perform the same elimination pattern on their eliminated/separator variable splits, and the leaf-to-root phase is completed once all messages reach node 6. For clarity, the message flow in leaf-to-root pahse is:

$$\Phi_1(a_2),\ \Phi_2(a_4) \to 3, \qquad \Phi_4(a_8) \to 5, \qquad \Phi_3(a_6),\ \Phi_5(a_{10}) \to 6.$$

In the root-to-leaf phase, values are passed from parent nodes to child nodes, and each node then recovers its own variables as follows:

- At the root, node 6 reconstructs $W_6^\star = \{a_6^\star, a_{10}^\star, a_{11}^\star\}$, keeps its local block, and passes the separator values $(a_6^\star, a_{10}^\star)$ to children $(3, 5)$.
- At internal nodes, node 3 uses $a_6^\star$ to reconstruct $W_3^\star = \{a_2^\star, a_4^\star, a_5^\star, a_6^\star\}$ according to (71) and then passes $(a_2^\star, a_4^\star)$ to children $(1, 2)$, while node 5 uses $a_{10}^\star$ to reconstruct $W_5^\star = \{a_8^\star, a_9^\star, a_{10}^\star\}$ and then passes $a_8^\star$ to child 4.
- At leaves, node 1 reconstructs $W_1^\star = \{a_1^\star, a_2^\star\}$ from $a_2^\star$ according to (60), node 2 reconstructs $W_2^\star = \{a_3^\star, a_4^\star\}$ from $a_4^\star$ according to (66), and node 4 reconstructs $W_4^\star = \{a_7^\star, a_8^\star\}$ from $a_8^\star$ similarly.

In summary, the root-to-leaf phase propagates only separator values downward and reconstructs each local block $W_i^\star$ in a top-down order.

F Detailed applications of NSMP

This appendix gives more details for the brief application discussion in Section 5.3. We now write out the three examples in a concrete form and show, for each one, what is shared, what is local, and how neighboring nodes are coupled on the graph.

We use one general model template for all three examples below. At each node i, we split parameters as $W_i = (u_i, v_i)$: $u_i \in \mathbb{R}^{d_s}$ is the shared part that should change smoothly across a graph $\mathcal{G}_{\mathrm{sm}} = (\mathcal{V}, \mathcal{E}_{\mathrm{sm}})$, and $v_i \in \mathbb{R}^{d_p}$ is the node-specific local part. Each node contributes a local loss $f_i(u_i, v_i)$. The full objective is

$$\min_{\{u_i, v_i\}_{i\in\mathcal{V}}} \sum_{i\in\mathcal{V}} f_i(u_i, v_i) \;+\; \frac{\lambda}{2} \sum_{(i,j)\in\mathcal{E}_{\mathrm{sm}}} w_{ij}\|u_i - u_j\|_2^2 \;+\; \frac{\mu_u}{2} \sum_{i\in\mathcal{V}} \|u_i\|_2^2 \;+\; \frac{\mu_v}{2} \sum_{i\in\mathcal{V}} \|v_i\|_2^2. \quad (72)$$

In this objective, the second term encourages neighboring nodes to have similar shared parts u_i, and the last two terms are standard ℓ_2 regularization. This is a generic model and will be reused in all three examples below. Across examples, we only change the meaning of (u_i, v_i), the local loss f_i, and the graph $\mathcal{G}_{\mathrm{sm}}$. Unless stated otherwise, $\mathcal{E}_{\mathrm{sm}}$ does not need to match the overlap edge set $\mathcal{E}$ in Section 2; it can be a geographic, referral, mobility, or interference graph.

F.1 Chain-store demand prediction with shared regional effects

The first example is a retail demand prediction task. Let i index stores. In practice, nearby stores often share similar regional patterns, such as weather effects, holidays, promotions, and district-level traffic. At the same time, each store still has its own characteristics, including store format, inventory, staffing, and daily operations.

Let $x_{i,t}^{(s)} \in \mathbb{R}^{d_s}$ denote shared/regional features at time t , and let $x_{i,t}^{(p)} \in \mathbb{R}^{d_p}$ denote store-specific local features. Let $y_{i,t} \in \mathbb{R}$ be realized sales. The linear predictor at node i is

$$\hat{y}_{i,t}(u_i, v_i) = u_i^\top x_{i,t}^{(s)} + v_i^\top x_{i,t}^{(p)}. \quad (73)$$

where u_i denotes the shared regional component and v_i denotes the store-specific local component. Node i observes a training index set $\mathcal{T}_i$. The local empirical risk is the sum of squared residuals between predictions and observed sales:

$$f_i(u_i, v_i) = \sum_{t\in\mathcal{T}_i} \left(\hat{y}_{i,t}(u_i, v_i) - y_{i,t}\right)^2 = \sum_{t\in\mathcal{T}_i} \left(u_i^\top x_{i,t}^{(s)} + v_i^\top x_{i,t}^{(p)} - y_{i,t}\right)^2. \quad (74)$$

Substituting (74) into (72) gives the final joint optimization objective, i.e., a concrete instance of the generalized federated optimization problem. It is worth noting that for each node i, the shared regional component u_i is coupled with its neighbors through the setting of $\mathcal{E}_{\mathrm{sm}}$. This resulting problem can then be solved by the proposed NSMP framework.

F.2 Hospital-level risk prediction with shared regional patterns

The second example considers hospital-level risk prediction. As in the retail case, we separate shared regional effects from institution-specific effects. Let i index hospitals. Nearby hospitals often share

regional signals, such as climate, demographics, and public-health conditions, while each hospital also has local factors, such as capacity, specialization, and patient mix.

Let $x_{i,t}^{(s)} \in \mathbb{R}^{d_s}$ denote shared/regional features at time t, and let $x_{i,t}^{(p)} \in \mathbb{R}^{d_p}$ denote hospital-specific local features. Let $y_{i,t} \in \mathbb{R}$ be a continuous risk-related target (e.g., normalized admissions or a risk score). The predictor at node i is

$$\hat{y}_{i,t}(u_i, v_i) = u_i^\top x_{i,t}^{(s)} + v_i^\top x_{i,t}^{(p)}. \tag{75}$$

where u_i denotes the shared regional component and v_i denotes the hospital-specific local component. Node i observes a training index set $\mathcal{T}_i$. The local empirical risk is

$$f_i(u_i, v_i) = \sum_{t \in \mathcal{T}_i} \left(\hat{y}_{i,t}(u_i, v_i) - y_{i,t}\right)^2. \tag{76}$$

Following the same template, substituting (76) into (72) gives the joint optimization objective for this healthcare setting. In this objective, the shared component u_i is coupled across neighboring hospitals through $\mathcal{E}_{\mathrm{sm}}$, and the resulting problem can be solved by the proposed NSMP framework.

F.3 Cell-tower traffic forecasting with shared spatial patterns

The third example focuses on cell-tower traffic forecasting. In a similar spirit, we decompose each station into a shared spatial component and a station-specific local component. Nearby cells often share spatial demand patterns due to commuting, daily cycles, weather, and events, while each station also has local factors such as hardware configuration, coverage, and subscriber mix.

Specifically, let i index base stations, and let $X_{i,t}$ collect local inputs such as historical load, time-of-day, weather, and event indicators. The predictor at node i is

$$\hat{y}_{i,t}(u_i, v_i) = g(X_{i,t}; u_i, v_i), \tag{77}$$

where u_i denotes the shared spatial component, v_i denotes the base-station-specific local component, and $y_{i,t} \in \mathbb{R}$ denotes the target traffic quantity. Unlike the previous two examples, the predictor g in (77) is not restricted to be linear and can represent a nonlinear model. The local empirical risk is

$$f_i(u_i, v_i) = \sum_{t \in \mathcal{T}_i} \ell\big(\hat{y}_{i,t}(u_i, v_i),\, y_{i,t}\big) = \sum_{t \in \mathcal{T}_i} \ell\big(g(X_{i,t}; u_i, v_i),\, y_{i,t}\big). \tag{78}$$

Following the same template, substituting (78) into (72) yields the joint objective for this wireless setting, which can also be solved by NSMP.

G Additional experimental details

This section provides experimental protocols and supplementary results for the numerical evaluation in Section 5.

We first detail the six-client and METR-LA experiments, including baseline configurations and communication costs. We then present additional evaluations on real-world distributed state-estimation datasets and in non-convex prediction with approximate messages.

Our code is available at `https://github.com/lan-weizhang/NSMP`. All experiments use the workstation configuration described in Appendix E.

G.1 Six-client distributed baselines

Section 5.1 shows that NSMP matches centralized least squares on this instance, as in Figure 4. In this part, using the same experimental setup, we compare the communication cost of NSMP, Gossip [25, 26], and ADMM [27]. The results are summarized in Table 3.

For NSMP, the spanning tree in Figure 5 has $M = 6$ clients and height $h = 2$. Following Section 4.3 and Appendix C, the leaf-to-root and root-to-leaf passes each require h synchronous hop-level rounds, giving $2h = 4$ communication rounds in total. Each of the $M - 1$ tree edges carries one upward and one downward message, yielding $2(M - 1) = 10$ directed messages. Since each

Table 3: Six-client communication under the stated evaluation rules. Objective-gap checkpoints are evaluated against the centralized reference; residual stopping does not require that reference.

Evaluation rule	Method	Rounds/iterations	Messages	Bytes
Exact completion	NSMP	4	10	160
Relative gap $\leq 10^{-15}$	Gossip	23	230	1,840
Residual tolerances 10^{-8}	ADMM	30	300	2,400

separator variable is scalar, (18) gives $4(M-1) = 20$ transmitted scalars. Each scalar is encoded in Float64 (eight bytes), yielding a total payload of 160 bytes, as shown in Table 3.

For Gossip, we use a variant of the distributed first-order method in (25) adapted to overlapping local coordinates. The stopping criterion is a relative objective gap of at most 10^{-15} with respect to the centralized optimum, which is first reached at iteration 23. The ADMM baseline follows the framework reviewed in Appendix A, using a peer-to-peer edge-consensus formulation whose communication cost is analyzed in Appendix C. The stopping criterion requires both primal and dual residuals to satisfy their standard thresholds, with absolute and relative tolerances set to 10^{-8}. It stops after 30 iterations, with relative parameter error 1.69×10^{-9}. As shown in Table 3, under these evaluation rules, NSMP obtains the exact solution in four communication rounds, with 11.5-fold and 15-fold reductions in transmitted bytes compared with Gossip and ADMM, respectively.

G.2 METR-LA prediction

This subsection details the METR-LA prediction experiments in Section 5.2, which compare the predictive performance and communication cost of NSMP with the conventional federated learning methods FedAvg [6] and FedRep [11].

In this experiment, we consider next-step traffic-speed prediction on METR-LA, using the data distributed with the DCRNN benchmark [17]. Each of the 206 sensors is treated as one client. Each client predicts the next speed value from its previous 12 speed measurements. Each sensor's time series is divided in time order: the first 70% is used for training, the next 10% for validation, and the final 20% for testing. As shown in Table 4, to evaluate the effectiveness of federated learning with limited training data, only 1% or 0.5% of each client's training data is available. All methods use the same data splits and training samples. Zeros are treated as missing, and the data are normalized using only the available training data.

Table 4: METR-LA test RMSE and numerical communication. The byte count is the same in both retained-data settings.

Method	RMSE (1%)	RMSE (0.5%)	Bytes
Local	4.2548	4.5508	0
FedAvg	4.1512	4.1627	12,854,400
FedRep	4.1595	4.1805	11,865,600
NSMP	**4.0742**	**4.1106**	**193,520**

Each client uses a 13-parameter linear model: a weighted sum of its previous 12 speed measurements plus a bias. The Local baseline in Table 4 learns all 13 parameters independently at each client. FedAvg shares all 13 parameters, so all clients use the same global model. In each round, clients take one full-batch gradient step locally; the server averages their updated parameters, weighted by local training-sample counts, and broadcasts the result. FedRep shares the 12 lag weights across all clients while keeping a private bias for each client. This setting achieves the lowest validation RMSE in both limited-data settings. Each round, clients update their weights and biases locally, and the server averages and broadcasts only the shared weights. FedAvg and FedRep run for up to 300 rounds, and the model with the lowest validation RMSE is used for testing.

For NSMP, we first construct a maximum-similarity spanning tree $\mathcal{T} = (\mathcal{V}, \mathcal{E}_{\mathcal{T}})$ connecting the 206 clients, using their pairwise similarities as edge weights. This tree has 205 edges, each connecting two neighboring clients whose model parameters are encouraged to be similar. To encourage this similarity, we add a penalty on parameter differences to the local regression loss. For a non-root

client i with coefficients $\beta_i \in \mathbb{R}^{13}$, assigning the penalty on its parent edge to this client gives the local objective

$$f_i(\beta_i, \beta_{p(i)}) = \frac{\|X_i\beta_i - y_i\|^2}{2N_i} + \frac{10^{-3}}{2}\|P\beta_i\|^2 + \frac{\lambda}{2}\bar{w}_{i,p(i)}\|\beta_i - \beta_{p(i)}\|^2, \tag{79}$$

where X_i and y_i contain client i's N_i training samples, $p(i)$ is its parent in the similarity tree, P excludes the bias, and $\bar{w}_{ij}$ is the edge weight divided by the mean weight of all tree edges. Since the root has no parent, its local objective contains only the first two terms of (79). Therefore, each non-root local function therefore depends on its own coefficients and its parent's coefficients, giving the overlapping local parameters required by NSMP.

As shown in Table 4, NSMP substantially reduces communication cost, transmitting only 193,520 bytes compared with 12,854,400 bytes for FedAvg and 11,865,600 bytes for FedRep. These correspond to reductions of 98.49% and 98.37%, respectively, in both limited-data settings. NSMP also achieves the lowest test RMSE, at 4.0742 and 4.1106 with 1% and 0.5% of the training data available, respectively. These results show that, in this experiment, NSMP achieves substantial communication savings while preserving predictive performance.

G.3 Real distributed state estimation

This subsection compares NSMP with two representative iterative distributed optimization methods, Gossip and ADMM, reviewed in Appendix A. We evaluate their solution accuracy and communication cost on the EPFL and SoCal state-estimation datasets.

We consider distributed state estimation in an electrical distribution grid, where clients use local voltage and current measurements to jointly estimate the voltages at all electrical nodes. Specifically, an electrical node (bus) is a physical connection point in the grid, whose three-phase voltage is to be estimated. For B such nodes, stacking the real and imaginary parts of their voltages gives the global state $x \in \mathbb{R}^d$, with $d = 6B$. Each client i holds local voltage and current observations $y_i \in \mathbb{R}^{m_i}$ involving a subset of electrical nodes.

Client i's observation vector y_i is related linearly to the global voltage state x by

$$y_i = H_i x + \varepsilon_i, \tag{80}$$

where $H_i \in \mathbb{R}^{m_i \times d}$ is the measurement matrix determined by the measurement locations and network parameters, and $\varepsilon_i \in \mathbb{R}^{m_i}$ is measurement noise. The task is to estimate x by minimizing the weighted measurement errors across all clients:

$$\min_x \; F(x) = \frac{1}{2}\sum_i \|\Omega_i^{1/2}(H_i x - y_i)\|^2, \tag{81}$$

where $\Omega_i \in \mathbb{R}^{m_i \times m_i}$ weights the reliability of client i's observations. In particular, the measurement matrix H_i is usually sparse, and its nonzero columns correspond to the voltage variables involved in client i's measurements. Overlapping nonzero columns of H_i across clients give rise to partially overlapping local parameters, making the problem suitable for NSMP. NSMP constructs the separator variables from these overlaps using (7).

Table 5: Real state-estimation results, averaged over the selected snapshots. EPFL reports a shifted quadratic objective gap; SoCal reports state RMSE relative to the centralized solution (per unit).

Dataset	Method	Error metric	Hop rounds/iterations	Bytes
EPFL	NSMP	5.01×10^{-13}	4	**3,296**
	ADMM	1.03×10^{-4}	1,000	1,248,192
	Gossip	8.03×10^{-1}	3,000	4,608,192
SoCal	NSMP	2.04×10^{-9}	8	**9,064**
	ADMM	4.65×10^{-4}	2,000	6,528,000
	Gossip	7.09×10^{-1}	5,000	21,120,000

EPFL. The EPFL dataset contains voltage and current measurements from the university's 20 kV campus distribution grid[3]. We use data from five synchronized phasor measurement units (PMUs), each treated as one client. The data contain 47,257 measurement records collected on 4 August 2014, each containing readings from the five devices at the same time. We select four records at intervals of 10,000 records and estimate the grid voltages independently using each selected record. The supplied network configuration determines each PMU client's measurement matrix in (80). Combining the five clients' local measurement errors gives the voltage-estimation objective in (81), where the measurement weights are based on preset noise levels.

To assess solver accuracy in the EPFL experiment, we use the objective gap relative to the centralized solution, where the objective $F(x)$ is defined in (81). For numerical stability, we avoid directly computing $F(x) - F(x_{\mathrm{cen}})$ and use the equivalent quadratic expression:

$$\Delta F(x) = \tfrac{1}{2}(x - x_{\mathrm{cen}})^\top H(x - x_{\mathrm{cen}}), \qquad H = \sum_i H_i^\top \Omega_i H_i. \tag{82}$$

Here, x is the voltage-state estimate returned by the method being evaluated, and x_{cen} is the reference estimate obtained by solving the same weighted least-squares problem centrally. A smaller $\Delta F(x)$ means that the objective value is closer to the centralized optimum. We use this metric because, unlike the unweighted state error $\|x - x_{\mathrm{cen}}\|$, it accounts for the measurement model and reliability weights through H. As shown in Table 5, ADMM and Gossip run for 1,000 and 3,000 iterations, respectively. NSMP achieves the smallest objective gap (5.01×10^{-13}) in only four communication rounds, transmitting 3,296 bytes and substantially reducing communication cost compared with both iterative methods.

SoCal. The SoCal 28-Bus dataset provides voltage and current measurements from a real distribution grid [29]. Following the supplied network model, the grid used in this experiment has 12 electrical nodes, with one client assigned to each node. We select three measurement records from the five-minute period starting at 07:00 on 14 November 2024 and estimate the grid voltages independently for each record. Similarly, for these 12 clients, the local observation equations follow (80), including known zero-injection conditions where applicable. The voltage-estimation task is formulated as the weighted least-squares problem in (81).

We measure solution accuracy using the voltage-state RMSE relative to the centralized solution:

$$\mathrm{RMSE}_{\mathrm{state}} = \sqrt{\frac{\|x - x_{\mathrm{cen}}\|^2}{72}}. \tag{83}$$

Here, x is the estimate returned by the evaluated method, and x_{cen} is the centralized reference estimate. The denominator 72 is the number of scalar state variables for the 12 three-phase nodes. A smaller RMSE indicates closer agreement with the centralized estimate.

As shown in Table 5, ADMM and Gossip are evaluated after 2,000 and 5,000 iterations, respectively. NSMP achieves the lowest state RMSE (2.04×10^{-9}) in eight communication rounds. It transmits only 9,064 bytes, reducing communication cost by over 99% compared with both iterative methods.

G.4 PEMS-BAY approximate non-convex messages

In this subsection, we evaluate NSMP with non-convex local objectives and approximate messages. Following Appendix D.2, clients use numerical optimization and function fitting to construct the messages, so both local minimization and message representation are approximate.

We use PEMS-BAY for next-step traffic-speed prediction. The 30 selected sensors are divided equally among six clients, with five sensors per client. For each sample, client i takes an input $X_{i,t} \in \mathbb{R}^{12\times 5}$ containing its sensors' speeds over the previous 12 time steps and predicts their next-step speeds $y_{i,t} \in \mathbb{R}^5$. Each client uses 128 training samples, 240 validation samples, and 480 test samples. The prediction network has a temporal encoder, a graph layer that combines information from the client's sensors, and an output layer, with hidden dimension four. Its nonlinear activations give a non-convex training objective.

Local trains all network parameters independently. FedAvg shares and averages all network weights and biases across the six clients. FedRep shares the temporal-encoder and graph-layer parameters,

[3]Original data source: `http://nanotera-stg2.epfl.ch/data/`.

while each client keeps its output-layer weights and bias private. Thus, FedAvg uses one global network, whereas FedRep combines a global representation with client-specific output layers.

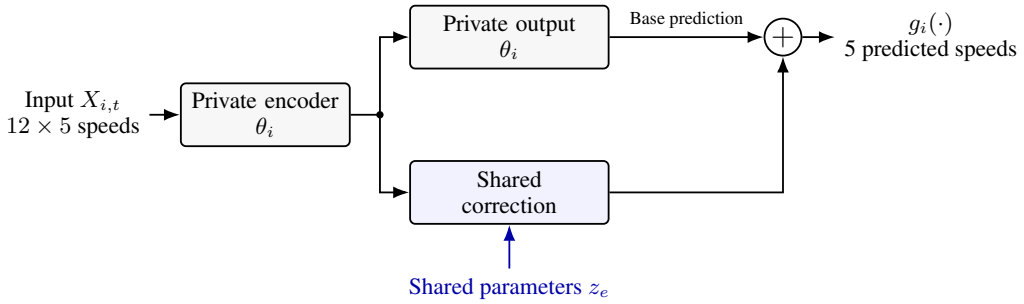


Figure 6: Input-output structure of client i's predictor g_i. The private network produces a base prediction, and shared edge variables provide an additive correction.

For NSMP, we construct a maximum-similarity spanning tree connecting the six clients, using correlations between their mean sensor-speed series over the training period. The tree has five edges. Client i uses private network parameters θ_i to produce hidden features and a base speed prediction. Each tree edge e has an additional parameter vector z_e shared by its two endpoint clients. Let $\mathcal{E}(i)$ denote the edges incident to client i. Figure 6 illustrates how the private network parameters θ_i and shared edge variables z_e jointly determine the predictor g_i. The shared-variable branch adjusts the private network's base prediction. The local objective is

$$f_i\big(\theta_i, \{z_e\}_{e\in\mathcal{E}(i)}\big) = \frac{1}{5N_i}\sum_{t=1}^{N_i} \big\| g_i\big(X_{i,t}; \theta_i, \{z_e\}_{e\in\mathcal{E}(i)}\big) - y_{i,t}\big\|_2^2 + \frac{\lambda_\theta}{2}\|\theta_i\|_2^2 + \frac{\lambda_z}{2}\sum_{e\in\mathcal{E}(i)} \|z_e\|_2^2, \tag{84}$$

where $N_i = 128$, $\lambda_\theta = 10^{-5}$, and $\lambda_z = 0.01$. The aggregate objective sums the local objectives, with each z_e shared by the two clients connected by edge e.

For NSMP, we construct approximate messages as follows. For a non-root client i, let e be its edge to the parent. The reduced message defined in (9) is written here as $\Phi_i(z_e)$, since its argument is the shared vector on this edge. At each sampled z_e, the client fixes this vector and uses Adam to optimize its private network parameters and child-edge vectors, approximately minimizing the local loss in (84) plus the received child messages, as in (51). The resulting samples are used to fit $\widehat{\Phi}_i(z_e)$ following (52). An MLP and simpler polynomial candidates are compared using validation error, and the selected approximation is sent to the parent. For comparison, FedAvg and FedRep are trained for 140 communication rounds. The mean results of all methods are reported in Table 6.

Table 6: PEMS-BAY with approximate non-convex messages. RMSE is mean $\pm$ sample standard deviation over three seeds. The communication bytes are averaged over three runs, with each transmitted value encoded in Float32 (4 bytes).

Method	Test RMSE	Bytes
Local	5.941 ± 0.875	0
FedAvg	5.781 ± 0.937	490,560
FedRep	5.681 ± 0.961	456,960
AdaptiveMsg-NSMP	$\mathbf{5.406 \pm 0.703}$	**9,568**

As shown in Table 6, AdaptiveMsg-NSMP achieves the lowest mean test RMSE of 5.406, compared with 5.941 for Local, 5.781 for FedAvg, and 5.681 for FedRep. It transmits only 9,568 bytes, reducing communication by 98.05% and 97.91% compared with FedAvg and FedRep, respectively. These results show that, even with approximate messages, the proposed NSMP achieves lower RMSE and transmits fewer bytes than the FedAvg and FedRep baselines in this experiment.

H Limitations

This work provides a framework for handling partially shared parameters in federated learning. Applying NSMP assumes that the local objectives and parameter-sharing relationships have been specified. In practical applications, however, models may not directly take a parameter-sharing form. A suitable formulation must therefore be constructed, including a characterization of which parameters are shared between neighboring clients and which remain private. How to derive these local parameter blocks and overlap relationships from application requirements remains an important modeling question for future research. Once a suitable parameter-sharing problem has been formulated, the proposed NSMP algorithm can be applied to the resulting optimization problem.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims match the exact-reduction theory in Section 4 and the experiments in Appendix G.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix H discusses modeling limitations; Section 4.3 and Appendix D.2 address communication costs and approximate messages.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The theoretical setting is specified in Sections 2–4, with proofs and approximation assumptions in Appendices B, C.1, and D.2.2.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Algorithm 1, the experimental appendices, and the released code provide the procedures and settings needed to reproduce the results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Appendix G links the public code repository, which provides execution commands, configurations, and data-access instructions.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5, Appendix G, and the released configurations specify the data, models, validation choices, and solver settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Table 6 reports mean RMSE and sample standard deviation over three seeds; no formal significance tests are reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix E specifies the workstation configuration: Intel i9-14900K CPU, 32 GB RAM, and NVIDIA GeForce RTX 4090 GPU.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The study uses synthetic and public benchmark data and involves no human-participant research or crowdsourcing.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Reduced communication costs can facilitate collaborative learning. Potential risks include information leakage through exchanged messages and unreliable downstream decisions when model estimates are inaccurate.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The work does not release high-risk models or newly scraped datasets requiring special safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Original data and code sources are credited, and reused assets are used in accordance with their applicable licenses and provider terms.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released implementation includes README files describing dependencies, data preparation, execution, and outputs.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing or experiments with human participants are involved.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human-subject research is involved, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not part of the core method or experimental pipeline.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.