Journal of Scientific Computing (2022) 91:15
https://doi.org/10.1007/s10915-022-01761-7

Interconnected Hierarchical Structures for Fast Direct Elliptic Solution

Xiao Liu[1] · Jianlin Xia[2] · Maarten V. de Hoop[1] · Xiaofeng Ou[2]

Received: 18 December 2020 / Revised: 9 November 2021 / Accepted: 29 December 2021

Abstract

We propose an interconnected hierarchical rank structure and use it to design a fast direct elliptic solver that can significantly reduce the amount of low-rank compression operations used in usual structured direct solvers. Interconnected structures within two hierarchical layers are exploited: the hierarchical partitioning of a large problem into subproblems that are local Schur complements on smaller subdomains, and the interconnected hierarchical structured approximations of the subproblems. The interconnected structures make it feasible to extensively reuse off-diagonal basis matrices produced in the rank-structured approximation of smaller local Schur complements. Such basis matrices are produced only once and then reused across multiple hierarchical levels of the sparse factorization. Unlike many existing rank-structured direct solvers where explicit low-rank compression is often the major computation, our new solver can then avoid most of the compression operations. This helps to both conveniently preserve the rank structures and reduce the cost. Under moderate conditions, the total factorization cost is $O(rn)$, where r is an appropriate off-diagonal numerical rank bound. The interconnected structures are further extended to accelerate a factorization update problem where many local coefficient updates are involved. Numerical tests on some PDE

The research of Jianlin Xia was supported in part by an NSF Grant DMS-1819166. Maarten V. de Hoop gratefully acknowledges support from the Simons Foundation under the MATH+X program, the NSF Grant DMS-1559587, the corporate members of the Geo-Mathematical Imaging Group at Rice University, and Total.

✉ Jianlin Xia
xiaj@math.purdue.edu

Xiao Liu
xiao.liu.rice@gmail.com

Maarten V. de Hoop
mvd2@rice.edu

Xiaofeng Ou
ou17@purdue.edu

[1] Department of Computational and Applied Mathematics, Rice University, Houston, TX 77005, USA

[2] Department of Mathematics, Purdue University, West Lafayette, IN 47907, USA

Published online: 26 February 2022

problems are used to demonstrate the efficiency and speedup. In particular, some reuse factors in our tests indicate dramatic reduction in the number of low-rank compression operations.

Keywords Interconnected hierarchical structure · Fast sparse direct solver · Elliptic equation · Basis reuse · Neighbor tree · Schur complement update

1 Introduction

This paper studies the design of a novel matrix structure that is useful for the fast direct solution of discretized problems such as elliptic PDEs of the form

$$Lu(x) = f(x), \quad x \in \Omega, \tag{1}$$

where L is a second-order elliptic partial differential operator in the domain of interest Ω. When the PDE is discretized with locally supported basis functions, a linear system with a sparse coefficient matrix A is obtained. Here, we are interested in exploring features from both the discretization and some intrinsic rank structures so as to show the feasibility of designing an interconnected rank structure that can greatly accelerate direct solutions of the relevant discretized linear systems.

Traditional direct solvers with nested dissection ordering [9] of the matrix A have complexities $O(n^{3/2})$ and $O(n^2)$ flops for two and three dimensions, respectively, where n is the order of A. The multifrontal method [7] is one of the most useful sparse direct solvers and it performs the sparse factorization in terms of multiple smaller local dense matrices called frontal matrices. Another type of popular direct solvers is supernodal methods such as SuperLU [6]. These direct solvers are generally expensive for large problems where the local dense matrices become challenging to store and factorize.

A lot of recent developments on sparse direct solvers focus on approximating the intermediate dense matrices by rank-structured representations. For elliptic problems, the intermediate dense matrices have off-diagonal blocks with small numerical ranks (see, e.g., [2, 4]). Low-rank approximations to these off-diagonal blocks yield rank-structured representations that effectively reduce the amount of data. Some frequently used rank-structured representations are $\mathscr{H}$-matrices [16], $\mathscr{H}^2$-matrices [15], and hierarchically semiseparable (HSS) matrices [5, 40]. Such representations often allow fast matrix–vector multiplication, factorization, inversion, and so on. Examples of rank-structured sparse direct solvers are $\mathscr{H}$-LU methods [14] and structured multifrontal methods [10, 37, 38, 42]. Specialized PDE solvers have also been proposed in [8, 17, 19, 20, 26, 30, 39] and can exploit additional properties of Cartesian coordinates and elliptic boundary value problems. Examples are Dirichlet-to-Neumann formulations in [12, 17, 26] and Robin-to-Robin formulations in [11, 24, 28].

When these structured solvers are applied to the discretized system for (1), they heavily rely on the construction of intermediate rank-structured representations. The rank-structured constructions are usually done through repeated use of low-rank compression techniques such as truncated SVDs, rank-revealing factorizations, and randomized SVDs. See [10, 21, 25, 40] for some examples. These constructions typically contribute to a significant cost of the sparse structured solvers [10, 33, 37, 38, 42]. They also make the implementation of the solvers sophisticated. Thus, some methods sacrifice certain efficiency to simplify the implementation by allowing some dense intermediate operations [1, 33, 38].

Here, we propose an interconnected hierarchical structure that makes it feasible to extensively *reuse information* during rank structured construction so as to significantly reduce

the cost of structured direct solution. Unlike many existing approaches, the new structure preserves off-diagonal basis matrices among local dense matrices across different levels of the sparse factorization. This avoids repetitive low-rank compression operations as well as dense intermediate Schur complements. Specifically, the novelties of this work include the following.

- *Interconnected tree structures* are designed to organize the sparse factorization. After repeated bisection of the domain, an *outer tree* structure is generated to govern the sparse factorization, and the tree nodes correspond to lists of interfaces for subdomains. Each node of the outer tree is further associated with an *inner tree* used for local structured operations. Inner trees at different outer levels *share subtrees* and are interconnected. Subtrees of the inner trees are preserved as much as possible when we traverse the outer tree, which is a major difference between our solver and existing solvers in [37, 38] that also use two layers of trees. The sharing of subtrees is natural following our discretizations and the levelwise elimination.
- *Interconnected hierarchically semiseparable (IHSS) matrix structures* are proposed to enable an efficient rank-structured factorization by taking advantage of both the intrinsic rank structures of the problem and the particular discretizations we follow (high-order finite element method with Robin-to-Robin formulations). Each inner tree is associated with an IHSS matrix corresponding to a local dense Schur complement. (The concept of a local Schur complement will be made precise later.) Following interconnected inner trees, IHSS matrices are also interconnected via the *reuse of basis matrices*. That is, IHSS matrices at different outer factorization levels share the same off-diagonal basis matrices. This avoids the majority of the cost needed for low-rank compression. Thus, along the outer tree, local dense factorizations can be performed in terms of IHSS structures.
- *A fast direct elliptic solver via IHSS structures* can then be designed. Unlike most other structured sparse direct solvers that need extensive off-diagonal compression, here the off-diagonal basis matrices are only computed once at a lower sparse factorization level and then *reused for later factorization levels*. This avoids the majority of the cost needed for low-rank compression. The basis reuse in IHSS structures helps to not only save the cost but also conveniently preserve the rank structures. For some discretized elliptic problems, with a suitable optimal choice of a so-called switching level in the outer tree (see Sect. 3.4), we can obtain a factorization cost of $O(rn)$ flops and an $O((\log r)n)$ storage count for factors, where r is a small numerical rank bound for appropriate off-diagonal blocks. In comparison, structured factorizations using explicit compression for all structured outer levels like those in [32, 38] cost $O(rn \log n)$.

We illustrate why such IHSS structures are feasible and how the off-diagonal bases are shared across different outer factorization levels. In fact, IHSS approximations at upper factorization levels can be conveniently obtained via fast updates of lower-level IHSS forms. Therefore, our solver does not need the expensive rank-structured construction used in other structured sparse direct solvers. The performance of the solver is confirmed in numerical tests on Poisson's equation and the Helmholtz equation. Our tests further show by how many times the basis matrices can be reused, which indicates significant reduction in the number of low-rank compression operations.

Our solver can also be extended to a challenging sparse factorization update problem, where there are multiple local changes to the coefficient of the PDE. We show how to use our IHSS algorithms to accelerate the factorization of some interior and exterior problems in a recent factorization update algorithm in [24].

Note that our purpose is not to design a general-purpose sparse direct solver or to provide a practical implementation of a sophisticated rank-structured solver that can apply to large complicated PDEs. Instead, we aim to show a *proof-of-concept idea* that illustrates the feasibility and the essential strategy of designing interconnected rank structures. We take advantage of some discretizations for elliptic PDEs so as to exploit the additional interconnected structures on top of known rank structures. Our discussions and tests are given in terms of some 2D elliptic PDEs. However, the key ideas can be extended to 3D and more practical problems, which is not our purpose.

The remaining sections are organized as follows. In Sect. 2, we give a basic framework that we follow for the design of our sparse direct elliptic solver. Section 3 presents the detailed interconnected hierarchical structured factorization, its advantages, and the complexity optimization. The extension to sparse factorization update is discussed in Sect. 4. Numerical examples are shown in Sect. 5 and some conclusions are drawn in Sect. 6.

2 Basic Framework of Our Sparse Direct Elliptic Solver

Sparse direct solvers often organize local factorizations following certain hierarchical tree structures. A typical example is the supernodal multifrontal factorization [7, 22] using the assembly tree structure. We first show a basic framework that we follow for the direct solution of some elliptic PDEs. The hierarchical factorization is organized based on certain features of the discretization and the bases that facilitate the design of the interconnected hierarchical structures later in Sect. 3.

2.1 Domain Partitioning, Interface Organization, and Neighbor Tree

We start from a hierarchical domain partitioning based on repeated bisection. Figure 1a illustrates a simple 2-level domain partitioning. Initially, a horizontal cut splits the domain into two level-1 subdomains. The *interfaces* or boundaries along the horizontal cut (marked as 5, 6, 7, 8 in red) are labeled as the level-1 interfaces. Here, the two sides of the same cut have distinct labels. Next, for each level-1 subdomain, a vertical cut splits it into two level-2 subdomains. The interfaces along the vertical cuts (marked as 1, 2 and 3, 4 in blue) are labeled as level-2 interfaces. This process then repeats. A 4-level partitioning is shown in Fig. 1b.

In the sparse factorization, all the finest level interior mesh points and the outermost boundary points of the entire domain are eliminated first. Thus, all our later discussions focus on processing the interfaces. Following the multilevel bisection, the interfaces can be organized into a binary tree $\mathbf{T}$ similarly to a separator tree from nested dissection [9]. Each node of $\mathbf{T}$ corresponds to a separator (or cut) of the mesh. Figure 1c shows the tree corresponding to Fig. 1b. $\mathbf{T}$ can essentially be used as an assembly tree [22] for sparse eliminations. The difference is that a separator or tree node here consists of a set of interfaces of subdomains. In addition, the leaf nodes of $\mathbf{T}$ are also interfaces instead of finest level subdomains. Later, we use a set of interfaces grouped inside a pair of parentheses to denote a separator as marked in Fig. 1c. Also, we do not distinguish between a node of $\mathbf{T}$ and a separator in the mesh.

The interfaces at different levels of the tree $\mathbf{T}$ correspond to intermediate Schur complements at different levels of sparse factorization. In the PDE solution, unknowns associated with lower-level interfaces are ordered before upper-level interfaces and are eliminated first.

The interfaces are also labeled consecutively at each level so that a convenient levelwise presentation can be used when we discuss the actual factorization later.

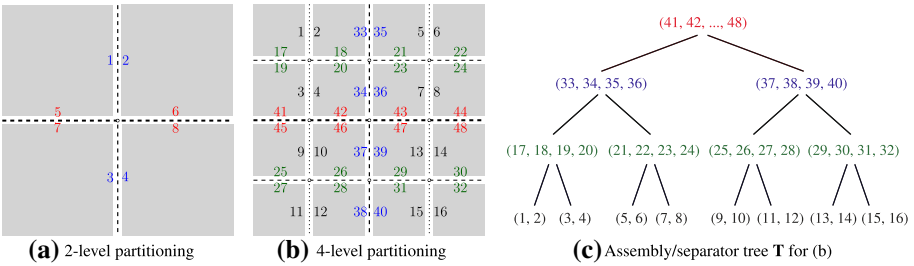
The elimination of lower-level separators mutually connects some upper-level separators and creates fill-in in the Schur complements [27]. To clearly identify such connections, we use the following terms and notation.

- For a separator S at level l of $\mathbf{T}$, its *neighbors* are those interfaces which are at upper levels and are connected to S due to the lower-level eliminations. A similar concept is used in [38, 39] for deriving some rank structured solvers. Accordingly, S is said to be a *pivot separator* (with respect to its neighbors). For example, the pivot separator (3, 4) (formed by interfaces 3, 4) in Fig. 1b–c has neighbors or neighbor interfaces 19, 20, 34, 41, 42.
- A *pivot interface* within a pivot separator at level l is the set of all interfaces belonging to one same level-l subdomain in the bisection process. We mark pivot interfaces in bold fonts. For example, during the elimination of the separator (17, 18, 19, 20) in Fig. 1b–c, (**17**, **18**) is a pivot interface and so is (**19**, **20**). Thus, each pivot separator includes two pivot interfaces associated with a pair of adjacent subdomains.
- For a pivot interface σ (within a pivot separator S at level l), the *neighbor list* associated with σ is the list consisting of σ itself and those interfaces that are neighbors of S and are also in the same level-l subdomain as σ. We use a pair of braces to identify a neighbor list. We also use a pair of parentheses to group interfaces from the same separator. For example, the pivot interface (**19**, **20**) in Fig. 1b–c has neighbor list {(**19**, **20**), 34, (41, 42)}.
- In a neighbor list, the interfaces that are from one same separator S in $\mathbf{T}$ are said to be *common-origin interfaces*, and the separator S is the *origin* of the interfaces. For example, in Fig. 1b, 19, 20 are common-origin interfaces in the neighbor list {(**19**, **20**), 34, (41, 42)} since they are from the same origin which is the separator (17, 18, 19, 20) of $\mathbf{T}$ in Fig. 1c. 41, 42 are also common-origin interfaces in this neighbor list.
- Finally, we introduce another tree $\mathscr{T}$ called *neighbor tree*. Each node of $\mathscr{T}$ is a neighbor list associated with a pivot interface, except the root which is empty. See Fig. 2. The neighbor tree $\mathscr{T}$ plays a key role in the design of our IHSS structures. In our later discussions, the sparse factorization will mainly be discussed in terms of the neighbor tree $\mathscr{T}$, with the aid of the assembly/separator tree $\mathbf{T}$. For two neighbor lists that are siblings at level l of $\mathscr{T}$, their pivot interfaces form one separator at level l of $\mathbf{T}$. Here, we assume $\mathbf{T}$ has levels $1, 2, \ldots, \mathbf{l}$ but $\mathscr{T}$ has levels $0, 1, \ldots, \mathbf{l}$. The reason is that $\mathscr{T}$ has an empty root at level 0.

2.2 Initial Schur Complements from the Finest Subdomains

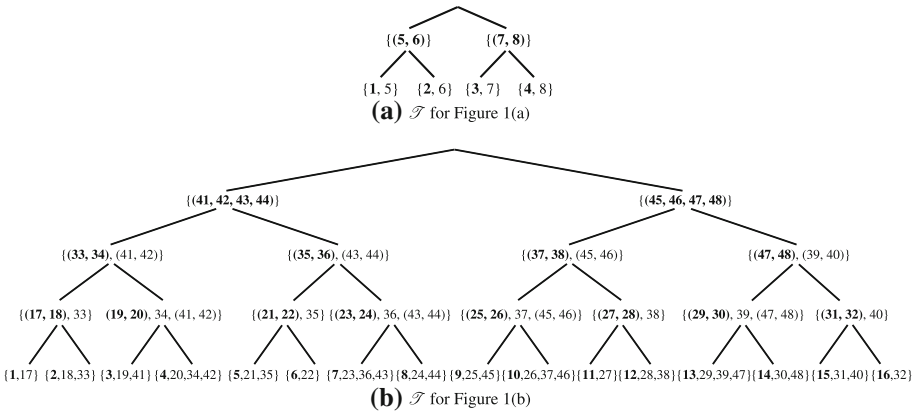
For elliptic PDEs like (1), the sparse direct factorization involves particular features in the Schur complements. We show the strategies we follow for processing the Schur complements. We first describe how to form an *initial Schur complement* that results from the elimination of the interior mesh points in all the finest subdomains as well as all exterior boundary points of the entire domain. Thus, the initial Schur complement corresponds to all the leaves in the neighbor tree $\mathscr{T}$ and is denoted $\mathbf{S}^{(\mathbf{l})}$.

$\mathbf{S}^{(\mathbf{l})}$ is computed from factorizing the discretized PDE restricted to the disjoint finest subdomains. Various discretization methods have been used for the restricted problems in subdomains, including finite difference methods in [19, 39], spectral collocation methods in [26], and finite element methods in [9, 17]. Here, we use a high-order finite element method with Lagrange bases set up according to [18]. (See [18, 24] and the references therein for



(a) 2-level partitioning **(b)** 4-level partitioning **(c)** Assembly/separator tree **T** for (b)

Fig. 1 Partitioning of a domain, where the interfaces of the subdomains introduced by the partitioning are indexed and the dashed lines are for the purpose of illustrating the partitioning and are not part of the domain. The numbering scheme of the interfaces is as follows. Smaller labels correspond to lower-level interfaces so as to reflect the order of elimination. Labels in the same color in the figure are associated with the same separator. We also keep consecutive numbering within each interface so as to simplify the lists of labels in (c) (Color figure online)



(a) $\mathscr{T}$ for Figure 1(a)

(b) $\mathscr{T}$ for Figure 1(b)

Fig. 2 Illustration of neighbor trees $\mathscr{T}$ for organizing neighbor lists

motivations of such a discretization.) Within each subdomain, the factorization usually yields a dense Schur complement that corresponds to a pseudo-differential operator [3,Section 1.2]. We choose the recent Robin-to-Robin formulation in [11, 28]. Note that unknowns on the corners of subdomains give rise to book-keeping issues [29] and can be removed by reducing the polynomial order on the boundary [11]. We use Legendre polynomials on the boundary and this reduction of order is an orthogonal projection. Thus, in our organization of the interfaces as in Fig. 1, there are no unknowns associated with the corner mesh points of the subdomains.

We follow the derivation of [28, Equation 2.9] to get the initial Schur complement $\mathbf{S}^{(\mathbf{l})}$. For simplicity, we illustrate the form of $\mathbf{S}^{(\mathbf{l})}$ using the 2-level partitioning in Fig. 1a. Here, $\mathbf{l} = 2$ and the neighbor tree $\mathscr{T}$ is shown in Fig. 2a. $\mathbf{S}^{(2)}$ has the following form:

$$\mathbf{S}^{(2)} = \begin{pmatrix} \mathbf{S}_{11}^{(2)} & I & & & \mathbf{S}_{15}^{(2)} & & & \\ I & \mathbf{S}_{22}^{(2)} & & & & \mathbf{S}_{26}^{(2)} & & \\ & & \mathbf{S}_{33}^{(2)} & I & & & \mathbf{S}_{37}^{(2)} & \\ & & I & \mathbf{S}_{44}^{(2)} & & & & \mathbf{S}_{48}^{(2)} \\ \mathbf{S}_{51}^{(2)} & & & & \mathbf{S}_{55}^{(2)} & & I & \\ & \mathbf{S}_{62}^{(2)} & & & & \mathbf{S}_{66}^{(2)} & & I \\ & & \mathbf{S}_{73}^{(2)} & & I & & \mathbf{S}_{77}^{(2)} & \\ & & & \mathbf{S}_{84}^{(2)} & & I & & \mathbf{S}_{88}^{(2)} \end{pmatrix}, \tag{2}$$

where the subscripts $1, 2, \ldots, 8$ correspond to the interfaces in Fig. 1a and the following submatrices correspond to the four leaves of $\mathscr{T}$:

$$\begin{pmatrix} \mathbf{S}_{11}^{(2)} & \mathbf{S}_{15}^{(2)} \\ \mathbf{S}_{51}^{(2)} & \mathbf{S}_{55}^{(2)} \end{pmatrix}, \quad \begin{pmatrix} \mathbf{S}_{22}^{(2)} & \mathbf{S}_{26}^{(2)} \\ \mathbf{S}_{62}^{(2)} & \mathbf{S}_{66}^{(2)} \end{pmatrix}, \quad \begin{pmatrix} \mathbf{S}_{33}^{(2)} & \mathbf{S}_{37}^{(2)} \\ \mathbf{S}_{73}^{(2)} & \mathbf{S}_{77}^{(2)} \end{pmatrix}, \quad \begin{pmatrix} \mathbf{S}_{44}^{(2)} & \mathbf{S}_{48}^{(2)} \\ \mathbf{S}_{84}^{(2)} & \mathbf{S}_{88}^{(2)} \end{pmatrix}, \tag{3}$$

These submatrices are the results of the elimination of the interior mesh points of the four subdomains and the boundary of the entire domain. The identity blocks in (2) introduce coupling to shared interfaces. This is due to the transmission condition on shared boundaries. See [11, 28] for the derivation.

2.3 Sparse Hierarchical Block LDU Factorization via Schur Complement Update

Once $\mathbf{S^{(l)}}$ is obtained, we can perform sparse hierarchical block LDU factorization and solution as in standard sparse direct solvers such as the multifrontal method. The factorization at a level $l = \mathbf{l}, \mathbf{l-1}, \ldots, 1$ can be written collectively as

$$\mathbf{S}^{(l)} = \begin{pmatrix} I & \\ \mathbf{L}^{(l)} & I \end{pmatrix} \begin{pmatrix} \mathbf{D}^{(l)} & \\ & \mathbf{S}^{(l-1)} \end{pmatrix} \begin{pmatrix} I & \mathbf{U}^{(l)} \\ & I \end{pmatrix}, \tag{4}$$

where $\mathbf{S}^{(l)}$ is the input corresponding to all the level-l nodes of $\mathscr{T}$, $\mathbf{L}^{(l)}$ and $\mathbf{U}^{(l)}$ are obtained from eliminating the pivot block $\mathbf{D}^{(l)}$ of $\mathbf{S}^{(l)}$, and $\mathbf{S}^{(l-1)}$ is the Schur complement to be factorized at level $l-1$. Here, the pivot block $\mathbf{D}^{(l)}$ is the submatrix of $\mathbf{S}^{(l)}$ corresponding to all pivot interfaces in the level-l neighbor lists of $\mathscr{T}$. With each level of factorization, the matrix size reduces and the tree $\mathscr{T}$ shrinks by one level. The process continues until level 1 of $\mathscr{T}$ is reached and the Schur complement $\mathbf{S}^{(1)}$ is factorized. For convenience, we refer to the computation of $\mathbf{S}^{(l-1)}$ from $\mathbf{S}^{(l)}$ via the sparse LDU factorization (4) as the *Schur complement update problem*, which is the main task for the sparse factorization.

After the sparse block LDU factorization, the solution can be computed by solving a sequence of intermediate linear systems. To solve an intermediate problem $\mathbf{S}^{(l)}x^{(l)} = b^{(l)}$ at level l, where $\mathbf{S}^{(l)}$ has the block factorization (4), the following three solution steps are needed:

$$\begin{pmatrix} I & \\ \mathbf{L}^{(l)} & I \end{pmatrix} \begin{pmatrix} y \\ b^{(l-1)} \end{pmatrix} = b^{(l)}, \tag{5}$$

$$\begin{pmatrix} \mathbf{D}^{(l)} & \\ & \mathbf{S}^{(l-1)} \end{pmatrix} \begin{pmatrix} z \\ x^{(l-1)} \end{pmatrix} = \begin{pmatrix} y \\ b^{(l-1)} \end{pmatrix}, \tag{6}$$

$$\begin{pmatrix} I & \mathbf{U}^{(l)} \\ & I \end{pmatrix} x^{(l)} = \begin{pmatrix} z \\ x^{(l-1)} \end{pmatrix}. \tag{7}$$

This is a recursive process since the second step needs to solve $\mathbf{S}^{(l-1)}x^{(l-1)} = b^{(l-1)}$ at level $l-1$.

As an example, consider the factorization of the matrix (2). After performing the elimination on $\mathbf{S}^{(2)}$ as in (4) with the leaf level of $\mathscr{T}$ (Fig. 2a) removed, the level-1 Schur complement is

$$\mathbf{S}^{(1)} = \begin{pmatrix} \begin{pmatrix} \mathbf{S}^{(1)}_{55} & \mathbf{S}^{(1)}_{56} \\ \mathbf{S}^{(1)}_{65} & \mathbf{S}^{(1)}_{66} \end{pmatrix} & I \\ I & \begin{pmatrix} \mathbf{S}^{(1)}_{77} & \mathbf{S}^{(1)}_{78} \\ \mathbf{S}^{(1)}_{87} & \mathbf{S}^{(1)}_{88} \end{pmatrix} \end{pmatrix}, \quad \text{with}$$

$$\begin{pmatrix} \mathbf{S}^{(1)}_{55} & \mathbf{S}^{(1)}_{56} \\ \mathbf{S}^{(1)}_{65} & \mathbf{S}^{(1)}_{66} \end{pmatrix} = \begin{pmatrix} \mathbf{S}^{(2)}_{55} & \\ & \mathbf{S}^{(2)}_{66} \end{pmatrix} - \begin{pmatrix} \mathbf{S}^{(2)}_{51} & \\ & \mathbf{S}^{(2)}_{62} \end{pmatrix} \begin{pmatrix} \mathbf{S}^{(2)}_{11} & I \\ I & \mathbf{S}^{(2)}_{22} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{S}^{(2)}_{15} & \\ & \mathbf{S}^{(2)}_{26} \end{pmatrix}, \tag{8}$$

$$\begin{pmatrix} \mathbf{S}^{(1)}_{77} & \mathbf{S}^{(1)}_{78} \\ \mathbf{S}^{(1)}_{87} & \mathbf{S}^{(1)}_{88} \end{pmatrix} = \begin{pmatrix} \mathbf{S}^{(2)}_{77} & \\ & \mathbf{S}^{(2)}_{88} \end{pmatrix} - \begin{pmatrix} \mathbf{S}^{(2)}_{73} & \\ & \mathbf{S}^{(2)}_{84} \end{pmatrix} \begin{pmatrix} \mathbf{S}^{(2)}_{33} & I \\ I & \mathbf{S}^{(2)}_{44} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{S}^{(2)}_{37} & \\ & \mathbf{S}^{(2)}_{48} \end{pmatrix}. \tag{9}$$

The representations (8)–(9) have a clear geometric interpretation. Take (8) as an example. $\mathbf{S}^{(2)}_{55}$ and $\mathbf{S}^{(2)}_{66}$ arise from two disjoint interfaces 5 and 6, but $\begin{pmatrix} \mathbf{S}^{(1)}_{55} & \mathbf{S}^{(1)}_{56} \\ \mathbf{S}^{(1)}_{65} & \mathbf{S}^{(1)}_{66} \end{pmatrix}$ joins the two interfaces into one pivot interface $(\mathbf{5}, \mathbf{6})$ of the separator $(5, 6, 7, 8)$, and the additional information is due to the update term contributed by the shared separator $(1, 2)$ via the coupling term $\begin{pmatrix} \mathbf{S}^{(2)}_{11} & I \\ I & \mathbf{S}^{(2)}_{22} \end{pmatrix}^{-1}$. In another word, the elimination of the separator $(1, 2)$ mutually connects the points in $(\mathbf{5}, \mathbf{6})$.

In general, in our direct elliptic solver, the Schur complement update problem is processed along the levelwise bottom-up traversal of the neighbor tree $\mathscr{T}$. Each $\mathbf{S}^{(l)}$ is processed in terms of the factorization of a sequence of local submatrices corresponding to the neighbor lists at level l of $\mathscr{T}$. For convenience, we call each such a submatrix a level-l *local Schur complement* (associated with a neighbor list). For example, for $\mathbf{S}^{(2)}$ in (2) with the neighbor tree $\mathscr{T}$ in Fig. 2a, the four submatrices in (3) are level-2 local Schur complements corresponding to the four leaf nodes of $\mathscr{T}$. The two submatrices of $\mathbf{S}^{(1)}$ in (8) and (9) are level-1 local Schur complements corresponding to the two level-1 nodes of $\mathscr{T}$. In the next section, we show how to accelerate the Schur complement update by applying structured methods to local Schur complements.

3 Interconnected Hierarchical Structured Factorization

We then consider the incorporation of rank structured methods into the basic factorization framework in the previous section. Directly performing the Schur complement update (4) is

expensive for large sizes. The cost can be reduced by converting local Schur complements into data-sparse forms, and the feasibility follows from [2, 4]. In this section, we design IHSS structures and construct IHSS approximations to the local Schur complements. The IHSS approximations at different levels of the sparse factorization are closely related via shared structures. We introduce IHSS approximations and factorizations based on interconnected tree structures.

3.1 Interconnected Tree Structures

Our sparse hierarchical factorization (4) follows the levelwise traversal of the neighbor tree $\mathscr{T}$. For each node (neighbor list) of $\mathscr{T}$, we construct an IHSS approximation to the corresponding local Schur complement. (The details are in Sect. 3.2.) Each IHSS approximation also corresponds to a tree called *IHSS tree*. Thus, we have two layers of trees: the neighbor tree $\mathscr{T}$ (as the *outer tree*) for organizing the sparse factorization, and an inner IHSS tree for the structured approximation of a local Schur complement. This two-layer tree structure is similar to structured multifrontal methods in [1, 37–39].

However, there are some differences between our new solver and those structured multifrontal methods.

- The structured multifrontal methods use the assembly tree as the outer-layer tree, while here our outer-layer tree is a neighbor tree.
- More importantly, the structured multifrontal methods study the two layers of trees separately, which may result in repetitive constructions of inner structures. In our new solver, the inner trees at different levels of the outer tree are interconnected such that subtrees of the inner trees are preserved and reused as much as possible when we traverse the outer tree. That is, our solver uses *interconnected tree structures*, which facilitate optimized data assembly and factorization.

A key mechanism of the interconnected tree structures is the organization of the inner IHSS trees with the aid of the neighbor lists in the neighbor tree $\mathscr{T}$.

1. A parent node (neighbor list) of $\mathscr{T}$ includes interfaces resulting from child level eliminations. At a child level, the pivot interfaces in two sibling neighbor lists form a complete separator which is eliminated. The remaining interfaces are collected to form the parent neighbor list. For example, in Figs. 1b and 2b, the elimination of the pivot interface (**21**, **22**) from {(**21**, **22**), 35} and the elimination of (**23**, **24**) from {(**23**, **24**), 36, (43, 44)} yield the parent node {(**35**, **36**), (43, 44)}.
2. Each separator in the separator tree **T** consists of a list of interfaces. Each interface serves as a leaf node in an inner IHSS tree. An IHSS tree is used to organize all the interfaces within a neighbor list in the neighbor tree. Common-origin interfaces correspond to one subtree of an IHSS tree. This makes sure that the IHSS structure fully respects the separators which the interfaces belong to. For example, for the neighbor list {(**23**, **24**), 36, (43, 44)} in Fig. 2b, the interfaces come from three different separators in Fig. 1c. The corresponding IHSS tree then has three subtrees (i.e., its root has three children). One subtree has the sibling interfaces 23 and 24 as leaves. Another subtree has one node corresponding to 36, and the third subtree has two leaves given by the sibling interfaces 43 and 44.
3. When common-origin interfaces are collected to form a parent neighbor list, leaf nodes or subtrees are put together to form larger subtrees of the IHSS tree associated with the parent. By keeping track of the domain partitioning around each separator, it is easy to see how these subtrees are formed. A parent IHSS tree can then be formed based on child IHSS trees.

4. The interfaces within each node of $\mathscr{T}$ are used for the partitioning of the associated local Schur complement. Such a partitioning is used in the construction of an IHSS approximation to the local Schur complement. For example, the four local Schur complements in (3) are partitioned precisely based on the interfaces inside the four leaf nodes in Fig. 2a. The corresponding IHSS tree is used to organize the block partitioning hierarchically.

These strategies ensure that structures passed from a lower level to the parent level are fully preserved in the IHSS approximation. This is elaborated in the next two subsections, where we show how IHSS forms are obtained along the traversal of the neighbor tree.

3.2 IHSS Representation

Suppose a node (neighbor list) at level l of $\mathscr{T}$ is $\alpha = \{c_1, c_2, \ldots, c_s\}$, where we assume all common-origin interfaces are already grouped into each c_j. Thus, the interfaces come from s different origins. Then the corresponding local Schur complement $S_\alpha^{(l)}$ within $\mathbf{S}^{(l)}$ can be represented as

$$S_\alpha^{(l)} = \begin{pmatrix} \mathbf{S}^{(l)}_{c_1,c_1} & \mathbf{S}^{(l)}_{c_1,c_2} & \cdots & \mathbf{S}^{(l)}_{c_1,c_s} \\ \mathbf{S}^{(l)}_{c_2,c_1} & \mathbf{S}^{(l)}_{c_2,c_2} & \cdots & \mathbf{S}^{(l)}_{c_2,c_s} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}^{(l)}_{c_s,c_1} & \mathbf{S}^{(l)}_{c_s,c_2} & \cdots & \mathbf{S}^{(l)}_{c_s,c_s} \end{pmatrix}. \tag{10}$$

We introduce an IHSS approximation to $S_\alpha^{(l)}$ via the following three key features.

1. *Low-rank off-diagonal forms*

   The off-diagonal blocks of $S_\alpha^{(l)}$ can be approximated by low-rank forms. Following the partitioning in (10), the IHSS approximation to $S_\alpha^{(l)}$ looks like

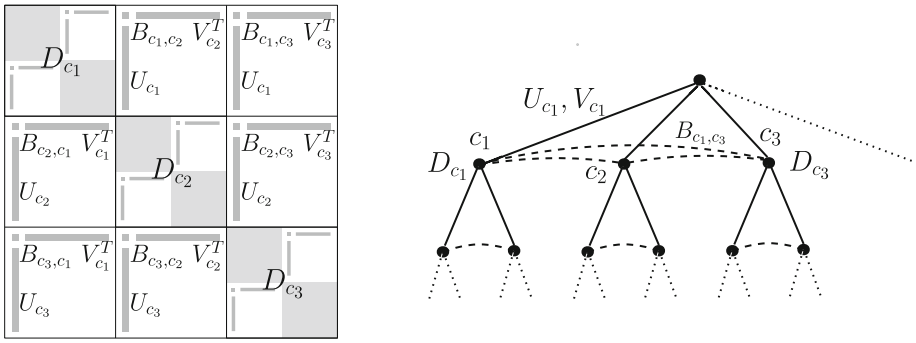
$$S_\alpha^{(l)} \approx \begin{pmatrix} D_{c_1} & U_{c_1} B_{c_1,c_2} V_{c_2}^T & \cdots & U_{c_1} B_{c_1,c_s} V_{c_s}^T \\ U_{c_2} B_{c_2,c_1} V_{c_1}^T & D_{c_2} & \cdots & U_{c_2} B_{c_2,c_s} V_{c_s}^T \\ \vdots & \vdots & \ddots & \vdots \\ U_{c_s} B_{c_s,c_1} V_{c_1}^T & U_{c_s} B_{c_s,c_2} V_{c_2}^T & \cdots & D_{c_s} \end{pmatrix}, \tag{11}$$

   where

   – each D_{c_j} is a diagonal block representing the self-interaction within the interface c_j;
   – each U_{c_j} (V_{c_j}) is the column (row) basis matrix representing the contribution of the interface c_j to the interaction between c_j and the other $s-1$ interfaces in the neighbor list;
   – each B_{c_j,c_k} describes the mutual interaction between a pair of interfaces c_j, c_k (ignoring the U_{c_j}, V_{c_k} basis matrices).

2. *Hierarchical structures*

   The structured approximation further appears in a hierarchical or nested form. Each c_i in α may be formed by grouping smaller common-origin interfaces pairwise and hierarchically.



(a) An IHSS matrix

(b) The corresponding IHSS tree with some generators marked

Fig. 3 Illustration of an IHSS matrix and the corresponding IHSS tree defined based on the interfaces c_1, c_2, c_3. The shaded boxes visualize the shapes of the D, B, U, V generators. The interfaces may consist of smaller lower level interfaces

For example, c_i may be formed by two common-origin interfaces z_1 and z_2, each of which may in turn be formed by two smaller common-origin interfaces. Thus, each c_i corresponds to a binary subtree T_{c_i} and the children of c_i are z_1 and z_2, which may also have children. The corresponding structured representation for D_{c_i} has a hierarchical form and is essentially a standard HSS form as in [5, 40]. The nested structures look like

$$D_{c_i} = \begin{pmatrix} D_{z_1} & U_{z_1} B_{z_1,z_2} V_{z_2}^T \\ U_{z_2} B_{z_2,z_1} V_{z_1}^T & D_{z_2} \end{pmatrix}, \quad U_{c_i} = \begin{pmatrix} U_{z_1} R_{z_1} \\ U_{z_2} R_{z_2} \end{pmatrix}, \quad V_{c_i} = \begin{pmatrix} V_{z_1} W_{z_1} \\ V_{z_2} W_{z_2} \end{pmatrix}, \tag{12}$$

where the R, W matrices are used to translate the basis matrices from child nodes to their parent. Thus, off-diagonal blocks at different hierarchical levels of T_{c_i} are low rank. T_{c_i} is also said to be an *HSS subtree*. All the HSS subtrees $T_{c_i}, i = 1, 2, \ldots, s$ are organized together under a root node which is the neighbor list α. This leads to a hierarchical tree T called *IHSS tree*. Figure 3 shows an example.

3. *Interconnected structures via shared off-diagonal bases*

The off-diagonal basis matrices U, V are preserved across different levels l of the outer tree $\mathscr{T}$. In the factorization (4), some of the off-diagonal basis matrices U, V of $S_\alpha^{(l)}$ are passed to level $l-1$ and are directly used to construct the IHSS approximations to the local Schur complements at level $l-1$. This will be explained in detail in Sect. 3.3.

The matrices D, B, U, V, R, W are called *generators* which are associated with the nodes of the IHSS tree T, as indicated by the subscripts of the generators. Due to the nested relation (12), only D, U, V generators associated with leaf nodes need to be stored. The U, V generators have small column sizes so that the representation is data sparse.

As compared with the HSS structure, the IHSS structure has the following features.

– Each nonleaf node in an HSS tree has two children, while the root of the IHSS tree may have more than two children, depending on the number of interfaces involved. See, e.g., Fig. 3.

– Corresponding to each interface c_i, a diagonal block of the IHSS form is a regular HSS block. Thus, the subtrees associated with the children of the root of the IHSS tree are binary HSS trees.
– The off-diagonal blocks corresponding to the interactions between interfaces are low-rank forms and the basis matrices also have nested forms like in the HSS case. Such nested forms are due to the shared off-diagonal bases across different levels of the tree $\mathscr{T}$ as mentioned above.

Remark 1 To simplify notation, we sometimes use $B_{c_i,*}$ to denote the B generator corresponding to an interface c_i and another unspecified interface. If an interface z_1 has only one sibling z_2 in the tree T, then we may use $\mathrm{sib}(z_1)$ to represent z_2 and write B_{z_1,z_2} as $B_{z_1,\mathrm{sib}(z_1)}$.

Remark 2 We can see that the discretization of the problem makes it feasible to take advantage of the interconnected structures across different outer layers. Accordingly, the IHSS solver relies on the discretization of the problem and is not as general as the structured solvers in [37, 38]. However, the IHSS idea provides a new way to explore additional structures (shared bases matrices) on top of known rank structures.

3.3 Schur Complement Update via IHSS Factorizations

We then use IHSS factorizations to perform the Schur complement update in (4). IHSS approximations like in (11)–(12) can be applied individually to each local Schur complement within $\mathbf{S}^{(l)}$ based on the associated neighbor list. IHSS approximations fully respect the interconnected tree structures by exploiting the connections of the neighbor lists at different levels of $\mathscr{T}$. That is, the U, V basis matrices of the IHSS forms used in $\mathbf{S}^{(l)}$ are preserved and reused in the computation of $\mathbf{S}^{(l-1)}$. The details are as follows.

Starting from a certain switching level $\mathbf{l}_s$ of $\mathscr{T}$ (the reason why the switching level is used will be explained in Sect. 3.4), an IHSS approximation is computed directly for each local Schur complement within $\mathbf{S}^{(\mathbf{l})}$. This may be based on direct off-diagonal block compression as in [40] or randomized construction like in [23, 25, 34, 41]. With the number of bisection levels $\mathbf{l}$ large enough, these local Schur complements have small sizes and the costs for the direct IHSS construction are low. Also, such direct IHSS construction is done only once at level $\mathbf{l}_s$ and the off-diagonal basis matrices will be reused at upper levels.

Then for $l = \mathbf{l}_s, \mathbf{l}_s - 1, \ldots, 1$, we perform the factorization of $\mathbf{S}^{(l)}$ in (4). We compute a structured approximation to $\mathbf{S}^{(l-1)}$ by constructing an IHSS approximation to each local Schur complement within $\mathbf{S}^{(l-1)}$. The interconnected tree structures are used to quickly get the IHSS approximations and to avoid expensive direct construction. Specifically, suppose two sibling nodes at level l of $\mathscr{T}$ are associated with the following two neighbor lists, respectively:

$$\alpha = \{\boldsymbol{\sigma}, c_1, c_2, \ldots, c_s\}, \quad \beta = \{\boldsymbol{\delta}, d_1, d_2, \ldots, d_t\}, \tag{13}$$

where $\boldsymbol{\sigma}$ and $\boldsymbol{\delta}$ are pivot interfaces to be eliminated, and we assume all common-origin interfaces have already been grouped into each $\boldsymbol{\sigma}, c_i, \boldsymbol{\delta}$, or d_j. (The grouping will be discussed at the end of Sect. 3.3.2 as part of the process for forming hierarchical representations.) Also, let the parent node of α and β in (13) be γ , which corresponds to a neighbor list formed by $c_1, \ldots, c_s, d_1, \ldots, d_t$. Examples of such interface lists in Figs. 1b and 2b are as follows:

$$\underset{\boldsymbol{\sigma}\quad c_1}{\alpha = \{\mathbf{(21,22)}, 35\}}, \quad \underset{\boldsymbol{\delta}\quad d_1\quad d_2}{\beta = \{\mathbf{(23,24)}, 36, (43,44)\}}, \quad \underset{c_1, d_1\quad d_2}{\gamma = \{\mathbf{(35,36)}, (43,44)\}}, \tag{14}$$

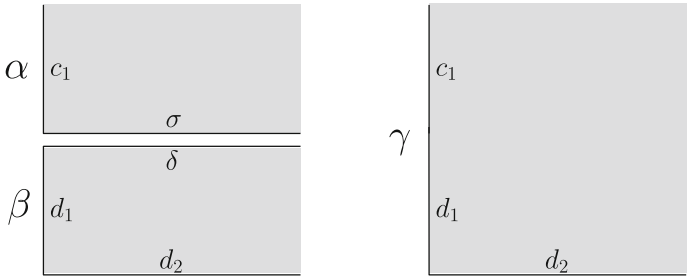


Fig. 4 Illustration of the neighbor lists in the example in (14) corresponding to the interfaces in Fig. 1(b). Each shaded area is a subdomain. σ and δ are pivot interfaces. Neighbor lists do not contain outermost domain boundaries

which are illustrated in Fig. 4.

Suppose the two neighbor lists α and β correspond to two local Schur complements $S_\alpha^{(l)}$ and $S_\beta^{(l)}$, respectively. In the factorization (4), we eliminate the pivot blocks of $S_\alpha^{(l)}$ and $S_\beta^{(l)}$ and form the local Schur complement $S_\gamma^{(l-1)}$ corresponding to γ at level $l-1$.

We first rewrite the local Schur complement updates like (8) or (9) in a general form (see [28, equations (2.9)–(2.14)] for the detailed derivation). That is, for $i, j = 1, 2, \ldots, \min\{s, t\}$,

$$\begin{pmatrix} \mathbf{S}_{c_i,c_j}^{(l-1)} & \mathbf{S}_{c_i,d_j}^{(l-1)} \\ \mathbf{S}_{d_i,c_j}^{(l-1)} & \mathbf{S}_{d_i,d_j}^{(l-1)} \end{pmatrix} = \begin{pmatrix} \mathbf{S}_{c_i,c_j}^{(l)} & \\ & \mathbf{S}_{d_i,d_j}^{(l)} \end{pmatrix} - \begin{pmatrix} \mathbf{S}_{c_i,\sigma}^{(l)} & \\ & \mathbf{S}_{d_j,\delta}^{(l)} \end{pmatrix} \begin{pmatrix} \mathbf{S}_{\sigma,\sigma}^{(l)} & I \\ I & \mathbf{S}_{\delta,\delta}^{(l)} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{S}_{\sigma,c_i}^{(l)} & \\ & \mathbf{S}_{\delta,d_j}^{(l)} \end{pmatrix}. \tag{15}$$

Let $\begin{pmatrix} G_{\sigma,\sigma} & G_{\sigma,\delta} \\ G_{\delta,\sigma} & G_{\delta,\delta} \end{pmatrix} \equiv \begin{pmatrix} \mathbf{S}_{\sigma,\sigma}^{(l)} & I \\ I & \mathbf{S}_{\delta,\delta}^{(l)} \end{pmatrix}^{-1}$. Then (15) can be written into the following individual equations:

$$\mathbf{S}_{c_i,c_j}^{(l-1)} = \mathbf{S}_{c_i,c_j}^{(l)} - \mathbf{S}_{c_i,\sigma}^{(l)} G_{\sigma,\sigma} \mathbf{S}_{\sigma,c_j}^{(l)}, \quad i, j \le s, \tag{16}$$

$$\mathbf{S}_{d_i,d_j}^{(l-1)} = \mathbf{S}_{d_i,d_j}^{(l)} - \mathbf{S}_{d_i,\delta}^{(l)} G_{\delta,\delta} \mathbf{S}_{\delta,d_j}^{(l)}, \quad i, j \le t, \tag{17}$$

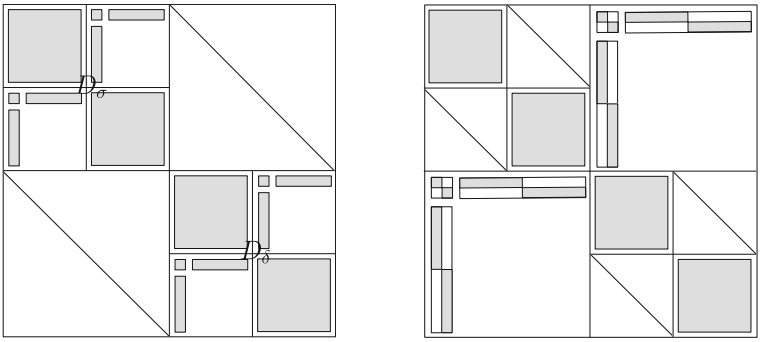
$$\mathbf{S}_{c_i,d_j}^{(l-1)} = -\mathbf{S}_{c_i,\sigma}^{(l)} G_{\sigma,\delta} \mathbf{S}_{\delta,d_j}^{(l)}, \quad i \le s, \quad j \le t, \tag{18}$$

$$\mathbf{S}_{d_j,c_i}^{(l-1)} = -\mathbf{S}_{d_j,\delta}^{(l)} G_{\delta,\sigma} \mathbf{S}_{\sigma,c_i}^{(l)}, \quad i \le s, \quad j \le t. \tag{19}$$

Note that (15) requires $i, j \le \min\{s, t\}$, but (16)–(19) apply to all $1 \le i \le s$, $1 \le j \le t$. For convenience, we will use (15) for some intuitive derivations and the results can be easily adapted to (16)–(19).

The blocks in (16)–(19) for all the interfaces c_i, d_j in the neighbor list associated with γ together form the local Schur complement $S_\gamma^{(l-1)}$. Each block represents the computation of the interaction between a pair of nodes at level $l-1$ of $\mathcal{T}$ based on the interactions at level l.

- (16) shows how $\mathbf{S}_{c_i,c_j}^{(l)}$ is updated to produce $\mathbf{S}_{c_i,c_j}^{(l-1)}$ due to the elimination of $\boldsymbol{\sigma}$. (17) can be similarly understood.
- (18) shows how new fill-in $\mathbf{S}_{c_i,d_j}^{(l-1)}$ is created due to the elimination of $\boldsymbol{\sigma}, \boldsymbol{\delta}$. (19) can be similarly understood.



(a) $\begin{pmatrix} \mathbf{S}^{(l)}_{\sigma,\sigma} & I \\ I & \mathbf{S}^{(l)}_{\delta,\delta} \end{pmatrix}$ **(b)** Matrix H after permutation

Fig. 5 Illustration of a pivot matrix and its permuted form H

We then derive the structured version of (15) and (16)–(19). This involves two steps: structured factorization of the pivot blocks and structured formation of $S^{(l-1)}_\gamma$.

3.3.1 Structured Factorization of the Pivot Matrix

In this step, we compute a structured LDU factorization of the pivot matrix $\begin{pmatrix} \mathbf{S}^{(l)}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} & I \\ I & \mathbf{S}^{(l)}_{\boldsymbol{\delta},\boldsymbol{\delta}} \end{pmatrix}$ in (15). In the IHSS approximations to $S^{(l)}_\alpha$ and $S^{(l)}_\beta$, the blocks $\mathbf{S}^{(l)}_{\boldsymbol{\sigma},\boldsymbol{\sigma}}$ and $\mathbf{S}^{(l)}_{\boldsymbol{\delta},\boldsymbol{\delta}}$ are approximated by standard HSS forms. We suppose the HSS approximations to $\mathbf{S}^{(l)}_{\boldsymbol{\sigma},\boldsymbol{\sigma}}$ and $\mathbf{S}^{(l)}_{\boldsymbol{\delta},\boldsymbol{\delta}}$ have generators

$$\{D_{\sigma_j}, B_{\sigma_j,\mathrm{sib}(\sigma_j)}, U_{\sigma_j}, V_{\sigma_j}, R_{\sigma_j}, W_{\sigma_j}\} \text{ and } \{D_{\delta_j}, B_{\delta_j,\mathrm{sib}(\delta_j)}, U_{\delta_j}, V_{\delta_j}, R_{\delta_j}, W_{\delta_j}\}, \quad (20)$$

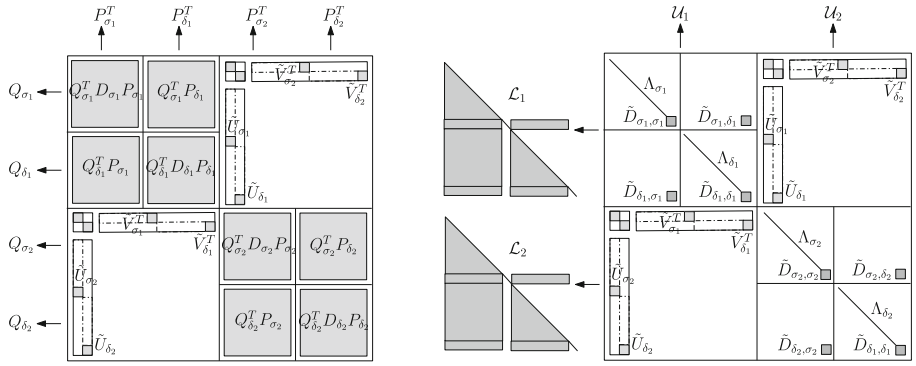
respectively, where σ_j, $j = 1, 2, \ldots$ and δ_j, $j = 1, 2, \ldots$ are the smaller interfaces that form $\boldsymbol{\sigma}$ and $\boldsymbol{\delta}$, respectively, and are from the leaf level of $\mathscr{T}$. (We omitted the superscript l in the D, B generators in this subsection. Also see Remark 1 about the subscripts of the B generators.) The pivot matrix $\begin{pmatrix} \mathbf{S}^{(l)}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} & I \\ I & \mathbf{S}^{(l)}_{\boldsymbol{\delta},\boldsymbol{\delta}} \end{pmatrix}$ can be permuted into an HSS form, and the permutation is defined by rearranging $\{\sigma_1, \sigma_2, \ldots, \delta_1, \delta_2, \ldots\}$ into $\{\sigma_1, \delta_1, \sigma_2, \delta_2, \ldots\}$. Suppose Π is the permutation matrix and the permuted matrix is

$$H = \Pi \begin{pmatrix} \mathbf{S}^{(l)}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} & I \\ I & \mathbf{S}^{(l)}_{\boldsymbol{\delta},\boldsymbol{\delta}} \end{pmatrix} \Pi^T.$$

See Fig. 5 for an example.

By combining the generators of $\mathbf{S}^{(l)}_{\boldsymbol{\sigma},\boldsymbol{\sigma}}$ and $\mathbf{S}^{(l)}_{\boldsymbol{\delta},\boldsymbol{\delta}}$ in (20), we get the generators of the HSS approximation to H as follows:

$$\begin{pmatrix} D_{\sigma_j} & I \\ I & D_{\delta_j} \end{pmatrix}, \begin{pmatrix} B_{\sigma_j,\mathrm{sib}(\sigma_j)} & 0 \\ 0 & B_{\delta_j,\mathrm{sib}(\delta_j)} \end{pmatrix}, \begin{pmatrix} U_{\sigma_j} & 0 \\ 0 & U_{\delta_j} \end{pmatrix}, \begin{pmatrix} V_{\sigma_j} & 0 \\ 0 & V_{\delta_j} \end{pmatrix}, \begin{pmatrix} R_{\sigma_j} & 0 \\ 0 & R_{\delta_j} \end{pmatrix}, \begin{pmatrix} W_{\sigma_j} & 0 \\ 0 & W_{\delta_j} \end{pmatrix}. \quad (21)$$



(a) Introducing zeros into off-diagonal blocks **(b)** Partial diagonal elimination

Fig. 6 Illustration of some major steps of the HSS LDU factorization

(Later, we will abuse notation and still use H to mean its HSS approximation.) Note that only the D generators are coupled. We can design a (nonsymmetric) HSS LDU factorization by modifying a (symmetric) HSS LDL factorization in [35]. However, since the U, V, R, W generators in (21) have block diagonal or decoupled forms, we can design the HSS LDU factorization to preserve these decoupled forms. We also use this factorization to produce results useful for the fast computation of (16)–(19). We traverse the HSS tree of H in a bottom-up order. (It is essentially to simultaneously traverse the HSS trees of $\mathbf{S}^{(l)}_{\boldsymbol{\sigma},\boldsymbol{\sigma}}$ and $\mathbf{S}^{(l)}_{\boldsymbol{\delta},\boldsymbol{\delta}}$.)

If σ_j and δ_j are leaf nodes, we introduce zeros into the corresponding basis matrices using QL factorizations

$$U_{\sigma_j} = Q_{\sigma_j}\begin{pmatrix} 0 \\ \tilde{U}_{\sigma_j} \end{pmatrix}, \quad V_{\sigma_j} = P_{\sigma_j}\begin{pmatrix} 0 \\ \tilde{V}_{\sigma_j} \end{pmatrix}, \quad U_{\delta_j} = Q_{\delta_j}\begin{pmatrix} 0 \\ \tilde{U}_{\delta_j} \end{pmatrix}, \quad V_{\delta_j} = P_{\delta_j}\begin{pmatrix} 0 \\ \tilde{V}_{\delta_j} \end{pmatrix}. \tag{22}$$

Then apply $Q^T_{\sigma_j}$ on the left to the block row of H corresponding to σ_j and apply P_{σ_j} on the right to the block column of H corresponding to σ_j. Similarly, apply $Q^T_{\delta_j}$ and P_{δ_j}. These introduce zero blocks into the corresponding off-diagonal blocks as in standard HSS ULV factorizations [5]. The corresponding diagonal blocks are transformed to

$$\begin{pmatrix} \bar{D}_{\sigma_j,\sigma_j} & \bar{D}_{\sigma_j,\delta_j} \\ \bar{D}_{\delta_j,\sigma_j} & \bar{D}_{\delta_j,\delta_j} \end{pmatrix} \equiv \begin{pmatrix} Q^T_{\sigma_j} D_{\sigma_j} P_{\sigma_j} & Q^T_{\sigma_j} P_{\delta_j} \\ Q^T_{\delta_j} P_{\sigma_j} & Q^T_{\delta_j} D_{\delta_j} P_{\delta_j} \end{pmatrix}. \tag{23}$$

Accordingly, suppose H is transformed into $\bar{H}$. See Fig. 6a for an illustration.

Then partition the blocks of (23) conformably following (22) and compute the following partial LDU factorization:

$$\begin{pmatrix} \bar{D}_{\sigma_j,\sigma_j} & \bar{D}_{\sigma_j,\delta_j} \\ \bar{D}_{\delta_j,\sigma_j} & \bar{D}_{\delta_j,\delta_j} \end{pmatrix} = \mathscr{L}_j \mathscr{D}_j \mathscr{U}_j, \quad \text{with} \quad \mathscr{D}_j = \begin{pmatrix} \Lambda_{\sigma_j} & & \tilde{D}_{\sigma_j,\delta_j} \\ & \Lambda_{\delta_j} & \\ \tilde{D}_{\delta_j,\sigma_j} & & \tilde{D}_{\delta_j,\delta_j} \end{pmatrix},$$

$$\mathscr{L}_j = \begin{pmatrix} \mathscr{L}_{\sigma_j,\sigma_j} & & \tilde{\mathscr{L}}_{\sigma_j,\delta_j} & \\ \tilde{\mathscr{L}}_{\sigma_j,\sigma_j} & I & & \\ \mathscr{L}_{\delta_j,\sigma_j} & & \mathscr{L}_{\delta_j,\delta_j} & \\ \tilde{\mathscr{L}}_{\delta_j,\sigma_j} & & \tilde{\mathscr{L}}_{\delta_j,\delta_j} & I \end{pmatrix}, \quad \mathscr{U}_j = \begin{pmatrix} \mathscr{U}_{\sigma_j,\sigma_j} & \tilde{\mathscr{U}}_{\sigma_j,\sigma_j} & \mathscr{U}_{\sigma_j,\delta_j} & \tilde{\mathscr{U}}_{\sigma_j,\delta_j} \\ & I & & \\ & \tilde{\mathscr{U}}_{\delta_j,\sigma_j} & \mathscr{U}_{\delta_j,\delta_j} & \tilde{\mathscr{U}}_{\delta_j,\delta_j} \\ & & & I \end{pmatrix},$$

where Λ_{σ_j} and Λ_{δ_j} are diagonal matrices and match the zero positions in (22). The purpose for such a factorization is partial elimination. That is, we apply $\mathscr{L}_j^{-1}$ on the left to the block row of $\bar{H}$ corresponding to (23) and apply $\mathscr{U}_j^{-1}$ on the right to the block column of $\bar{H}$ corresponding to (23), and suppose the resulting matrix is $\tilde{H}$. Then Λ_{σ_j} and Λ_{δ_j} can be eliminated from $\tilde{H}$. In addition, this partial elimination does not impact the modified basis matrices $\tilde{U}_{\sigma_j}, \tilde{V}_{\sigma_j}, \tilde{U}_{\delta_j}, \tilde{V}_{\delta_j}$ previously computed in (22). See Fig. 6b.

If σ_i and δ_i are non-leaf nodes, suppose they have children μ_1, μ_2 and ν_1, ν_2, respectively. Partial eliminations have been performed when these child nodes are visited. The remaining blocks that are not eliminated at the child level can be merged to form some new HSS generators. Let

$$\hat{D}_{\sigma_j} = \begin{pmatrix} \tilde{D}_{\mu_1,\mu_1} & \tilde{U}_{\mu_1} B_{\mu_1,\mu_2} \tilde{V}_{\mu_2}^T \\ \tilde{U}_{\mu_2} B_{\mu_2,\mu_1} \tilde{V}_{\mu_1}^T & \tilde{D}_{\mu_2,\mu_2} \end{pmatrix}, \quad \hat{U}_{\sigma_j} = \begin{pmatrix} \tilde{U}_{\mu_1} R_{\mu_1} \\ \tilde{U}_{\mu_2} R_{\mu_2} \end{pmatrix}, \quad \hat{V}_{\sigma_j} = \begin{pmatrix} \tilde{V}_{\mu_1} W_{\mu_1} \\ \tilde{V}_{\mu_2} W_{\mu_2} \end{pmatrix},$$

$$\hat{D}_{\delta_j} = \begin{pmatrix} \tilde{D}_{\nu_1,\nu_1} & \tilde{U}_{\nu_1} B_{\nu_1,\nu_2} \tilde{V}_{\nu_2}^T \\ \tilde{U}_{\nu_2} B_{\nu_2,\nu_1} \tilde{V}_{\nu_1}^T & \tilde{D}_{\nu_2,\nu_2} \end{pmatrix}, \quad \hat{U}_{\delta_j} = \begin{pmatrix} \tilde{U}_{\nu_1} R_{\nu_1} \\ \tilde{U}_{\nu_2} R_{\nu_2} \end{pmatrix}, \quad \hat{V}_{\delta_j} = \begin{pmatrix} \tilde{V}_{\nu_1} W_{\nu_1} \\ \tilde{V}_{\nu_2} W_{\nu_2} \end{pmatrix},$$

Associate σ_j with HSS generators $\hat{D}_{\sigma_j}, B_{\sigma_j,\mathrm{sib}(\sigma_j)}, \hat{U}_{\sigma_j}, \hat{V}_{\sigma_j}, R_{\sigma_j}, W_{\sigma_j}$ and associate δ_j with HSS generators $\hat{D}_{\delta_j}, B_{\delta_j,\mathrm{sib}(\delta_j)}, \hat{U}_{\delta_j}, \hat{V}_{\delta_j}, R_{\delta_j}, W_{\delta_j}$. Then we can remove the children of σ_j and δ_j from their associated HSS trees respectively. Accordingly, their associated HSS forms can be merged and permuted into a larger HSS form with generators similar to (21) but with small differences:

$$\begin{pmatrix} \hat{D}_{\sigma_j} & \mathrm{diag}(\tilde{D}_{\mu_1,\nu_1}, \tilde{D}_{\mu_2,\nu_2}) \\ \mathrm{diag}(\tilde{D}_{\nu_1,\mu_1}, \tilde{D}_{\nu_2,\mu_2}) & \hat{D}_{\delta_j} \end{pmatrix}, \begin{pmatrix} B_{\sigma_j,\mathrm{sib}(\sigma_j)} & 0 \\ 0 & B_{\delta_j,\mathrm{sib}(\delta_j)} \end{pmatrix},$$
$$\begin{pmatrix} \hat{U}_{\sigma_j} & 0 \\ 0 & \hat{U}_{\delta_j} \end{pmatrix}, \begin{pmatrix} \hat{V}_{\sigma_j} & 0 \\ 0 & \hat{V}_{\delta_j} \end{pmatrix}, \begin{pmatrix} R_{\sigma_j} & 0 \\ 0 & R_{\delta_j} \end{pmatrix}, \begin{pmatrix} W_{\sigma_j} & 0 \\ 0 & W_{\delta_j} \end{pmatrix}.$$

where diag() is used to denote a block diagonal matrix. Then the elimination process above can be similarly applied to σ_j and δ_j.

This process is repeated in a bottom-up sweep. If σ_j and δ_j are root nodes, after the factorizations, we also store the following matrix:

$$\tilde{G} = \begin{pmatrix} \tilde{V}_{\sigma_j}^T & \\ & \tilde{V}_{\delta_j}^T \end{pmatrix} \begin{pmatrix} \tilde{D}_{\sigma_j,\sigma_j} & \tilde{D}_{\sigma_j,\delta_j} \\ \tilde{D}_{\delta_j,\sigma_j} & \tilde{D}_{\delta_j,\delta_j} \end{pmatrix}^{-1} \begin{pmatrix} \tilde{U}_{\sigma_j} & \\ & \tilde{U}_{\delta_j} \end{pmatrix}. \tag{24}$$

$\tilde{G}$ is a small matrix and will be used a little later in (28) to facilitate the fast computation of (16)–(19).

3.3.2 Structured Formation of $S_\gamma^{(l-1)}$

With the pivot separators eliminated from $S_\alpha^{(l)}$ and $S_\beta^{(l)}$, we then show how to find an IHSS approximation to the local Schur complement $S_\gamma^{(l-1)}$ based on (15) or (16)–(19). Suppose

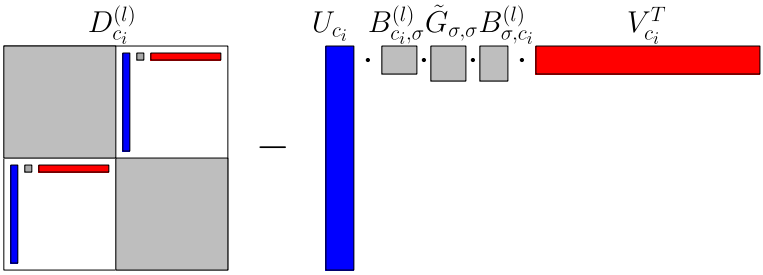


Fig. 7 Illustration of using (29) to update D generators of local Schur complements

the generators of the IHSS approximations to $S_\alpha^{(l)}$ and $S_\beta^{(l)}$ associated with c_j and d_j in (13) are

$$\{D_{c_j}^{(l)}, B_{c_j,*}^{(l)}, U_{c_j}, V_{c_j}, R_{c_j}, W_{c_j}\} \quad \text{and} \quad \{D_{d_j}^{(l)}, B_{d_j,*}^{(l)}, U_{d_j}, V_{d_j}, R_{d_j}, W_{d_j}\}, \tag{25}$$

respectively, where the superscript l in the D, B generators is used to indicate the level dependence. The U, V, R, W generators are related to the off-diagonal bases and will be reused across different levels l in our interconnected structured method.

With the generators in (25), the second term on the right-hand side of (15) can be approximated by a structured form

$$\begin{aligned}
&\begin{pmatrix} U_{c_i} B_{c_i,\boldsymbol{\sigma}}^{(l)} V_{\boldsymbol{\sigma}}^T & \\ & U_{d_j} B_{d_j,\boldsymbol{\delta}}^{(l)} V_{\boldsymbol{\delta}}^T \end{pmatrix} \begin{pmatrix} \mathbf{S}_{\boldsymbol{\sigma},\boldsymbol{\sigma}}^{(l)} & I \\ I & \mathbf{S}_{\boldsymbol{\delta},\boldsymbol{\delta}}^{(l)} \end{pmatrix}^{-1} \begin{pmatrix} U_{\boldsymbol{\sigma}} B_{\boldsymbol{\sigma},c_i}^{(l)} V_{c_i}^T & \\ & U_{\boldsymbol{\delta}} B_{\boldsymbol{\delta},d_j}^{(l)} V_{d_j}^T \end{pmatrix} \\
&= \begin{pmatrix} U_{c_i} & \\ & U_{d_j} \end{pmatrix} \begin{pmatrix} B_{c_i,\boldsymbol{\sigma}}^{(l)} & \\ & B_{d_j,\boldsymbol{\delta}}^{(l)} \end{pmatrix} \begin{pmatrix} \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} & \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\delta}} \\ \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\sigma}} & \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\delta}} \end{pmatrix} \begin{pmatrix} B_{\boldsymbol{\sigma},c_i}^{(l)} & \\ & B_{\boldsymbol{\delta},d_j}^{(l)} \end{pmatrix} \begin{pmatrix} V_{c_i}^T & \\ & V_{d_j}^T \end{pmatrix},
\end{aligned} \tag{26}$$

where

$$\begin{pmatrix} \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} & \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\delta}} \\ \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\sigma}} & \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\delta}} \end{pmatrix} = \begin{pmatrix} V_{\boldsymbol{\sigma}}^T & \\ & V_{\boldsymbol{\delta}}^T \end{pmatrix} \begin{pmatrix} \mathbf{S}_{\boldsymbol{\sigma},\boldsymbol{\sigma}}^{(l)} & I \\ I & \mathbf{S}_{\boldsymbol{\delta},\boldsymbol{\delta}}^{(l)} \end{pmatrix}^{-1} \begin{pmatrix} U_{\boldsymbol{\sigma}} & \\ & U_{\boldsymbol{\delta}} \end{pmatrix}. \tag{27}$$

Note that there is no need to use extra computations to directly form (27). Instead, following the HSS LDU factorization in Sect. 3.3.1, an idea of reduced matrices in [38,Theorem 3.1] and [37,Theorem 3.2] can be used to show that (27) is actually given by $\tilde{G}$ in (24):

$$\begin{pmatrix} \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} & \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\delta}} \\ \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\sigma}} & \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\delta}} \end{pmatrix} = \tilde{G}. \tag{28}$$

This avoids the use of HSS solutions and multiplications to form (27).

We then discuss how to quickly obtain the structured form of the local Schur complement $S_\gamma^{(l-1)}$ in $\mathbf{S}^{(l-1)}$. There are three cases to consider.

I. For the diagonal blocks $\mathbf{S}_{c_i,c_i}^{(l-1)}$ and $\mathbf{S}_{d_j,d_j}^{(l-1)}$ corresponding to interactions within the interfaces c_i and d_j, respectively, by substituting the IHSS approximations given by (25) into

(16) and (17), we get

$$\begin{aligned}\mathbf{S}_{c_i,c_i}^{(l-1)} &\approx D_{c_i}^{(l)} - U_{c_i} B_{c_i,\boldsymbol{\sigma}}^{(l)} \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} B_{\boldsymbol{\sigma},c_i}^{(l)} V_{c_i}^T \equiv D_{c_i}^{(l-1)}, \quad i \le s,\\ \mathbf{S}_{d_j,d_j}^{(l-1)} &\approx D_{d_j}^{(l)} - U_{d_j} B_{d_j,\boldsymbol{\delta}}^{(l)} \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\delta}} B_{\boldsymbol{\delta},d_j}^{(l)} V_{d_j}^T \equiv D_{d_j}^{(l-1)}, \quad j \le t.\end{aligned} \tag{29}$$

The two equations describe how to compute a generator $D^{(l-1)}$ from $D^{(l)}$ in the form of a low-rank update. Note that c_i is allowed to include lower level interfaces so that $D_{c_i}^{(l)}$ in (29) itself is in an HSS form. (It is similar for d_i and $D_{d_i}^{(l)}$.) Figure 7 illustrates this. In this case, the off-diagonal blocks of $D_{c_i}^{(l)}$ have U and V basis matrices related to U_{c_i} and V_{c_i}, respectively. This is due to the nested bases in the HSS structure [40]. A result in [37,Proposition 3.3] can then be used to quickly update the lower level D, B generators within $D_{c_i}^{(l)}$ to get those of $D_{c_i}^{(l-1)}$. The basis generators U, V (and R, W) remain the same.

II. For the off-diagonal block $\mathbf{S}_{c_i,c_j}^{(l-1)}$ corresponding to interactions between c_i and c_j (and also $\mathbf{S}_{d_i,d_j}^{(l-1)}$ corresponding to interactions between d_i and d_j), where $i \ne j$, we get also from (16) and (17) that

$$\begin{aligned}\mathbf{S}_{c_i,c_j}^{(l-1)} &\approx U_{c_i} \left(B_{c_i,c_j}^{(l)} - B_{c_i,\boldsymbol{\sigma}}^{(l)} \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} B_{\boldsymbol{\sigma},c_j}^{(l)} \right) V_{c_j}^T \equiv U_{c_i} B_{c_i,c_j}^{(l-1)} V_{c_j}^T, \quad i,j \le s, \quad i \ne j,\\ \mathbf{S}_{d_i,d_j}^{(l-1)} &\approx U_{d_i} \left(B_{d_i,d_j}^{(l)} - B_{d_i,\boldsymbol{\delta}}^{(l)} \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\delta}} B_{\boldsymbol{\delta},d_j}^{(l)} \right) V_{d_j}^T \equiv U_{d_i} B_{d_i,d_j}^{(l-1)} V_{d_j}^T, \quad i,j \le t, \quad i \ne j.\end{aligned}$$

Thus, we obtain low-rank approximations to $\mathbf{S}_{c_i,c_j}^{(l-1)}$ and $\mathbf{S}_{d_i,d_j}^{(l-1)}$ by simply reusing existing U, V basis matrices. We just need to update the B generators as follows:

$$\begin{aligned}B_{c_i,c_j}^{(l-1)} &= B_{c_i,c_j}^{(l)} - B_{c_i,\boldsymbol{\sigma}}^{(l)} \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} B_{\boldsymbol{\sigma},c_j}^{(l)}, \quad i,j \le s, \quad i \ne j,\\ B_{d_i,d_j}^{(l-1)} &= B_{d_i,d_j}^{(l)} - B_{d_i,\boldsymbol{\delta}}^{(l)} \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\delta}} B_{\boldsymbol{\delta},d_j}^{(l)}, \quad i,j \le t, \quad i \ne j.\end{aligned} \tag{30}$$

Again, the U, V (and R, W) generators remain the same. See Fig. 8 for an illustration.

III. For the off-diagonal blocks $\mathbf{S}_{c_i,d_j}^{(l-1)}$ and $\mathbf{S}_{d_j,c_i}^{(l-1)}$ corresponding to new fill-in, we get from (18) and (19) that

$$\begin{aligned}\mathbf{S}_{c_i,d_j}^{(l-1)} &\approx -U_{c_i} B_{c_i,\boldsymbol{\sigma}}^{(l)} \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\delta}} B_{\boldsymbol{\delta},d_j}^{(l)} V_{d_j}^T \equiv U_{c_i} B_{c_i,d_j}^{(l-1)} V_{d_j}^T, \quad i \le s, \quad j \le t,\\ \mathbf{S}_{d_j,c_i}^{(l-1)} &\approx -U_{d_j} B_{d_j,\boldsymbol{\delta}}^{(l)} \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\sigma}} B_{\boldsymbol{\sigma},c_i}^{(l)} V_{c_i}^T \equiv U_{d_j} B_{d_j,c_i}^{(l-1)} V_{c_i}^T, \quad i \le s, \quad j \le t.\end{aligned}$$

Thus, we also obtain low-rank approximations to $\mathbf{S}_{c_i,d_j}^{(l-1)}$ and $\mathbf{S}_{d_j,c_i}^{(l-1)}$ by simply reusing existing U, V basis matrices. We just need to create new B generators as follows:

$$\begin{aligned}B_{c_i,d_j}^{(l-1)} &= -B_{c_i,\boldsymbol{\sigma}}^{(l)} \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\delta}} B_{\boldsymbol{\delta},d_j}^{(l)}, \quad i \le s, \quad j \le t,\\ B_{d_j,c_i}^{(l-1)} &= -B_{d_j,\boldsymbol{\delta}}^{(l)} \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\sigma}} B_{\boldsymbol{\sigma},c_i}^{(l)}, \quad i \le s, \quad j \le t.\end{aligned} \tag{31}$$

See Fig. 9.

From the three cases (29)–(31) we can observe that existing U, V basis matrices for relevant off-diagonal blocks of $S_\alpha^{(l)}$ and $S_\beta^{(l)}$ are used directly for the off-diagonal blocks of $S_\gamma^{(l-1)}$ and only the D, B generators need to be updated. That is, *the off-diagonal basis information is reused for local Schur complements across different levels l without extra*

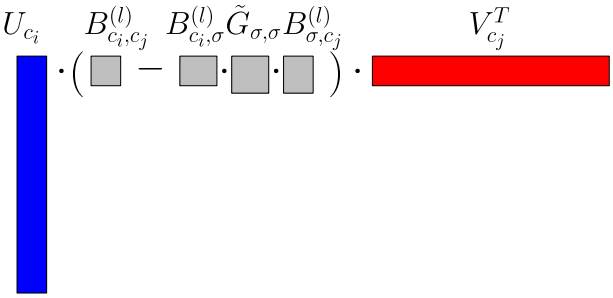


Fig. 8 Illustration of using (30) to update B generators of local Schur complements

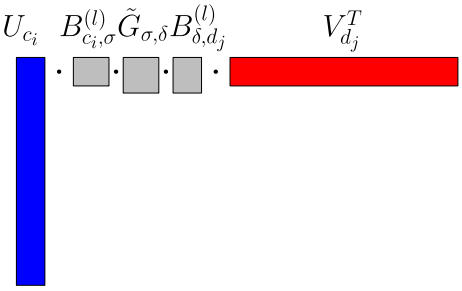


Fig. 9 Illustration of using (31) to create B generators of new fill-in in local Schur complements

compression. This confirms the third feature in Sect. 3.2 when IHSS representations are introduced.

We then finalize the IHSS representation for $S_\gamma^{(l-1)}$. The neighbor list associated with γ can be obtained by collecting the interfaces $c_1, \ldots, c_s$ and $d_1, \ldots, d_t$ in (13). Accordingly, the corresponding HSS subtrees together with the associated U, V basis matrices are preserved. The collection of the HSS subtrees is used to form the IHSS tree for $S_\gamma^{(l-1)}$. There is only one thing left. That is, we need to merge common-origin interfaces c_j and d_k that are in α and β separately. The corresponding HSS subtrees also need to be merged into a larger HSS subtree by introducing one additional hierarchical level. Without loss of generality, suppose c_1 and d_1 are common-origin interfaces (belonging to a single separator at level $l-1$ of the separator tree $\mathbf{T}$) and is to be combined into an interface $e \equiv (c_1, d_1)$. According to [40,Algorithm 1], this just needs to introduce the generators $\begin{pmatrix} R_{c_1} \\ R_{d_1} \end{pmatrix}$ and $\begin{pmatrix} W_{c_1} \\ W_{d_1} \end{pmatrix}$ into the nested basis structure. For example, $\begin{pmatrix} R_{c_1} \\ R_{d_1} \end{pmatrix}$ is an approximate column basis matrix of

$$\begin{pmatrix} B^{(l-1)}_{c_1,c_2} & \ldots & B^{(l-1)}_{c_1,c_s} & B^{(l-1)}_{c_1,d_2} & \ldots & B^{(l-1)}_{c_1,d_t} \\ B^{(l-1)}_{d_1,c_2} & \ldots & B^{(l-1)}_{d_1,c_s} & B^{(l-1)}_{d_1,d_2} & \ldots & B^{(l-1)}_{d_1,d_t} \end{pmatrix}. \tag{32}$$

Since s and t are usually very small, the size of the matrix in (32) is small. A rank-revealing QR factorization can be applied to (32):

$$\begin{pmatrix} B^{(l-1)}_{c_1,c_2} & \ldots & B^{(l-1)}_{c_1,c_s} & B^{(l-1)}_{c_1,d_2} & \ldots & B^{(l-1)}_{c_1,d_t} \\ B^{(l-1)}_{d_1,c_2} & \ldots & B^{(l-1)}_{d_1,c_s} & B^{(l-1)}_{d_1,d_2} & \ldots & B^{(l-1)}_{d_1,d_t} \end{pmatrix} \approx \begin{pmatrix} R_{c_1} \\ R_{d_1} \end{pmatrix} \begin{pmatrix} B^{(l-1)}_{e,c_2} & \ldots & B^{(l-1)}_{e,c_s} & B^{(l-1)}_{e,d_2} & \ldots & B^{(l-1)}_{e,d_t} \end{pmatrix}. \tag{33}$$

The new B generators on the right-hand side are introduced due to the creation of e. After all such merging steps, we obtain an IHSS approximation to $S_\gamma^{(l-1)}$.

Applying the above local Schur complement updates to all the interface lists at level l of $\mathscr{T}$ produces the IHSS approximations to local Schur complements at level $l-1$. That is, we get a structured approximation to $\mathbf{S}^{(l-1)}$. This completes the Schur complement update problem (4) at level l. The process can then be repeated, until level 1 of $\mathscr{T}$ is reached. At this point, we only need to compute an HSS LDU factorization as in Sect. 3.3.1.

This factorization process produces structured factors associated with each node of $\mathscr{T}$. The factors can be used to quickly solve linear systems in (5)–(7). The structured solution algorithms can be designed conveniently and the details are omitted.

3.4 Advantages and Complexity

Our solver performs the Schur complement update via the fast structured updates (29)–(31). Relevant off-diagonal basis matrices U, V are computed only once (for small blocks) and then reused across different outer factorization levels without the need for repeated direct low-rank compression. This feature helps to both save the cost and guarantee that the rank structure is fully preserved in the sparse factorization. In comparison, rank-structured multifrontal solvers like those in [10, 37, 39] cannot conveniently keep consistent block partitioning or reuse HSS generators due to a nontrivial process (called extend-add) for assembling intermediate structured Schur complements. This process needs to reorder, extend, and match different separators. The assembled frontal matrices are then approximated by rank-structured matrices based on algebraic or randomized compression. Therefore, it is not convenient to explore IHSS structures in rank-structured multifrontal solvers. On the other hand, rank-structured multifrontal methods can solve more general problems and do not rely on specific boundary conditions. The Dirichlet-to-Neumann formulation [12, 26] is closely related to the Robin-to-Robin formulation. It is more commonly used since it is easier to derive. However, it is more complicated for structured factorizations using HSS matrices because the addition and recompression of HSS forms are required in order to factorize pivot blocks [12,Section 7.2]. HSS recompression is a tricky process and comes with additional costs and approximation errors. With more efforts, we believe that IHSS structures may be explored within the Dirichlet-to-Neumann formulation as well. The factorization of pivot matrices in Sect. 3.3.1 will be more technical in order to have structured addition and recompression. After that, the formation of the local Schur compliments in Sect. 3.3.2 might be similar.

The complexity of our algorithm can be studied as follows. Let n be the size of the discretized elliptic problem in two dimensions and $\mathbf{l}$ be the total number of levels in $\mathscr{T}$. Assume all the finest-level subdomains have constant problem sizes independent of n, so that factorizing all the corresponding subproblems at level $\mathbf{l}$ has total linear complexity. That is, we let

$$2^{\mathbf{l}} = O(n). \tag{34}$$

Like the methods in [38, 39], to achieve the optimal complexity, the structured construction typically does not immediately start from the finest level. Instead, a switching level $\mathbf{l}_s (\leq \mathbf{l})$ is chosen. That is, structured factorizations are applied just to the upper $\mathbf{l}_s$ levels and dense operations are applied to the bottom $\mathbf{l} - \mathbf{l}_s$ levels. This also avoids applying rank-structured operations to very small dense matrices. To estimate the total complexity, suppose N_l is the largest size of the local Schur complement at level l, and r is the maximum size of all the B generators in the IHSS approximations in the sparse factorization.

At a level l below the switching level, the factorization cost associated with each local Schur complement is $O(N_l^3)$. At the switching level $l = \mathbf{l}_s$, a (one-time) compression cost is used to approximate each local Schur complement as an IHSS form. The cost is $O(rN_l^2)$ like in the HSS construction in [40]. Then for $l \le \mathbf{l}_s$, structured factorizations are performed as in Sects. 3.3.1 and 3.3.2. Note that the U, V generators are reused for upper levels so the only extra compression is for a small matrix as in (33). The cost associated with each local Schur complement is $O(r^2 N_l)$. Without loss of generality, we assume $N_l = O(\sqrt{n/2^l})$, and the total factorization complexity is

$$\underbrace{O\left(2^{\mathbf{l}}\right)}_{\text{factorizations for finest-level subdomains}} + \underbrace{\sum_{l=\mathbf{l}_s+1}^{\mathbf{l}} 2^l O\left(N_l^3\right)}_{\text{dense Schur complement factorizations}} + \underbrace{2^{\mathbf{l}_s} O\left(rN_{\mathbf{l}_s}^2\right)}_{\substack{\text{IHSS constructions} \\ \text{at level } \mathbf{l}_s}}$$

$$+ \underbrace{\sum_{l=1}^{\mathbf{l}_s} 2^l O(r^2 N_l)}_{\text{IHSS factorizations}} = O(n) + O(2^{-\mathbf{l}_s/2} n^{3/2}) + O(rn) + O(r^2 2^{\mathbf{l}_s/2} n^{1/2}).$$

To minimize this count, we choose

$$2^{\mathbf{l}_s} = O(r^{-2} n). \tag{35}$$

(34) and (35) indicate $\mathbf{l} - \mathbf{l}_s = O(\log r)$. This leads to the optimal factorization complexity $O(rn)$. Accordingly, the storage count (number of nonzeros in structured factors) looks like

$$\underbrace{O(2^{\mathbf{l}})}_{\text{factors for finest-level subdomains}} + \underbrace{\sum_{l=\mathbf{l}_s+1}^{\mathbf{l}} 2^l O\left(N_l^2\right)}_{\text{factors of dense Schur complements}} + \underbrace{\sum_{l=1}^{\mathbf{l}_s} 2^l O(rN_l)}_{\text{IHSS factors}}$$

$$= O(n) + O((\mathbf{l} - \mathbf{l}_s)n) + O(n) = O((\log r)n).$$

In comparison, if a structured multifrontal method like in [32, 38] is applied, the optimal factorization cost is $O(rn \log n)$, where r is the maximum size of the B generators in the HSS approximations to the local Schur complements. Thus, assuming the same r is used, our factorization cost is lower by a factor of $O(\log n)$.

If r is bounded, then the method has a linear factorization complexity. In practice, r is allowed to slightly increase with n as $r = O(\log^p n_l)$. Then the total complexity remains $O(n)$ based on a rank relaxation study in [36, 38]. For 2D elliptic equations and Helmholtz equations with small or complex-valued wavenumbers, we expect r to be small [2, 4, 8].

4 Extension to Structured Factorization Update Under Multiple Coefficient Changes

Our algorithm can be extended to a more challenging situation where the factorization needs to be updated frequently due to various local changes to the coefficient of the PDE. The magnitudes of the local changes may also be large. For such a situation, conventional sparse hierarchical factorization becomes too expensive since any local update will be propagated upward along the assembly tree. Recently in [24], a fast factorization update method was proposed that uses a set of interior and exterior factorizations. Whenever a subdomain is

updated, only a small interior factorization in that subdomain needs to be done. The updated interior factor is combined with a set of existing exterior factors to produce the updated sparse factorization.

The factorization update method in [24] uses a formulation consistent with Sect. 2. Thus, we can use our interconnected hierarchical structured methods to accelerate the factorization update. We sketch the main ideas here without going into the tedious technical details and show some numerical tests later.

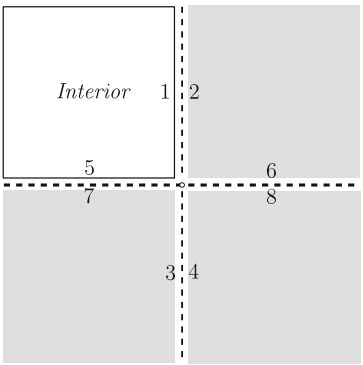
- The method in [24] starts with the hierarchical direct factorization described in Sect. 2 here. To distinguish from additional steps needed for the factorization update, the subdomains generated by the hierarchical domain partitioning as in Sect. 2.1 are called *interior subdomains*, and the factorization or elimination of unknowns associated with each interior subdomain is called an *interior factorization*. As discussed in Sect. 3, the IHSS factorization can be used to replace the standard interior factorization.
- The complement of an interior subdomain is called an *exterior subdomain*. In [24], an *exterior factorization* method is developed to take advantage of shared factors during the elimination of unknowns in exterior subdomains. It is shown in [24] that every exterior subdomain is a union of certain interior subdomains. Therefore, an exterior factorization can always reuse the results of certain interior factorizations to save the cost. A neighbor tree can also be constructed for the exterior factorization.
- For the factorization update due to coefficient updates in an interior subdomain, we can compute the new factorization by combining the new factors in that interior subdomain and existing factors in the corresponding exterior subdomain. This avoid the propagation of local updates to the entire sparse factorization. See [24,Section 3] for more details.
- IHSS approximations can be used to accelerate the exterior factorizations as well. When an existing factor is involved, the existing tree structures and IHSS approximations are reused. The existing formulas (29)–(31) still hold. On the boundary of a subdomain, the intermediate Schur complements from the interior and exterior factorization have the same size and the same block partitioning.

As an example, consider the matrix in (2) with the domain partitioning shown in Fig. 1a. Suppose there are coefficient updates in the upper-left subdomain which is the interior subdomain (Fig. 10a). The exterior subdomain is the union of the three remaining subdomains. We can transform the neighbor tree in Fig. 2a to get a new neighbor tree $\widetilde{\mathscr{T}}$ to organize the factorizations (Fig. 10b). The exterior factorization can be computed by eliminating the separators (3, 4) and (6, 8) following $\widetilde{\mathscr{T}}$. The elimination of (3, 4) has been performed in (9) during the interior factorization and is reused. Thus, the corresponding subtree of $\mathscr{T}$ is reused in $\widetilde{\mathscr{T}}$. See the dash-dotted subtree in Fig. 10b. The elimination of (6, 8) can be performed similarly afterwards by

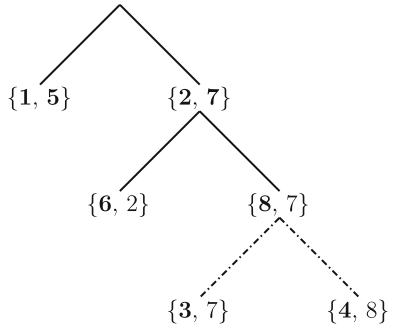
$$\begin{pmatrix} \mathbf{S}_{22}^{(1)} & \mathbf{S}_{27}^{(1)} \\ \mathbf{S}_{72}^{(1)} & \mathbf{S}_{77}^{(1)} \end{pmatrix} = \begin{pmatrix} \mathbf{S}_{22}^{(2)} & \\ & \mathbf{S}_{77}^{(1)} \end{pmatrix} - \begin{pmatrix} \mathbf{S}_{26}^{(2)} & \\ & \mathbf{S}_{78}^{(1)} \end{pmatrix} \begin{pmatrix} \mathbf{S}_{66}^{(2)} & I \\ I & \mathbf{S}_{88}^{(1)} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{S}_{62}^{(2)} & \\ & \mathbf{S}_{87}^{(1)} \end{pmatrix}.$$

After changing the coefficients in the interior subdomain, we only need to update the interior factorization in that subdomain, which corresponds to the left child of the root in Fig. 10b. No update is needed for the exterior factorization (corresponding to the entire right subtree of the root in Fig. 10b). The updated interior factor is then combined with the existing exterior factors to get the new overall factor. This avoids the global propagation of local updates. The reader is referred to [24] for more details. We would just like to point out that our IHSS strategies can be used to accelerate all these elimination steps just like in

$$\begin{pmatrix} \mathbf{S}_{22}^{(1)} & \mathbf{S}_{27}^{(1)} \\ \mathbf{S}_{72}^{(1)} & \mathbf{S}_{77}^{(1)} \end{pmatrix} = \begin{pmatrix} \mathbf{S}_{22}^{(2)} & \\ & \mathbf{S}_{77}^{(1)} \end{pmatrix} - \begin{pmatrix} \mathbf{S}_{26}^{(2)} & \\ & \mathbf{S}_{78}^{(1)} \end{pmatrix} \begin{pmatrix} \mathbf{S}_{66}^{(2)} & I \\ I & \mathbf{S}_{88}^{(1)} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{S}_{62}^{(2)} & \\ & \mathbf{S}_{87}^{(1)} \end{pmatrix}.$$



(a) Interior and exterior subdomains



(b) New neighbor tree $\widetilde{\mathscr{T}}$

Fig. 10 A pair of interior and exterior subdomains and the corresponding neighbor tree $\widetilde{\mathscr{T}}$ for organizing the factorizations. The exterior subdomain is formed by the shaded areas. The neighbor tree is transformed from the tree $\mathscr{T}$ in Fig. 2a. The dash-dotted subtree is reused from $\mathscr{T}$

Sect. 3. The complexity improvement is similar to that in Sect. 3. In the next section, a test example will be included to show the performance improvement.

5 Numerical Examples

We apply our interconnected hierarchical structured method to solve some elliptic PDEs. The PDEs are discretized with a continuous Galerkin method and the coefficients may contain large jump discontinuities between elements. We also test a challenging local factorization update problem. As mentioned before, our main purpose is to verify the feasibility of IHSS structures in a proof-of-concept study. Thus, the tests are done for some 2D problems, which is sufficient to demonstrate the benefits. The ideas can be extended to 3D problems but the implementation would be much more technical. We compare the following three methods:

- NEW: our new sparse factorization based on IHSS methods;
- STD: the standard supernodal multifrontal factorization using nested dissection (without intermediate structured approximations);
- SMF: a structured multifrontal factorization with HSS approximations of the intermediate frontal matrices like in [38];

In the actual implementations of NEW, we do not need to assemble the entire matrix explicitly. Instead, we follow the geometric multifrontal patterns as illustrated in Sects. 2 and 3, where only local PDE problems and Schur complements are formed and factorized. Due to the use of Robin-to-Robin formations, the local problems serve as approximations to part of the global problem. In order to be able to apply STD and SMF so as to perform comparison, the global matrix is assembled explicitly from the subproblems (using finite element methods). Solutions from STD applied to the global matrix are treated as the exact

solutions. The approximation natures contribute to some accuracy differences between NEW and SMF with respect to STD. This overall should not affect our illustration of the reduction in compression operations with the IHSS strategy. Note that the global matrix has a smaller number of nonzeros than all the local subproblems together due to overlapping interfaces in the subproblems, which brings a slight advantage to SMF.

In our IHSS approximations, we set a 2-norm relative tolerance $\tau = 10^{-6}$ in the compression at the switching level $\mathbf{l}_s$ and the compression step (33). This tolerance is also used for the HSS construction in SMF. All the compression is based on an adaptive randomized rank-revealing QR routine like those in [13, 35] and with τ as a relative tolerance. The implementation involves economy-sized SVDs of some small skinny matrices and their flop count follows [31].

We test the algorithms in Matlab R2021a and report the following results.

- Factorization costs in terms of flops.
- Storage of factors in terms of the number of nonzeros.
- Solution accuracy as the relative 2-norm error of the solutions from NEW and SMF compared against the solution from STD.
- IHSS rank which is the size r mentioned in Sect. 3.4.
- *IHSS reuse factor* defined as the ratio between the number of low-rank compression operations (rank-revealing QR factorizations) in a direct structured construction and that in the IHSS construction for the largest three separators (in the top two levels of the separator tree). More specifically, let k_1 be the number of row and column basis matrices in the IHSS form, which would be the number of off-diagonal compression steps if it needs to directly produce these basis matrices in the structured approximation of a dense local Schur complement. With basis reuse, the basis matrices produced at the switching level are reused at upper levels, as mentioned in Sect. 3.4. Above the switching level, only few extra compression steps (like in (33)) are needed to produce the additional basis matrices. Let k_2 be the number of these extra compression steps. k_1 and k_2 are counted for the top two levels. The reuse factor is then the ratio of the sum of the k_1 counts over the sum of the k_2 counts. This factor measures by how many times the IHSS method reduces the number of low-rank compression operations during the elimination of those separators.

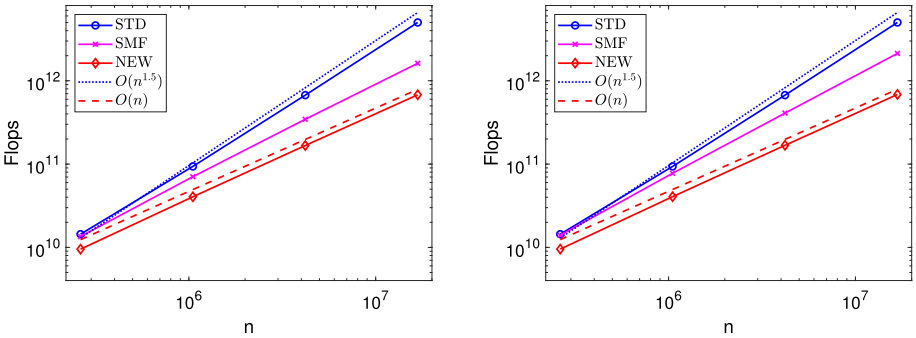
Example 1 The first example is Poisson's equation $-\Delta u(x) = f(x),\ x \in \Omega$, where each local Schur complement like in (3) is a discretized Robin-to-Robin map that describes the linear relation from $\partial_n u + au$ to $\partial_n u - au$ on the boundary. Here, $\partial_n u$ denotes the Neumann boundary value and a is a positive constant to ensure well-posedness. The computational domain is $\Omega = [0, 1]^2$ with a uniform triangulation. For the tests, we choose $a = 1/h$, where h is the mesh spacing. The number of unknowns is $n = (p/h + 1)^2$, where p is the polynomial order of the nodal Lagrange basis functions and is chosen to be 4. The right-hand side function is chosen as $f(x) = \exp\left(-\|x - (0.5, 0.5)\|^2/h^2\right)$.

For this problem, it is known that the intermediate Schur complements are rank structured and SMF is efficient. We use this problem to show how IHSS structures help to further reuse computations in the structured factorization. Table 1 provides the IHSS rank and the IHSS reuse factors. Table 2 gives the performance of the solvers as the matrix size increases. The corresponding scaling plot is shown in Fig. 11a. We have the following observations.

- Significant basis reuse can be observed in all the tests. For the largest problem size $n = 4097^2$, the reuse factor is already 8.0 times, which indicates a dramatic reduction in the number of low-rank compression operations. (Note that this reuse factor is not the

Table 1 Examples 1 and 2: IHSS ranks and reuse factors

Problem size	513^2	1025^2	2049^2	4097^2
IHSS rank				
Example 1	23	28	32	36
Example 2	25	33	53	92
IHSS reuse factor	$\frac{148}{52} \approx 2.8$	$\frac{516}{116} \approx 4.4$	$\frac{1508}{244} \approx 6.2$	$\frac{4004}{500} \approx 8.0$



(a) Example 1 **(b)** Example 2

Fig. 11 Examples 1 and 2: Factorization costs of STD, SMF, and NEW

difference of total factorization costs for NEW and SMF, since the factor is only for the 3 largest local Schur complements and also both methods still need the local structured factorizations other than the structured constructions.)

- NEW costs less than both STD and SMF. For the largest problem size $n = 4097^2$, the factorization cost of STD is 7.4 times of that of NEW. The SMF cost is also about 2.4 times of that of NEW. The factorization costs of NEW scale like nearly $O(n)$. This is consistent with the theoretical estimates. As indicated in Fig. 11a, the advantage of NEW gets bigger for larger n.
- NEW also requires lower storage. For $n = 4097^2$, NEW needs less than 60% of the storage of STD. The difference in the storage is less significant since all the three methods have roughly linear storage.
- With lower costs and storage, NEW achieves satisfactory accuracy as controlled by the compression tolerance. For the two largest problem sizes, SMF has worse accuracy than NEW, possibly due to the large number of rank-revealing approximations at many levels. If a slightly smaller tolerance is used for SMF, then comparable accuracy can be achieved, at the cost of slightly increased costs. For example, with $\tau = 10^{-7}$, SMF yields solution accuracies $8.5e{-}7$ and $7.7e{-}7$ with factorization costs $3.77e11$ and $1.80e12$ for $n = 2049^2$ and $n = 4097^2$, respectively. The impact of this smaller tolerance on the SMF storage is very small.

Since the implementation is in Matlab, the timing is not reported and the advantage of NEW in timing is not as significant. We expect to see larger differences in timing with more efficient implementations for bigger problem sizes. Again, we would like to emphasize that the benefit of IHSS structures can already be clearly observed from the reuse factors, and the potential can be fully utilized in practical implementations.

Table 2 Example 1: Test results with STD, SMF, and NEW, where the number of nonzeros is the total number of nonzeros of all leaf-level local problems

Problem size	513^2	1025^2	2049^2	4097^2
Number of nonzeros	6, 234, 240	24, 936, 960	99, 747, 840	398, 991, 360
$(\mathbf{l}, \mathbf{l}_s)$	(8, 5)	(10, 7)	(12, 9)	(14, 11)
Factorization flops				
STD	$1.44e10$	$9.37e10$	$6.72e11$	$5.01e12$
SMF	$1.35e10$	$7.05e10$	$3.45e11$	$1.62e12$
NEW	$9.52e9$	$4.05e10$	$1.67e11$	$6.79e11$
Factor storage				
STD	$2.86e7$	$1.34e8$	$6.18e8$	$2.77e9$
SMF	$2.58e7$	$1.08e8$	$4.41e8$	$1.78e9$
NEW	$2.27e7$	$9.76e7$	$4.05e8$	$1.65e9$
Solution accuracy $\frac{\|x-x_0\|_2}{\|x_0\|_2}$				
SMF	$4.7e{-7}$	$2.7e{-6}$	$5.9e{-6}$	$1.2e{-5}$
NEW	$2.3e{-6}$	$1.2e{-6}$	$5.9e{-7}$	$3.0e{-7}$

Example 2 The following Helmholtz equation with jump discontinuities in the coefficient is a more challenging Hermitian indefinite problem:

$$-\Delta u(x) - k^2(x)u(x) = f(x), \quad x \in \Omega, \tag{36}$$

where $k(x)$ is the wavenumber with a 300% jump, similar to an example in [24]. We follow the impedance-to-impedance formulation [11, 28] and each local Schur complement (3) maps from $\partial_n u - \mathrm{i}\eta u$ to $\partial_n u + \mathrm{i}\eta u$, where η is chosen as the average value of $k(x)$. Neumann boundary condition is imposed on the physical boundary $\partial\Omega$. As the problem size increases, we fix the sampling rate: there are 4 points for the smallest side length among those rectangles; comparing with the shortest wavelength, there are 24 points per wavelength. The choices of Ω, h, n, and $f(x)$ are the same as in Example 1.

The test results are reported in Table 3. The advantages of NEW in the cost can be further observed from Fig. 11b. Note that the cost of STD is the same for both Examples 1 and 2 since the discretization and nonzero patterns are the same. The factorization costs of NEW are similar to those in the previous example. On the other hand, the factorization costs of SMF are higher than in the previous example and the SMF cost for $n = 4097^2$ is now about 3.1 times of that of NEW. The IHSS ranks and reuse factors are given in Table 1, where the reuse factors are the same as in the previous example.

Figure 12 illustrates a solution, where a jump corresponding to the jump location in the wavenumber $k(x)$ can be observed.

Example 3 For the Helmholtz problem in Example 2, we also demonstrate the performance of IHSS methods for accelerating the coefficient update problems as mentioned in Sect. 4. We follow the setting in [24] and introduce a local update to the coefficient by reducing the wavenumber by $1/2$ in an interior subdomain containing 160^2 unknowns.

In order to perform quick factorization update after local coefficient updates, we precompute the interior and exterior factorizations as mentioned in Sect. 4. IHSS operations are

Table 3 Example 2: Test results with STD, SMF, and NEW

Problem size $(\mathbf{l}, \mathbf{l}_s)$	513^2 (8, 5)	1025^2 (10, 7)	2049^2 (12, 9)	4097^2 (14, 11)
Factorization flops				
STD	$1.44e10$	$9.37e10$	$6.72e11$	$5.01e12$
SMF	$1.40e10$	$7.75e10$	$4.09e11$	$2.13e12$
NEW	$9.54e9$	$4.06e10$	$1.68e11$	$6.85e11$
Factor storage				
STD	$2.86e7$	$1.34e8$	$6.18e8$	$2.77e9$
SMF	$2.59e7$	$1.08e8$	$4.42e8$	$1.78e9$
NEW	$2.27e7$	$9.79e7$	$4.07e8$	$1.66e9$
Solution accuracy $\frac{\|x-x_0\|_2}{\|x_0\|_2}$				
SMF	$8.7e-7$	$2.0e-6$	$9.4e-6$	$4.1e-5$
NEW	$9.3e-6$	$8.1e-6$	$8.9e-6$	$1.4e-5$

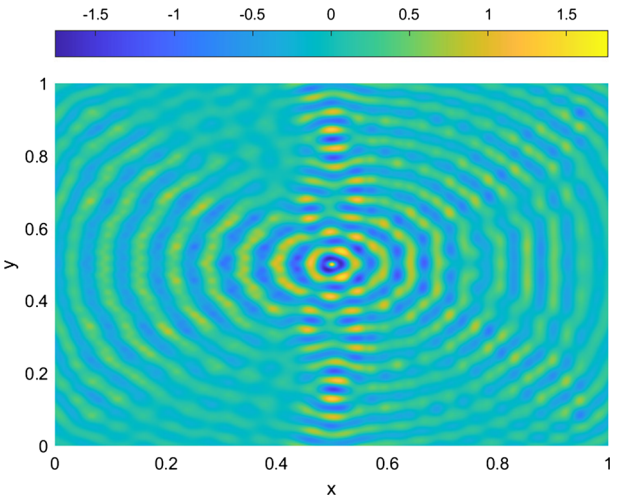


Fig. 12 Example 2: Visualization of a solution corresponding to the problem size 1025^2

used to accelerate both types of factorizations. The results are shown in Table 4, where we compare the factorization update based on NEW (denoted NEW_UPD) and that based on the previous factorization in [24] (denoted UPD).

SMF is not tested since it is significantly more expensive to be used for local updates due to the need to perform nearly refactorizations instead of local factorization updates. (The local factorization update with SMF or a regular multifrontal factorization has cost essentially in the same order of magnitude as the original factorization cost. See [24] for more details.)

We have the following observations.

– As compared with the update method in [24] , the IHSS method greatly reduces the factorization costs. For the largest problem size $n = 2561^2$, NEW_UPD reduces the cost for the interior factorization by about 2.7 times and reduces the cost for the exterior factorization by about 10.4 times. Figure 13 further shows the scaling plots and indicates the significant speedup of NEW_UPD over UPD.

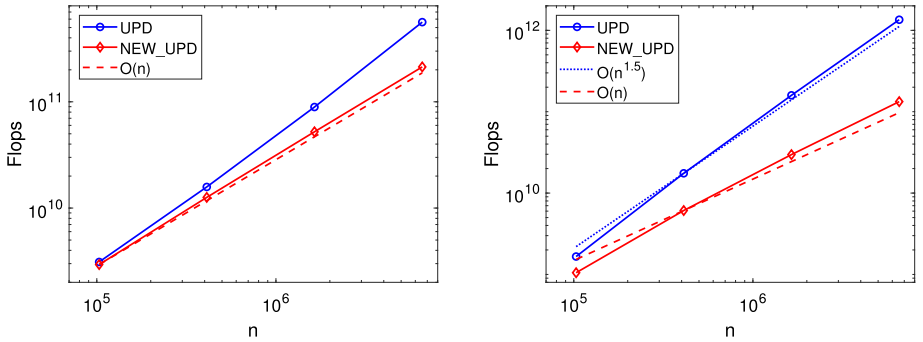
Table 4 Example 3: Results for interior and exterior factorizations

Problem size	321^2	641^2	1281^2	2561^2
Number of nonzeros	2,437,184	9,748,736	38,994,994	155,979,776
$(\mathbf{l}, \mathbf{l}_s)$	(7, 5)	(9, 7)	(11, 9)	(13, 11)
Factorization flops				
Interior				
UPD	$3.11e9$	$1.58e10$	$8.93e10$	$5.62e11$
NEW_UPD	$2.94e9$	$1.26e10$	$5.22e10$	$2.13e11$
Exterior				
UPD	$1.66e9$	$1.75e10$	$1.59e11$	$1.35e12$
NEW_UPD	$1.05e9$	$6.10e9$	$2.97e10$	$1.33e11$
Factor storage				
Interior				
UPD	$9.03e6$	$4.65e7$	$2.31e8$	$1.11e9$
NEW_UPD	$7.58e6$	$3.27e7$	$1.36e8$	$5.56e8$
Exterior				
UPD	$3.87e6$	$2.56e7$	$1.46e8$	$7.66e8$
NEW_UPD	$3.43e6$	$1.89e7$	$8.98e7$	$3.96e8$
IHSS reuse factor of NEW_UPD	$\frac{296}{52} \approx 5.7$	$\frac{1032}{116} \approx 8.9$	$\frac{3016}{244} \approx 12.4$	$\frac{8008}{500} \approx 16.0$

- The sizes of interior and exterior factors are reduced simultaneously by NEW_UPD. For example, for $n = 2561^2$, the storage of the exterior factor from NEW_UPD is about half of that from UPD.
- The basis reuse with NEW_UPD is even more significant than just in the interior factorization. The exterior factorization can also reuse basis matrices from the interior factorization. For the largest problem size, the reuse factor is already about 16.
- Since the factorization update just needs the refactorization in the local subdomain where the coefficient changes, the factorization update cost is nearly independent of the matrix size n, as shown in Table 5. The total solution update cost is roughly proportional to n.
- The updated solutions of NEW_UPD are compared against UPD. The solution accuracy after the factorization update is also well controlled by the compression accuracy.

6 Conclusions

We have designed a type of interconnected hierarchical rank structures called IHSS structures and developed fast IHSS methods for solving some elliptic PDEs. Our structured sparse factorization is organized in terms of an outer neighbor tree with inner IHSS trees. Unlike many existing rank-structured direct solvers, our IHSS strategies can reuse off-diagonal basis matrices across the outer sparse factorization levels. It thus not only eliminates large dense intermediate matrix operations, but also avoids excessive low-rank compression operations during the structured factorization. The extensive basis reuse in IHSS structures helps to both save the cost and conveniently preserve the rank structures. The factorization complexity of the solver is $O(rn)$ (Sect. 3.4), which is lower than some other solvers by a factor of $O(\log n)$. We have also extended the solver to a challenging sparse factorization update problems. The



(a) Interior factorization **(b)** Exterior factorization

Fig. 13 Examples 3: Costs of interior and exterior factorizations with UPD and NEW_UPD

Table 5 Example 3: Results for factorization and solution updates with NEW_UPD

Problem size	321^2	641^2	1281^2	2561^2
Factorization update flops	7.41*e*8	8.57*e*8	8.57*e*8	8.57*e*8
Solution update flops				
Interior	6.17*e*6	6.70*e*6	6.70*e*6	6.70*e*6
Exterior	9.15*e*6	4.86*e*7	2.13*e*8	8.85*e*8
Solution accuracy	7.9*e*−8	1.4*e*−7	2.4*e*−7	2.0*e*−7

large reuse factors in the numerical tests indicate significant reductions in the need of low-rank compression operations.

We expect similar ideas can be generalized to three dimensional problems, although the implementation and the management of the discretizations and tree structures would be much more involved and need to handle substantially more technical details. This will be studied and tested in future work. The current proof-of-concept work already clearly demonstrates the potential of the key ideas. In addition, since the IHSS structure relies on certain discretizations, it remains open to explore the feasibility of extending the IHSS framework to more general discretizations and even more general sparse matrices. Also, more practical implementations are expected to be done based on this work.

Acknowledgements Thank the two anonymous referees for the helpful suggestions.

Declarations

Conflict of interest The source codes and data for the tests are available upon request. The authors declare that there is no conflict of interest.

References

1. Amestoy, P., Ashcraft, C., Boiteau, O., Buttari, A., L'Excellent, J.-Y., Weisbecker, C.: Improving multifrontal methods by means of block low-rank representations. SIAM J. Sci. Comput. **37**, A1451–A1474 (2015)

2. Bebendorf, M., Hackbusch, W.: Existence of $\mathscr{H}$-matrix approximants to the inverse FE-matrix of elliptic operator with L^∞-coefficients. Numer. Math. **95**, 1–28 (2003)
3. Chan, T.F., Mathew, T.P.: Domain decomposition algorithms. Acta Numer. **3**, 61–143 (1994)
4. Chandrasekaran, S., Dewilde, P., Gu, M., Somasunderam, N.: On the numerical rank of the off-diagonal blocks of Schur complements of discretized elliptic PDEs. SIAM J. Matrix Anal. Appl. **31**, 2261–2290 (2010)
5. Chandrasekaran, S., Gu, M., Pals, T.: A fast ULV decomposition solver for hierarchically semiseparable representations. SIAM J. Matrix Anal. Appl. **28**, 603–622 (2006)
6. Demmel, J.W., Gilbert, J.R., Li, X.S.: SuperLU users' guide. http://crd.lbl.gov/~xiaoye/SuperLU/superlu_ug.pdf
7. Duff, I.S., Reid, J.K.: The multifrontal solution of indefinite sparse symmetric linear equations. ACM Trans. Math. Softw. **9**, 302–325 (1983)
8. Engquist, B., Ying, L.: Sweeping preconditioner for the Helmholtz equation: hierarchical matrix representation. Commun. Pure Appl. Math. **64**, 697–735 (2011)
9. George, A.: Nested dissection of a regular finite element mesh. SIAM J. Numer. Anal. **10**, 345–363 (1973)
10. Ghysels, P., Li, X.S., Rouet, F.H., Williams, S., Napov, A.: An efficient multicore implementation of a novel HSS-structured multifrontal solver using randomized sampling. SIAM J. Sci. Comput. **38**, S358–S384 (2016)
11. Gillman, A., Barnett, A.H., Martinsson, P.G.: A spectrally accurate direct solution technique for frequency-domain scattering problems with variable media. BIT Numer. Math. **55**, 141–170 (2015)
12. Gillman, A., Martinsson, P.G.: A direct solver with $O(n)$ complexity for variable coefficient elliptic PDEs discretized via a high-order composite spectral collocation method. SIAM J. Sci. Comput. **36**, A2023–A2046 (2014)
13. Gorman, C., Chávez, G., Ghysels, P., Mary, T., Rouet, F.-H., Li, X.S.: Robust and accurate stopping criteria for adaptive randomized sampling in matrix-free hierarchically semiseparable construction. SIAM J. Sci. Comput. **41**, S61–S85 (2019)
14. Grasedyck, L., Kriemann, R., Le Borne, S.: Domain-decomposition based $mathcal H$-LU preconditioners. In: Widlund, O.B., Keyes, D.E. (eds.) Domain Decomposition Methods in Science and Engineering XVI, vol. 55, pp. 661–668. Springer LNCSE, Berlin (2006)
15. Hackbusch, W., Börm, S.: Data-sparse approximation by adaptive $\mathscr{H}^2$-matrices. Computing **69**, 1–35 (2002)
16. Hackbusch, W., Grasedyck, L., Börm, S.: An introduction to hierarchical matrices. Math. Bohem. **127**, 229–241 (2002)
17. Hackbusch, W., Khoromskij, B.N., Kriemann, R.: Direct Schur complement method by domain decomposition based on $\mathscr{H}$-matrix approximation. Comput. Vis. Sci. **8**, 179–188 (2005)
18. Hesthaven, J.S., Warburton, T.: Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications. Springer, Berlin (2007)
19. Ho, K.L., Ying, L.: Hierarchical interpolative factorization for elliptic operators: differential equations. Commun. Pure Appl. Math. **69**, 1415–1451 (2016)
20. Li, Y., Ying, L.: Distributed-memory hierarchical interpolative factorization. Res. Math. Sci. **4**, 1–23 (2017)
21. Lin, L., Lu, J., Ying, L.: Fast construction of hierarchical matrix representation from matrix–vector multiplication. J. Comput. Phys. **230**, 4071–4087 (2011)
22. Liu, J.W.: The multifrontal method for sparse matrix solution: theory and practice. SIAM Rev. **34**, 82–109 (1992)
23. Liu, X., Xia, J., de Hoop, M.V.: Parallel randomized and matrix-free direct solvers for large structured dense linear systems. SIAM J. Sci. Comput. **38**, S508–S538 (2016)
24. Liu, X., Xia, J., de Hoop, M.V.: Fast factorization update for general elliptic equations under multiple coefficient updates. SIAM J. Sci. Comput. **42**, A1174–A1199 (2020)
25. Martinsson, P.G.: A fast randomized algorithm for computing hierarchically semiseparable representation of a matrix. SIAM. J. Matrix Anal. Appl. **32**, 1251–1274 (2011)
26. Martinsson, P.G.: A direct solver for variable coefficient elliptic PDEs discretized via a composite spectral collocation method. J. Comput. Phys. **242**, 460–479 (2013)
27. Parter, S.: The use of linear graphs in Gauss elimination. SIAM Rev. **3**, 119–130 (1961)
28. Pedneault, M., Catalin, T., Boubendir, Y.: Schur complement domain decomposition methods for the solution of multiple scattering problems. IMA J. Appl. Math. **82**, 1104–1134 (2017)
29. Schmitz, P.G., Ying, L.: A fast direct solver for elliptic problems on general meshes in 2D. J. Comput. Phys. **231**, 1314–1338 (2012)
30. Schmitz, P.G., Ying, L.: A fast nested dissection solver for Cartesian 3D elliptic problems using hierarchical matrices. J. Comput. Phys. **258**, 227–245 (2014)

31. Trefethen, L.N., David, B.: Numerical Linear Algebra, vol. 50. SIAM, Philadelphia (1997)
32. Wang, S., de Hoop, M.V., Xia, J.: Acoustic inverse scattering via Helmholtz operator factorization and optimization. J. Comput. Phys. **229**, 8445–8462 (2010)
33. Wang, S., de Hoop, M.V., Xia, J.: On 3D modeling of seismic wave propagation via a structured parallel multifrontal direct Helmholtz solver. Geophys. Prospect. **59**, 857–873 (2011)
34. Xi, Y., Xia, J., Cauley, S., Balakrishnan, V.: Superfast and stable structured solvers for Toeplitz least squares via randomized sampling. SIAM J. Matrix Anal. Appl. **35**, 44–72 (2014)
35. Xi, Y., Xia, J., Chan, R.: A fast randomized eigensolver with structured LDL factorization update. SIAM J. Matrix Anal. Appl. **35**, 974–996 (2014)
36. Xia, J.: On the complexity of some hierarchical structured matrix algorithms. SIAM J. Matrix Anal. Appl. **33**, 388–410 (2012)
37. Xia, J.: Randomized sparse direct solvers. SIAM J. Matrix Anal. Appl. **34**, 197–227 (2013)
38. Xia, J.: Efficient structured multifrontal factorization for general large sparse matrices. SIAM J. Sci. Comput. **35**, A832–A860 (2013)
39. Xia, J., Chandrasekaran, S., Gu, M., Li, X.S.: Superfast multifrontal method for large structured linear systems of equations. SIAM J. Matrix Anal. Appl. **31**, 1382–1411 (2009)
40. Xia, J., Chandrasekaran, S., Gu, M., Li, X.S.: Fast algorithms for hierarchically semiseparable matrices. Numer. Linear Algebra Appl. **17**, 953–976 (2010)
41. Xia, J., Xi, Y., Gu, M.: A superfast structured solver for Toeplitz linear systems via randomized sampling. SIAM J. Matrix Anal. Appl. **33**, 837–858 (2012)
42. Xin, Z., Xia, J., de Hoop, M.V., Cauley, S., Balakrishnan, V.: A distributed-memory randomized structured multifrontal method for sparse direct solutions. SIAM J. Sci. Comput. **39**, C292–C318 (2017)