Tree Quasi-Separable matrices

A simultaneous generalization of sequentially and hierarchically semiseparable representations

Govindarajan, Nithin; Chandrasekaran, Shivkumar; Dewilde, Patrick

DOI
10.1137/24M1682348

Publication date
2025

Document Version
Final published version

Published in
SIAM Journal on Matrix Analysis and Applications

Citation (APA)
Govindarajan, N., Chandrasekaran, S., & Dewilde, P. (2025). Tree Quasi-Separable matrices: A simultaneous generalization of sequentially and hierarchically semiseparable representations. *SIAM Journal on Matrix Analysis and Applications*, *46*(2), 1562-1586. https://doi.org/10.1137/24M1682348

SIAM J. MATRIX ANAL. APPL. © 2025 Society for Industrial and Applied Mathematics
Vol. 46, No. 2, pp. 1562–1586

TREE QUASI-SEPARABLE MATRICES: A SIMULTANEOUS GENERALIZATION OF SEQUENTIALLY AND HIERARCHICALLY SEMISEPARABLE REPRESENTATIONS[*]

NITHIN GOVINDARAJAN[†], SHIVKUMAR CHANDRASEKARAN[‡], AND PATRICK DEWILDE[§]

Abstract. We present a unification and generalization of what is known in the literature as sequentially and hierarchically semiseparable (SSS and HSS) representations for matrices. These so-called tree quasi-separable (TQS) matrices contain sparse matrices with tree-structured adjacency graphs as an important subcase. TQS matrices inherit all the favorable algebraic properties of SSS and HSS under addition, products, and inversion. To arrive at these properties, we prove a key result that characterizes the conversion of any dense matrix into a TQS representation. Here, we specifically show through an explicit construction that the size of the representation is dictated by the ranks of certain Hankel blocks of the matrix. Analogous to SSS and HSS, TQS matrices admit fast matrix-vector products and direct solvers. A sketch of the associated algorithms is provided.

Key words. TQS matrices, HSS matrices, SSS matrices, rank-structured matrices, quasi-separable matrices, GIRS

MSC code. 15A23

DOI. 10.1137/24M1682348

See reproducibility of computational results at end of the article.

1. Introduction.

Matrices in applied problems of interest often exhibit a structure of low rank in their off-diagonal blocks. These structures have, for instance, been observed in the discretization of integral equations [21], Schur complements of discretizations of PDEs [9, 9, 26], certain Cauchy-like matrices [14, 28], evaluations of potentials [18], and companion matrices [3], among others. Many frameworks have been proposed to efficiently represent these low-rank structures so that efficient linear algebra operations can be performed with such matrices. This includes the fast multiple method [18], semiseparable and quasi-separable matrices [4, 17, 22], sequentially semiseparable (SSS) matrices [8, 7], hierarchically semiseparable (HSS) matrices [13, 27], $\mathcal{H}$- and $\mathcal{H}^2$-matrices [5, 19, 20], and hierarchically off-diagonal low-rank matrices [1, 2]. These frameworks are related and have their specific benefits, pitfalls, and special use cases. A complete review of the subject goes beyond the scope of this paper.

This paper examines the low-rank structures that are preserved during the inversion of a (block-)sparse matrix with a tree-structured adjacency graph. These structures induce a family of typically *dense* matrices that possess certain low-rank properties on their submatrices. Although sparse matrices whose adjacency graphs are trees form, along with their inverses, a special subcategory of this family, the

[*]Received by the editors August 5, 2024; accepted for publication (in revised form) by R. Vandebril February 13, 2025; published electronically June 9, 2025.
https://doi.org/10.1137/24M1682348

[†]Electrical Engineering (ESAT), KU Leuven, Leuven, Belgium (Nithin.Govindarajan@kuleuven.be).
[‡]Electrical and Computer Engineering, University of California–Santa Barbara, Santa Barbara, CA 93106 USA (shiv@ece.ucsb.edu).
[§]Electrical Engineering, Mathematics and Computer Science, TU Delft, Delft, The Netherlands, and Institute of Advanced Study, Technical University Munich, Munich, Germany (p.dewilde@me.com).

family contains dense matrices that are not necessarily the inverse of a sparse matrix. In this paper, we present a new class of representations for rank-structured matrices that can capture these structures exactly. In fact, we show that these representations satisfy a graph-induced rank structure (GIRS) property if the corresponding graph of the associated graph-partitioned matrix is a tree (see [10]). The representations are referred to as *tree quasi-separable* (TQS) matrices.

Interestingly, TQS matrices simultaneously unify and generalize SSS and HSS matrices. Apart from introducing a new family of rank-structured matrices, an important technical contribution of our paper is an algorithm that realizes a minimal TQS representation for any dense matrix. This algorithm is, in effect, a unification and generalization of the algorithms for doing the same with SSS and HSS matrices. The algorithm allows us to prove a result (Theorem 4.4) that fully characterizes the properties of a TQS representation of any matrix. Specifically, for a given tree structure and block partitioning of the matrix, the dimensions of the TQS generators are dictated by the ranks of certain matrix subblocks referred to as Hankel blocks. Through this fact, we show that TQS inherits the favorable algebraic properties of SSS and HSS matrices under addition, products, and inversion. Importantly, the inverse of a TQS matrix is again a TQS matrix of exactly the same rank-profile if the input and output dimensions of the partitions are chosen equally. TQS matrices allow for fast inversion algorithms and matrix-vector products. We present one such fast direct solver using a sparse embedding technique introduced in [6] for HSS matrices.

It is well-known that SSS and HSS matrices are used extensively in practice. TQS matrices are a more flexible drop-in replacement for HSS and SSS matrices. We anticipate that this flexibility can lead to even further improvements in many applications, particularly for networked dynamical systems on graphs [23]. Although applications are the subject of future work, we present some illustrative examples of sparse matrices where the choice of a TQS representation is both natural and efficient.

The remainder of this paper is outlined as follows. In section 2, we give a quick review of SSS and HSS matrices. Section 3 introduces TQS matrices in detail. The algebraic properties of TQS matrices are discussed in section 4. In section 5, we discuss the construction algorithm, and along with it the proof of Theorem 4.4. Section 6 covers the fast matrix-vector multiplication algorithm and the procedure for efficiently solving a linear system involving TQS matrices.

2. A brief recap on SSS and HSS matrices.

This section briefly reviews the definitions of SSS and HSS matrices. We emphasize that we use a rather unconventional, and admittedly redundant, notation in the definitions of both SSS and HSS matrices. However, this is done intentionally to establish a direct link with TQS matrices that will require more detailed notation. Subsection 2.1 covers SSS matrices and subsection 2.2 covers HSS matrices. Subsection 2.3 briefly summarizes performing algebra with SSS and HSS matrices. For more details on the algorithms themselves, we recommend [7, 8] for SSS and [13, 24, 25, 27] for HSS. A more tutorial-styled introduction to the subject is given in [12].

2.1. SSS matrices.

SSS matrices were first introduced in [15] in a study to generalize systems theory to the time-varying case. To define SSS matrices, let $m_i, n_i \in \mathbb{N} \cup \{0\}$ for $i \in \{1, \dots, K\}$ and $\rho_{(i,i+1)} \in \mathbb{N}\{0\}$ for $i \in \{1, \dots, K-1\}$ and introduce the matrices $\mathrm{D}^i \in \mathbb{F}^{m_i \times n_i}$ for $i \in \{1, \dots, K\}$, $\mathrm{B}^i_{i+1} \in \mathbb{F}^{\rho_{(i,i+1)} \times n_i}$ for $i \in \{1, \dots, K-1\}$, $\mathrm{U}^i_{i+1,i-1} \in \mathbb{F}^{\rho_{(i,i+1)} \times \rho_{(i-1,i)}}$ for $i \in \{2, \dots, K-1\}$, $\mathrm{C}^i_{i-1} \in \mathbb{F}^{m_i \times \rho_{(i-1,i)}}$ for $i \in \{2, \dots, K\}$, $\mathrm{P}^i_{i-1} \in \mathbb{F}^{\rho_{(i,i-1)} \times n_i}$ for $i \in \{2, \dots, K\}$, $\mathrm{W}^i_{i-1,i+1} \in \mathbb{F}^{\rho_{(i,i-1)} \times \rho_{(i+1,i)}}$ for $i \in \{2, \dots, K-1\}$, and $\mathrm{Q}^i_{i+1} \in \mathbb{F}^{m_i \times \rho_{(i+1,i)}}$ for $i \in \{1, \dots, K\}$. Next, define

$$\mathrm{A}_{ij} := \begin{cases} \mathrm{D}^i, & i = j, \\ \mathrm{C}^i_{i-1}\mathrm{U}^{i-1}_{i,i-2}\cdots\mathrm{U}^{j+1}_{j+2,j}\mathrm{B}^j_{j+1}, & i > j, \\ \mathrm{Q}^i_{i+1}\mathrm{W}^{i+1}_{i,i+2}\cdots\mathrm{W}^{j-1}_{j-2,j}\mathrm{P}^j_{j-1}, & i < j. \end{cases} \tag{2.1}$$

An SSS matrix $\mathrm{A} \in \mathbb{F}^{M\times N}$, with $M = \sum_{i=1}^K m_i$ and $N = \sum_{i=1}^K n_i$, is the block-partitioned matrix with block entries specified by (2.1). For example, in the case of $K = 4$, an SSS matrix takes on the form

$$\mathrm{A} = \begin{bmatrix} \mathrm{D}^1 & \mathrm{Q}^1_2\mathrm{P}^2_1 & \mathrm{Q}^1_2\mathrm{W}^2_{1,3}\mathrm{P}^3_2 & \mathrm{Q}^1_2\mathrm{W}^2_{1,3}\mathrm{W}^3_{2,4}\mathrm{P}^4_3 \\ \mathrm{C}^2_1\mathrm{B}^1_2 & \mathrm{D}^2 & \mathrm{Q}^2_3\mathrm{P}^3_2 & \mathrm{Q}^2_3\mathrm{W}^3_{2,4}\mathrm{P}^4_3 \\ \mathrm{C}^3_2\mathrm{U}^2_{3,1}\mathrm{B}^1_2 & \mathrm{C}^3_2\mathrm{B}^2_3 & \mathrm{D}^3 & \mathrm{Q}^3_4\mathrm{P}^4_3 \\ \mathrm{C}^4_2\mathrm{U}^3_{4,2}\mathrm{U}^2_{3,1}\mathrm{B}^1_2 & \mathrm{C}^4_2\mathrm{U}^3_{4,2}\mathrm{B}^2_3 & \mathrm{C}^4_3\mathrm{B}^3_2 & \mathrm{D}^4 \end{bmatrix}. \tag{2.2}$$

The entries of the SSS representation (2.1) can be seen as the result of decomposing A as the sum of a causal and anticausal linear-time-variant (LTV) system on a line graph. At this point of our discussion, it may be important to remark that the phrase "sequentially semiseparable" has been chosen rather inconveniently in the literature since SSS matrices are effectively a representation for the much richer class of quasi-separable matrices [4, 17]. The prefix "sequentially quasi-separable" would have been more appropriate.

2.2. HSS matrices. While SSS matrices have their origins in systems theory, HSS matrices arose independently to simplify the algebra of the fast multipole method so that it has favorable properties under inversion. To define HSS matrices, we recursively partition a matrix $\mathrm{A} \in \mathbb{F}^{M\times N}$ in a hierarchic manner. This process is best described through a binary tree.

Assume that the nodes of the binary tree are indexed by a *postordered* traversal of the tree and let r denote its root node. Furthermore, let $\mathrm{L}(l)$ (respectively, $\mathrm{R}(l)$) denote the left (respectively, right) child of node l in the binary tree. If l is a terminating point of the recursion, or equivalently, a leaf node of the binary tree, we set $\mathrm{A}^l = \mathrm{D}^l \in \mathbb{F}^{m_l\times n_l}$. On the other hand, if l is a nonterminating point of the recursion, we set

$$\mathrm{A}^l = \begin{bmatrix} \mathrm{A}^{\mathrm{L}(l)} & \mathcal{Q}^{\mathrm{L}(l)}_l \mathrm{V}^l_{\mathrm{L}(l),\mathrm{R}(l)} \mathcal{B}^{\mathrm{R}(l)}_l \\ \mathcal{Q}^{\mathrm{R}(l)}_l \mathrm{V}^l_{\mathrm{R}(l),\mathrm{L}(l)} \mathcal{B}^{\mathrm{L}(l)}_l & \mathrm{A}^{\mathrm{R}(l)} \end{bmatrix}, \tag{2.3}$$

where $\mathrm{A}^{\mathrm{L}(l)} \in \mathbb{F}^{m_{\mathrm{L}(l)}\times n_{\mathrm{L}(l)}}$, $\mathrm{A}^{\mathrm{R}(l)} \in \mathbb{F}^{m_{\mathrm{R}(l)}\times n_{\mathrm{R}(l)}}$, $m_l = m_{\mathrm{L}(l)} + m_{\mathrm{R}(l)}$, and $n_l = n_{\mathrm{L}(l)} + n_{\mathrm{R}(l)}$. Furthermore, $\mathcal{Q}^{\mathrm{L}(l)}_l = \mathrm{Q}^{\mathrm{L}(l)}_l \in \mathbb{F}^{m_{\mathrm{L}(l)}\times \rho_{(l,\mathrm{L}(l))}}$ (respectively, $\mathcal{Q}^{\mathrm{R}(l)}_l = \mathrm{Q}^{\mathrm{R}(l)}_l \in \mathbb{F}^{m_{\mathrm{R}(l)}\times \rho_{(l,\mathrm{R}(l))}}$) if $\mathrm{L}(l)$ (respectively, $\mathrm{R}(l)$) is a leaf node of the binary tree. Otherwise,

$$\mathcal{Q}^{\mathrm{L}(l)}_l = \begin{bmatrix} \mathcal{Q}^{\mathrm{L}(\mathrm{L}(l))}_{\mathrm{L}(l)} \mathrm{W}^{\mathrm{L}(l)}_{\mathrm{L}(\mathrm{L}(l)),l} \\ \mathcal{Q}^{\mathrm{R}(\mathrm{L}(l))}_{\mathrm{L}(l)} \mathrm{W}^{\mathrm{L}(l)}_{\mathrm{R}(\mathrm{L}(l)),l} \end{bmatrix}, \qquad \mathcal{Q}^{\mathrm{R}(l)}_l = \begin{bmatrix} \mathcal{Q}^{\mathrm{L}(\mathrm{R}(l))}_{\mathrm{R}(l)} \mathrm{W}^{\mathrm{R}(l)}_{\mathrm{L}(\mathrm{R}(l)),l} \\ \mathcal{Q}^{\mathrm{R}(\mathrm{R}(l))}_{\mathrm{R}(l)} \mathrm{W}^{\mathrm{R}(l)}_{\mathrm{R}(\mathrm{R}(l)),l} \end{bmatrix}$$

with $\mathrm{W}^{\mathrm{L}(l)}_{\mathrm{L}(\mathrm{L}(l)),l} \in \mathbb{F}^{\rho_{(\mathrm{L}(l),\mathrm{L}(\mathrm{L}(l)))}\times \rho_{(l,\mathrm{L}(l))}}$, $\mathrm{W}^{\mathrm{L}(l)}_{\mathrm{R}(\mathrm{L}(l)),l} \in \mathbb{F}^{\rho_{(\mathrm{L}(l),\mathrm{R}(\mathrm{L}(l)))}\times \rho_{(l,\mathrm{L}(l))}}$, $\mathrm{W}^{\mathrm{R}(l)}_{\mathrm{R}(\mathrm{L}(l)),l} \in \mathbb{F}^{\rho_{(\mathrm{R}(l),\mathrm{R}(\mathrm{L}(l)))}\times \rho_{(l,\mathrm{R}(l))}}$, $\mathrm{W}^{\mathrm{R}(l)}_{\mathrm{R}(\mathrm{R}(l)),l} \in \mathbb{F}^{\rho_{(\mathrm{R}(l),\mathrm{R}(\mathrm{R}(l)))}\times \rho_{(l,\mathrm{R}(l))}}$. Similarly, $\mathcal{B}^{\mathrm{L}(l)}_l = \mathrm{B}^{\mathrm{L}(l)}_l \in \mathbb{F}^{\rho_{(\mathrm{L}(l),l)}\times n_{\mathrm{L}(l)}}$ (respectively, $\mathcal{B}^{\mathrm{R}(l)}_l = \mathrm{B}^{\mathrm{R}(l)}_l \in \mathbb{F}^{\rho_{(\mathrm{R}(l),l)}\times n_{\mathrm{R}(l)}}$) if $\mathrm{L}(l)$ (respectively, $\mathrm{R}(l)$) is a leaf node of the binary tree. Otherwise,

$$\mathcal{B}^{\mathrm{L}(l)}_l = \begin{bmatrix} \mathrm{U}^{\mathrm{L}(l)}_{l,\mathrm{L}(\mathrm{L}(l))} \mathcal{B}^{\mathrm{L}(\mathrm{L}(l))}_{\mathrm{L}(l)} & \mathrm{U}^{\mathrm{L}(l)}_{l,\mathrm{R}(\mathrm{L}(l))} \mathcal{B}^{\mathrm{R}(\mathrm{L}(l))}_{\mathrm{L}(l)} \end{bmatrix},$$
$$\mathcal{B}^{\mathrm{R}(l)}_l = \begin{bmatrix} \mathrm{U}^{\mathrm{R}(l)}_{l,\mathrm{L}(\mathrm{R}(l))} \mathcal{B}^{\mathrm{L}(\mathrm{R}(l))}_{\mathrm{R}(l)} & \mathrm{U}^{\mathrm{R}(l)}_{l,\mathrm{R}(\mathrm{R}(l))} \mathcal{B}^{\mathrm{R}(\mathrm{R}(l))}_{\mathrm{R}(l)} \end{bmatrix}$$

with $\mathrm{U}^{\mathrm{L}(l)}_{l,\mathrm{L}(\mathrm{L}(l))} \in \mathbb{F}^{\rho(\mathrm{L}(l),l)\times\rho(\mathrm{L}(\mathrm{L}(l)),\mathrm{L}(l))}$, $\mathrm{U}^{\mathrm{L}(l)}_{\mathrm{R}(\mathrm{L}(l)),l} \in \mathbb{F}^{\rho(\mathrm{L}(l),l)\times\rho(\mathrm{R}(\mathrm{L}(l)),\mathrm{L}(l))}$, $\mathrm{U}^{\mathrm{R}(l)}_{\mathrm{L}(\mathrm{R}(l)),l} \in \mathbb{F}^{\rho(\mathrm{R}(l),l)\times\rho(\mathrm{L}(\mathrm{R}(l)),\mathrm{R}(l))}$, $\mathrm{U}^{\mathrm{R}(l)}_{l,\mathrm{R}(\mathrm{R}(l))} \in \mathbb{F}^{\rho(\mathrm{R}(l),l)\times\rho(\mathrm{R}(\mathrm{R}(l)),\mathrm{R}(l))}$. At the root level, we set $\mathrm{A} = \mathrm{A}^r \in \mathbb{F}^{m_r\times n_r}$ with $m_r =: M$ and $n_r =: N$. For example, the matrix

$$\mathrm{A} = \begin{bmatrix} \mathrm{D}^1 & \mathrm{Q}^1_3\mathrm{V}^3_{1,2}\mathrm{B}^2_3 & \mathrm{Q}^1_5\mathrm{W}^3_{1,5}\mathrm{V}^5_{3,4}\mathrm{B}^4_5 \\ \mathrm{Q}^1_3\mathrm{V}^3_{2,1}\mathrm{B}^1_3 & \mathrm{D}^2 & \mathrm{Q}^2_5\mathrm{W}^3_{1,5}\mathrm{V}^5_{3,4}\mathrm{B}^4_5 \\ \mathrm{Q}^4_5\mathrm{V}^5_{4,3}\mathrm{U}^3_{5,1}\mathrm{B}^1_3 & \mathrm{Q}^4_5\mathrm{V}^5_{4,3}\mathrm{U}^3_{5,2}\mathrm{B}^2_3 & \mathrm{D}^4 \end{bmatrix} \tag{2.4}$$
$$= \begin{bmatrix} \mathrm{A}^3 & \mathcal{Q}^3_5\mathrm{V}^5_{3,4}\mathcal{B}^4_5 \\ \mathcal{Q}^4_5\mathrm{V}^5_{4,3}\mathcal{B}^3_5 & \mathrm{A}^4 \end{bmatrix}$$
$$= \mathrm{A}^5$$

is an HSS matrix.

2.3. Algebra with SSS and HSS matrices. SSS and HSS matrices share common algebraic properties. Any dense matrix can be converted into an SSS or HSS matrix. The dimensions of the generators, and thus the efficiency of the representation, are specified by the ranks of the off-diagonal (i.e., so-called Hankel) blocks. Sums and products of SSS (respectively, HSS) matrices are again SSS (respectively, HSS) but with a doubling in the size of the generators. The inverse of an SSS (respectively, HSS) is an SSS (respectively, HSS) matrix of the same generator dimensions. SSS (respectively, HSS) representations of matrices that have small Hankel block ranks can be multiplied with vectors in linear time. The same holds for the solve operation. For the latter, one common approach to both representations is a lifting technique that solves the linear system as a larger block-sparse system [6]. All these commonalities do not come from nowhere, since SSS and HSS belong to the same family of a more general class of matrices.

3. Tree quasi-separable matrices.

This section formally introduces TQS matrices. To do so, we first introduce some terminology in subsections 3.1 and 3.2. The actual definition of TQS matrices is given in subsection 3.3. Finally, subsection 3.4 describes SSS and HSS as special cases of TQS matrices.

3.1. Tree graphs. Let $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, with node set $\mathbb{V} = \{1, 2, \ldots, K\}$, be a *connected acyclic undirected graph*. Here, for reasons that will be clear later, the edge set $\mathbb{E}$ is *unconventionally* interpreted as a collection of *ordered pairs* of nodes (instead of unordered pairs!), but with the property that $(i, j) \in \mathbb{E}$ if and only if $(j, i) \in \mathbb{E}$, i.e., the edges in both directions are included. The number of edges incident to a node $i \in \mathbb{V}$, i.e., its degree, is denoted by $\deg(i)$. Since $\mathbb{G}$ is acyclic, every pair of nodes $i, j \in \mathbb{G}$ is connected by one, and only one, *path*[1]

$$\mathbb{P}(i, j) = i - w_2 - \cdots - w_{p-1} - j$$

of specific length p.

The graph $\mathbb{G}$ may be interpreted as a rooted tree.[2] Indeed, if $\mathcal{N}(i; k)$ denotes the *set of k-neighbors* of node $i \in \mathbb{V}$, i.e., the set of nodes that can be reached by i through

[1] Here we do not allow for self-intersecting paths. A "path" passing from node i to node j by passing through i and going to k, to subsequently come back to node j, is *not* considered a valid path!

[2] Although one may work without such an interpretation, this viewpoint shall be useful for deriving some of the results in the paper. Specifically, it allows the labeling of the generating matrices, which in turn simplifies validating the correctness of Algorithm 1.

traversing a path of length k, then a tree $\mathbb{G}(r)$ is induced by picking any $r \in \mathbb{V}$ and setting

$$\mathbb{V}_0 = \{r\}, \quad \mathbb{V}_1 = \mathcal{N}(r;1), \quad \mathbb{V}_2 = \mathcal{N}(r;2), \quad \ldots, \quad \mathbb{V}_L = \mathcal{N}(r;L).$$

The depth of the tree specified by the partition $\mathbb{V} = \mathbb{V}_0 \cup \mathbb{V}_1 \cup \cdots \cup \mathbb{V}_L$ equals L and is defined as the smallest number with the property $\mathcal{N}(r;L+1) = \emptyset$. The level l at which a node $i \in \mathbb{V}_l$ resides within the tree is denoted by $\mathcal{L}(i)$ (note that this depends implicitly on the choice of root node made earlier!). An edge $(i,j) \in \mathbb{E}$ with $\mathcal{L}(i) = \mathcal{L}(j) - 1$ is called an *up-edge* since it is directed toward the root node. Likewise, if $\mathcal{L}(i) = \mathcal{L}(j) + 1$ it is called a *down-edge* since it is directed away from the root node.

Within the setting of the tree $\mathbb{G}(r)$, a node $j \in \mathbb{V}$ is considered to be a child of $i \in \mathcal{N}(j) = \mathcal{N}(j;1)$ if $i \in \mathbb{V}_l$ and $j \in \mathbb{V}_{l+1}$. Vice versa, i is the parent of node j. The notion of children and parents may be further generalized: $j \in \mathbb{V}$ is a k-child (with $k > 0$) of $i \in \mathcal{N}(j;k)$ (and vice versa i is the k-parent of node j) if $i \in \mathbb{V}_l$ and $j \in \mathbb{V}_{l+k}$. Two nodes $i, j \in \mathbb{V}_l$ are siblings (k-siblings) if they both share the same parent node $w \in \mathbb{V}_{l-1}$ ($w \in \mathbb{V}_{l-k}$). Naturally, a node will have at most one k-parent, but it may have many k-children or k-siblings. The parent (k-parent) of a node $i \in \mathbb{V}$ is denoted by $\mathcal{P}(i)$ ($\mathcal{P}(i;k)$) and this set may be empty (e.g., for the root node). The set of children (k-children) of node $i \in \mathbb{V}$ is denoted by $\mathcal{C}(i)$ ($\mathcal{C}(i;k)$). The set of siblings (k-siblings) of node $i \in \mathbb{V}$ is denoted by $\mathcal{S}(i)$ ($\mathcal{S}(i;k)$). The set of descendants of node $i \in \mathbb{V}$ and including i itself is denoted by

$$\mathcal{D}(i) := i \cup \mathcal{C}(i;1) \cup \mathcal{C}(i;2) \cup \cdots \cup \mathcal{C}(i;L - \mathcal{L}(i)).$$

and we set $\bar{\mathcal{D}}(i) := \mathbb{V} \setminus \mathcal{D}(i)$. A node $i \in \mathbb{V}$ is called a *leaf node* if its set of descendants is empty. Note that leaf nodes may reside at any level of the tree. The set of leaf nodes of an acyclic fully connected graph is denoted by $\mathscr{L}(\mathbb{G})$.

3.2. Graph-partitioned matrices. We can associate the graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ with a block-partitioned matrix. To do this, associate each node with an input of size $n_i \in \mathbb{N} \cup \{0\}$ and an output of size $m_i \in \mathbb{N} \cup \{0\}$. Note that we allow the sizes of the inputs and outputs to be *empty*! Now, let

$$M = \sum_{i \in \mathbb{V}} m_i, \quad N = \sum_{i \in \mathbb{V}} n_i.$$

We may then introduce a matrix $\mathrm{T} \in \mathbb{F}^{M \times N}$ that is assembled from the system of linear equations

$$\boldsymbol{b}_i = \sum_{j \in \mathbb{V}} \mathrm{T}\{i,j\} \boldsymbol{x}_j, \quad i \in \mathbb{V}. \tag{3.1}$$

That is,

$$\mathrm{T} := \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ K \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \cdots & K \end{array} \\ \begin{bmatrix} \mathrm{T}\{1,1\} & \mathrm{T}\{1,2\} & \cdots & \mathrm{T}\{1,K\} \\ \mathrm{T}\{2,1\} & \mathrm{T}\{2,2\} & \cdots & \mathrm{T}\{2,K\} \\ \vdots & \vdots & & \vdots \\ \mathrm{T}\{K,1\} & \mathrm{T}\{K,2\} & \cdots & \mathrm{T}\{K,K\} \end{bmatrix} \end{array}, \tag{3.2}$$

where $\mathrm{T}\{i,j\} \in \mathbb{F}^{m_i \times n_j}$ is the matrix associated with the contribution of the input at node $i \in \mathbb{V}$ to the output at node $j \in \mathbb{V}$. Given two subsets $\mathbb{A} = \{i_1, \ldots, i_A\} \subset \mathbb{V}$ and $\mathbb{B} = \{j_1, \ldots, j_B\} \subset \mathbb{V}$ we have submatrices of T with the notation

$$\mathrm{T}\{\mathbb{A},\mathbb{B}\} = \begin{array}{c} \\ i_1 \\ \vdots \\ i_A \end{array} \begin{array}{c} \begin{array}{ccc} j_1 & \cdots & j_B \end{array} \\ \begin{bmatrix} \mathrm{T}\{i_1,j_1\} & \cdots & \mathrm{T}\{i_1,j_B\} \\ \vdots & & \vdots \\ \mathrm{T}\{i_A,j_1\} & \cdots & \mathrm{T}\{i_A,j_B\} \end{bmatrix} \end{array}.$$

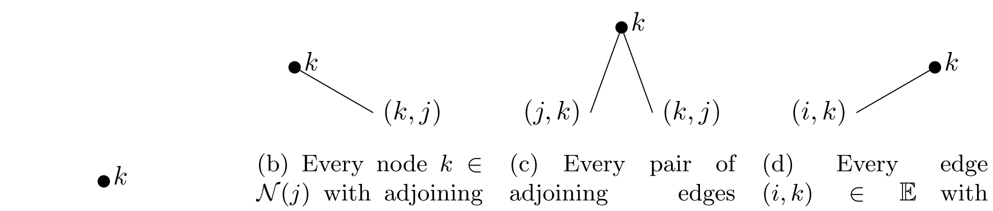
Actual entries of the matrix T or its block-components $\mathrm{T}\{i,j\}$ are accessed using a square-bracket notation, i.e., $\mathrm{T}[k,l]$ and $\mathrm{T}\{i,j\}[k,l]$ refer to the (k,l)th entry of T and $\mathrm{T}\{i,j\}$, respectively.

The matrix T alongside with $\mathbb{G}$ is referred to as a *graph-partitioned matrix.* Note that this definition is agnostic to whether $\mathbb{G}$ is a tree or not. Nonetheless, in this paper, we will limit ourselves to only acyclic undirected graphs that may also be viewed as trees.

3.3. Definition of TQS matrices. A TQS matrix is a specific construction of a graph-partitioned matrix T associated with a fully connected and acyclic graph $\mathbb{G}=(\mathbb{V},\mathbb{E})$. To form a TQS matrix from $\mathbb{G}$, we equip the graph with weights on the edges of the graph: every edge $(i,j)\in\mathbb{E}$ is associated with a weight $\rho_{(i,j)}\in\mathbb{N}\cup\{0\}$. Furthermore, every edge $(i,j)\in\mathbb{E}$ is associated with a state vector $\boldsymbol{h}_{(i,j)}\in\mathbb{F}^{\rho_{(i,j)}}$. Specifically, $\boldsymbol{h}_{(i,j)}$ describes a state computed in node i and transmitted to node j. The set $\{\rho_e\}_{e\in\mathbb{E}}$ is referred to as the *rank-profile* on $\mathbb{G}$. It collectively describes the state dimensions of all the edges. The TQS matrix is formed by running an input-driven LTV dynamical system on the graph. To describe this LTV system, the following matrices are introduced (see also Figure 1):

- an *input-to-output* operator $\mathrm{D}^k\in\mathbb{F}^{m_k\times n_k}$ for every node $k\in\mathbb{V}$ describing a mapping from input $\boldsymbol{x}_k$ to output $\boldsymbol{b}_k$,
- an *input-to-edge* operator $\mathrm{Inp}_j^k\in\mathbb{F}^{\rho_{(k,j)}\times n_k}$ for every node $k\in\mathcal{N}(j)$ and adjoining edge $(k,j)\in\mathbb{E}$ describing a mapping from input $\boldsymbol{x}_k$ to state $\boldsymbol{h}_{(k,j)}$,
- an *edge-to-edge* operator $\mathrm{Trans}_{i,j}^k\in\mathbb{F}^{\rho_{(j,k)}\times\rho_{(j,k)}}$ for every pair of adjoining edges $(j,k)\in\mathbb{E}$ and $(k,i)\in\mathbb{E}$ (with $i,j\in\mathcal{N}(k)$) describing a mapping from state $\boldsymbol{h}_{(j,k)}$ to state $\boldsymbol{h}_{(k,i)}$,
- and an *edge-to-output* operator $\mathrm{Out}_i^k\in\mathbb{F}^{m_k\times\rho_{(i,k)}}$ for every edge $(i,k)\in\mathbb{E}$ and adjoining node $i\in\mathcal{N}(k)$ describing a mapping from state $\boldsymbol{h}_{(j,k)}$ to output $\boldsymbol{b}_k$.

Notice that matrices (operators) belong to a node indicated by the superscript, while the subscripts implicitly refer to edges attached to the respective node, indicating the destination of the data being computed in that node. A TQS matrix is then defined as follows.



(a) Every node $k\in\mathbb{V}$ is equipped with an *input-to-output* operator.

(b) Every node $k\in\mathcal{N}(j)$ with adjoining edge $(k,j)\in\mathbb{E}$ is equipped with an *input-to-edge* operator.

(c) Every pair of adjoining edges $(j,k)\in\mathbb{E}$ and $(k,i)\in\mathbb{E}$ is equipped with an *edge-to-edge* operator.

(d) Every edge $(i,k)\in\mathbb{E}$ with adjoining node $i\in\mathcal{N}(k)$ is equipped with an *edge-to-output* operator.

FIG. 1. *TQS operators associated with elements of the graph* $\mathbb{G}$.

DEFINITION 3.1 (TQS matrices). *Let $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ be a connected acyclic undirected graph. A TQS matrix with input dimensions $\{n_i\}_{i\in\mathbb{V}}$, output dimensions $\{m_i\}_{i\in\mathbb{V}}$, and rank-profile $\{\rho_e\}_{e\in\mathbb{E}}$ is a graph-partitioned matrix $\mathrm{T} \in \mathbb{F}^{M\times N}$ of size $M = \sum_{i\in\mathbb{V}} m_i$ by $N = \sum_{i\in\mathbb{V}} n_i$ whose block entries are specified by*

$$\mathrm{T}\{i,j\} = \begin{cases} \mathrm{D}^i, & i = j, \\ \mathrm{Out}^i_{p_{\nu-1}} \mathrm{Trans}^{p_{\nu-1}}_{i,p_{\nu-2}} \cdots \mathrm{Trans}^{p_2}_{p_3,p_1} \mathrm{Trans}^{p_1}_{p_2,j} \mathrm{Inp}^j_{p_1}, & i \neq j, \end{cases} \tag{3.3}$$

with $j = p_0 - p_1 - \cdots - p_\nu = i$ being the unique path from node $j \in \mathbb{V}$ to node $i \in \mathbb{V}$.

The notation of the generating matrices of the TQS representation in Definition 3.1 is *agnostic* to a tree order. Later on, in section 5, we shall address the realization problem *or* the problem of constructing a TQS representation from a dense graph-partitioned matrix. To aid this construction, it shall be useful to "color" the generating matrices with respect to their orientation in the graph. Particularly, once a root node $r \in \mathbb{V}$ is chosen for the graph, we introduce additional *nomenclature* to distinguish between different input-to-edge, edge-to-edge, and edge-to-output operators:[3]

- An *input-to-edge* operator Inp^k_j is denoted by B^k_j if $j \in \mathbb{V}$ is a parent of $k \in \mathbb{V}$ (i.e., $j = \mathcal{P}(k)$) and by P^k_j if $j \in \mathbb{V}$ is a child of $k \in \mathbb{V}$ (i.e., $j \in \mathcal{C}(k)$).
- An *edge-to-edge* operator $\mathrm{Trans}^k_{i,j}$ is denoted by $\mathrm{U}^k_{i,j}$ if $j \in \mathbb{V}$ and $i \in \mathbb{V}$ are respectively a child and a parent of $k \in \mathbb{V}$ (i.e., $j \in \mathcal{C}(k)$ and $i = \mathcal{P}(k)$), by $\mathrm{V}^k_{i,j}$ if $i, j \in \mathbb{V}$ are siblings of $k \in \mathbb{V}$ (i.e. $i, j \in \mathcal{S}(k)$), and by $\mathrm{W}^k_{i,j}$ if $j \in \mathbb{V}$ and $i \in \mathbb{V}$ are respectively a parent and a child of $k \in \mathbb{V}$ (i.e., $j = \mathcal{P}(k)$ and $i \in \mathcal{C}(k)$).
- An *edge-to-output* operator Out^k_i is denoted by C^k_i if $i \in \mathbb{V}$ is a *child* of $k \in \mathbb{V}$ (i.e., $i \in \mathcal{C}(k)$) and by Q^k_i if $i \in \mathbb{V}$ is a *parent* of $k \in \mathbb{V}$ (i.e., $i = \mathcal{P}(k)$).

The input-to-output, input-to-edge, edge-to-edge, and edge-to-output operators can systematically be organized in so-called spinner matrices or transition maps at every node $k \in \mathbb{V}$. Let $i_1, i_2, \ldots, i_p \in \mathcal{C}(k)$ be an enumeration of the set of children and $j = \mathcal{P}(k)$ the parent node (see Figure 2). At every node $k \in \mathbb{V}$, the following relations must be satisfied:

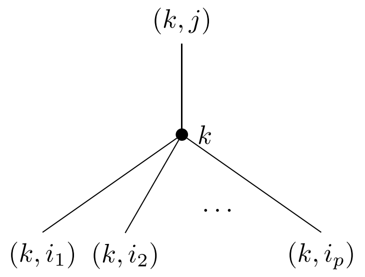


FIG. 2. *Node $k \in \mathbb{V}$ with its children $i_1, i_2, \ldots, i_p \in \mathcal{C}(k)$ and parent $j = \mathcal{P}(k)$.*

[3]Note that these operators are well-defined irrespective of the choice of the root.

$$
\begin{bmatrix} \boldsymbol{h}_{(k,i_1)} \\ \boldsymbol{h}_{(k,i_2)} \\ \vdots \\ \boldsymbol{h}_{(k,i_p)} \\ \boldsymbol{h}_{(k,j)} \\ \boldsymbol{b}_k \end{bmatrix} = \begin{bmatrix} 0 & \mathrm{V}^k_{i_1,i_2} & \cdots & \mathrm{V}^k_{i_1,i_p} & \mathrm{W}^k_{i_1,j} & \mathrm{P}^k_{i_1} \\ \mathrm{V}^k_{i_2,i_1} & 0 & & \mathrm{V}^k_{i_2,i_p} & \mathrm{W}^k_{i_2,i_1} & \mathrm{P}^k_{i_2} \\ \vdots & \vdots & \ddots & & \vdots & \vdots \\ \mathrm{V}^k_{i_p,i_1} & \mathrm{V}^k_{i_p,i_2} & \cdots & 0 & \mathrm{W}^k_{i_p,j} & \mathrm{P}^k_{i_p} \\ \mathrm{U}^k_{j,i_i} & \mathrm{U}^k_{j,i_2} & \cdots & \mathrm{U}^k_{j,i_p} & 0 & \mathrm{B}^k_j \\ \mathrm{C}^k_{i_1} & \mathrm{C}^k_{i_2} & \cdots & \mathrm{C}^k_{i_p} & \mathrm{Q}^k_j & \mathrm{D}^k \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_{(i_1,k)} \\ \boldsymbol{h}_{(i_2,k)} \\ \vdots \\ \boldsymbol{h}_{(i_p,k)} \\ \boldsymbol{h}_{(j,k)} \\ \boldsymbol{x}_k \end{bmatrix}. \tag{3.4}
$$

The spinner matrices of the TQS representation are

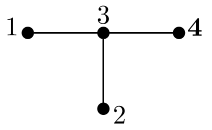
$$
\mathrm{S}^k := \begin{array}{c} \\ i_1 \\ i_2 \\ \vdots \\ i_p \\ j \\ \mathrm{Out} \end{array} \begin{array}{c} \begin{array}{cccccc} i_1 & i_2 & \cdots & i_p & j & \mathrm{Inp} \end{array} \\ \begin{bmatrix} 0 & \mathrm{V}^k_{i_1,i_2} & \cdots & \mathrm{V}^k_{i_1,i_p} & \mathrm{W}^k_{i_1,j} & \mathrm{P}^k_{i_1} \\ \mathrm{V}^k_{i_2,i_1} & 0 & & \mathrm{V}^k_{i_2,i_p} & \mathrm{W}^k_{i_2,i_1} & \mathrm{P}^k_{i_2} \\ \vdots & & \ddots & \vdots & \vdots & \vdots \\ \mathrm{V}^k_{i_p,i_1} & \mathrm{V}^k_{i_p,i_2} & \cdots & 0 & \mathrm{W}^k_{i_p,j} & \mathrm{P}^k_{i_p} \\ \mathrm{U}^k_{j,i_i} & \mathrm{U}^k_{j,i_2} & \cdots & \mathrm{U}^k_{j,i_p} & 0 & \mathrm{B}^k_j \\ \mathrm{C}^k_{i_1} & \mathrm{C}^k_{i_2} & \cdots & \mathrm{C}^k_{i_p} & \mathrm{Q}^k_j & \mathrm{D}^k \end{bmatrix} \end{array}, \qquad k \in \mathbb{V}. \tag{3.5}
$$

The spinner matrices reveal how many numerical entries are involved in a TQS matrix. Specifically, this number equals

$$
\sum_{i\in\mathbb{V}} \left(m_i + \sum_{j\in\mathcal{N}(i)} r_{(i,j)} \right) \left(n_i + \sum_{j\in\mathcal{N}(i)} r_{(j,i)} \right) - \sum_{j\in\mathcal{N}(i)} r_{(i,j)} r_{(j,i)}. \tag{3.6}
$$

To illustrate our definition with an example, consider the following tree:

$\mathbb{G}_a(4)$:



with node 4 (highlighted in bold) designated to be the root node. The TQS matrix for the corresponding tree $\mathbb{G}_a(4)$ has the structure

$$
\mathrm{T} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & \mathrm{D}^1 & \mathrm{Q}^1_3\mathrm{V}^3_{1,2}\mathrm{B}^2_3 & \mathrm{Q}^1_3\mathrm{P}^3_1 & \mathrm{Q}^1_3\mathrm{W}^3_{1,4}\mathrm{P}^4_3 \\ 2 & \mathrm{Q}^2_3\mathrm{V}^3_{2,1}\mathrm{B}^1_3 & \mathrm{D}^2 & \mathrm{Q}^2_3\mathrm{P}^3_2 & \mathrm{Q}^2_3\mathrm{W}^3_{2,4}\mathrm{P}^4_3 \\ 3 & \mathrm{C}^3_1\mathrm{B}^1_3 & \mathrm{C}^3_2\mathrm{B}^2_3 & \mathrm{D}^3 & \mathrm{Q}^3_4\mathrm{P}^4_3 \\ 4 & \mathrm{C}^4_3\mathrm{U}^3_{4,1}\mathrm{B}^1_3 & \mathrm{C}^4_3\mathrm{U}^3_{4,2}\mathrm{B}^2_3 & \mathrm{C}^4_3\mathrm{B}^3_4 & \mathrm{D}^4 \end{array}.
$$

The spinner matrices take on the form

$$
\mathrm{S}^1 = \begin{array}{c|cc} & \mathbf{3} & \mathrm{Inp} \\ \hline \mathbf{3} & 0 & \mathrm{B}^1_3 \\ \mathrm{Out} & \mathrm{Q}^1_3 & \mathrm{D}^1 \end{array}, \quad \mathrm{S}^2 = \begin{array}{c|cc} & \mathbf{3} & \mathrm{Inp} \\ \hline \mathbf{3} & 0 & \mathrm{B}^2_3 \\ \mathrm{Out} & \mathrm{Q}^2_3 & \mathrm{D}^2 \end{array}, \quad \mathrm{S}^4 = \begin{array}{c|cc} & 3 & \mathrm{Inp} \\ \hline 3 & 0 & \mathrm{P}^4_3 \\ \mathrm{Out} & \mathrm{C}^4_3 & \mathrm{D}^4 \end{array},
$$

$$
\mathrm{S}^3 = \begin{array}{c|cccc} & 1 & 2 & \mathbf{4} & \mathrm{Inp} \\ \hline 1 & 0 & \mathrm{V}^3_{1,2} & \mathrm{W}^3_{1,4} & \mathrm{P}^3_1 \\ 2 & \mathrm{V}^3_{2,1} & 0 & \mathrm{W}^3_{2,4} & \mathrm{P}^3_2 \\ \mathbf{4} & \mathrm{U}^3_{4,1} & \mathrm{U}^3_{4,2} & 0 & \mathrm{B}^3_4 \\ \mathrm{Out} & \mathrm{C}^3_1 & \mathrm{C}^3_2 & \mathrm{Q}^3_4 & \mathrm{D}^4 \end{array},
$$

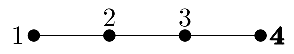
where the parent nodes of each spinner matrix are highlighted in bold.

In general, one can expect a TQS representation to be efficient if the state dimensions are small and the degrees of the nodes are not large. For example, if $M = N = K$, and

$$\rho_{\max} := \max_{e\in\mathbb{E}} \rho_e \ll N, \qquad \deg_{\max} := \max_{i\in\mathbb{V}} \deg(i) \ll N, \tag{3.7}$$

consideration of (3.6) reveals that the number of parameters in the representation is of the order $\Theta((1+d\rho)^2N)$. Later on, in subsection 4.1, we shall see that the set of state dimensions $\{\rho_e\}_{e\in\mathbb{E}}$ will allow us to put bounds on the ranks of certain submatrices of a TQS matrix.

3.4. HSS and SSS matrices as special subcases. While SSS and HSS matrices have their origins in disparately different fields, TQS matrices bring them under one roof as it includes SSS matrices and HSS matrices as special subcases. To further elaborate on this, consider line graph

$\mathbb{G}_b(4)$:


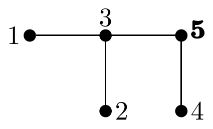
with node 4 chosen as the root node. The spinner matrices associated with $\mathbb{G}_b(4)$ are of the form

$$\mathrm{S}^1 = \begin{array}{c|cc} & \mathbf{2} & \mathrm{Inp} \\ \hline \mathbf{2} & 0 & \mathrm{B}^1_2 \\ \mathrm{Out} & \mathrm{Q}^1_2 & \mathrm{D}^1 \end{array}, \quad \mathrm{S}^2 = \begin{array}{c|ccc} & 1 & \mathbf{3} & \mathrm{Inp} \\ \hline 1 & 0 & \mathrm{W}^2_{1,3} & \mathrm{P}^2_1 \\ \mathbf{3} & \mathrm{U}^2_{3,1} & 0 & \mathrm{B}^2_3 \\ \mathrm{Out} & \mathrm{C}^2_1 & \mathrm{Q}^2_3 & \mathrm{D}^2 \end{array},$$

$$\mathrm{S}^3 = \begin{array}{c|ccc} & 2 & \mathbf{4} & \mathrm{Inp} \\ \hline 2 & 0 & \mathrm{W}^3_{2,4} & \mathrm{P}^3_2 \\ \mathbf{4} & \mathrm{U}^3_{4,2} & 0 & \mathrm{B}^3_4 \\ \mathrm{Out} & \mathrm{C}^3_2 & \mathrm{Q}^3_4 & \mathrm{D}^3 \end{array}, \quad \mathrm{S}^4 = \begin{array}{c|cc} & 3 & \mathrm{Inp} \\ \hline 3 & 0 & \mathrm{P}^4_3 \\ \mathrm{Out} & \mathrm{C}^4_3 & \mathrm{D}^4 \end{array},$$

and generate the SSS matrix shown in (2.2). In general, all linearly ordered line graphs produce a TQS matrix of the type described in subsection 2.1 if the final node is chosen as the root node.

HSS matrices, on the other hand, are TQS matrices associated with binary trees. For example, consider the postordered binary tree

$\mathbb{G}_c(5)$:


with node 5 as the root node. Furthermore, suppose that $n_3 = m_3 = n_5 = m_5 = 0$, i.e., all the nonleaf nodes are "empty nodes" with *zero* input and output dimensions. The spinner matrices of $\mathbb{G}_c(5)$ are of the form

$$\mathrm{S}^1 = \begin{array}{c|cc} & \mathbf{3} & \mathrm{Inp} \\ \hline \mathbf{3} & 0 & \mathrm{B}^1_3 \\ \mathrm{Out} & \mathrm{Q}^1_3 & \mathrm{D}^1 \end{array}, \quad \mathrm{S}^2 = \begin{array}{c|cc} & \mathbf{3} & \mathrm{Inp} \\ \hline \mathbf{3} & 0 & \mathrm{B}^2_3 \\ \mathrm{Out} & \mathrm{Q}^2_3 & \mathrm{D}^2 \end{array}, \quad \mathrm{S}^4 = \begin{array}{c|cc} & \mathbf{5} & \mathrm{Inp} \\ \hline \mathbf{5} & 0 & \mathrm{B}^4_5 \\ \mathrm{Out} & \mathrm{Q}^4_5 & \mathrm{D}^4 \end{array},$$

$$\mathrm{S}^3 = \begin{array}{c|cccc} & 1 & 2 & \mathbf{5} & \mathrm{Inp} \\ \hline 1 & 0 & \mathrm{V}^3_{1,2} & \mathrm{W}^3_{1,5} & | \\ 2 & \mathrm{V}^3_{2,1} & 0 & \mathrm{W}^3_{2,5} & | \\ \mathbf{5} & \mathrm{U}^3_{5,1} & \mathrm{U}^3_{5,2} & 0 & | \\ \mathrm{Out} & - & - & - & . \end{array}, \quad \mathrm{S}^5 = \begin{array}{c|ccc} & 3 & 4 & \mathrm{Inp} \\ \hline 3 & 0 & \mathrm{V}^5_{3,4} & | \\ 4 & \mathrm{V}^5_{4,3} & 0 & | \\ \mathrm{Out} & - & - & . \end{array},$$

and generate the HSS matrix in (2.4). In general, all postordered binary tree graphs produce a TQS matrix of the type described in subsection 2.2, provided that all nonleaf nodes are empty nodes of zero dimensions.

In all other cases, the TQS matrices are neither SSS nor HSS. The example of the previous section and the upcoming example in subsection 5.1 belong to this category.

3.5. TQS representations of sparse matrices. The construction (or realization) of a TQS representation will be treated in section 5; however, for sparse matrices whose adjacency graph is a tree, the TQS representation can be written down directly from inspection. For example, consider the family of sparse matrices whose adjacency graph corresponds to a regular k-level binary tree (see Figure 3). Using a "nested dissection"-styled ordering of the entries, we obtain the sequence of matrices

$$\mathrm{T}_1 = [d^1], \quad \mathrm{T}_2 = \begin{bmatrix} d^1 & & p^3_1 \\ & d^2 & p^3_2 \\ b^1_3 & b^2_3 & d^3 \end{bmatrix}, \quad \mathrm{T}_3 = \begin{bmatrix} d^1 & & p^3_1 & & & & \\ & d^2 & p^3_2 & & & & \\ b^1_3 & b^2_3 & d^3 & & & & p^7_3 \\ & & & d^4 & & p^6_4 & \\ & & & & d^5 & p^6_5 & \\ & & & b^4_6 & b^5_6 & d^6 & p^7_6 \\ & & b^3_7 & & & b^6_7 & d^7 \end{bmatrix},$$

$$\mathrm{T}_4 = \begin{bmatrix} d^1 & & p^3_1 & & & & & & & & & & & & \\ & d^2 & p^3_2 & & & & & & & & & & & & \\ b^1_3 & b^2_3 & d^3 & & & & p^7_3 & & & & & & & & \\ & & & d^4 & & p^6_4 & & & & & & & & & \\ & & & & d^5 & p^6_5 & & & & & & & & & \\ & & & b^4_6 & b^5_6 & d^6 & p^7_6 & & & & & & & & \\ & & b^3_7 & & & b^6_7 & d^7 & & & & & & & & p^{15}_7 \\ & & & & & & & d^8 & & p^{10}_8 & & & & & \\ & & & & & & & & d^9 & p^{10}_9 & & & & & \\ & & & & & & & b^8_{10} & b^9_{10} & d^{10} & & & & p^{14}_{10} & \\ & & & & & & & & & & d^{11} & & p^{13}_{11} & & \\ & & & & & & & & & & & d^{12} & p^{13}_{12} & & \\ & & & & & & & & & & b^{11}_{13} & b^{12}_{13} & d^{13} & b^{14}_{13} & \\ & & & & & & & & & b^{10}_{14} & & & b^{13}_{14} & d^{14} & p^{15}_{14} \\ & & & & & & b^7_{15} & & & & & & & b^{14}_{15} & d^{15} \end{bmatrix},$$

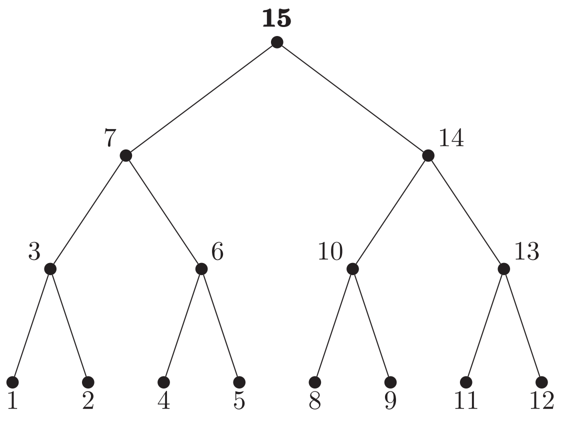


FIG. 3. *A level $k = 4$ binary tree with a "nested dissection"-styled ordering of the nodes.*

..., etc. For the specified ordering of the entries,[4] the memory complexity of storing the matrices $\{\mathrm{T}_k\}_{k=1}^{\infty}$ with a TQS representation depends on the choice of the partition and the graph. For standard HSS constructions, the matrices $\{\mathrm{T}_k\}_{k=1}^{\infty}$ are compressed using the hierarchical partitioning delineated above. These constructions lead to an $O(k)$ growth in the HSS Hankel block ranks and the representation will require $\Theta(K \log K)$ parameters, where $K \sim 2^k$ denotes the number of nodes in tree. The same asymptotic storage complexity is also attained with an SSS representation. Alternatively, one can use the adjacency graphs of T_k as the corresponding graph for the TQS representation. Doing so, the *root node* and *leaf nodes* will have the spinner matrices[5]

$$(3.8) \qquad \mathrm{S}^r = \begin{array}{c} \\ \mathrm{L}(r) \\ \mathrm{R}(r) \\ \mathrm{Out} \end{array} \begin{array}{c} \begin{array}{ccc} \mathrm{L}(r) & \mathrm{R}(r) & \mathrm{Inp} \end{array} \\ \begin{bmatrix} 0 & 0 & p^r_{\mathrm{L}(r)} \\ 0 & 0 & p^r_{\mathrm{R}(r)} \\ 1 & 1 & d^r \end{bmatrix} \end{array} \quad \text{and} \quad \mathrm{S}^i = \begin{array}{c} \\ \boldsymbol{\mathcal{P}(i)} \\ \mathrm{Out} \end{array} \begin{array}{c} \begin{array}{cc} \boldsymbol{\mathcal{P}(i)} & \mathrm{Inp} \end{array} \\ \begin{bmatrix} 0 & b^i_{\mathcal{P}(i)} \\ 1 & d^i \end{bmatrix} \end{array},$$

respectively, while the *interior nodes* (i.e., nonleaf and nonroot nodes) take on the form

$$(3.9) \qquad S_i = \begin{array}{c} \\ \mathrm{L}(i) \\ \mathrm{R}(i) \\ \boldsymbol{\mathcal{P}(i)} \\ \mathrm{Out} \end{array} \begin{array}{c} \begin{array}{cccc} \mathrm{L}(i) & \mathrm{R}(i) & \boldsymbol{\mathcal{P}(i)} & \mathrm{Inp} \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & p^i_{\mathrm{L}(i)} \\ 0 & 0 & 0 & p^i_{\mathrm{R}(i)} \\ 0 & 0 & 0 & b^i_{\mathcal{P}(i)} \\ 1 & 1 & 1 & d^i \end{bmatrix} \end{array}.$$

Notice that all the edge-to-edge operators are *zero* while the edge-to-output operators are equal to *one.* As we shall see in Proposition 4.8 in subsection 4.3, the inverse of T_k will also have a TQS representation of exactly the same dimensions. The edge-to-edge operators will no longer be zero.

The resulting TQS representations from the spinner matrices (3.8) and (3.9) are neither SSS nor HSS. With this representation, the complexity of storing T_k has become linear (i.e., $\Theta(K)$), which is a logarithmic improvement over SSS and HSS. It should be noted that picking the adjacency graph as the corresponding graph for the TQS representation is *not* always a sensible choice. A canonical example is the K-by-K arrowhead matrix

$$\begin{bmatrix} d^1 & & & & p^K_1 \\ & d^2 & & & p^K_2 \\ & & \ddots & & \vdots \\ & & & d^{K-1} & p^K_{K-1} \\ b^1_K & b^2_K & \cdots & b^{K-1}_K & d^K \end{bmatrix}.$$

The complexity for the TQS representation would be comparable to the dense matrix itself. This is because the adjacency graph contains a node for which the degree grows with $K-1$. The line graph is a far better choice, resulting in a linear complexity.

[4]Note that the recovery of the optimal ordering which enables the most efficient TQS representations involves combinatorial search!

[5]Recall that L(l) (respectively, R(l)) denotes the left (respectively, right) child of node l a binary tree.

4. Algebraic properties of TQS matrices.

This section discusses important algebraic properties of TQS matrices. Subsection 4.1 describes the GIRS property of TQS matrices. Subsection 4.2 introduces the theorem that characterizes the minimal TQS representation for any graph-partitioned matrix whose graph is a tree. Subsection 4.3 discusses the properties of TQS matrices under addition, products, and inversion.

4.1. Graph-induced rank structure of TQS matrices.

Inverses of tridiagonal matrices are examples of dense matrices whose off-diagonal blocks possess low-rank qualities. For graph-partitioned matrices $(\mathrm{T},\mathbb{G})$, this property of off-diagonal low rank can be expressed in a more general framework through the notion of GIRS. To describe GIRS, we first introduce the concept of a Hankel block. Let $\mathbb{A}\subset\mathbb{V}$ and let $\bar{\mathbb{A}}=\mathbb{V}\setminus\mathbb{A}$ denote its complement. We may (block-)permute T such that

$$\Pi_1 T\Pi_2=\begin{bmatrix}\mathrm{T}\{\mathbb{A},\mathbb{A}\} & \mathrm{T}\{\mathbb{A},\bar{\mathbb{A}}\}\\ \mathrm{T}\{\bar{\mathbb{A}},\mathbb{A}\} & \mathrm{T}\{\bar{\mathbb{A}},\bar{\mathbb{A}}\}\end{bmatrix}.$$

We call $\mathrm{T}\{\bar{\mathbb{A}},\mathbb{A}\}$ the *Hankel block induced by* $\mathbb{A}$. An edge $(i,j)\in\mathbb{E}$ is called a *border edge* with respect to $\mathbb{A}$ if $i\in\mathbb{A}$ and $j\in\bar{\mathbb{A}}$. The number of border edges, or the *edge count,* induced by $\mathbb{A}$ is denoted $\mathcal{E}(\mathbb{A})$. The GIRS property is defined as follows.

DEFINITION 4.1 (GIRS property). *The pair* $(\mathrm{T},\mathbb{G})$ *is said to satisfy the graph-induced (low-)rank structure for a constant* $c\geq 0$ *if* $\forall\mathbb{A}\subset\mathbb{V}$, *we have*

$$\operatorname{rank}\mathrm{T}\{\bar{\mathbb{A}},\mathbb{A}\}\leq c\mathcal{E}(\mathbb{A}).$$

TQS matrices turn out to satisfy a GIRS property. To see this, one must exploit the direct correspondence between the path followed from node $j\in\mathbb{V}$ to $i\in\mathbb{V}$ in $\mathbb{G}$ and the associated matrix expression at block-entry $\mathrm{T}\{i,j\}$. We may prove the following result.

PROPOSITION 4.2. *A TQS matrix* $\mathrm{T}\in\mathbb{F}^{M\times N}$ *with rank-profile* $\{\rho_e\}_{e\in\mathbb{E}}$ *satisfies the GIRS property for* $c=\max_{e\in\mathbb{E}}\rho_e$.

Proof. Let $\mathbb{A}\subset\mathbb{V}$ and $\bar{\mathbb{A}}=\mathbb{V}\setminus\mathbb{A}$. We must show that the rank of $\mathrm{T}\{\bar{\mathbb{A}},\mathbb{A}\}$ is bounded by $\mathcal{E}(\mathbb{A})\cdot\max_{e\in\mathbb{E}}\rho_e$. Suppose $i\in\mathbb{A}$ and $j\in\bar{\mathbb{A}}$. Since $\mathbb{G}$ is acyclic, there exists a unique path $\mathbb{P}(i,j)$ connecting i to j. By construction, $\mathbb{P}(i,j)$ first starts at a node in $\mathbb{A}$ to eventually transition into a node in $\bar{\mathbb{A}}$. It may be possible that $\mathbb{P}(i,j)$ leaves and enters $\mathbb{A}$ multiple times before it finally enters back into $\bar{\mathbb{A}}$ one last time to reach node j. Write

$$\mathbb{P}(i,j)=i-\cdots-s-t-\cdots-v-w-\cdots-j,$$

where $(s,t)\in\mathbb{E}$ (with $s\in\mathbb{A}$, $t\in\bar{\mathbb{A}}$) marks the first time $\mathbb{P}(i,j)$ entering $\bar{\mathbb{A}}$ and $(v,w)\in\mathbb{E}$ (with $s\in\mathbb{A}$, $t\in\bar{\mathbb{A}}$) marks the last time $\mathbb{P}(i,j)$ leaving $\mathbb{A}$. We may factor

$$\mathrm{T}\{j,i\}=\Gamma^{j}_{(v,w)}\Phi_{(v,w),(s,t)}\Psi^{i}_{(s,t)}, \tag{4.1}$$

where

$$\Psi^{i}_{(s,t)} := \begin{cases} \mathrm{Inp}^{s}_{t}, & i = s, \\ \mathrm{Trans}^{s}_{t,*} \cdots \mathrm{Trans}^{*}_{*,j}\mathrm{Inp}^{i}_{*}, & i \neq s, \end{cases}$$

$$\Phi_{(v,w),(s,t)} := \begin{cases} \mathrm{Id}, & (s,t) = (v,w), \\ \mathrm{Trans}^{v}_{w,*} \cdots \mathrm{Trans}^{t}_{*,s}, & (s,t) \neq (v,w), \end{cases}$$

$$\Gamma^{j}_{(v,w)} := \begin{cases} \mathrm{Out}^{w}_{v}, & j = w, \\ \mathrm{Out}^{j}_{*}\mathrm{Trans}^{*}_{j,*} \cdots \mathrm{Trans}^{v}_{*,w}, & j \neq t. \end{cases}$$

Let $\{e_i\}_{i=1}^{\mathcal{E}(A)} \subset \mathbb{E}$ denote the set of border edges, and

$$\Phi := \begin{bmatrix} \Phi_{e_1,e_1} & \cdots & \Phi_{e_1,e_{\mathcal{E}(A)}} \\ \vdots & & \vdots \\ \Phi_{e_{\mathcal{E}(A)},e_1} & \cdots & \Phi_{e_{\mathcal{E}(A)},e_{\mathcal{E}(A)}} \end{bmatrix}.$$

Given (4.1), it becomes evident that we may factor $\mathrm{T}\{\bar{\mathbb{A}}, \mathbb{A}\} = \Gamma\Phi\Psi$. Thus,

$$\operatorname{rank} \mathrm{T}\{\bar{\mathbb{A}}, \mathbb{A}\} = \sum_{i=1}^{\mathcal{E}(\mathbb{A})} \rho_{e_i} \leq \mathcal{E}\,(\mathbb{A}) \cdot \max_{e\in\mathbb{E}} \rho_e. \qquad \square$$

4.2. Minimal TQS representations. Proposition 4.2 shows that TQS matrices satisfy the GIRS property for $c = \max_{e\in\mathbb{E}} \rho_e$. A question arises as to whether the converse also holds. If a tree-graph-partitioned matrix satisfies the GIRS property for $c > 0$, does this then imply the existence of a TQS representation whose rank-profile is bounded by the GIRS constant? For SSS and HSS matrices both these questions can be answered in the affirmative. Interestingly, the same holds also for the more general TQS matrices. To answer this question, one must study the problem of constructing a minimal TQS representation.

Definition 4.3 (minimal TQS representation). *Let* $(\mathrm{T}, \mathbb{G})$ *be a graph-partitioned matrix with* $\mathbb{G}$ *acyclic and connected. A TQS representation for* T *with rank-profile* $\{\rho_e\}_{e\in\mathbb{E}}$ *is called minimal if any other TQS representation for* T *with rank-profile* $\{\rho'_e\}_{e\in\mathbb{E}}$ *satisfies* $\rho'_e \geq \rho_e \;\forall\; e \in \mathbb{E}$.

Given a graph-partitioned matrix $(\mathrm{T}, \mathbb{G})$ with G acyclic and connected, the rank-profile of the corresponding TQS representation can be derived from the ranks of Hankel blocks whose edge count is unity. For a tree, these so-called unit Hankel blocks are quite straightforward to enlist as every edge $(i, j) \in \mathbb{E}$ can be uniquely paired with one such unit Hankel block. Indeed, once a root node $r \in \mathbb{V}$ for the graph has been picked, it must hold that either $j = \mathcal{P}(i)$ or $i = \mathcal{P}(j)$ for the corresponding tree $\mathbb{G}(r)$. Let

$$\mathbb{H}_{(i,j)} = \begin{cases} \mathcal{D}(i), & j = \mathcal{P}(i), \\ \mathbb{V} \setminus \mathcal{D}(j), & i = \mathcal{P}(j), \end{cases}$$

and let $\mathrm{H}_{(i,j)}$ denote the Hankel block corresponding with subset $\mathbb{H}_{(i,j)}$, i.e., $\mathrm{H}_{(i,j)} = \mathrm{T}\{\mathbb{H}_{(i,j)}, \bar{\mathbb{H}}\}_{(i,j)}$ with $\bar{\mathbb{H}}_{(i,j)} = \mathbb{V} \setminus \mathbb{H}_{(i,j)}$. We have the following result.

THEOREM 4.4. *Let* $(\mathrm{T},\mathbb{G})$ *be a graph-partitioned matrix with* $\mathbb{G}$ *acyclic and connected. Then* $\mathrm{T}\in\mathbb{F}^{M\times N}$ *admits a TQS representation with rank-profile*

$$\rho_e = \operatorname{rank}\mathrm{H}_e, \qquad e\in\mathbb{E}.$$

Furthermore, such a TQS representation is minimal.

The proof of Theorem 4.4 is postponed to subsection 5.4, where we shall introduce an explicit algorithm to convert a dense matrix into a TQS representation. Proposition 4.2 and Theorem 4.4 yield the following corollary.

COROLLARY 4.5. *Let* $(\mathrm{T},\mathbb{G})$ *be a graph-partitioned matrix with* $\mathbb{G}$ *acyclic and connected. Then* $\mathrm{T}\in\mathbb{F}^{M\times N}$ *satisfies the GIRS property for* $c>0$ *if and only if* $\mathrm{T}\in\mathbb{F}^{M\times N}$ *admits a TQS representation with a rank-profile* $\{\rho_e\}_{e\in\mathbb{E}}$ *satisfying* $\rho_e\le c\,\forall\, e\in\mathbb{E}$.

4.3. Sums, products, and inverses of TQS matrices. The algebraic properties of SSS and HSS matrices under addition, multiplication, and inversion generalize to TQS matrices. The following three propositions may be established from Theorem 4.4.

PROPOSITION 4.6 (TQS addition). *Let* $\mathrm{T}_1,\mathrm{T}_2\in\mathbb{F}^{M\times N}$*, with* $M=\sum_{i\in\mathbb{V}}m_i$ *and* $N=\sum_{i\in\mathbb{V}}n_i$*, be TQS matrices of rank-profiles* $\{\rho_{1,e}\}_{e\in\mathbb{E}}$ *and* $\{\rho_{2,e}\}_{e\in\mathbb{E}}$ *associated with the acyclic connected graph* $\mathbb{G}=(\mathbb{V},\mathbb{E})$*. Then,* $\mathrm{T}_3=\mathrm{T}_1+\mathrm{T}_2$ *is a TQS matrix of rank-profile* $\{\rho_{1,e}+\rho_{2,e}\}_{e\in\mathbb{E}}$.

Proof. Since $\operatorname{rank}\mathrm{T}_3\{\bar{\mathbb{A}},\mathbb{A}\}\le\operatorname{rank}\mathrm{T}_1\{\bar{\mathbb{A}},\mathbb{A}\}+\operatorname{rank}\mathrm{T}_2\{\bar{\mathbb{A}},\mathbb{A}\}$ for any Hankel block induced by $\mathbb{A}$, the result directly follows from Theorem 4.4. ☐

PROPOSITION 4.7 (TQS product). *Let* $\mathrm{T}_1\in\mathbb{F}^{M\times N}$ *and* $\mathrm{T}_2\in\mathbb{F}^{N\times P}$*, with* $M=\sum_{i\in\mathbb{V}}m_i$*,* $N=\sum_{i\in\mathbb{V}}n_i$*, and* $P=\sum_{i\in\mathbb{V}}p_i$*, be TQS matrices of rank-profiles* $\{\rho_{1,e}\}_{e\in\mathbb{E}}$ *and* $\{\rho_{2,e}\}_{e\in\mathbb{E}}$ *associated with the acyclic connected graph* $\mathbb{G}=(\mathbb{V},\mathbb{E})$*. Then,* $\mathrm{T}_3=\mathrm{T}_1\mathrm{T}_2$ *is a TQS matrix of rank-profile* $\{\rho_{1,e}+\rho_{2,e}\}_{e\in\mathbb{E}}$.

Proof. Since

$$\begin{aligned}\operatorname{rank}\mathrm{T}_3\{\bar{\mathbb{A}},\mathbb{A}\}&=\operatorname{rank}\left(\mathrm{T}_1\{\bar{\mathbb{A}},\mathbb{A}\}\mathrm{T}_2\{\mathbb{A},\mathbb{A}\}+\mathrm{T}_1\{\bar{\mathbb{A}},\bar{\mathbb{A}}\}\mathrm{T}_2\{\bar{\mathbb{A}},\mathbb{A}\}\right)\\&\le\operatorname{rank}\mathrm{T}_1\{\bar{\mathbb{A}},\mathbb{A}\}+\operatorname{rank}\mathrm{T}_2\{\bar{\mathbb{A}},\mathbb{A}\}\end{aligned}$$

for any Hankel block induced by $\mathbb{A}$, the result directly follows from Theorem 4.4. ☐

PROPOSITION 4.8 (TQS inverse). *Let* $\mathrm{T}\in\mathbb{F}^{N\times N}$*, with* $N=\sum_{i\in\mathbb{V}}n_i$*, be a nonsingular TQS matrix of rank-profile* $\{\rho_e\}_{e\in\mathbb{E}}$ *associated with the acyclic connected graph* $\mathbb{G}=(\mathbb{V},\mathbb{E})$*. Then,* T^{-1} *is also a TQS matrix of rank-profile* $\{\rho_e\}_{e\in\mathbb{E}}$.

Proof. Consider the Hankel block $\mathrm{T}\{\mathbb{A},\mathbb{A}\}$ and note that, under the hypothesis of Proposition 4.8, $\mathrm{T}\{\mathbb{A},\mathbb{A}\}$ is a square matrix. Observe that under the hypothesis, let $\mathrm{T}\{\mathbb{A},\mathbb{A}\}=\mathrm{U}\Sigma\mathrm{V}$ denote singular value decomposition and $\mathrm{B}(\epsilon)=\mathrm{U}(\Sigma+\epsilon\mathrm{Id})\mathrm{V}$ for $\epsilon>0$. By construction the inverse of $\mathrm{B}(\epsilon)$ exists, and

$$\begin{aligned}&\begin{bmatrix}\mathrm{B}(\epsilon) & \mathrm{T}\{\mathbb{A},\bar{\mathbb{A}}\}\\ \mathrm{T}\{\bar{\mathbb{A}},\mathbb{A}\} & \mathrm{T}\{\bar{\mathbb{A}},\bar{\mathbb{A}}\}\end{bmatrix}^{-1}\\&\quad=\begin{bmatrix} * & *\\ -\left(\mathrm{T}\{\bar{\mathbb{A}},\bar{\mathbb{A}}\}-\mathrm{T}\{\bar{\mathbb{A}},\mathbb{A}\}\mathrm{B}^{-1}(\epsilon)\mathrm{T}\{\mathbb{A},\bar{\mathbb{A}}\}\right)^{-1}\mathrm{T}\{\bar{\mathbb{A}},\mathbb{A}\}\mathrm{B}^{-1}(\epsilon) & *\end{bmatrix}^{-1}.\end{aligned}$$

By taking limits for $\epsilon \to 0$, it becomes straightforward to show that $\operatorname{rank} \mathrm{T}^{-1}\{\bar{\mathbb{A}}, \mathbb{A}\} = \operatorname{rank} \mathrm{T}\{\bar{\mathbb{A}}, \mathbb{A}\}$ for any Hankel block induced by $\mathbb{A}$. The proposition then follows from Theorem 4.4 and a limiting argument on the rank of determinants and minors. ∎

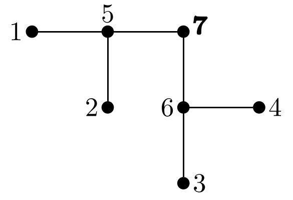
5. TQS matrix construction.

This section discusses the construction (or realization in the language of systems theory) of a TQS representation from a dense matrix provided a tree $\mathbb{G}(r)$ and an accompanying partitioning of the matrix. In subsection 5.1 we describe the construction on an illustrative example. The general algorithm is described in subsection 5.2. The construction or realization algorithm is naturally a generalization of the SSS and HSS realization algorithms. The presented algorithm will allow us to prove Theorem 4.4 in subsection 4.2. This is done in subsection 5.4.

5.1. An illustrative example.

Before describing the general algorithm, we first illustrate the construction of a TQS representation on an illustrative example. Consider the tree

$\mathbb{G}_d(7)$:



with node 7 picked as the root node. $\mathbb{G}_b(7)$ is a tree of depth 2 and comprises the levels $\mathbb{V}_0 = \{7\}$, $\mathbb{V}_1 = \{5, 6\}$, and $\mathbb{V}_2 = \{1, 2, 3, 4\}$. The corresponding spinner matrices and TQS form are shown in Figure 4. It turns out that the generating matrices of the TQS representation can be retrieved from computing low-rank factorizations of the unit Hankel blocks $\mathrm{H}_{(i,j)}$ in a particular sequence.

To start, we begin with the unit Hankel blocks associated with the edges at the deepest level of the tree. Specifically, from the low-rank factorizations of the unit Hankel blocks $\mathrm{H}_{(i,j)} = \mathrm{X}_{(i,j)} \mathrm{Y}_{(i,j)}$ with $i \in \mathbb{V}_2$ and $j = \mathcal{P}(i) \in \mathbb{V}_1$, we shall be able to obtain the B's of the spinner matrices corresponding to the nodes in $\mathbb{V}_2$ and the C's of the spinner matrices corresponding to the nodes in $\mathbb{V}_1$. For example, for $i = 1 \in \mathbb{V}_2$ and $j = \mathcal{P}(1) = 5 \in \mathbb{V}_1$, we may write

$$\mathrm{H}_{(1,5)} = \begin{array}{c} 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{bmatrix} \mathrm{X}_{(1,5)}\{2\} \\ \mathrm{X}_{(1,5)}\{3\} \\ \mathrm{X}_{(1,5)}\{4\} \\ \mathrm{X}_{(1,5)}\{5\} \\ \mathrm{X}_{(1,5)}\{6\} \\ \mathrm{X}_{(1,5)}\{7\} \end{bmatrix} \begin{array}{c} 1 \\ \left[\mathrm{Y}_{(1,5)}\{1\}\right] \end{array}$$

since we know that $\mathrm{H}_{(1,5)}$ should factor into

$$\begin{array}{c} 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{bmatrix} \mathrm{Q}_5^2 \mathrm{V}_{2,1}^5 \\ \mathrm{Q}_6^3 \mathrm{W}_{3,7}^6 \mathrm{V}_{6,5}^7 \mathrm{U}_{7,1}^5 \\ \mathrm{Q}_6^4 \mathrm{W}_{4,7}^6 \mathrm{V}_{6,5}^7 \mathrm{U}_{7,1}^5 \\ \mathrm{C}_1^5 \\ \mathrm{Q}_7^6 \mathrm{V}_{6,5}^7 \mathrm{U}_{7,1}^5 \\ \mathrm{C}_5^7 \mathrm{U}_{7,1}^5 \end{bmatrix} \begin{array}{c} 1 \\ \left[B_5^1\right] \end{array},$$

$$
\mathrm{S}^1 = \begin{array}{c} \\ \mathbf{5} \\ o \end{array}\begin{array}{c} \begin{array}{cc} \mathbf{5} & i \end{array} \\ \begin{bmatrix} 0 & \mathrm{B}_5^1 \\ \mathrm{Q}_5^1 & \mathrm{D}^1 \end{bmatrix} \end{array}
\quad
\mathrm{S}^2 = \begin{array}{c} \\ \mathbf{5} \\ o \end{array}\begin{array}{c} \begin{array}{cc} \mathbf{5} & i \end{array} \\ \begin{bmatrix} 0 & \mathrm{B}_5^2 \\ \mathrm{Q}_5^2 & \mathrm{D}^2 \end{bmatrix} \end{array}
\quad
\mathrm{S}^3 = \begin{array}{c} \\ 0 \\ o \end{array}\begin{array}{c} \begin{array}{cc} \mathbf{6} & i \end{array} \\ \begin{bmatrix} \mathrm{B}_6^3 & \mathbf{6} \\ \mathrm{Q}_6^3 & \mathrm{D}^3 \end{bmatrix} \end{array},
\quad
\mathrm{S}^4 = \begin{array}{c} \\ \mathbf{6} \\ o \end{array}\begin{array}{c} \begin{array}{cc} \mathbf{6} & i \end{array} \\ \begin{bmatrix} 0 & \mathrm{B}_6^4 \\ \mathrm{Q}_6^4 & \mathrm{D}^4 \end{bmatrix} \end{array}
$$

$$
\mathrm{S}^5 = \begin{array}{c|cccc} & 1 & 2 & \mathbf{7} & i \\ \hline 1 & 0 & \mathrm{V}_{1,2}^5 & \mathrm{W}_{1,7}^5 & \mathrm{P}_1^5 \\ 2 & \mathrm{V}_{2,1}^5 & 0 & \mathrm{W}_{2,7}^5 & \mathrm{P}_2^5 \\ \mathbf{7} & \mathrm{U}_{7,1}^5 & \mathrm{U}_{7,2}^5 & 0 & \mathrm{B}_7^5 \\ o & \mathrm{C}_1^5 & \mathrm{C}_2^5 & \mathrm{Q}_7^5 & \mathrm{D}^5 \end{array},
\quad
\mathrm{S}^6 = \begin{array}{c|cccc} & 3 & 4 & \mathbf{7} & i \\ \hline 3 & 0 & \mathrm{V}_{3,4}^6 & \mathrm{W}_{3,7}^6 & \mathrm{P}_3^6 \\ 4 & \mathrm{V}_{4,3}^6 & 0 & \mathrm{W}_{4,7}^6 & \mathrm{P}_4^6 \\ \mathbf{7} & \mathrm{U}_{7,3}^6 & \mathrm{U}_{7,4}^6 & 0 & \mathrm{B}_7^6 \\ o & \mathrm{C}_3^6 & \mathrm{C}_4^6 & \mathrm{Q}_7^6 & \mathrm{D}^6 \end{array}
$$

$$
\mathrm{S}^7 = \begin{array}{c|ccc} & 5 & 6 & i \\ \hline 5 & 0 & \mathrm{W}_{5,6}^7 & \mathrm{P}_5^7 \\ 6 & \mathrm{W}_{6,5}^7 & 0 & \mathrm{P}_6^7 \\ o & \mathrm{C}_5^7 & \mathrm{C}_6^7 & \mathrm{D}^7 \end{array}
$$

$$
\mathrm{T} = \begin{array}{ccccccc|c}
1 & 2 & 3 & 4 & 5 & 6 & 7 & \\ \hline
\mathrm{D}^1 & \mathrm{Q}_5^1\mathrm{V}_{1,2}^5\mathrm{B}_5^2 & \mathrm{Q}_5^1\mathrm{W}_{1,7}^5\mathrm{V}_{5,6}^7\mathrm{U}_{7,3}^6\mathrm{B}_6^3 & \mathrm{Q}_5^1\mathrm{W}_{1,7}^5\mathrm{V}_{5,6}^7\mathrm{U}_{7,4}^6\mathrm{B}_6^4 & \mathrm{Q}_5^1 P_1^5 & \mathrm{Q}_5^1\mathrm{U}_{1,7}^5\mathrm{V}_{5,6}^7\mathrm{B}_7^6 & \mathrm{Q}_5^1\mathrm{W}_{1,7}^5\mathrm{P}_5^7 & 1 \\
\mathrm{Q}_5^2\mathrm{V}_{2,1}^5\mathrm{B}_5^1 & \mathrm{D}^2 & \mathrm{C}_5^2\mathrm{W}_{2,7}^5\mathrm{V}_{6,5}^7\mathrm{U}_{7,3}^6\mathrm{B}_6^3 & \mathrm{Q}_5^2\mathrm{W}_{1,7}^5\mathrm{V}_{5,6}^7\mathrm{U}_{7,4}^6\mathrm{B}_6^4 & \mathrm{Q}_5^2\mathrm{P}_2^5 & \mathrm{Q}_5^2\mathrm{W}_{2,7}^5\mathrm{V}_{5,6}^7\mathrm{B}_7^6 & \mathrm{Q}_5^2\mathrm{W}_{2,7}^5\mathrm{P}_5^7 & 2 \\
\mathrm{Q}_6^3\mathrm{W}_{3,7}^6\mathrm{V}_{6,5}^7\mathrm{U}_{7,1}^5\mathrm{B}_5^1 & \mathrm{Q}_6^3\mathrm{W}_{3,7}^6\mathrm{V}_{6,5}^7\mathrm{U}_{7,2}^5\mathrm{B}_5^2 & \mathrm{D}^3 & \mathrm{Q}_6^3\mathrm{V}_{3,4}^6\mathrm{B}_6^4 & \mathrm{Q}_6^3\mathrm{W}_{3,7}^6\mathrm{V}_{6,5}^7\mathrm{B}_7^5 & \mathrm{Q}_6^3\mathrm{P}_3^6 & \mathrm{Q}_6^3\mathrm{W}_{3,7}^6\mathrm{P}_6^7 & 3 \\
\mathrm{Q}_6^4\mathrm{W}_{4,7}^6\mathrm{V}_{6,5}^7\mathrm{U}_{7,1}^5\mathrm{B}_5^1 & \mathrm{Q}_6^4\mathrm{W}_{4,7}^6\mathrm{V}_{6,5}^7\mathrm{U}_{7,2}^5\mathrm{B}_5^2 & \mathrm{Q}_6^4\mathrm{V}_{4,3}^6\mathrm{B}_6^3 & \mathrm{D}^4 & \mathrm{Q}_6^4\mathrm{W}_{4,7}^6\mathrm{V}_{6,5}^7\mathrm{B}_7^5 & \mathrm{Q}_6^4\mathrm{P}_4^6 & \mathrm{Q}_6^4\mathrm{W}_{1,7}^6\mathrm{P}_6^7 & 4 \\
\mathrm{C}_1^5\mathrm{B}_5^1 & \mathrm{C}_2^5\mathrm{B}_5^2 & \mathrm{Q}_7^5\mathrm{V}_{5,6}^7\mathrm{U}_{7,3}^6\mathrm{B}_6^3 & \mathrm{Q}_7^5\mathrm{V}_{5,6}^7\mathrm{U}_{7,4}^6\mathrm{B}_6^4 & \mathrm{D}^5 & \mathrm{Q}_7^5\mathrm{V}_{5,6}^7\mathrm{B}_7^6 & \mathrm{Q}_7^5\mathrm{P}_5^7 & 5 \\
\mathrm{Q}_7^6\mathrm{V}_{6,5}^7\mathrm{U}_{7,1}^5\mathrm{B}_5^1 & \mathrm{Q}_7^6\mathrm{U}_{6,5}^7\mathrm{V}_{7,2}^5\mathrm{B}_5^2 & \mathrm{C}_3^6\mathrm{B}_6^3 & \mathrm{C}_4^6\mathrm{B}_6^4 & \mathrm{Q}_7^6\mathrm{V}_{7,5}^7\mathrm{B}_7^5 & \mathrm{D}^6 & \mathrm{Q}_7^6\mathrm{P}_6^7 & 6 \\
\mathrm{C}_5^7\mathrm{U}_{7,1}^5\mathrm{B}_5^1 & \mathrm{C}_5^7\mathrm{U}_{7,2}^5\mathrm{B}_5^2 & \mathrm{C}_6^7\mathrm{U}_{7,3}^6\mathrm{B}_6^3 & \mathrm{C}_6^7\mathrm{U}_{7,4}^6\mathrm{B}_6^4 & \mathrm{C}_5^7\mathrm{B}_7^5 & \mathrm{C}_6^7\mathrm{B}_7^6 & \mathrm{D}^7 & 7
\end{array}
$$

FIG. 4. *The spinner matrices and TQS form associated with the tree $\mathbb{G}_b(7)$.*

and we may set $C_1^5 = X_{(1,5)}\{5\} \in \mathbb{F}^{m_5 \times \rho_{(1,5)}}$ and $B_1^5 = Y_{(1,5)}\{1\} \in \mathbb{F}^{\rho_{(1,5)} \times n_1}$ with $\rho_{(1,5)} = \operatorname{rank} H_{(1,5)}$. With similar reasoning, we may set $C_2^5 = X_{(2,5)}\{5\} \in \mathbb{F}^{m_5\times\rho_{(2,5)}}$, $B_5^2 = Y_{(2,5)}\{2\} \in \mathbb{F}^{\rho_{(2,5)}\times n_2}$, $C_3^6 = X_{(3,6)}\{6\} \in \mathbb{F}^{m_6\times\rho_{(3,6)}}$, $B_6^3 = Y_{(3,6)}\{3\} \in \mathbb{F}^{\rho_{(3,6)}\times n_3}$, $C_4^6 = X_{(4,6)}\{6\} \in \mathbb{F}^{m_6\times\rho_{(4,6)}}$, $B_6^4 = Y_{(4,6)}\{4\} \in \mathbb{F}^{\rho_{(4,6)}\times n_4}$ with $\rho_{(2,5)} = \operatorname{rank} H_{(2,5)}$, $\rho_{(3,6)} = \operatorname{rank} H_{(3,6)}$, and $\rho_{(4,6)} = \operatorname{rank} H_{(4,6)}$.

Next, moving one level up the tree by computing low-rank factorizations $H_{(i,j)} = X_{(i,j)}Y_{(i,j)}$ with $i \in \mathbb{V}_1$ and $j = \mathcal{P}(i) \in \mathbb{V}_0$, we can obtain the U's and B's of the spinner matrices corresponding to the nodes in $\mathbb{V}_1$ and the C's of the spinner matrices corresponding to the nodes in $\mathbb{V}_0$. For example, for $i = 5 \in \mathbb{V}_1$ and $j = \mathcal{P}(5) = 7 \in \mathbb{V}_0$, we may write

$$H_{(5,7)} = \begin{array}{c} 3\\4\\6\\7 \end{array}\begin{bmatrix} X_{(1,5)}\{3\} & X_{(2,5)}\{3\} & T\{3,5\} \\ X_{(1,5)}\{4\} & X_{(2,5)}\{4\} & T\{4,5\} \\ X_{(1,5)}\{6\} & X_{(2,5)}\{6\} & T\{6,5\} \\ X_{(1,5)}\{7\} & X_{(2,5)}\{7\} & T\{7,5\} \end{bmatrix} \begin{array}{c} \begin{array}{ccc} \mathcal{D}(1) & \mathcal{D}(2) & 5 \end{array} \\ \begin{bmatrix} Y_{(1,5)} & & \\ & Y_{(2,5)} & \\ & & \mathrm{Id} \end{bmatrix} \end{array}$$

with the help of previously computed factorizations. The low-rank factorization $H_{(5,7)} = X_{(5,7)}Y_{(5,7)}$ can be obtained by compressing the matrix

$$\begin{array}{c} 3\\4\\6\\7 \end{array}\begin{bmatrix} X_{(1,5)}\{3\} & X_{(2,5)}\{3\} & T\{3,5\} \\ X_{(1,5)}\{4\} & X_{(2,5)}\{4\} & T\{4,5\} \\ X_{(1,5)}\{6\} & X_{(2,5)}\{6\} & T\{6,5\} \\ X_{(1,5)}\{7\} & X_{(2,5)}\{7\} & T\{7,5\} \end{bmatrix} = \begin{array}{c} 3\\4\\6\\7 \end{array}\begin{bmatrix} X_{(5,7)}\{3\} \\ X_{(5,7)}\{4\} \\ X_{(5,7)}\{6\} \\ X_{(5,7)}\{7\} \end{bmatrix} \begin{bmatrix} Z_{(5,7)}^{\mathcal{D}(1)} & Z_{(5,7)}^{\mathcal{D}(2)} & Z_{(5,7)}^{5} \end{bmatrix},$$

which sets

$$\begin{array}{ccc} \mathcal{D}(1) & \mathcal{D}(2) & 5 \\ [Y_{(5,7)}\{\mathcal{D}(1)\} & Y_{(5,7)}\{\mathcal{D}(2)\} & Y_{(5,7)}\{5\}] \end{array} = \begin{array}{c} \begin{array}{ccc} \mathcal{D}(1) & \mathcal{D}(2) & 5 \end{array} \\ \begin{bmatrix} Z_{(5,7)}^{\mathcal{D}(1)}Y_{(1,5)} & Z_{(5,7)}^{\mathcal{D}(2)}Y_{2,5)} & Z_{(5,7)}^{5} \end{bmatrix} \end{array}.$$

Since $Y_{(1,5)} = B_5^1$, $Y_{(2,5)} = B_5^2$, and $H_{(5,7)}$ should factor into

$$\begin{array}{c} 3\\4\\6\\7 \end{array}\begin{bmatrix} Q_6^3 W_{3,7}^6 V_{6,5}^7 \\ Q_6^4 W_{4,7}^6 V_{6,5}^7 \\ Q_7^6 V_{6,5}^7 \\ C_5^7 \end{bmatrix} \begin{array}{ccc} \mathcal{D}(1) & \mathcal{D}(2) & 5 \\ [U_{7,1}^5 B_5^1 & U_{7,2}^5 B_5^2 & B_7^5] \end{array},$$

we see that $C_5^7 = X_{(5,7)}\{7\} \in \mathbb{F}^{m_7\times\rho_{(5,7)}}$, $U_{7,1}^5 = Z_{(5,7)}^{\mathcal{D}(1)} \in \mathbb{F}^{\rho_{(5,7)}\times\rho_{(1,5)}}$, $U_{7,2}^5 = Z_{(5,7)}^{\mathcal{D}(2)} \in \mathbb{F}^{\rho_{(5,7)}\times\rho_{(2,5)}}$, $B_7^5 = Z_{(5,7)}^5 \in \mathbb{F}^{\rho_{(5,7)}\times n_5}$ with $\rho_{(5,7)} = \operatorname{rank} H_{(5,7)}$. With similar reasoning, we may set $C_6^7 = X_{(6,7)}\{7\} \in \mathbb{F}^{m_7\times\rho_{(6,7)}}$, $U_{7,3}^6 = Z_{(6,7)}^{\mathcal{D}(3)} \in \mathbb{F}^{\rho_{(6,7)}\times\rho_{(3,6)}}$, $U_{7,4}^6 = Z_{(6,7)}^{\mathcal{D}(4)} \in \mathbb{F}^{\rho_{(6,7)}\times\rho_{(4,6)}}$, $B_7^6 = Z_{(6,7)}^6 \in \mathbb{F}^{\rho_{(6,7)}\times n_6}$ with $\rho_{(6,7)} = \operatorname{rank} H_{(6,7)}$.

By now, we have fully climbed up the tree and arrived at the root node. In this process, we have computed all the B's, U's, and C's of the TQS representation. This was done by peeling off the terms from the low-rank factorizations of the unit Hankel blocks. To compute the remaining P's, V's, W's, and Q's, we proceed in the same way. However, the main difference is that we start at the root and work ourselves down the tree toward the leaves. To start, through computing the low-rank factorizations $H_{(j,i)} = X_{(j,i)}Y_{(j,i)}$ with $i \in \mathbb{V}_1$ and $j = \mathcal{P}(i) \in \mathbb{V}_0$, we will be able to retrieve the P's and V's of the spinner matrices corresponding to the nodes in $\mathbb{V}_0$ and the Q's of the spinner matrices corresponding to the nodes in $\mathbb{V}_1$. For example, for $i = 5 \in \mathbb{V}_1$ and $j = \mathcal{P}(5) = 7 \in \mathbb{V}_0$, we may write

$$\mathrm{H}_{(7,5)} = \begin{array}{c}1\\2\\5\end{array}\begin{bmatrix}\mathrm{X}_{(6,7)}\{1\} & \mathrm{T}\{1,7\}\\ \mathrm{X}_{(6,7)}\{2\} & \mathrm{T}\{2,7\}\\ \mathrm{X}_{(6,7)}\{5\} & \mathrm{T}\{5,7\}\end{bmatrix}\begin{array}{cc}\mathcal{D}(6) & 7\\ \left[\begin{matrix}\mathrm{Y}_{(6,7)} & \\ & \mathrm{Id}\end{matrix}\right.\end{array}\Bigg].$$

Compressing the matrix

$$\begin{array}{c}1\\2\\5\end{array}\begin{bmatrix}\mathrm{X}_{(6,7)}\{1\} & \mathrm{T}\{1,7\}\\ \mathrm{X}_{(6,7)}\{2\} & \mathrm{T}\{2,7\}\\ \mathrm{X}_{(6,7)}\{5\} & \mathrm{T}\{5,7\}\end{bmatrix} = \begin{array}{c}1\\2\\5\end{array}\begin{bmatrix}\mathrm{X}_{(7,5)}\{1\}\\ \mathrm{X}_{(7,5)}\{2\}\\ \mathrm{X}_{(7,5)}\{5\}\end{bmatrix}\begin{bmatrix}\mathrm{Z}^{\mathcal{D}(6)}_{(7,5)} & \mathrm{Z}^{7}_{(7,5)}\end{bmatrix}$$

allows us to produce the low-rank factorization $\mathrm{H}_{(7,5)} = \mathrm{X}_{(7,5)}\mathrm{Y}_{(7,5)}$ with

$$\begin{array}{cc}\mathcal{D}(6) & 7\\ [\mathrm{Y}_{(7,5)}\{\mathcal{D}(6)\} & \mathrm{Y}_{(7,5)}\{7\}]\end{array} = \begin{array}{cc}\mathcal{D}(6) & 7\\ \big[\mathrm{Z}^{\mathcal{D}(6)}_{(7,5)}\mathrm{Y}_{(6,7)} & \mathrm{Z}^{7}_{(7,5)}\big]\end{array}.$$

Since $\mathrm{Y}_{(6,7)} = \begin{array}{ccc}3 & 4 & 6\\ [\mathrm{U}^6_{7,3}\mathrm{B}^3_6 & \mathrm{U}^6_{7,4}\mathrm{B}^4_6 & \mathrm{B}^6_7]\end{array}$ and $\mathrm{H}_{(7,5)}$ should factor into

$$\begin{array}{c}1\\2\\5\end{array}\begin{bmatrix}\mathrm{Q}^1_5\mathrm{W}^5_{1,7}\\ \mathrm{Q}^2_5\mathrm{W}^5_{2,7}\\ \mathrm{Q}^5_7\end{bmatrix}\begin{array}{cc} & \mathcal{D}(6) \qquad\qquad 7\\ \big[\mathrm{V}^7_{5,6} & \big[\mathrm{U}^6_{7,3}\mathrm{B}^3_6 \quad \mathrm{U}^6_{7,4}\mathrm{B}^4_6 \quad \mathrm{B}^6_7\big] \quad \mathrm{P}^7_5\big]\end{array},$$

we may set $\mathrm{Q}^5_7 = \mathrm{X}_{(7,5)}\{5\} \in \mathbb{F}^{m_5\times\rho_{(7,5)}}$, $\mathrm{V}^7_{5,6} = \mathrm{Z}^{\mathcal{D}(6)}_{(7,5)} \in \mathbb{F}^{\rho_{(7,5)}\times\rho_{(6,7)}}$, $\mathrm{P}^7_5 = \mathrm{Z}^7_{(7,5)} \in \mathbb{F}^{\rho_{(7,5)}\times n_7}$ with $\rho_{(7,5)} = \operatorname{rank}\mathrm{H}_{(7,5)}$. With similar reasoning, we may set $\mathrm{Q}^6_7 = \mathrm{X}_{(7,6)}\{6\} \in \mathbb{F}^{m_6\times\rho_{(7,5)}}$, $\mathrm{V}^7_{5,6} = \mathrm{Z}^{\mathcal{D}(5)}_{(7,6)} \in \mathbb{F}^{\rho_{(7,6)}\times\rho_{(5,7)}}$, $\mathrm{P}^7_6 = \mathrm{Z}^7_{(7,6)} \in \mathbb{F}^{\rho_{(7,6)}\times n_7}$ with $\rho_{(7,6)} = \operatorname{rank}\mathrm{H}_{(7,6)}$.

At last, moving one level down, we reach the bottom of the tree. By computing the low-rank factorizations $\mathrm{H}_{(j,i)} = \mathrm{X}_{(j,i)}\mathrm{Y}_{(j,i)}$ with $i \in \mathbb{V}_2$ and $j = \mathcal{P}(i) \in \mathbb{V}_1$, we will be able to compute all the remaining terms of TQS representation. Specifically, we will be able to retrieve all the P's, W's, and V's of the spinner matrices corresponding to the nodes in $\mathbb{V}_1$ and the Q's of the spinner matrices corresponding to the nodes in $\mathbb{V}_2$. For example, for $i = 1 \in \mathbb{V}_2$ and $j = \mathcal{P}(1) = 5 \in \mathbb{V}_1$, we may write

$$\mathrm{H}_{(5,1)} = \begin{array}{c}1\end{array}\begin{bmatrix}\mathrm{X}_{(7,5)}\{1\} & \mathrm{X}_{(2,5)}\{1\} & \mathrm{T}\{1,5\}\end{bmatrix}\begin{array}{ccc}\bar{\mathcal{D}}(5) & \mathcal{D}(2) & 5\\ \hline \mathrm{Y}_{(7,5)} & & \\ & \mathrm{Y}_{(2,5)} & \\ & & \mathrm{Id}\end{array}.$$

Compressing the matrix

$$1\ \begin{bmatrix}\mathrm{X}_{(7,5)}\{1\} & \mathrm{X}_{(2,5)}\{1\} & \mathrm{T}\{1,5\}\end{bmatrix} = 1\ \begin{bmatrix}\mathrm{X}_{(5,1)}\{1\}\end{bmatrix}\begin{bmatrix}\mathrm{Z}^{\bar{\mathcal{D}}(5)}_{(5,1)} & \mathrm{Z}^{\mathcal{D}(2)}_{(5,1)} & \mathrm{Z}^{5}_{(5,1)}\end{bmatrix}$$

allows us to produce the low-rank factorization $\mathrm{H}_{(7,5)} = \mathrm{X}_{(7,5)}\mathrm{Y}_{(7,5)}$ with

$$\begin{array}{ccc}\bar{\mathcal{D}}(5) & \mathcal{D}(2) & 5\\ [\mathrm{Y}_{(5,2)}\{\bar{\mathcal{D}}(5)\} & \mathrm{Y}_{(5,2)}\{\mathcal{D}(2)\} & \mathrm{Y}_{(5,2)}\{5\}]\end{array} = \begin{array}{ccc}\bar{\mathcal{D}}(5) & \mathcal{D}(2) & 5\\ \big[\mathrm{Z}^{\bar{\mathcal{D}}(5)}_{(5,1)}\mathrm{Y}_{(7,5)} & \mathrm{Z}^{\mathcal{D}(2)}_{(5,1)}\mathrm{Y}_{(2,5)} & \mathrm{Z}^{5}_{(7,5)}\big]\end{array}.$$

Since $\mathrm{Y}_{(7,5)} = \big[\mathrm{V}^7_{5,6}\mathrm{U}^6_{7,3}\mathrm{B}^3_6 \quad \mathrm{V}^7_{6,5}\mathrm{U}^6_{7,4}\mathrm{B}^4_6 \quad \mathrm{V}^7_{5,6}\mathrm{B}^6_7 \quad \mathrm{P}^7_5\big]$, $\mathrm{Y}_{(2,5)} = \mathrm{B}^2_5$, and $\mathrm{H}_{(5,1)}$ should factor into

$$1\ [\mathrm{Q}^1_5]\ \begin{array}{ccc} \bar{\mathcal{D}}(5) & \mathcal{D}(2) & 5\\ \big[\mathrm{W}^5_{1,7}\big[\mathrm{V}^7_{5,6}\mathrm{U}^6_{7,3}\mathrm{B}^3_6 \quad \mathrm{V}^7_{6,5}\mathrm{U}^6_{7,4}\mathrm{B}^4_6 \quad \mathrm{V}^7_{5,6}\mathrm{B}^6_7 \quad \mathrm{P}^7_5\big] & \mathrm{V}^5_{1,2}\mathrm{B}^2_5 & \mathrm{P}^5_1\big]\end{array},$$

we see that $\mathrm{Q}_5^1 = \mathrm{X}_{(5,1)}\{1\} \in \mathbb{F}^{m_1 \times \rho_{(5,1)}}$, $\mathrm{W}_{1,7}^5 = \mathrm{Z}_{(5,1)}^{\bar{\mathcal{D}}(5)} \in \mathbb{F}^{\rho_{(5,1)} \times \rho_{(7,5)}}$, $\mathrm{V}_{1,2}^5 = \mathrm{Z}_{(5,1)}^{\mathcal{D}(2)} \in \mathbb{F}^{\rho_{(5,1)} \times \rho_{(2,5)}}$, $\mathrm{P}_1^5 = \mathrm{Z}_{(7,5)}^5 \in \mathbb{F}^{\rho_{(5,1)} \times n_5}$ with $\rho_{(5,1)} = \operatorname{rank} \mathrm{H}_{(5,1)}$. The remaining terms of the TQS representation are obtained in the same way.

5.2. The general construction algorithm. The approach taken for the example of the previous section generalizes for a generic TQS representation. This process is described in Algorithm 1. The process of converting a dense matrix into TQS form consists of two phases: an upsweep and a downsweep phase. In the upsweep phase, one starts at the leaves of the tree and works up toward the root. In this process, all the B's, C's, and U's are computed. In the downsweep phase that follows, one starts at the root of the tree and then works down toward the leaves. In this second leg, the P's, Q's, W's, and V's are computed. The additional nomenclature introduced in subsection 3.3 reveals that all of the generators are computed exactly once, thus alluding to any inconsistencies that may occur.

ALGORITHM 1 (TQS construction algorithm). *Let $\mathbb{G}(r)$ be a tree with root node $r \in \mathbb{V}$ and let $\mathrm{T} \in \mathbb{F}^{M \times N}$ be the associated graph-partitioned matrix with $M = \sum_{i \in \mathbb{V}} m_i$ and $N = \sum_{i \in \mathbb{V}} n_i$. A set of generators for the TQS representation of T is obtained by following the steps outlined below.*

1. ***Diagonal stage.*** *Set* $\mathrm{D}^i = \mathrm{T}\{i,i\}$ for $i \in \mathbb{V}$.
2. ***Upsweep stage.*** *For $l = L, L-1, \ldots, 1$ do the following:*
   (a) *For every $i \in \mathbb{V}_l$ with parent node $j = \mathcal{P}(i) \in \mathbb{V}_{l-1}$ and children $\mathcal{C}(i) = \{w_1, w_2, \ldots, w_\alpha\} \subset \mathbb{V}$, write $\mathrm{H}_{(i,j)} = \mathrm{F}_{(i,j)}\mathrm{G}_{(i,j)}$, where*

$$\mathrm{F}_{(i,j)} := \begin{array}{c} \\ \bar{\mathcal{D}}(i) \end{array} \begin{bmatrix} \mathrm{X}_{(w_1,i)}\{\bar{\mathcal{D}}(i)\} & \cdots & \mathrm{X}_{(w_\alpha,i)}\{\bar{\mathcal{D}}(i)\} & \mathrm{T}\{\bar{\mathcal{D}}(i), i\} \end{bmatrix},$$

$$\mathrm{G}_{(i,j)} := \begin{array}{c} \begin{array}{cccc} \mathcal{D}(w_1) & \cdots & \mathcal{D}(w_\beta) & i \end{array} \\ \begin{bmatrix} \mathrm{Y}_{(w_1,i)} & & & \\ & \ddots & & \\ & & \mathrm{Y}_{(w_\beta,i)} & \\ & & & \mathrm{Id} \end{bmatrix} \end{array}.$$

   (b) *Let $\rho_{(i,j)} = \operatorname{rank} \mathrm{F}_{(i,j)} = \operatorname{rank} \mathrm{H}_{(i,j)}$ and compute the low-rank compression $\mathrm{F}_{(i,j)} = \mathrm{X}_{(i,j)}\mathrm{Z}_{(i,j)}$.*
   (c) *Set*

$$\mathrm{B}_j^i := \mathrm{Z}_{(i,j)}^i, \quad \mathrm{C}_i^j := \mathrm{X}_{(i,j)}\{j\}, \quad \mathrm{U}_{j,w_t}^i := \mathrm{Z}_{(i,j)}^{\mathcal{D}(w_t)}$$

   for $t = 1, 2, \ldots, \alpha$.
   (d) *Define $\mathrm{Y}_{(i,j)} = \mathrm{Z}_{(i,j)}\mathrm{G}_{(i,j)}$ so that $\mathrm{X}_{(i,j)}\mathrm{Y}_{(i,j)}$ is a low-rank factorization for* $\mathrm{H}_{(i,j)}$.
3. ***Downsweep stage.*** *For $l = 1, 2, \ldots, L$ do the following:*
   (a) *For every $i \in \mathbb{V}_l$ with parent node $j = \mathcal{P}(i) \in \mathbb{V}_{l-1}$, grandparent node[6] $k = \mathcal{P}(i;2) \in \mathbb{V}_{l-2}$, and siblings $\mathcal{S}(i) = \{v_1, v_2, \ldots, v_\beta\} \subset \mathbb{V}$, write $\mathrm{H}_{(j,i)} = \mathrm{F}_{(j,i)}\mathrm{G}_{(j,i)}$, where*

[6]For $l = 1$ there will be no grandparent node, in which case the corresponding terms associated with it can be ignored.

$$\mathrm{F}_{(j,i)} := \begin{array}{c} \mathcal{D}(i) \end{array} \begin{bmatrix} \mathrm{X}_{(k,j)}\{\mathcal{D}(i)\} & \mathrm{X}_{(v_1,j)}\{\mathcal{D}(i)\} & \cdots & \mathrm{X}_{(v_\beta,j)}\{\mathcal{D}(i)\} & \mathrm{T}\{\mathcal{D}(i),j\} \end{bmatrix}$$

$$\mathrm{G}_{(j,i)} := \begin{array}{ccccc} \bar{\mathcal{D}}(j) & \mathcal{D}(v_1) & \cdots & \mathcal{D}(v_\beta) & j \\ \left[\mathrm{Y}_{(k,j)}\right. & & & & \\ & \mathrm{Y}_{(v_1,j)} & & & \\ & & \ddots & & \\ & & & \mathrm{Y}_{(v_\beta,j)} & \\ & & & & \left.\mathrm{Id}\right] \end{array}$$

(b) *Let* $\rho_{(j,i)} = \operatorname{rank} \mathrm{F}_{(j,i)} = \operatorname{rank} \mathrm{H}_{(j,i)}$ *and compute the low-rank compression* $\mathrm{F}_{(j,i)} = \mathrm{X}_{(j,i)}\mathrm{Z}_{(j,i)}$.

(c) *Set*

$$\mathrm{P}_i^j = \mathrm{Z}_{(j,i)}^j, \quad \mathrm{W}_{(j,k)}^i = \mathrm{Z}_{(j,i)}^{\bar{\mathcal{D}}(j)}, \quad \mathrm{Q}_j^i = \mathrm{X}_{(j,i)}\{i\}, \quad \mathrm{V}_{j,v_t}^i = \mathrm{Z}_{(j,i)}^{\mathcal{D}(v_t)}$$

for $t = 1, 2, \ldots, \beta$.

(d) *Set* $\mathrm{Y}_{(j,i)} = \mathrm{Z}_{(j,i)}\mathrm{G}_{(j,i)}$ *so that* $\mathrm{X}_{(j,i)}\mathrm{Y}_{(j,i)}$ *is a low-rank factorization for* $\mathrm{H}_{(j,i)}$.

We remark that Algorithm 1 is not the only approach for constructing a TQS matrix. There exists some flexibility in algorithmic design choices that could be optimized for parallelism and memory consumption. A more detailed analysis goes outside the scope of this paper. For now, we stay contented that Algorithm 1 presents a valid construction/realization algorithm.

5.3. Numerical tests. To further validate the construction algorithm for correctness, we have implemented Algorithm 1 in the Julia language.[7] Table 1 showcases the numerical results obtained with the implemented algorithm for two experiments. In the first experiment, Algorithm 1 is applied to reconstruct minimal TQS representations of randomly generated TQS matrices for the tree graphs $\mathbb{G}_a(4)$, $\mathbb{G}_b(4)$, $\mathbb{G}_c(5)$, and $\mathbb{G}_d(7)$. The TQS matrices are first converted into dense matrices, after which Algorithm 1 is applied to reconstruct the TQS matrix. In the second experiment, Algorithm 1 is used to construct TQS representations for the inverse of the matrices T_k, $k = 1, 2, 3, \ldots$, from subsection 3.5. The parameters used to generate T_k are again chosen randomly. Using the adjacency graph as the corresponding tree, it follows from Proposition 4.8 that T_k^{-1} admit a scalar TQS representation, i.e., $\rho_e = 1$ $\forall e \in \mathbb{E}$. Our numerical experiment also confirmed this property. Tables 1(a) and 1(b) show the relative 2-norm error of the constructed TQS matrices (with respect to the original dense matrix) for experiments 1 and 2, respectively. The results suggest that Algorithm 1 can generate TQS approximations up to machine precision accuracy.

5.4. Proof of Theorem 4.4. We are now ready to prove Theorem 4.4.

Proof of Theorem 4.4. Algorithm 1 presents a constructive proof for the existence of a TQS representation with rank-profile $\{\rho_e := \operatorname{rank} \mathrm{H}_e\}_{e\in\mathbb{E}}$. The only thing left is

[7]This code is made available at https://github.com/nithingovindarajan/TQSmatrices.

Table 1
Numerical results obtained with Algorithm 1. The mean and standard deviation of the relative 2-norm error of the constructed TQS matrix (w.r.t. the original dense matrix) is computed. The statistics are computed from 10 random trials.

(a) Computed error statistics for the TQS reconstruction of randomly generated TQS matrices for the tree graphs $\mathbb{G}_a(4)$, $\mathbb{G}_b(4)$, $\mathbb{G}_c(5)$, and $\mathbb{G}_d(7)$.

	rel. error	
	mean	std
$\mathbb{G}_a(4)$	2.2431e-15	2.0166e-15
$\mathbb{G}_b(4)$	7.6436e-16	1.4924e-16
$\mathbb{G}_c(5)$	6.5400e-16	2.4239e-16
$\mathbb{G}_d(7)$	4.0584e-15	2.8004e-15

(b) Computed error statistics for the TQS construction of the inverse of the matrices T_k, $k = 1, 2, 3, \ldots$ from subsection 3.5.

k	*rel. error*	
	mean	std
1	0.0	0.0
2	1.41199e-16	4.87742e-17
3	2.78289e-16	8.81678e-17
4	3.6505e-16	1.66578e-16
5	4.71926e-16	2.48319e-16
6	6.95086e-16	2.30473e-16
7	5.68185e-16	1.56358e-16
8	1.03728e-15	3.58987e-16
9	1.1168e-15	4.68181e-16

to show that the TQS representation produced by Algorithm 1 is minimal. This can be verified by setting up a contradiction. Suppose there exists a TQS representation with rank-profile $\{\rho'_e\}_{e\in\mathbb{E}}$ and $\rho'_e < \rho_e$ for some edge $e \in \mathbb{E}$; then the unit Hankel H_e admits a low-rank factorization of rank $\rho'_e < \rho_e$, which is not possible. ∎

6. TQS linear systems.

In this section, we examine linear systems $\mathrm{T}\boldsymbol{x} = \boldsymbol{b}$, where $\mathrm{T} \in \mathbb{F}^{M\times N}$ is a TQS matrix on $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ of dimensions $M = \sum_{i\in\mathbb{V}} m_i$ by $N = \sum_{i\in\mathbb{V}} n_i$. By conformally partitioning $\boldsymbol{b} \in \mathbb{F}^M$ and $\boldsymbol{x} \in \mathbb{F}^N$ into subvectors $\boldsymbol{b}_i \in \mathbb{F}^{m_i}$ and $\boldsymbol{x}_i \in \mathbb{F}^{n_i}$, respectively, we first describe in subsection 6.1 how the matrix-vector product $\boldsymbol{b} = \mathrm{T}\boldsymbol{x}$ is evaluated efficiently. Then, in subsection 6.2, we proceed and use the relations derived for the matrix-vector product to formulate an efficient method to solve $\mathrm{T}\boldsymbol{x} = \boldsymbol{b}$ for $\boldsymbol{x}$ given $\boldsymbol{b}$.

6.1. Evaluation of matrix-vector product.

The block entries of a TQS matrix originate from evolving a dynamical system over a graph. In the special case of an SSS matrix, the lower and upper triangular parts of the matrix are the result of running a causal and anticausal LTV dynamical system. This is exploited in the matrix-vector product. For the general case, the dynamics evolve over a tree, and it becomes much harder to provide a simple characterization. Nonetheless, formulating the "state-space" equations shall produce a fast matrix-vector product algorithm for the general case as well. Recall that each edge $(i, j) \in \mathbb{E}$ of the acyclic connected graph $\mathbb{G}$ a state vector $\boldsymbol{h}_{(i,j)} \in \mathbb{F}^{\rho_{(i,j)}}$. The transition maps (3.4) yield the state equations

$$\boldsymbol{h}_{(i,j)} = \sum_{w\in\mathcal{N}(i)\backslash\{j\}} \mathrm{Trans}^i_{j,w}\boldsymbol{h}_{(w,i)} + \mathrm{Inp}^i_j\boldsymbol{x}_i, \qquad (i,j) \in \mathbb{E}, \tag{6.1}$$

along with the output equations

$$\boldsymbol{b}_j = \sum_{i\in\mathcal{N}(j)} \mathrm{Out}^j_i\boldsymbol{h}_{(i,j)} + \mathrm{D}^j\boldsymbol{x}_i, \qquad j \in \mathbb{V}. \tag{6.2}$$

By picking a root node $r \in \mathbb{V}$, the state and output equations for the corresponding tree $\mathbb{G}(r)$ may be further refined to

$$\boldsymbol{h}_{(i,j)} = \begin{cases} \sum_{w\in\mathcal{C}(i)} \mathrm{U}^i_{j,w}\boldsymbol{h}_{(w,i)} + \mathrm{B}^i_j\boldsymbol{x}_i, & j=\mathcal{P}(i), \\ \mathrm{W}^i_{j,\mathcal{P}(i)}\boldsymbol{h}_{(\mathcal{P}(i),i)} + \sum_{w\in\mathcal{S}(j)} \mathrm{V}^i_{j,w}\boldsymbol{h}_{(w,i)} + \mathrm{P}^i_j\boldsymbol{x}_i, & i=\mathcal{P}(j), \end{cases} \tag{6.3}$$

and

$$\boldsymbol{b}_j = \mathrm{Q}^j_{\mathcal{P}(j)}\boldsymbol{h}_{(\mathcal{P}(j),j)} + \sum_{i\in\mathcal{C}(j)} \mathrm{B}^j_i\boldsymbol{h}_{(i,j)} + \mathrm{D}^j\boldsymbol{x}_i. \tag{6.4}$$

A careful examination of (6.3) reveals a natural causal ordering on the state variables. Specifically, for a leaf $i \in \mathscr{L}(\mathbb{G})$, the state equations are simply $\boldsymbol{h}_{(i,j)} = \mathrm{B}^i_j\boldsymbol{x}_i$. One may thus start with computing the state vectors at the leaves and then work up toward the interiors of the graph. Once the root node is reached, the reverse process can be initiated by flowing outward toward the leaves. Algorithm 2 exactly describes this process. The TQS matrix-vector product involves

$$\mathcal{O}\left(\sum_{i\in\mathbb{V}}\left(m_i + \sum_{j\in\mathcal{N}(i)} r_{(i,j)}\right)\left(n_i + \sum_{j\in\mathcal{N}(i)} r_{(j,i)}\right)\right)$$

floating point operations (flops). In particular, the complexity becomes a linear time w.r.t. the matrix dimensions if, for instance, the properties in (3.7) are applicable.

ALGORITHM 2 (TQS matrix-vector product). *Given a TQS matrix* $\mathrm{T} \in \mathbb{F}^{M\times N}$ *on* $\mathbb{G}(r)$ *of dimensions* $M = \sum_{i\in\mathbb{V}} m_i$ *by* $N = \sum_{i\in\mathbb{V}} n_i$*, the matrix-vector product* $\boldsymbol{b}_i = \sum_{j\in\mathbb{V}} \mathrm{T}\{i,j\}\boldsymbol{x}_j$ *for* $i \in \mathbb{V}$ *is obtained by following the steps outlined below.*

1. ***Diagonal stage.*** *Initialize* $\boldsymbol{b}_i = \mathrm{D}^i\boldsymbol{x}_i$ *for* $i \in \mathbb{V}$.
2. ***Upsweep stage.*** *For* $l = L, L-1, \ldots, 1$ *do the following:*
   (a) *For every* $i \in \mathbb{V}_l$ *with parent node* $j = \mathcal{P}(i) \in \mathbb{V}_{l-1}$ *and children* $\mathcal{C}(i) = \{w_1, w_2, \ldots, w_\alpha\} \subset \mathbb{V}$*, evaluate*

$$\boldsymbol{h}_{(i,j)} = \sum_{p=1}^{\alpha} \mathrm{U}^i_{j,w_p}\boldsymbol{h}_{(w_p,i)} + \mathrm{B}^i_j\boldsymbol{x}_i.$$

   (b) *Update*

$$\boldsymbol{b}_j \leftarrow \boldsymbol{b}_j + \mathrm{C}^j_i\boldsymbol{h}_{(i,j)}.$$

3. ***Downsweep stage.*** *For* $l = 1, 2, \ldots, L$ *do the following:*
   (a) *For every* $i \in \mathbb{V}_l$ *with parent node* $j = \mathcal{P}(i) \in \mathbb{V}_{l-1}$*, grandparent node* $k = \mathcal{P}(i;2) \in \mathbb{V}_{l-2}$*, and siblings* $\mathcal{S}(i) = \{v_1, v_2, \ldots, v_\beta\} \subset \mathbb{V}$*, evaluate*

$$\boldsymbol{h}_{(j,i)} = \mathrm{W}^j_{i,k}\boldsymbol{h}_{(k,j)} + \sum_{p=1}^{\beta} \mathrm{V}^j_{i,w_p}\boldsymbol{h}_{(w_p,i)} + \mathrm{B}^j_i\boldsymbol{x}_j.$$

   (b) *Update*

$$\boldsymbol{b}_i \leftarrow \boldsymbol{b}_i + \mathrm{Q}^i_j\boldsymbol{h}_{(j,i)}.$$

6.2. Solving linear systems. The (possibly dense) linear system $\mathrm{T}\boldsymbol{x} = \boldsymbol{b}$ is efficiently solved for $\boldsymbol{x}$ given $\boldsymbol{b}$ by using the same reasoning introduced in [6]. By treating $\{\boldsymbol{h}_e\}_{e\in\mathbb{E}}$ and $\{\boldsymbol{x}_i\}_{i\in\mathbb{V}}$ as unknowns, and $\{\boldsymbol{b}_i\}_{i\in\mathbb{V}}$ as knowns, (6.1) and (6.2) collectively yield the sparse linear system

$$\boldsymbol{h}_{(i,j)} - \sum_{w\in\mathcal{N}(i)\backslash\{j\}} \mathrm{Trans}^{i}_{j,w}\boldsymbol{h}_{(w,i)} - \mathrm{Inp}^{i}_{j}\boldsymbol{x}_i = 0, \qquad (i,j)\in\mathbb{E}, \tag{6.5a}$$

$$\sum_{i\in\mathcal{N}(j)} \mathrm{Out}^{j}_{i}\boldsymbol{h}_{(i,j)} + \mathrm{D}^{j}\boldsymbol{x}_i = \boldsymbol{b}_j, \qquad j\in\mathbb{V}. \tag{6.5b}$$

Particularly, if we let $\mathcal{N}(j) = \{i_1, i_2, \ldots, i_p\}$ and define

$$\boldsymbol{\theta}_j := \begin{bmatrix} \boldsymbol{x}_j \\ \boldsymbol{h}_{(i_1,j)} \\ \boldsymbol{h}_{(i_2,j)} \\ \vdots \\ \boldsymbol{h}_{(i_p,j)} \end{bmatrix}, \qquad \boldsymbol{\beta}_j = \begin{bmatrix} \boldsymbol{b}_j \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

the adjacency graph of the matrix expression $\Xi\boldsymbol{\theta} = \boldsymbol{\beta}$ that describes (6.5) coincides with $\mathbb{G}$. That is, $\Xi\{i,j\} \neq 0$ if and only if $(i,j)\in\mathbb{E}$. Acyclic graphs are chordal graphs and have a perfect elimination order without any fill-in (see [16]). Thus, (6.5) may be efficiently solved with any standard (block-)sparse solver. Solving (6.5) involves roughly

$$\mathcal{O}\left(\sum_{i\in\mathbb{V}}\left(n_i + \sum_{j\in\mathcal{N}(i)} r_{(i,j)}\right)^3\right)$$

flops. In particular, the complexity becomes linear w.r.t. the matrix dimensions if, for instance, the properties in (3.7) are applicable.

7. Conclusions and future work.

We introduced a new class of representations for rank-structured matrices called tree quasi-separable (TQS) matrices. It was shown that TQS matrices unify and generalize SSS and HSS matrices. Furthermore, by deriving an explicit construction algorithm, we characterized the properties of a minimal TQS representation for a given tree graph-partitioned matrix. Subsequently, we showed that TQS inherits many of the well-known properties of SSS and HSS matrices concerning matrix-vector multiplication, matrix-matrix multiplication, matrix-matrix addition, and inversion.

Future work will be geared toward the efficient implementation and generalization of many algorithms associated with SSS and HSS matrices. Specifically, in a future paper, we shall derive expressions for the LU factorization and (pseudo-)inverse of TQS matrices. The potential applications of TQS (and the greater flexibility that is offered by them) will also be explored. Finally, we note that the results associated with TQS may form an essential building block for constructing representations associated with more general GIRS matrices (see [11]).

Reproducibility of computational results. This paper has been awarded the "SIAM Reproducibility Badge: Code and data available" as a recognition that the authors have followed reproducibility principles valued by SIMAX and the scientific computing community. Code and data that allow readers to reproduce the results in this paper are available at https://github.com/nithingovindarajan/TQSmatrices and in the supplementary materials (Supp_Materials.zip [local/web 16.4KB]).

REFERENCES

[1] S. Ambikasaran and E. Darve, *An $\mathcal{O}(N \log N)$ fast direct solver for partial hierarchically semi-separable matrices*, J. Sci. Comput., 57 (2013), pp. 477–501, https://doi.org/10.1007/s10915-013-9714-z.

[2] A. Aminfar, S. Ambikasaran, and E. Darve, *A fast block low-rank dense solver with applications to finite-element matrices*, J. Comput. Phys., 304 (2016), pp. 170–188, https://doi.org/10.1016/j.jcp.2015.10.012.

[3] J. L. Aurentz, T. Mach, R. Vandebril, and D. S. Watkins, *Fast and backward stable computation of roots of polynomials*, SIAM J. Matrix Anal. Appl., 36 (2015), pp. 942–973, https://doi.org/10.1137/140983434.

[4] T. Bella, Y. Eidelman, I. Gohberg, and V. Olshevsky, *Computations with quasiseparable polynomials and matrices*, Theoret. Comput. Sci., 409 (2008), pp. 158–179, https://doi.org/10.1016/j.tcs.2008.09.008.

[5] S. Börm, L. Grasedyck, and W. Hackbusch, *Introduction to hierarchical matrices with applications*, Eng. Anal. Bound. Elem., 27 (2003), pp. 405–422, https://doi.org/10.1016/S0955-7997(02)00152-2.

[6] S. Chandrasekaran, P. Dewilde, M. Gu, W. Lyons, and T. Pals, *A fast solver for HSS representations via sparse matrices*, SIAM J. Matrix Anal. Appl., 29 (2006), pp. 67–81, https://doi.org/10.1137/050639028.

[7] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, X. Sun, A.-J. van der Veen, and D. White, *Some fast algorithms for sequentially semiseparable representations*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 341–364, https://doi.org/10.1137/S0895479802405884.

[8] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, and A.-J. van der Veen, *Fast stable solver for sequentially semi-separable linear systems of equations*, in International Conference on High-Performance Computing, Springer, Cham, 2002, pp. 545–554.

[9] S. Chandrasekaran, P. Dewilde, M. Gu, and N. Somasunderam, *On the numerical rank of the off-diagonal blocks of Schur complements of discretized elliptic PDEs*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2261–2290, https://doi.org/10.1137/090775932.

[10] S. Chandrasekaran, E. N. Epperly, and N. Govindarajan, *Graph-Induced Rank Structures and Their Representations*, preprint, arXiv:1911.05858, 2019.

[11] S. Chandrasekaran and N. Govindarajan, *Graph Induced Rank Structured Matrices and Their Representations*, presented at Computational and Applied Mathematics, Selva di Fasano, Italy, 2023.

[12] S. Chandrasekaran, N. Govindarajan, and A. Rajagopal, *Fast algorithms for displacement and low-rank structured matrices*, in Proceedings of the International Symposium on Symbolic and Algebraic Computation, ACM, New York, 2018, pp. 17–22.

[13] S. Chandrasekaran, M. Gu, and T. Pals, *A fast ULV decomposition solver for hierarchically semiseparable representations*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 603–622, https://doi.org/10.1137/S0895479803436652.

[14] S. Chandrasekaran, M. Gu, X. Sun, J. Xia, and J. Zhu, *A superfast algorithm for Toeplitz systems of linear equations*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 1247–1266, https://doi.org/10.1137/040617200.

[15] P. Dewilde and A.-J. Van der Veen, *Time-Varying Systems and Computations*, Springer, Dordrecht, The Netherlands, 1998.

[16] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, Oxford, UK, 2017.

[17] Y. Eidelman and I. Gohberg, *On a new class of structured matrices*, Integral Equations Operator Theory, 34 (1999), pp. 293–324, https://link.springer.com/article/10.1007/BF01300581.

[18] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 135 (1997), pp. 280–292, https://doi.org/10.1006/jcph.1997.5706.

[19] W. Hackbusch, *Hierarchical Matrices: Algorithms and Analysis*, Springer Ser. Comput. Math. 49, Springer, New York, 2015.

[20] W. Hackbusch and S. Börm, *Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices*, Computing, 69 (2002), pp. 1–35, https://link.springer.com/article/10.1007/s00607-002-1450-4.

[21] V. Rokhlin, *Rapid solution of integral equations of classical potential theory*, J. Comput. Phys., 60 (1985), pp. 187–207, https://doi.org/10.1016/0021-9991(85)90002-6.

[22] R. Vandebril, M. Van Barel, and N. Mastronardi, *Matrix Computations and Semiseparable Matrices: Linear Systems*, Vol. 1, JHU Press, Baltimore, 2008.

[23] M. Verhaegen, C. Yu, and B. Sinquin, *Data-Driven Identification of Networks of Dynamic Systems*, Cambridge University Press, Cambridge, UK, 2022.

[24] S. Wang, X. S. Li, J. Xia, Y. Situ, and M. V. De Hoop, *Efficient scalable algorithms for solving dense linear systems with hierarchically semiseparable structures*, SIAM J. Sci. Comput., 35 (2013), pp. C519–C544, https://doi.org/10.1137/110848062.

[25] J. Xia, *On the complexity of some hierarchical structured matrix algorithms*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 388–410, https://doi.org/10.1137/110827788.

[26] J. Xia, *Robust and efficient multifrontal solver for large discretized PDEs*, in High-Performance Scientific Computing, Springer, London, (2012), pp. 199–217, https://link.springer.com/chapter/10.1007/978-1-4471-2437-5_10.

[27] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, *Fast algorithms for hierarchically semiseparable matrices*, Numer. Linear Algebra Appl., 17 (2010), pp. 953–976, https://doi.org/10.1002/nla.691.

[28] J. Xia, Y. Xi, and M. Gu, *A superfast structured solver for Toeplitz linear systems via randomized sampling*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 837–858, https://doi.org/10.1137/110831982.