Approximately Optimal Core Shapes for Tensor Decompositions

Mehrdad Ghadiri [*1] **Matthew Fahrbach** [*2] **Gang Fu** [2] **Vahab Mirrokni** [2]

Abstract

This work studies the combinatorial optimization problem of finding an optimal *core tensor shape*, also called multilinear rank, for a size-constrained Tucker decomposition. We give an algorithm with provable approximation guarantees for its reconstruction error via connections to higher-order singular values. Specifically, we introduce a novel *Tucker packing problem*, which we prove is NP-hard, and give a polynomial-time approximation scheme based on a reduction to the 2-dimensional knapsack problem with a matroid constraint. We also generalize our techniques to *tree tensor network decompositions*. We implement our algorithm using an integer programming solver, and show that its solution quality is competitive with (and sometimes better than) the greedy algorithm that uses the true Tucker decomposition loss at each step, while also running up to 1000x faster.

1. Introduction

Low-rank tensor decomposition is a powerful tool in the modern machine learning toolbox. Like low-rank matrix factorization, it has countless applications in scientific computing, data mining, and signal processing (Kolda & Bader, 2009; Sidiropoulos et al., 2017), e.g., anomaly detection in data streams (Jang & Kang, 2021) and compressing convolutional neural networks on mobile devices for faster inference while reducing power consumption (Kim et al., 2016).

The most widely used tensor decompositions are the canonical polyadic (CP) decomposition, Tucker decomposition, and tensor-train decomposition (Oseledets, 2011)—the last two being instances of *tree tensor networks* (Krämer, 2020). CP decomposition factors a tensor into the sum of r rank-one tensors. Tucker decomposition, however, specifies the rank R_n in each dimension n and relies on a core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$ for reconstructing the decomposition. The notion of *multilinear rank* $\mathbf{r} = (R_1, \ldots, R_N)$ puts practitioners in a challenging spot because the set of feasible core shapes can be exponentially large. Furthermore, searching in this state space can be prohibitively expensive because evaluating the true quality of a core shape requires computing a Tucker decomposition, which for large tensors can take hours and consume hundreds of GB of RAM. For example, in the MATLAB Tensor Toolbox (Bader & Kolda, 2022), we need to specify the core shape parameter `ranks` in advance before computing a size-constrained Tucker decomposition.

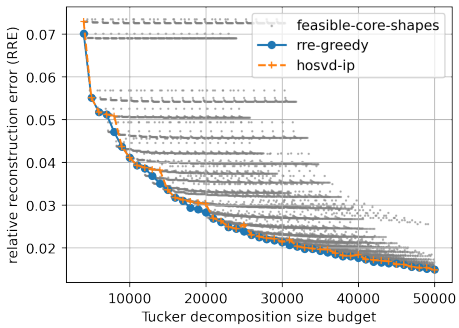

Figure 1. Pareto frontier of core shapes $\mathbf{r} \in [20]^3$ for hyperspectral tensor $\mathcal{X} \in \mathbb{R}^{1024 \times 1344 \times 33}$. Plots the RRE, i.e., $L(\mathcal{X}, \mathbf{r})/\|\mathcal{X}\|_{\mathrm{F}}^2$, as a function of compression rate. RRE-greedy builds core shapes by computing Tucker decompositions at each step. HOSVD-IP is Algorithm 2 with integer programming, which builds core shapes via a surrogate packing problem on higher-order singular values.

In practice, the most popular Tucker decomposition algorithms are the $\mathbf{r}$-truncated higher-order singular value decomposition (HOSVD) in De Lathauwer et al. (2000a), sequentially truncated ST-HOSVD in Vannieuwenhoven et al. (2012), and higher-order orthogonal iteration (HOOI), which is a structured alternating least squares algorithm.

We explore the simple but fundamental discrete optimization problem for low-rank tensor decompositions:

> *If a Tucker decomposition of $\mathcal{X}$ can use at most c parameters, which core tensor shape minimizes the reconstruction error?*

This is a multilinear generalization of the best rank-r matrix approximation problem. While there are many parallels to low-rank matrix factorization, tensor rank-related problems can be thoroughly different and more challenging than their

[*]Equal contribution [1]Georgia Tech [2]Google Research. Correspondence to: Mehrdad Ghadiri <mehrdad29@gmail.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

matrix counterparts. For example, computing the CP rank of a real-valued tensor is NP-hard (Hillar & Lim, 2013).

1.1. Our contributions and techniques

We summarize the main contributions of this work below:

1. We formalize the core tensor shape problem for size-constrained Tucker decompositions and introduce the *Tucker packing problem*, which we prove is NP-hard. The approximation algorithms we develop build on a relationship between the optimal reconstruction error of a rank-$\mathbf{r}$ Tucker decomposition and a multi-dimensional tail sum of its higher-order singular values (De Lathauwer et al., 2000a; Hackbusch, 2019).
2. We design a polynomial-time approximation scheme (PTAS) for the surrogate Tucker packing problem (Theorem 4.6) by showing that it suffices to consider a small number of budget splits between the cost of the core tensor and the cost of the factor matrices. Each budget split subproblem reduces to a *2-dimensional knapsack problem with a partition matroid constraint* after minor transformations. We solve these knapsack problems using the PTAS of Grandoni et al. (2014), or in practice with integer linear programming.
3. We extend our approach to *tree tensor networks*, which generalize the Tucker decomposition, tensor-train decomposition, and hierarchical Tucker decomposition. In doing so, we synthesize several works on tree tensors from the mathematics and physics communities, and give a succinct introduction for computer scientists.
4. Finally, we demonstrate the effectiveness of our Tucker packing-based core shape solvers on four real-world tensors. Our HOSVD-IP algorithm is competitive with (and sometimes outperforms) the greedy algorithm that uses the true RRE, while running up to 1000x faster.

1.2. Related works

Core shape constraints. De Lathauwer et al. (2000b) introduced the problem of computing the best rank-$\mathbf{r}$ tensor approximation for a *prespecified* core shape $\mathbf{r}$, and demonstrated the benefit of initializing the decomposition with a truncated HOSVD and then running iterative methods such as HOOI. Eldén & Savas (2009); Ishteva et al. (2009; 2011; 2013); Eldén & Dehghan (2022) consider this problem for rank-(r_1, r_2, r_3) decompositions and develop a suite of advanced algorithms: a Newton method on Grassmannian manifolds, a trust-region method on Riemannian manifolds, Jacobi rotations for symmetric tensors, and a Krylov-type iterative method. All these works, however, are concerned with optimizing the tensor decomposition for a fixed core shape—not with optimizing the core tensor shape itself.

Ehrlacher et al. (2021) and Xiao & Yang (2021) recently explored *rank-adaptive* methods for HOOI that find minimal core shapes such that the Tucker decomposition achieves a target reconstruct error. They also leverage properties of the HOSVD, but they *do not impose a hard constraint on the size of the returned Tucker decomposition*. Hashemizadeh et al. (2020) generalized the RRE-greedy algorithm in Figure 1 to tensor networks for both rank and size constraints.

Low-rank tensor decomposition. Song et al. (2019) gave polynomial-time $(1+\varepsilon)$-approximation algorithms for many types of low-rank tensor decompositions with respect to the Frobenius norm, including CP and Tucker decompositions. Frandsen & Ge (2022) showed that if a third-order tensor has an exact Tucker decomposition, then all local minima of an appropriately regularized loss landscape are globally optimal. Several works recently studied Tucker decomposition in streaming models (Traoré et al., 2019; Sun et al., 2020) and a sliding window model (Jang & Kang, 2021). Fast randomized low-rank tensor decomposition algorithms based on sketching have been proposed in Zhou et al. (2014); Cheng et al. (2016); Battaglino et al. (2018); Malik & Becker (2018); Che & Wei (2019); Ma & Solomonik (2021); Larsen & Kolda (2022); Fahrbach et al. (2022); Malik (2022).

2. Preliminaries

Notation. The *order* of a tensor is its number of dimensions. We denote scalars by normal lowercase letters $x \in \mathbb{R}$, vectors by boldface lower letters $\mathbf{x} \in \mathbb{R}^n$, matrices by boldface uppercase letters $\mathbf{X} \in \mathbb{R}^{m \times n}$, and higher-order tensors by boldface script letters $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$. We use normal uppercase letters for the size of an index set, e.g., $[N] = \{1, 2, \ldots, N\}$. We denote the i-th entry of vector $\mathbf{x}$ by x_i, the (i, j)-th entry of matrix $\mathbf{X}$ by x_{ij}, and the (i, j, k)-th entry of a third-order tensor $\mathcal{X}$ by x_{ijk}.

Tensor products. The *fibers* of a tensor are the vectors we get by fixing all but one index. For example, if $\mathcal{X} \in \mathbb{R}^3$, we denote the column, row, and tube fibers by $\mathbf{x}_{:jk}$, $\mathbf{x}_{i:k}$, and $\mathbf{x}_{ij:}$, respectively. The *mode-n unfolding* of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is the matrix $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times (I_1 \ldots I_{n-1} I_{n+1} \ldots I_N)}$ that arranges the mode-n fibers of $\mathcal{X}$ as columns of $\mathbf{X}_{(n)}$ ordered lexicographically by index.

We denote the *n-mode product* of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and matrix $\mathbf{A} \in \mathbb{R}^{J \times I_N}$ by $\mathcal{Y} = \mathcal{X} \times_n \mathbf{A}$, where $\mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N}$. This operation multiplies each mode-n fiber of $\mathcal{X}$ by $\mathbf{A}$, and can be expressed elementwise as

$$(\mathcal{X} \times_n \mathbf{A})_{i_1 \ldots i_{n-1} j i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \ldots i_N} a_{j i_n}.$$

The inner product of two tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is the

sum of the products of their entries:

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \ldots i_N} y_{i_1 i_2 \ldots i_N}.$$

The Frobenius norm of a tensor $\mathcal{X}$ is $\|\mathcal{X}\|_{\mathrm{F}} = \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle}$.

Tucker decomposition. The *Tucker decomposition* of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ decomposes $\mathcal{X}$ into a *core tensor* $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$ and N *factor matrices* $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$. We refer to $\mathbf{r} = (R_1, \ldots, R_N)$ as the *core shape*, which is also called the *multilinear rank* or *truncation* of the decomposition. We denote the loss of an optimal rank-$\mathbf{r}$ Tucker decomposition by

$$L(\mathcal{X}, \mathbf{r}) \stackrel{\text{def}}{=} \min_{\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_N}} \|\mathcal{X} - \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \cdots \times_N \mathbf{A}^{(N)}\|_{\mathrm{F}}^2.$$

3. Reduction to HOSVD Tucker packing

3.1. Higher-order singular value decomposition

We start with a recap of the seminal work on higher-order singular value decompositions (HOSVD) by De Lathauwer, De Moor, and Vandewalle (2000a).

Theorem 3.1 (De Lathauwer et al. (2000a, Theorem 2))**.** *Any tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ can be written as*

$$\mathcal{X} = \mathcal{S} \times_1 \mathbf{U}^{(1)} \times_2 \cdots \times_N \mathbf{U}^{(N)},$$

where each $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times I_n}$ is an orthogonal matrix and $\mathcal{S} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is a tensor with subtensors $\mathcal{S}_{i_n = \alpha}$, obtained by fixing the n-th index to α, that have the properties:

1. all-orthogonality: *for all possible values of n, α and β subject to $\alpha \neq \beta$, two subtensors $\mathcal{S}_{i_n=\alpha}$ and $\mathcal{S}_{i_n=\beta}$ are orthgonal, i.e., $\langle \mathcal{S}_{i_n=\alpha}, \mathcal{S}_{i_n=\beta} \rangle = 0$ when $\alpha \neq \beta$;*
2. ordering: *for all values of n, $\|\mathcal{S}_{i_n=1}\|_{\mathrm{F}} \geq \|\mathcal{S}_{i_n=2}\|_{\mathrm{F}} \geq \cdots \geq \|\mathcal{S}_{i_n=I_n}\|_{\mathrm{F}} \geq 0$.*

Furthermore, the values $\|\mathcal{S}_{i_n=i}\|_{\mathrm{F}}$, denoted by $\sigma_i^{(n)}$, are the singular values of the mode-n unfolding $\mathbf{X}_{(n)}$, and the columns of $\mathbf{U}^{(n)}$ are the the left singular vectors.

Next, we present the `TuckerHOSVD` algorithm. This is a widely used initialization strategy when computing rank-$\mathbf{r}$ Tucker decompositions (Kolda & Bader, 2009), i.e., if the core shape $\mathbf{r}$ is predetermined.

Algorithm 1 `TuckerHOSVD`

Input: $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, core shape $\mathbf{r} = (R_1, \ldots, R_N)$

1: **for** $n = 1$ to N **do**
2: $\quad \mathbf{A}^{(n)} \leftarrow R_n$ top left singular vectors of $\mathbf{X}_{(n)}$
3: **end for**
4: $\mathcal{G} \leftarrow \mathcal{X} \times_1 \mathbf{A}^{(1)} \times_2 \cdots \times_N \mathbf{A}^{(N)}$
5: **return** $\mathcal{G}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}$

The output of `TuckerHOSVD` has the following error guarantees (De Lathauwer et al., 2000a; Hackbusch, 2019). These bounds suggest a less expensive *surrogate loss function* to minimize instead when optimizing the core tensor shape subject to a Tucker decomposition size constraint. We give self-contained proofs of these results in Appendix A.1 that build on Theorem 3.1.

Theorem 3.2 (De Lathauwer et al. (2000a, Property 10); Hackbusch (2019, Theorem 10.2))**.** *For any tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and core shape $\mathbf{r} \in [I_1] \times \cdots \times [I_N]$, let the output of $\texttt{TuckerHOSVD}(\mathcal{X}, \mathbf{r})$ be $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$ and $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, for each $n \in [N]$. If we let*

$$\widehat{\mathcal{X}}_{\mathrm{HOSVD}(\mathbf{r})} \stackrel{\text{def}}{=} \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \cdots \times_N \mathbf{A}^{(N)} \tag{1}$$

denote the reconstructed $\mathbf{r}$-truncated tensor, then

$$\begin{aligned} \|\mathcal{X} - \widehat{\mathcal{X}}_{\mathrm{HOSVD}(\mathbf{r})}\|_{\mathrm{F}}^2 &\leq \sum_{n=1}^{N} \sum_{i_n=R_n+1}^{I_n} \left(\sigma_{i_n}^{(n)}\right)^2 \\ &\leq N \cdot L(\mathcal{X}, \mathbf{r}). \end{aligned}$$

Furthermore, we have $L(\mathcal{X}, \mathbf{r}) \leq \|\mathcal{X} - \widehat{\mathcal{X}}_{\mathrm{HOSVD}(\mathbf{r})}\|_{\mathrm{F}}^2$.

Theorem 3.2 implies that the following function is a meaningful proxy for the reconstruction error of an optimal rank-$\mathbf{r}$ Tucker decomposition.

Definition 3.3. Define the *surrogate loss* of core shape $\mathbf{r}$ as

$$\widetilde{L}(\mathcal{X}, \mathbf{r}) \stackrel{\text{def}}{=} \sum_{n=1}^{N} \sum_{i_n=R_n+1}^{I_n} \left(\sigma_{i_n}^{(n)}\right)^2. \tag{2}$$

To summarize so far, for any core shape $\mathbf{r} \in [I_1] \times \cdots \times [I_N]$, we are guaranteed that $1/N \cdot \widetilde{L}(\mathcal{X}, \mathbf{r}) \leq L(\mathcal{X}, \mathbf{r}) \leq \widetilde{L}(\mathcal{X}, \mathbf{r})$. We refer the reader to Appendix A.2 for the full details.

3.2. Tucker packing problem

Next, observe that the sum of squared singular values across all mode-n unfoldings of $\mathcal{X}$ is

$$\sum_{n=1}^{N} \sum_{i_n=1}^{I_n} \left(\sigma_{i_n}^{(n)}\right)^2 = \sum_{n=1}^{N} \|\mathbf{X}_{(n)}\|_{\mathrm{F}}^2 = N \|\mathcal{X}\|_{\mathrm{F}}^2.$$

This means we can solve a singular value packing problem instead by considering the complement of the surrogate loss. The following lemma is a wrapper for the truncated HOSVD error guarantees in Theorem 3.2.

Lemma 3.4. *For any tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and budget $c \geq 1 + \sum_{n=1}^{N} I_n$ for the size of the Tucker decomposition, let the set of feasible core shapes be*

$$F = \left\{ \mathbf{r} \in [I_1] \times \cdots \times [I_N] : \prod_{n=1}^{N} R_n + \sum_{n=1}^{N} I_n R_n \leq c \right\}.$$

Then, we have

$$\widetilde{\mathbf{r}}^* \stackrel{\text{def}}{=} \arg\min_{\mathbf{r}\in F} \widetilde{L}(\mathcal{X}, \mathbf{r}) = \arg\max_{\mathbf{r}\in F} \sum_{n=1}^{N} \sum_{i_n=1}^{R_n} \left(\sigma_{i_n}^{(n)}\right)^2.$$

Further, if $\mathbf{r}^* \stackrel{\text{def}}{=} \arg\min_{\mathbf{r}\in F} L(\mathcal{X}, \mathbf{r})$ *is an optimal budget-constrained core shape, then* $L(\mathcal{X}, \widetilde{\mathbf{r}}^*) \le N \cdot L(\mathcal{X}, \mathbf{r}^*)$.

To find a core shape whose optimal Tucker decomposition approximates the optimal loss $L(\mathcal{X}, \mathbf{r}^*)$ subject to a size constraint, we solve the maximization problem in Lemma 3.4. Optimizing this proxy objective is substantially less expensive than methods that rely on rank-$\mathbf{r}$ Tucker decomposition solvers as a subroutine. We formalize this idea by introducing the more general problem below.

Definition 3.5 (Tucker packing problem)**.** Given a shape $(I_1, \ldots, I_N) \in \mathbb{Z}_{\ge 1}^N$, N non-increasing sequences $\mathbf{a}^{(n)} \in \mathbb{R}_{\ge 0}^{I_n}$, and a budget $c \ge 1$, the *Tucker packing problem* asks to find a core shape $\mathbf{r} \in [I_1] \times \cdots \times [I_N]$ that solves:

$$\text{maximize} \quad \sum_{n=1}^{N} \sum_{i_n=1}^{R_n} a_{i_n}^{(n)} \tag{3}$$

$$\text{subject to} \quad \prod_{n=1}^{N} R_n + \sum_{n=1}^{N} I_n R_n \le c \tag{4}$$

We also denote the objective by $f(\mathbf{r}) \stackrel{\text{def}}{=} \sum_{n=1}^{N} \sum_{i_n=1}^{R_n} a_{i_n}^{(n)}$.

Theorem 3.6. *The Tucker packing problem is* NP-hard.

We prove this result in Appendix A.3 with an intricate reduction from the EQUIPARTITION problem (see, e.g., Garey & Johnson (1979, SP12)).

NP-hardness motivates the need for efficient approximation algorithms. In Section 4, we develop a *polynomial-time approximation scheme* (PTAS) for the Tucker packing problem. We leave the existence of a fully-polynomial time approximation scheme (FPTAS) as a challenging open question for future works.

To conclude, since Tucker packing is the complement of surrogate loss minimization, we must quantify how a $(1-\varepsilon)$-approximation for the packing problem can affect the error incurred in the surrogate loss. We explain this in detail in Appendix A.4 and present the main idea below.

Lemma 3.7. *Let* $\mathbf{r} \in [I_1] \times \cdots \times [I_N]$ *be any core shape that achieves a* $(1 - \varepsilon/N)$*-approximation to the Tucker packing problem. Then, we have* $\mathrm{RRE}(\mathcal{X}, \mathbf{r}) \le N \cdot \mathrm{RRE}(\mathcal{X}, \mathbf{r}^*) + \varepsilon$, *where* $\mathrm{RRE}(\mathcal{X}, \mathbf{r}) := L(\mathcal{X}, \mathbf{r}) / \|\mathcal{X}\|_{\mathrm{F}}^2$.

Remark 3.8. We can obtain global approximation guarantees for Tucker decomposition reconstruction error by (1) finding an approximately optimal core shape, (2) running `TuckerHOSVD` to initialize the Tucker decomposition, and (3) using alternating least squares (ALS) to improve the tensor decomposition. This is analogous to how k-means++ enhances Lloyd's algorithm (Arthur & Vassilvitskii, 2006).

4. Algorithm

4.1. Warm-up: Connections to multiple-choice knapsack

To start, consider a simplified version of the Tucker packing problem that only accounts for the size of the core tensor, i.e., the factor matrices do not use any of the budget. We show that after two simple transformations this new problem reduces to the *multiple-choice knapsack problem*,[1] which is NP-hard (Pisinger, 1995) but has an FPTAS (Lawler, 1977).

Concretely, the optimization problem is

$$\text{maximize} \quad \sum_{n=1}^{N} \sum_{i_n=1}^{R_n} a_{i_n}^{(n)} \tag{5}$$

$$\text{subject to} \quad \prod_{n=1}^{N} R_n \le c \tag{6}$$

Prefix sums transformation. To get closer to a 0-1 knapsack problem, define new coefficients by taking the prefix sums of the $a_{i_n}^{(n)}$'s, for each $n \in [N]$ and $i_n \in [I_n]$:

$$p_{i_n}^{(n)} \stackrel{\text{def}}{=} \sum_{j_n=1}^{i_n} a_{j_n}^{(n)}.$$

This "core size-only" Tucker packing problem can be reformulated as the following integer program:

$$\begin{aligned} \text{maximize} \quad & \sum_{n=1}^{N} \sum_{i_n=1}^{I_n} p_{i_n}^{(n)} x_{i_n}^{(n)} \\ \text{subject to} \quad & \prod_{n=1}^{N} \sum_{i_n=1}^{I_n} i_n x_{i_n}^{(n)} \le c \qquad (7) \\ & \sum_{i_n=1}^{I_n} x_{i_n}^{(n)} = 1 \qquad \forall n \in [N] \\ & x_{i_n}^{(n)} \in \{0, 1\} \qquad \forall n \in [N], i_n \in [I_n] \end{aligned}$$

We optimize over i_n instead of R_n for notational brevity.

Log transformation. Next, replace constraint (7) with the linear inequality

$$\sum_{n=1}^{N} \sum_{i_n=1}^{I_n} \log(i_n) x_{i_n}^{(n)} \le \log(c).$$

[1] The multiple-choice knapsack problem is a 0-1 knapsack problem where the items are partitioned into N classes and exactly one item must be taken from each class (Sinha & Zoltners, 1979).

This substitution is valid because in any feasible solution, for each $n \in [N]$, exactly one of $x_1^{(n)}, x_2^{(n)}, \ldots, x_{I_n}^{(n)}$ is equal to one and the rest are zero. Putting everything together, this core size-only Tucker packing problem is the following multiple-choice knapsack problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{n=1}^{N} \sum_{i_n=1}^{I_n} p_{i_n}^{(n)} x_{i_n}^{(n)} && (8) \\ \text{subject to} \quad & \sum_{n=1}^{N} \sum_{i_n=1}^{I_n} \log(i_n) x_{i_n}^{(n)} \le \log(c) \\ & \sum_{i_n=1}^{I_n} x_{i_n}^{(n)} = 1 && \forall n \in [N] \\ & x_{i_n}^{(n)} \in \{0,1\} && \forall n \in [N], i_n \in [I_n] \end{aligned}$$

Theorem 4.1 (Lawler 1977)**.** *There exists an algorithm that computes a $(1-\varepsilon)$-approximation to problem (8) in time and space $O(N^2 \varepsilon^{-1} \sum_{n=1}^{N} I_n)$.*

The FPTAS in Theorem 4.1 for multiple-choice knapsack uniformly downscales all coefficients $p_{i_n}^{(n)}$ in the objective, rounds them, and then uses dynamic programming.

4.2. PTAS for the Tucker packing problem

Now we consider the true cost of a Tucker decomposition, i.e., the size of the core tensor and the factor matrices. We first introduce a simple grid-search algorithm that solves approximate Tucker packing for a general class of feasible solutions (i.e., *downwards closed sets*). This captures the Tucker packing problem and will be useful for extending our results to tree tensor networks in Section 5.

Definition 4.2. For any $N \ge 1$ and $(I_1, \ldots, I_N) \in \mathbb{Z}_{\ge 1}^N$, let $F \subseteq [I_1] \times \cdots \times [I_N]$. The set F is *downward closed* if for any pair $(R_1, \ldots, R_N), (R'_1, \ldots, R'_N) \in [I_1] \times \cdots \times [I_N]$ such that $R'_n \le R_n$ for all $n \in [N]$, $(R_1, \ldots, R_N) \in F$ implies that $(R'_1, \ldots, R'_N) \in F$.

Lemma 4.3. *Let $0 < \varepsilon \le 1$ and $F \subseteq [I_1] \times \cdots \times [I_N]$ be downwards closed. For each $n \in [N]$, define*

$$S_n^{(\varepsilon)} = \left\{ \lceil (1+\varepsilon)^k \rceil : k \in \mathbb{Z}_{\ge 0}, \lceil (1+\varepsilon)^k \rceil \le I_n \right\}.$$

Let $\mathbf{r}^*$ be an optimal solution to the generalized problem

$$\begin{aligned} \text{maximize} \quad & \sum_{n=1}^{N} \sum_{i_n=1}^{R_n} a_{i_n}^{(n)} && (9) \\ \text{subject to} \quad & (R_1, \ldots, R_N) \in F \end{aligned}$$

and let $\mathbf{r}^{(\varepsilon)} = (R_1^{(\varepsilon)}, \ldots, R_N^{(\varepsilon)})$ be an optimal solution to

$$\begin{aligned} \text{maximize} \quad & \sum_{n=1}^{N} \sum_{i_n=1}^{R_n} a_{i_n}^{(n)} && (10) \\ \text{subject to} \quad & (R_1, \ldots, R_N) \in (S_1^{(\varepsilon)} \times \cdots \times S_N^{(\varepsilon)}) \cap F \end{aligned}$$

Then, $f(\mathbf{r}^{(\varepsilon)}) \ge (1+\varepsilon)^{-1} f(\mathbf{r}^*)$. Further, there is an algorithm that finds an optimal solution of (10) with running time $O(\sum_{n=1}^{N} I_n + \varepsilon^{-N} \prod_{n=1}^{N} (1 + \log_2(I_n)))$.

The time complexity in Lemma 4.3 is exponential in the order of the tensor, so we now focus on designing our main algorithm `TuckerPackingSolver`, whose running time is $O(\text{poly}(N, \log c, I_n))$ for constant values of $\varepsilon > 0$.

Algorithm description. There are two phases in this algorithm: (1) exhaustively search over all "small" core shapes by trying all budget allocation splits when the factor matrix cost is low; and (2) try coarser splits between the core tensor size and factor matrix costs. In the large phase, we show that it is sufficient to consider $O(\log_{1+\varepsilon} c)$ such splits. Each budget split induces a problem of the form (13), which after applying prefix sum and log transformations becomes a familiar *2-dimensional knapsack problem with a partition matroid constraint*:

$$\begin{aligned} \text{maximize} \quad & \sum_{n=1}^{N} \sum_{i_n=1}^{I_n} p_{i_n}^{(n)} x_{i_n}^{(n)} && (11) \\ \text{subject to} \quad & \sum_{n=1}^{N} \sum_{i_n=1}^{I_n} \log(i_n) x_{i_n}^{(n)} \le \log(c_{\text{core}}) \\ & \sum_{n=1}^{N} \sum_{i_n=1}^{I_n} I_n i_n x_{i_n}^{(n)} \le c - c_{\text{core}} \\ & \sum_{i_n=1}^{I_n} x_{i_n}^{(n)} = 1 && \forall n \in [N] \\ & x_{i_n}^{(n)} \in \{0,1\} && \forall n \in [N], i_n \in [I_n] \end{aligned}$$

More generally, (11) is a *d-budgeted matroid independent set problem* (see, e.g., Grandoni et al. (2014)), in which a linear objective function is maximized subject to d knapsack constraints and a matroid constraint. Recall that the multi-dimensional knapsack problem (even without any matroid constraints) does not admit an FPTAS unless P = NP (Gens & Levner, 1979; Korte & Schrader, 1981). It does, however, have a PTAS as shown by the next theorem.

Theorem 4.4 (Grandoni et al. 2014, Corollary 4.4)**.** *There is a PTAS (i.e., a $(1-\varepsilon)$-approximation algorithm) for the d-budgeted matroid independent set problem with running time $O(m^{O(d^2/\varepsilon)})$, where m is the number of items.*

The number of items in (11) is $m = \sum_{n=1}^{N} I_n$, one for each core shape dimension choice. Thus, since $d = 2$, this gives a running time of $O((\sum_{n=1}^{N} I_n)^{O(1/\varepsilon)})$. This in turn allows us to bound the overall running time of Algorithm 2.

Remark 4.5. We can use integer linear programming solvers for (11) instead of Grandoni et al. (2014), but this is possible only because we decouple the core shape cost and the factor matrix cost, i.e., because of the c_{factor} and c_{core} budget splits.

Algorithm 2 `TuckerPackingSolver`

Input: shape $(I_1, \dots, I_N) \in \mathbb{Z}^N_{\geq 1}$, N non-increasing sequences $\mathbf{a}^{(n)} \in \mathbb{R}^{I_n}_{\geq 0}$, budget $c \geq 1 + \sum_{n=1}^N I_n$, error $\varepsilon > 0$

1: Initialize $S \leftarrow \varnothing$
2: **for** $c_{\text{factor}} \in [\lceil 1/\varepsilon \rceil \sum_{n=1}^N I_n]$ **do**
3: Let $\mathbf{r}'$ be a $(1-\varepsilon)$-approximate solution to:

$$\begin{aligned} \text{maximize} \quad & f(\mathbf{r}) && (12)\\ \text{subject to} \quad & \textstyle\prod_{n=1}^N R_n \leq c - c_{\text{factor}} \\ & \textstyle\sum_{n=1}^N I_n R_n \leq c_{\text{factor}} \\ & R_n \in [\min(\lceil 1/\varepsilon \rceil, I_n)] \qquad \forall n \in [N] \end{aligned}$$

4: Update $S \leftarrow S \cup \{\mathbf{r}'\}$
5: **end for**
6: **for** $k = 0$ to $\lfloor \log_{1+\varepsilon} c \rfloor$ **do**
7: Set $c_{\text{core}} \leftarrow (1+\varepsilon)^k$
8: Let $\mathbf{r}^{(k)}$ be a $(1-\varepsilon)$-approximate solution to:

$$\begin{aligned} \text{maximize} \quad & f(\mathbf{r}) && (13)\\ \text{subject to} \quad & \textstyle\prod_{n=1}^N R_n \leq c_{\text{core}} \\ & \textstyle\sum_{n=1}^N I_n R_n \leq c - c_{\text{core}} \\ & R_n \in [I_n] \qquad \forall n \in [N] \end{aligned}$$

9: Update $S \leftarrow S \cup \{\mathbf{r}^{(k)}\}$
10: **end for**
11: **return** $\arg\max_{\mathbf{r} \in S} f(\mathbf{r})$

Theorem 4.6. *If $0 < \varepsilon < 1/3$, then Algorithm 2 returns a $(1-3\varepsilon)$-approximate solution to Problem (9) in time*

$$O\left(\left(\log_{1+\varepsilon}(c) + \lceil 1/\varepsilon \rceil \sum_{n=1}^N I_n\right)\left(\sum_{n=1}^N I_n\right)^{O(1/\varepsilon)}\right).$$

Proof. `TuckerPackingSolver` solves for two types of shapes: (1) "small" solutions where each $R_n \leq \lceil 1/\varepsilon \rceil$, and (2) "large" solutions where $R_n > \lceil 1/\varepsilon \rceil$ for some $n \in [N]$.

In the small phase, observe that since $R_n \leq \min(\lceil 1/\varepsilon \rceil, I_n)$, the factor matrix cost is $\sum_{n=1}^N I_n R_n \leq \lceil 1/\varepsilon \rceil \sum_{n=1}^N I_n$. Therefore, we can exhaustively check all small budget splits of the form $c_{\text{factor}} \in [\lceil 1/\varepsilon \rceil \sum_{n=1}^N I_n]$. Each split induces a 2-dimensional knapsack problem with a partition matroid (but for a smaller set of items), so use Theorem 4.4 to obtain a $(1-\varepsilon)$-approximation for each subproblem. If an optimal solution $\mathbf{r}^*$ to the Tucker packing problem is small, then Algorithm 2 recovers an approximately optimal objective.

For the large phase, assume that the optimal core shape $\mathbf{r}^*$ has a dimension $m \in [N]$ such that $R^*_m > \lceil 1/\varepsilon \rceil$. The algorithm searches over large shapes indirectly by splitting the budget c between the size of the core tensor and the total cost of factor matrices. A crucial observation is that we only need to check $O(\log_{1+\varepsilon} c)$ different splits because *there is a sufficient amount of slack in the large dimension R^*_m.*

To proceed, let $c^*_{\text{core}} = \prod_{n=1}^N R^*_n$, and let k^* be the largest integer such that $(1+\varepsilon)^k \leq c^*_{\text{core}}$. Define $\widehat{c}_{\text{core}} = (1+\varepsilon)^{k^*}$, and let $\widehat{\mathbf{r}} = (\widehat{R}_1, \dots, \widehat{R}_N)$ where

$$\widehat{R}_n = \begin{cases} R^*_n & \text{if } n \neq m, \\ \lfloor R^*_n/(1+\varepsilon) \rfloor & \text{if } n = m. \end{cases}$$

Since we assumed $R^*_m \geq \lceil 1/\varepsilon \rceil$ is a large dimension, $\widehat{R}_m = \lfloor R^*_m/(1+\varepsilon) \rfloor \geq \lfloor \lceil 1/\varepsilon \rceil/(1+\varepsilon) \rfloor$. Next, observe that

$$\begin{aligned} \prod_{n=1}^N \widehat{R}_n &\leq \frac{1}{1+\varepsilon} \prod_{n=1}^N R^*_n = \frac{c_{\text{core}}}{1+\varepsilon} \\ &< \frac{(1+\varepsilon)^{k^*+1}}{1+\varepsilon} = (1+\varepsilon)^{k^*} = \widehat{c}_{\text{core}}, \end{aligned}$$

and $\sum_{n=1}^N I_n \widehat{R}_n \leq \sum_{n=1}^N I_n R^*_n \leq c - c^*_{\text{core}} \leq c - \widehat{c}_{\text{core}}$. Therefore, $\widehat{\mathbf{r}}$ is a feasible solution of (13) for $k = k^*$. Furthermore, $f(\mathbf{r}^{(k^*)}) \geq (1-\varepsilon) f(\widehat{\mathbf{r}})$.

Next, we show that $f(\widehat{\mathbf{r}}) \geq (1-2\varepsilon) f(\mathbf{r}^*)$. For any $n \neq m$, we have $\widehat{R}_n = R^*_n$, so it follows that

$$\sum_{i_n=1}^{\widehat{R}_n} a^{(n)}_{i_n} = \sum_{i_n=1}^{R^*_n} a^{(n)}_{i_n}. \tag{14}$$

For the large dimension m, we have

$$\widehat{R}_m = \left\lfloor \frac{R^*_m}{1+\varepsilon} \right\rfloor \geq \frac{R^*_m}{1+\varepsilon} - 1 = \frac{R^*_m - (1+\varepsilon)}{1+\varepsilon} \cdot \frac{R^*_m}{R^*_m}.$$

The assumption $R^*_m \geq \lceil 1/\varepsilon \rceil$ then gives

$$\frac{R^*_m - (1+\varepsilon)}{R^*_m} \geq 1 - \frac{1+\varepsilon}{\lceil 1/\varepsilon \rceil} \geq 1 - \varepsilon - \varepsilon^2.$$

It follows for any $\varepsilon \geq 0$ that

$$\widehat{R}_m \geq \frac{1-\varepsilon-\varepsilon^2}{1+\varepsilon} R^*_m \geq (1-2\varepsilon) R^*_m.$$

Since $a^{(m)}_{i_1} \geq \cdots \geq a^{(m)}_{I_m}$ is non-increasing, we have

$$\sum_{i_m=1}^{\widehat{R}_m} a^{(m)}_{i_m} \geq (1-2\varepsilon) \sum_{i_m=1}^{R^*_m} a^{(m)}_{i_m}. \tag{15}$$

Combining (14) and (15) gives $f(\widehat{\mathbf{r}}) \geq (1-2\varepsilon) f(\mathbf{r}^*)$, so then $f(\mathbf{r}^{(k^*)}) \geq (1-\varepsilon)(1-2\varepsilon) f(\mathbf{r}^*) \geq (1-3\varepsilon) f(\mathbf{r}^*)$, which proves the approximation guarantee.[2]

Finally, the running time follows from our reductions to the 2-dimensional knapsack problem with a partition matroid constraint, and using Theorem 4.4 for each budget split. □

[2] To obtain a PTAS, it is enough to consider the budget splits $c_{\text{factor}} \in [\sum_{n=1}^N I_n^2]$, i.e., line 2 of `TuckerPackingSolver`, but this gives a worse running time as explained in Appendix B.

5. Tree tensor network decompositions

Here we consider a general decomposition called tree tensor network (Oseledets & Tyrtyshnikov, 2009; Krämer, 2020), which includes Tucker decomposition, tensor-train decomposition, and hierarchical Tucker decomposition as special cases.

Definition 5.1 (Tree tensor network)**.** Let $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ be any tensor. Let $G = (V, E)$ be a rooted tree with N leaves where each node v corresponds to a subset $S_v \subseteq [N]$. The leaves are the N singletons of $[N]$, and internal nodes v are recursively defined by $S_v = \cup_{u \in C_v} S_u$, where C_v is the set of children of v.

Each edge $e \in E$ is endowed an integer $R_e \geq 1$. Then, a (truncated) *tree tensor network* decomposition of $\mathcal{X}$ for tree G is the following collection of tensors, each corresponding to a $v \in V$. For each leaf, the tensor is $\mathbf{A}^{(v)} \in \mathbb{R}^{I_n \times R_e}$, where e is the edge connecting v to its parent. For each internal node v, its tensor is $\mathcal{T}^{(v)} \in \mathbb{R}^{R_{e_1} \times \cdots \times R_{e_k}}$, where $E_v = \{e_1, \ldots, e_k\}$ is the set of edges incident to v.

The output tensor $\widehat{\mathcal{X}}$ is constructed by taking the mode-wise products of all the "node tensors" in G over their corresponding edges. These products commute and are associative (see, e.g., Proposition 2.17 in Krämer (2020)).

Remark 5.2. The tree tensor network for Tucker decomposition corresponds to a tree of depth one, and for hierarchical Tucker decomposition it is an (almost) balanced binary tree (Oseledets & Tyrtyshnikov, 2009; Grasedyck, 2010).

We give an example with figures in Appendix C. Now we generalize the definition of tensor unfolding.

Definition 5.3. For any $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $S \subseteq [N]$, the *matricization* $\mathbf{X}_{(S)} \in \mathbb{R}^{P \times Q}$, where $P = \prod_{n \in S} I_n$ and $Q = \prod_{n \in [N] \setminus S} I_n$, is the matrix with the entries of $\mathcal{X}$ arranged lexicographically by their original index tuples.

The next theorem gives a polynomial-time algorithm for finding a tree tensor network decomposition that achieves bounded reconstruction error for specified R_e values.

Theorem 5.4 (Grasedyck (2010); Krämer (2020))**.** *Let $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $G = (V, E)$, R_e for $e \in E$ be the tree tensor network parameters as in Definition 5.1. There exists a polynomial-time algorithm that finds $\mathcal{T}^{(v)}$ for $v \in V$ (for leaves these tensors are the matrices $\mathbf{A}^{(v)}$) such that*

$$\|\mathcal{X} - \widehat{\mathcal{X}}\|_{\mathrm{F}}^2 \leq \sum_{v \in V \setminus \{r\}} \sum_{i_v = R_v + 1}^{P_v} \left(\sigma_{i_v}^{(v)}\right)^2 \leq (|V| - 1) \cdot \|\mathcal{X} - \mathcal{X}_{\text{best}}\|_{\mathrm{F}}^2,$$

where $\mathcal{X}_{\text{best}}$ is the best tree tensor network decomposition for G and the values R_e, r is the root node, $P_v = \prod_{n \in S_v} I_n$, and $\sigma_i^{(v)}$ is the i-th singular value of $\mathbf{X}_{(S_v)}$.

Further, the size of the tree tensor network decomposition is $\sum_{v \in L} I_v R_v + \sum_{v \in J} \prod_{e \in E_v} R_e$, where L is the set of leaves, J is the set of internal nodes, and R_v is value on the edge that connects v to its parent.

It follows that we can define the NP-hard tree tensor network packing problem, which generalizes Tucker packing.

Definition 5.5 (Tree tensor network packing)**.** Given a shape $(I_1, \ldots, I_N) \in \mathbb{Z}_{\geq 1}^N$, tree $G = (V, E)$ with leaves $L = [N]$ and internal nodes J, $|V| - 1$ non-increasing sequences (corresponding to non-root nodes) $\mathbf{a}^{(v)} \in \mathbb{R}_{\geq 0}^{P_v}$ with $P_v = \prod_{n \in S_v} I_n$, and the budget $c \geq 1$, the *tree tensor network packing* problem asks to find $R_v \in \mathbb{Z}_{\geq 1}$ for $v \in V \setminus \{r\}$, where r is the root and N_v is the set of neighbors of node v, that solves:

$$\text{maximize} \quad \sum_{v \in V \setminus \{r\}} \sum_{i=1}^{R_v} a_i^{(v)} \tag{16}$$

$$\text{subject to} \quad \sum_{v \in J} \prod_{u \in N_v} R_u + \sum_{v \in L} I_v R_v \leq c \tag{17}$$

Theorem 5.6. *There is a $(1 - \varepsilon)$-approximation algorithm for the tree tensor network packing problem the runs in time*

$$O\left(\sum_{v \in V \setminus \{t\}} P_v + \varepsilon^{-(|V|-1)} \prod_{v \in V \setminus \{t\}} (1 + \log_2(P_v))\right).$$

6. Experiments

We compare several algorithms for computing the core shapes of size-constrained Tucker decompositions on four real-world tensors (see Table 1). These experiments demonstrate the effectiveness of using the surrogate loss $\widetilde{L}(\mathcal{X}, \mathbf{r})$ in place of the true relative reconstruction error (RRE), both in terms of solution quality and running time. All experiments use NumPy (Harris et al., 2020) with an Intel Xeon W-1235 processor (3.7 GHz, 8.25MB cache) and 128GB of RAM. The source code is available at `https://github.com/fahrbach/approximately-optimal-core-shapes`.

Table 1. Statistics for tensor datasets used in experiments.

TENSOR	SHAPE	SIZE
CARDIAC MRI	$256 \times 256 \times 14 \times 20$	$18,350,080$
HYPERSPECTRAL	$1024 \times 1344 \times 33$	$45,416,488$
VICROADS	$1084 \times 2033 \times 96$	$211,562,112$
COIL-100	$7200 \times 128 \times 128 \times 3$	$353,894,400$

6.1. Algorithms

The first four algorithms we consider are based on HOSVD Tucker packing: they compute the mode-n singular values $\sigma_{i_n}^{(n)}$ and take $\mathbf{a}^{(n)} = \{(\sigma_{i_n}^{(n)})^2\}_{i_n=1}^{I_n}$ as input to their Tucker

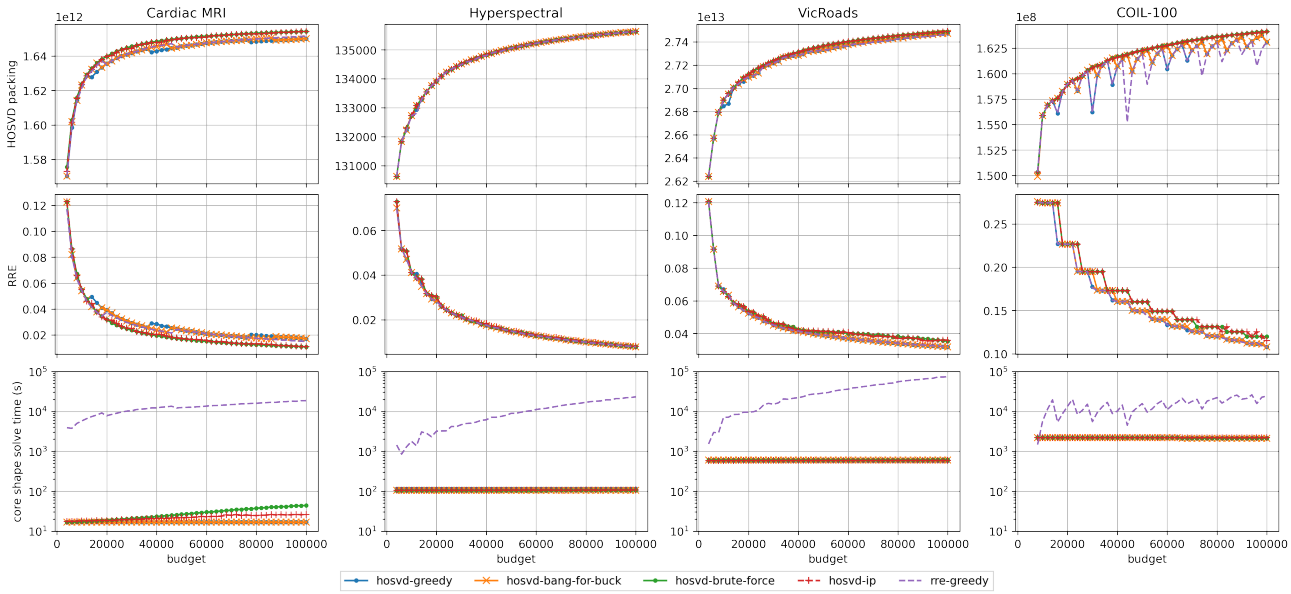

Figure 2. Comparison of five core shape solvers on four real-world tensors (columns) for increasing values of the Tucker decomposition size budget $c \leq 100,000$. The plots in the top row are the HOSVD Tucker packing objective value $f(\mathbf{r})$ for the core shape solutions $\mathbf{r}$, the middle row is the RRE, and the bottom row is the running time of each algorithm in seconds.

packing instance. The fifth is a commonly used greedy algorithm that computes true losses $L(\mathcal{X}, \mathbf{r})$ at each step.

HOSVD-IP is the `TuckerPackingSolver` algorithm with $\varepsilon = 0.25$, but we use the integer programming solver in `scipy.optimize.mlip` to solve each budget split subproblem instead of the PTAS in Grandoni et al. (2014).

HOSVD-greedy maximizes the same packing objective by repeating $\mathbf{r} \leftarrow \arg\max_{\mathbf{r}' \in N(\mathbf{r})} f(\mathbf{r}')$, where $N(\mathbf{r})$ is the set of neighboring feasible core shapes $\mathbf{r}' = \mathbf{r} + \mathbf{e}_n$ and $\mathbf{e}_n \in \{0, 1\}^N$ is a standard unit vector. This is Algorithm 3.1 in Ehrlacher et al. (2021) with additional budget constraints.

HOSVD-bang-for-buck is analogous to HOSVD-greedy, but it increments the dimension in each step that maximizes $(f(\mathbf{r}') - f(\mathbf{r}))/(\text{cost}(\mathbf{r}') - \text{cost}(\mathbf{r}))$, where $\text{cost}(\mathbf{r})$ is the size of the rank-$\mathbf{r}$ Tucker decomposition.

HOSVD-brute-force exhaustively checks all feasible core shapes and outputs the maximum Tucker packing objective.

RRE-greedy constructs the core shape by computing $O(N)$ rank-$\mathbf{r}'$ Tucker decompositions in each step and incrementing the dimension that most improves the RRE, Concretely, the update is $\mathbf{r} \leftarrow \arg\min_{\mathbf{r}' \in N(\mathbf{r})} L(\mathcal{X}, \mathbf{r}')$, similar to the Greedy-TL algorithm of Hashemizadeh et al. (2020).

6.2. Results

We consider the budgets $c \leq 100,000$ for all tensor datasets. For each c, we run each algorithm to get core shape $\mathbf{r}$. Then in Figure 2, we plot the packing objective $f(\mathbf{r})$, the RRE, i.e., $L(\mathcal{X}, \mathbf{r})/\|\mathcal{X}\|_{\text{F}}^2$, and the algorithm running time (including the mode-n singular value computations) as a function of c. Each $L(\mathcal{X}, \mathbf{r})$ computation uses 20 iterations of HOOI.

Cardiac MRI shows that maximizing the HOSVD Tucker packing objective $f(\mathbf{r})$ can give noticeably better RRE than RRE-greedy, while also running 1000x faster. If we take a closer look at the core shapes these algorithms output, HOSVD-{brute-force, IP} always return core shapes of the form $(x, y, z, 1)$, whereas the greedy algorithms allocate budget to the time dimension as c increases, e.g., $(x, y, z, 3)$. Increases in the fourth dimension correspond to points of degradation in $f(\mathbf{r})$ and RRE in Figure 2.

This tensor is also small enough to see differences in the running times of the HOSVD packing algorithms. In particular, we see (1) that there is a fixed cost for computing the $\sigma_{i_n}^{(n)}$'s, and (2) that HOSVD-{greedy, bang-for-buck} are faster than HOSVD-IP, which is faster than HOSVD-brute-force. All algorithms are significantly faster than RRE-greedy.

Hyperspectral shows that the surrogate loss $\widetilde{L}(\mathcal{X}, \mathbf{r})$ and RRE can guide greedy algorithms to the same core shapes, and that HOSVD-greedy can achieve maximum the Tucker packing objective. We see that computing the higher-order singular values becomes the bottleneck for the HOSVD solvers, not solving the packing instances themselves.

VicRoads shows a clear gap between RRE and the surrogate loss. While HOSVD-{greedy, bang-for-buck} are suboptimal in the packing objective, they achieve the same RRE as the 100x slower RRE-greedy algorithm. This data demonstrates a shortcoming of the surrogate loss, but also shows that higher-order singular values can still be effective.

COIL-100 is perhaps the most interesting tensor because it shows the *non-monotonic* behavior of greedy HOSVD Tucker packing algorithms. Similar to cardiac MRI, every time a greedy core shape solver increases the dimension of the first index (corresponding to the number of objects), the packing objective $f(\mathbf{r})$. This effect also appears in the RRE plots, but it happens in the opposite direction.

Acknowledgements

We thank the anonymous reviewers of an earlier version of this work for directing us to the multiple-choice knapsack literature. We also want to thank Mohit Singh and Anupam Gupta for fruitful discussions about our main algorithm.

References

Arthur, D. and Vassilvitskii, S. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

Bader, B. W. and Kolda, T. G. Tensor toolbox for MATLAB, version 3.4. https://www.tensortoolbox.org/, 2022.

Battaglino, C., Ballard, G., and Kolda, T. G. A practical randomized CP tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 39(2):876–901, 2018.

Che, M. and Wei, Y. Randomized algorithms for the approximations of tucker and the tensor train decompositions. *Advances in Computational Mathematics*, 45(1):395–428, 2019.

Cheng, D., Peng, R., Liu, Y., and Perros, I. SPALS: Fast alternating least squares via implicit leverage scores sampling. *Advances in Neural Information Processing Systems*, 29, 2016.

De Lathauwer, L., De Moor, B., and Vandewalle, J. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000a.

De Lathauwer, L., De Moor, B., and Vandewalle, J. On the best rank-1 and rank-$(r_1, r_2, \ldots, r_n)$ approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000b.

Eckart, C. and Young, G. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Ehrlacher, V., Grigori, L., Lombardi, D., and Song, H. Adaptive hierarchical subtensor partitioning for tensor compression. *SIAM Journal on Scientific Computing*, 43(1): A139–A163, 2021.

Eldén, L. and Dehghan, M. A Krylov–Schur-like method for computing the best rank-(r_1, r_2, r_3) approximation of large and sparse tensors. *Numerical Algorithms*, pp. 1–33, 2022.

Eldén, L. and Savas, B. A Newton–Grassmann method for computing the best multilinear rank-(r_1, r_2, r_3) approximation of a tensor. *SIAM Journal on Matrix Analysis and Applications*, 31(2):248–271, 2009.

Fahrbach, M., Fu, G., and Ghadiri, M. Subquadratic kronecker regression with applications to tensor decomposition. *Advances in Neural Information Processing Systems*, 35:28776–28789, 2022.

Frandsen, A. and Ge, R. Optimization landscape of Tucker decomposition. *Mathematical Programming*, 193(2):687–712, 2022.

Garey, M. R. and Johnson, D. S. *Computers and Intractability*. W. H. Freeman, 1979.

Gens, G. V. and Levner, E. V. Computational complexity of approximation algorithms for combinatorial problems. In *International Symposium on Mathematical Foundations of Computer Science*, pp. 292–300. Springer, 1979.

Grandoni, F., Ravi, R., Singh, M., and Zenklusen, R. New approaches to multi-objective optimization. *Mathematical Programming*, 146(1):525–554, 2014.

Grasedyck, L. Hierarchical singular value decomposition of tensors. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2029–2054, 2010.

Hackbusch, W. *Tensor Spaces and Numerical Tensor Calculus*, volume 56. Springer, 2nd edition, 2019.

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

Hashemizadeh, M., Liu, M., Miller, J., and Rabusseau, G. Adaptive tensor learning with tensor networks. *Workshop on Quantum Tensor Networks in Machine Learning, 34th Conference on Neural Information Processing Systems*, 2020.

Hillar, C. J. and Lim, L.-H. Most tensor problems are NP-hard. *Journal of the ACM (JACM)*, 60(6):1–39, 2013.

Ishteva, M., De Lathauwer, L., Absil, P.-A., and Van Huffel, S. Differential-geometric newton method for the best rank-(r_1, r_2, r_3) approximation of tensors. *Numerical Algorithms*, 51(2):179–194, 2009.

Ishteva, M., Absil, P.-A., Van Huffel, S., and De Lathauwer, L. Best low multilinear rank approximation of higher-order tensors, based on the Riemannian trust-region scheme. *SIAM Journal on Matrix Analysis and Applications*, 32(1):115–135, 2011.

Ishteva, M., Absil, P.-A., and Van Dooren, P. Jacobi algorithm for the best low multilinear rank approximation of symmetric tensors. *SIAM Journal on Matrix Analysis and Applications*, 34(2):651–672, 2013.

Jang, J.-G. and Kang, U. Fast and memory-efficient Tucker decomposition for answering diverse time range queries. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 725–735, 2021.

Kim, Y.-D., Park, E., Yoo, S., Choi, T., Yang, L., and Shin, D. Compression of deep convolutional neural networks for fast and low power mobile applications. In *ICLR*, 2016.

Kolda, T. G. and Bader, B. W. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

Korte, B. and Schrader, R. On the existence of fast approximation schemes. In *Nonlinear Programming 4*, pp. 415–437. Elsevier, 1981.

Krämer, S. *Tree tensor networks, associated singular values and high-dimensional approximation*. PhD thesis, RWTH Aachen University, 2020.

Larsen, B. W. and Kolda, T. G. Practical leverage-based sampling for low-rank tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 43(3):1488–1517, 2022.

Lawler, E. L. Fast approximation algorithms for knapsack problems. In *18th Annual Symposium on Foundations of Computer Science*, pp. 206–213. IEEE, 1977.

Ma, L. and Solomonik, E. Fast and accurate randomized algorithms for low-rank tensor decompositions. *Advances in Neural Information Processing Systems*, 34:24299–24312, 2021.

Malik, O. A. More efficient sampling for tensor decomposition with worst-case guarantees. In *International Conference on Machine Learning*, pp. 14887–14917. PMLR, 2022.

Malik, O. A. and Becker, S. Low-rank Tucker decomposition of large tensors using TensorSketch. *Advances in Neural Information Processing Systems*, 31:10096–10106, 2018.

Nascimento, S. M., Amano, K., and Foster, D. H. Spatial distributions of local illumination color in natural scenes. *Vision Research*, 120:39–44, 2016.

Nene, S. A., Nayar, S. K., Murase, H., et al. Columbia object image library (COIL-20). 1996.

Oseledets, I. V. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

Oseledets, I. V. and Tyrtyshnikov, E. E. Breaking the curse of dimensionality, or how to use svd in many dimensions. *SIAM Journal on Scientific Computing*, 31(5):3744–3759, 2009.

Pisinger, D. A minimal algorithm for the multiple-choice knapsack problem. *European Journal of Operational Research*, 83(2):394–410, 1995.

Schimbinschi, F., Nguyen, X. V., Bailey, J., Leckie, C., Vu, H., and Kotagiri, R. Traffic forecasting in complex urban networks: Leveraging big data and machine learning. In *2015 IEEE International Conference on Big Data*, pp. 1019–1024. IEEE, 2015.

Sidiropoulos, N. D., De Lathauwer, L., Fu, X., Huang, K., Papalexakis, E. E., and Faloutsos, C. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, 2017.

Sinha, P. and Zoltners, A. A. The multiple-choice knapsack problem. *Operations Research*, 27(3):503–515, 1979.

Song, Z., Woodruff, D. P., and Zhong, P. Relative error tensor low rank approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2772–2789. SIAM, 2019.

Sun, Y., Guo, Y., Luo, C., Tropp, J., and Udell, M. Low-rank Tucker approximation of a tensor from streaming data. *SIAM Journal on Mathematics of Data Science*, 2 (4):1123–1150, 2020.

Traoré, A., Berar, M., and Rakotomamonjy, A. Singleshot: A scalable Tucker tensor decomposition. *Advances in Neural Information Processing Systems*, 32, 2019.

Vannieuwenhoven, N., Vandebril, R., and Meerbergen, K. A new truncation strategy for the higher-order singular value decomposition. *SIAM Journal on Scientific Computing*, 34(2):A1027–A1052, 2012.

Xiao, C. and Yang, C. A rank-adaptive higher-order orthogonal iteration algorithm for truncated Tucker decomposition. *arXiv preprint arXiv:2110.12564*, 2021.

Zhou, G., Cichocki, A., and Xie, S. Decomposition of big tensors with low multilinear rank. *arXiv preprint arXiv:1412.1885*, 2014.

A. Missing analysis from Section 3

A.1. Proof of Theorem 3.2

A.1.1. Upper bounding the reconstruction error

For any Tucker decomposition $\mathcal{X} = \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \cdots \times_N \mathbf{A}^{(N)} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, the mode-$n$ unfolding of $\mathcal{X}$ can be written as

$$\mathbf{X}_{(n)} = \mathbf{A}^{(n)} \mathbf{G}_{(n)} \left(\mathbf{A}^{(1)} \otimes \cdots \mathbf{A}^{(n-1)} \otimes \mathbf{A}^{(n+1)} \otimes \cdots \otimes \mathbf{A}^{(N)} \right)^{\intercal}. \tag{18}$$

This is the corrected version of Equation (4.2) in Kolda & Bader (2009).

Lemma A.1 (De Lathauwer et al. (2000a, Property 10)). *For any $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $n \in [N]$, let $\sigma_1^{(n)} \geq \sigma_2^{(n)} \cdots \geq \sigma_{I_n}^{(n)}$ denote the singular values of $\mathbf{X}_{(n)}$. Then, for any core shape $\mathbf{r} = (R_1, R_2, \ldots, R_N) \in [I_1] \times [I_2] \times \cdots \times [I_N]$, we have*

$$\begin{aligned} L(\mathcal{X}, \mathbf{r}) &\leq \|\mathcal{X} - \widehat{\mathcal{X}}_{\text{HOSVD}(\mathbf{r})}\|_{\text{F}}^2 \\ &\leq \sum_{i_1=R_1+1}^{I_1} \left(\sigma_{i_1}^{(1)}\right)^2 + \sum_{i_2=R_2+1}^{I_2} \left(\sigma_{i_2}^{(2)}\right)^2 + \cdots + \sum_{i_N=R_N+1}^{I_N} \left(\sigma_{i_N}^{(N)}\right)^2. \end{aligned}$$

Proof. Let the HOSVD of $\mathcal{X} = \mathcal{S} \times_1 \mathbf{U}^{(1)} \times_2 \cdots \times_N \mathbf{U}^{(N)}$ as in Theorem 3.1. Let $\overline{\mathcal{S}}$ denote the truncated version of $\mathcal{S}$ with respect to core shape $\mathbf{r}$ such that

$$\overline{s}_{i_1 \ldots i_N} = \begin{cases} s_{i_1 \ldots i_N} & \text{if } i_n \leq R_n \text{ for all } n \in [N], \\ 0 & \text{otherwise.} \end{cases}$$

Let $\overline{\mathcal{X}} = \overline{\mathcal{S}} \times_1 \mathbf{U}^{(1)} \times_2 \cdots \times_N \mathbf{U}^{(N)}$. Summing over all dimensions $n \in [N]$ and using the HOSVD results in Theorem 3.1,

$$\begin{aligned} \|\mathcal{S} - \overline{\mathcal{S}}\|_{\text{F}}^2 &\leq \sum_{n=1}^{N} \sum_{j=R_n+1}^{I_n} \|(\mathcal{S} - \overline{\mathcal{S}})_{i_n=j}\|_{\text{F}}^2 \\ &= \sum_{n=1}^{N} \sum_{j=R_n+1}^{I_n} \|\mathcal{S}_{i_n=j}\|_{\text{F}}^2 \\ &= \sum_{n=1}^{N} \sum_{i_n=R_n+1}^{I_n} \left(\sigma_{i_n}^{(n)}\right)^2. \end{aligned}$$

We have $\|\mathcal{S} - \overline{\mathcal{S}}\|_{\text{F}}^2 = \|\mathbf{S}_{(n)} - \overline{\mathbf{S}}_{(n)}\|_{\text{F}}^2$ for all values of n. Further, since the Kronecker product of two orthogonal matrices is also orthogonal and multiplication by an orthogonal matrix does not affect the Frobenius norm, Equation (18) implies that

$$\begin{aligned} \|\mathbf{S}_{(n)} - \overline{\mathbf{S}}_{(n)}\|_{\text{F}}^2 &= \|\mathbf{U}_{(n)}(\mathbf{S}_{(n)} - \overline{\mathbf{S}}_{(n)})(\mathbf{U}_{(1)} \otimes \ldots \otimes \mathbf{U}_{(n-1)} \otimes \mathbf{U}_{(n+1)} \otimes \cdots \otimes \mathbf{U}_{(N)})^{\intercal}\|_{\text{F}}^2 \\ &= \|\mathcal{X} - \overline{\mathcal{X}}\|_{\text{F}}^2. \end{aligned}$$

Observing that $\overline{\mathcal{X}} = \widehat{\mathcal{X}}_{\text{HOSVD}(\mathbf{r})}$ by the definition of `TuckerHOSVD` in Algorithm 1 and putting everything together,

$$L(\mathcal{X}, \mathbf{r}) \leq \|\mathcal{X} - \widehat{\mathcal{X}}_{\text{HOSVD}(\mathbf{r})}\|_{\text{F}}^2 = \|\mathcal{X} - \overline{\mathcal{X}}\|_{\text{F}}^2 = \|\mathcal{S} - \overline{\mathcal{S}}\|_{\text{F}}^2 \leq \sum_{n=1}^{N} \sum_{i_n=R_n+1}^{I_n} \left(\sigma_{i_n}^{(n)}\right)^2,$$

which completes the proof. □

A.1.2. Lower bounding the reconstruction error

Theorem A.2 (Eckart & Young (1936)). *Let $\mathbf{A} \in \mathbb{R}^{n \times d}$ with $n \geq d$ and singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d \geq 0$. Let $\mathbf{A}_k$ be the best rank-k approximation of $\mathbf{A}$ in the Frobenius norm. Then*

$$\|\mathbf{A} - \mathbf{A}_k\|_{\text{F}}^2 = \sum_{i=k+1}^{d} \sigma_i^2.$$

Lemma A.3. *For any $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $n \in [N]$, let $\sigma_1^{(n)} \geq \cdots \geq \sigma_{I_n}^{(n)}$ denote the singular values of $\mathbf{X}_{(n)}$. For every core shape $\mathbf{r} = (R_1, R_2, \ldots, R_N) \in [I_1] \times [I_2] \times \cdots \times [I_N]$, we have*

$$L(\mathcal{X}, \mathbf{r}) \geq \max_{n \in [N]} \sum_{i_n = R_n + 1}^{I_n} \left(\sigma_{i_n}^{(n)}\right)^2.$$

Proof. Let the core tensor and factors that minimize $L(\mathcal{X}, \mathbf{r})$ be $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$ and $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, i.e.,

$$L(\mathcal{X}, \mathbf{r}) = \|\mathcal{X} - \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \cdots \times_N \mathbf{A}^{(N)}\|_{\mathrm{F}}^2.$$

Let $\widehat{\mathcal{X}} = \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \cdots \times_N \mathbf{A}^{(N)}$. Equation (18) and the dimensions of $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ imply that

$$\operatorname{rank}\left(\widehat{\mathbf{X}}_{(n)}\right) \leq \operatorname{rank}\left(\mathbf{A}^{(n)}\right) \leq R_n,$$

since $R_n \leq I_n$. The Eckart–Young–Mirsky theorem (Theorem A.2) with the characterization of $\sigma_i^{(n)}$ in Theorem 3.1 gives

$$\|\mathcal{X} - \widehat{\mathcal{X}}\|_{\mathrm{F}}^2 = \|\mathbf{X}_{(n)} - \widehat{\mathbf{X}}_{(n)}\|_{\mathrm{F}}^2 \geq \sum_{i_n = R_n + 1}^{I_n} \left(\sigma_{i_n}^{(n)}\right)^2. \tag{19}$$

Equation (19) holds for all values of n, so take the equation that maximizes the right-hand side. □

A.1.3. Combining the results

Now we combine Lemma A.1 and Lemma A.3 to give an approximation inequality that is true for all core shapes.

Lemma A.4. *For any $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and any core shape $\mathbf{r} = (R_1, R_2, \ldots, R_N) \in [I_1] \times [I_2] \times \cdots \times [I_N]$, we have*

$$\frac{1}{N} \cdot \widetilde{L}(\mathcal{X}, \mathbf{r}) \leq L(\mathcal{X}, \mathbf{r}) \leq \widetilde{L}(\mathcal{X}, \mathbf{r}),$$

where $\widetilde{L}(\mathcal{X}, \mathbf{r})$ is defined in Equation (2).

Proof. Summing Equation (19) in the proof of Lemma A.3 over all values of $n \in [N]$ gives

$$\sum_{n=1}^{N} \sum_{i_n = R_n + 1}^{I_n} \left(\sigma_{i_n}^{(n)}\right)^2 \leq N \cdot L(\mathcal{X}, \mathbf{r}) \implies \frac{1}{N} \cdot \widetilde{L}(\mathcal{X}, \mathbf{r}) \leq L(\mathcal{X}, \mathbf{r}). \tag{20}$$

The upper bound $L(\mathcal{X}, \mathbf{r}) \leq \widetilde{L}(\mathcal{X}, \mathbf{r})$ is a restatement of Lemma A.1. □

Theorem 3.2 (De Lathauwer et al. (2000a, Property 10); Hackbusch (2019, Theorem 10.2))**.** *For any tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and core shape $\mathbf{r} \in [I_1] \times \cdots \times [I_N]$, let the output of $\texttt{TuckerHOSVD}(\mathcal{X}, \mathbf{r})$ be $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$ and $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, for each $n \in [N]$. If we let*

$$\widehat{\mathcal{X}}_{\mathrm{HOSVD}(\mathbf{r})} \stackrel{\mathrm{def}}{=} \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \cdots \times_N \mathbf{A}^{(N)} \tag{1}$$

denote the reconstructed $\mathbf{r}$-truncated tensor, then

$$\begin{aligned} \|\mathcal{X} - \widehat{\mathcal{X}}_{\mathrm{HOSVD}(\mathbf{r})}\|_{\mathrm{F}}^2 &\leq \sum_{n=1}^{N} \sum_{i_n = R_n + 1}^{I_n} \left(\sigma_{i_n}^{(n)}\right)^2 \\ &\leq N \cdot L(\mathcal{X}, \mathbf{r}). \end{aligned}$$

Furthermore, we have $L(\mathcal{X}, \mathbf{r}) \leq \|\mathcal{X} - \widehat{\mathcal{X}}_{\mathrm{HOSVD}(\mathbf{r})}\|_{\mathrm{F}}^2$.

Proof. The proof follows by combining Lemma A.1 and Equation (20). □

A.2. Proof of Lemma 3.4

Lemma A.5. *For any tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and budget $c \geq 1 + \sum_{n=1}^{N} I_n$ for the size of the Tucker decomposition, let the set of feasible core shapes be*

$$F = \left\{ \mathbf{r} \in [I_1] \times \cdots \times [I_N] : \prod_{n=1}^{N} R_n + \sum_{n=1}^{N} I_n R_n \leq c \right\}.$$

Then, we have

$$\widetilde{\mathbf{r}}^* \stackrel{\text{def}}{=} \arg\min_{\mathbf{r} \in F} \widetilde{L}(\mathcal{X}, \mathbf{r}) = \arg\max_{\mathbf{r} \in F} \sum_{n=1}^{N} \sum_{i_n=1}^{R_n} \left(\sigma_{i_n}^{(n)}\right)^2.$$

Further, if $\mathbf{r}^* \stackrel{\text{def}}{=} \arg\min_{\mathbf{r} \in F} L(\mathcal{X}, \mathbf{r})$ is an optimal budget-constrained core shape, then $L(\mathcal{X}, \widetilde{\mathbf{r}}^*) \leq N \cdot L(\mathcal{X}, \mathbf{r}^*)$.

Proof. For any n, we have

$$\sum_{i_n=1}^{I_n} \left(\sigma_{i_n}^{(n)}\right)^2 = \|\mathbf{X}_{(n)}\|_{\mathrm{F}}^2 = \|\mathcal{X}\|_{\mathrm{F}}^2.$$

Therefore, for any choice of $\mathbf{r} = (R_1, R_2, \ldots, R_N)$, we have

$$\left[\sum_{n=1}^{N} \sum_{i_n=1}^{R_n} \left(\sigma_{i_n}^{(n)}\right)^2\right] + \left[\sum_{n=1}^{N} \sum_{i_n=R_n+1}^{I_n} \left(\sigma_{i_n}^{(n)}\right)^2\right] = N\|\mathcal{X}\|_{\mathrm{F}}^2.$$

This is a constant value that only depends on $\mathcal{X}$, so minimizing $\widetilde{L}(\mathcal{X}, \mathbf{r})$ is equivalent to maximizing the packing version since both problems optimize over the same set F.

Lastly, we have

$$\widetilde{L}(\mathcal{X}, \widetilde{\mathbf{r}}^*) \leq \widetilde{L}(\mathcal{X}, \mathbf{r}^*) \leq N \cdot L(\mathcal{X}, \mathbf{r}^*),$$

where the first inequality follows from optimizing the surrogate loss and the second inequality follows from Lemma A.4 since that result holds for all core shapes. □

A.3. Hardness

Definition A.6. Let $N \geq 2$ be an even integer and $w_1, \ldots, w_N \geq 1$ be integers. The EQUIPARTITION problem asks to determine whether there exists a subset $S \subseteq [N]$ of size $n/2$ such that

$$\sum_{i \in S} w_i = \sum_{i \in [N] \setminus S} w_i.$$

Lemma A.7 (Garey & Johnson 1979, SP12)**.** EQUIPARTITION *is* NP-complete.

We now give a reduction from the equipartition problem to the Tucker packing problem.

Theorem 3.6. *The Tucker packing problem is* NP-hard.

Proof. Let $T, w_1, \ldots, w_T$ be an instance of EQUIPARTITION where $w_n \geq 2$ for all $n \in [T]$. Notice that the assumption $w_n \geq 2$ is without loss of generality because we can multiply all of the values $w_1, \ldots, w_T$ by two.

Let $M = \sum_{n \in [T]} w_n$ be the sum of all weights, and let $N \geq T$ be the smallest integer such that $2^{N-T/2} > 4(N-T)+3M/2$. Now we construct an instance of the Tucker packing problem. For each $n \in [T]$, let:

- $I_n = w_n$

- $a_1^{(n)} = 2M$
- $a_2^{(n)} = M + w_n$
- $a_{i_n}^{(n)} = 0$, for all $i_n \in [I_n] \setminus \{1, 2\}$

Next, for each $n \in [N] \setminus [T]$, let:

- $I_n = 2$
- $a_1^{(n)} = a_2^{(n)} = 2M$

Finally, set the budget to be $c = 2^{N-T/2} + 4(N-T) + 3M/2$.

First, notice that this is a valid instance of the Tucker packing problem since $a_1^{(n)} \geq a_2^{(n)} \geq \cdots \geq a_{I_n}^{(n)}$ for all $n \in [N]$. Further, since $N = O(T \cdot \log_2(3M/2))$ and $a_{i_n}^{(n)} = 0$ for $i_n \in [I_n] \setminus \{1, 2\}$, the size of the description of this problem is polynomial in the size of the description of the corresponding EQUIPARTITION problem.

Now we consider a decision version of this Tucker packing problem in which we are asked to determine whether there exists a feasible solution $(R_1, \ldots, R_N)$ such that

$$\sum_{n=1}^{N} \sum_{i_n=1}^{R_n} a_{i_n}^{(n)} \geq M(4N - 3T/2) + M/2. \tag{21}$$

We show that a positive answer to the decision version of the Tucker packing problem in (21) implies a positive answer to the EQUIPARTITION problem and vice versa.

Suppose the answer to the decision version of the Tucker packing problem is YES, and $\mathbf{r}^*$ is an optimal solution such that

$$\sum_{n=1}^{N} \sum_{i_n=1}^{R_n^*} a_{i_n}^{(n)} \geq M(4N - 3T/2) + M/2 \quad \text{and} \quad \prod_{n=1}^{N} R_n^* + \sum_{n=1}^{N} I_n R_n^* \leq c.$$

Since

$$c = 2^{N-T/2} + 4(N-T) + 3M/2 < 2 \cdot 2^{N-T/2} = 2^{N-T/2+1},$$

there are at most $N - T/2$ values of R_n^* such that $R_n^* \geq 2$. Further, since $a_{i_n}^{(n)} = 0$ for all $i_n \geq 3$, we never have $R_n^* > 2$ in a minimal optimal solution. It follows that $R_n^* \in \{1, 2\}$ for all $n \in [N]$, and

$$\prod_{n=1}^{N} R_n^* \leq 2^{N-T/2}.$$

Next, we establish the structure of an optimal solution to this Tucker packing instance. Observe that $\widehat{\mathbf{r}} = (\widehat{R}_1, \ldots, \widehat{R}_N)$ with $\widehat{R}_1 = \cdots = \widehat{R}_T = 1$ and $\widehat{R}_{T+1} = \cdots = \widehat{R}_N = 2$ is a feasible solution $\mathbf{r}$ that achieves an objective value of $M(4N - 2T)$. Now consider any feasible solution in which there exists $i \in [T]$ and $j \in [N] \setminus [T]$ such that $R_i = 2$ and $R_j = 1$. If we switch the values of R_i and R_j, then the cost decreases by $w_i - 2 \geq 0$ (i.e., the solution is still feasible), and the objective value increases by $M - w_i > 0$. Therefore, since $\widehat{\mathbf{r}}$ is feasible, in an optimal solution we have $R_n = 2$ for all $n \in [N] \setminus [T]$ and at most $T/2$ of the R_n's for $n \in [T]$ are equal to two.

Let $S = \{i \in [T] : R_i^* = 2\}$. Then by construction we have

$$\sum_{n \in S} a_2^{(n)} = \sum_{n \in S} (M + w_n) = M|S| + \sum_{n \in S} w_n < M(|S| + 1).$$

Moreover, since the answer to the decision problem is YES and in an optimal solution we have $R_n^* = 2$ for all $n \in [N] \setminus [T]$, it follows that

$$\begin{aligned}\sum_{n\in S} a_2^{(n)} &= f(\mathbf{r}^*) - \sum_{n=1}^{N} a_1^{(n)} - \sum_{n=T+1}^{N} a_2^{(n)} \\ &= f(\mathbf{r}^*) - 2NM - 2(N-T)M \\ &\geq M(4N - 3T/2) + M/2 - 2NM - 2(N-T)M \\ &= MT/2 + M/2.\end{aligned} \tag{22}$$

Therefore,

$$M(|S| + 1) > MT/2 + M/2,$$

which implies $|S| > T/2 - 1/2$, so $|S| \geq T/2$ since $|S|$ and $T/2$ are integers. Using the characterization above about an optimal solution together with the fact that the budget is strictly less than $2^{N-T/2+1}$ gives us $|S| \leq T/2$. Thus, a YES to the decision problem implies that $|S| = T/2$, which further implies $\prod_{n=1}^{N} R_n^* = 2^{N-T/2}$.

It then follows from our choice of budget c that

$$\sum_{n=1}^{N} I_n R_n^* \leq c - 2^{N-T/2} = 4(N-T) + 3M/2,$$

which then by the definition of I_n implies that

$$\left(\sum_{n=1}^{T} w_n R_n^* + \sum_{n=T+1}^{N} 2R_n^*\right) = \left(M + \sum_{n\in S} w_n\right) + 4(N-T) \leq 4(N-T) + 3M/2 \implies \sum_{n\in S} w_n \leq M/2.$$

Furthermore, using (22), the definition of the $a_{i_n}^{(n)}$'s, and the fact that $|S| = T/2$, we have

$$\sum_{n\in S} a_2^{(n)} = \sum_{n\in S}(M + w_n) = |S|M + \sum_{n\in S} w_n \geq MT/2 + M/2 \implies \sum_{n\in S} w_n \geq M/2.$$

Putting everything together, we get $\sum_{n\in S} w_n = M/2$. Thus, S is a solution for the EQUIPARTITION problem.

Now suppose the answer to the EQUIPARTITION problem is YES. Let $S \subseteq [T]$ such that $|S| = T/2$ and $\sum_{n\in S} w_n = M/2$. Construct $\mathbf{r}^*$ as follows: For each $n \in S \cup ([N] \setminus [T])$, set $R_n^* = 2$; for each $n \in [T] \setminus S$, set $R_n^* = 1$.

Then, by the definitions of I_n and $a_{i_n}^{(n)}$ above, we have

$$\sum_{n=1}^{N}\sum_{i_n=1}^{R_n^*} a_{i_n}^{(n)} \geq M(4N - 3T/2) + M/2 \quad \text{and} \quad \prod_{n=1}^{N} R_n^* + \sum_{n=1}^{N} I_n R_n^* \leq c,$$

which completes the proof. □

A.4. Translating between approximate maximization and minimization

We prove an additive-error guarantee that shows how a $(1-\varepsilon')$-approximate solution to the Tucker packing problem, i.e., a core shape $\mathbf{r} \in [I_1] \times \cdots \times [I_N]$, can lead to an increase in the surrogate loss objective.

Lemma A.8. *Let $\mathbf{r} \in [I_1] \times \cdots \times [I_N]$ be any core shape that achieves a $(1-\varepsilon/N)$-approximation to the Tucker packing problem. Then, we have* $\mathrm{RRE}(\mathcal{X}, \mathbf{r}) \leq N \cdot \mathrm{RRE}(\mathcal{X}, \mathbf{r}^*) + \varepsilon$, *where* $\mathrm{RRE}(\mathcal{X}, \mathbf{r}) := L(\mathcal{X}, \mathbf{r})/\|\mathcal{X}\|_{\mathrm{F}}^2$.

Proof. Let $\varepsilon' = \varepsilon/N$ and $\widetilde{\mathbf{r}}^*$ be the optimal shape for the surrogate loss $\widetilde{L}$. If $\mathbf{r} = (R_1, \ldots, R_N)$ is a $(1-\varepsilon')$-approximation

to the Tucker packing problem, it follows that

$$\begin{aligned}\frac{\widetilde{L}(\mathcal{X},\mathbf{r})}{\|\mathcal{X}\|_{\mathrm{F}}^2} &= \frac{N\|\mathcal{X}\|_{\mathrm{F}}^2 - \sum_{n=1}^{N}\sum_{i_n=1}^{R_n}\left(\sigma_{i_n}^{(n)}\right)^2}{\|\mathcal{X}\|_{\mathrm{F}}^2} \\ &\leq \frac{N\|\mathcal{X}\|_{\mathrm{F}}^2 - (1-\varepsilon')\sum_{n=1}^{N}\sum_{i_n=1}^{\widetilde{R}_n^*}\left(\sigma_{i_n}^{(n)}\right)^2}{\|\mathcal{X}\|_{\mathrm{F}}^2} \\ &= \frac{\widetilde{L}(\mathcal{X},\widetilde{\mathbf{r}}^*)}{\|\mathcal{X}\|_{\mathrm{F}}^2} + \varepsilon'\left(N - \frac{\widetilde{L}(\mathcal{X},\widetilde{\mathbf{r}}^*)}{\|\mathcal{X}\|_{\mathrm{F}}^2}\right) \\ &\leq \frac{\widetilde{L}(\mathcal{X},\widetilde{\mathbf{r}}^*)}{\|\mathcal{X}\|_{\mathrm{F}}^2} + \varepsilon.\end{aligned}$$

Theorem 3.2 gives us $L(\mathcal{X},\mathbf{r}) \leq \widetilde{L}(\mathcal{X},\mathbf{r}) \leq N \cdot L(\mathcal{X},\mathbf{r})$. By definition $\widetilde{L}(\mathcal{X},\widetilde{\mathbf{r}}^*) \leq \widetilde{L}(\mathcal{X},\mathbf{r}^*)$, so we have

$$\mathrm{RRE}(\mathcal{X},\mathbf{r}) = \frac{L(\mathcal{X},\mathbf{r})}{\|\mathcal{X}\|_{\mathrm{F}}^2} \leq \frac{\widetilde{L}(\mathcal{X},\mathbf{r})}{\|\mathcal{X}\|_{\mathrm{F}}^2} \leq \frac{\widetilde{L}(\mathcal{X},\widetilde{\mathbf{r}}^*)}{\|\mathcal{X}\|_{\mathrm{F}}^2} + \varepsilon \leq \frac{\widetilde{L}(\mathcal{X},\mathbf{r}^*)}{\|\mathcal{X}\|_{\mathrm{F}}^2} + \varepsilon \leq N \cdot \frac{L(\mathcal{X},\mathbf{r}^*)}{\|\mathcal{X}\|_{\mathrm{F}}^2} + \varepsilon = N \cdot \mathrm{RRE}(\mathcal{X},\mathbf{r}^*) + \varepsilon,$$

as desired. □

B. Missing analysis from Section 4

B.1. Proof of Lemma 4.3

Lemma B.1. *Let $0 < \varepsilon \leq 1$ and $F \subseteq [I_1] \times \cdots \times [I_N]$ be downwards closed. For each $n \in [N]$, define*

$$S_n^{(\varepsilon)} = \left\{\lceil (1+\varepsilon)^k \rceil : k \in \mathbb{Z}_{\geq 0}, \lceil (1+\varepsilon)^k \rceil \leq I_n\right\}.$$

Let $\mathbf{r}^*$ be an optimal solution to the generalized problem

$$\begin{aligned}&\text{maximize} && \sum_{n=1}^{N}\sum_{i_n=1}^{R_n} a_{i_n}^{(n)} \\ &\text{subject to} && (R_1,\ldots,R_N) \in F\end{aligned} \tag{9}$$

and let $\mathbf{r}^{(\varepsilon)} = (R_1^{(\varepsilon)},\ldots,R_N^{(\varepsilon)})$ be an optimal solution to

$$\begin{aligned}&\text{maximize} && \sum_{n=1}^{N}\sum_{i_n=1}^{R_n} a_{i_n}^{(n)} \\ &\text{subject to} && (R_1,\ldots,R_N) \in (S_1^{(\varepsilon)} \times \cdots \times S_N^{(\varepsilon)}) \cap F\end{aligned} \tag{10}$$

Then, $f(\mathbf{r}^{(\varepsilon)}) \geq (1+\varepsilon)^{-1} f(\mathbf{r}^*)$. Further, there is an algorithm that finds an optimal solution of (10) with running time $O(\sum_{n=1}^{N} I_n + \varepsilon^{-N}\prod_{n=1}^{N}(1+\log_2(I_n)))$.

Proof. Let $k_n \geq 0$ be the largest integer such that $(1+\varepsilon)^{k_n} \leq R_n^*$ for each $n \in [N]$. Further, let

$$\widehat{R}_n = \lceil (1+\varepsilon)^{k_n} \rceil.$$

Since R_n^* is an integer, we know that $\widehat{R}_n \leq R_n^*$. Therefore, because F is downwards closed, $\widehat{\mathbf{r}} = (\widehat{R}_1,\ldots,\widehat{R}_N)$ is a feasible solution to (10). It follows that

$$f(\widehat{\mathbf{r}}) \leq f(\mathbf{r}^{(\varepsilon)}). \tag{23}$$

Now we will show that $f(\mathbf{r}^*) \leq (1+\varepsilon) f(\widehat{\mathbf{r}})$. Since $a_1^{(n)} \geq \cdots \geq a_{I_n}^{(n)} \geq 0$ for all $n \in [N]$, we have

$$(1+\varepsilon) \sum_{i_n=1}^{\widehat{R}_n} a_{i_n}^{(n)} \geq \sum_{i_n=1}^{\lfloor (1+\varepsilon)\widehat{R}_n \rfloor} a_{i_n}^{(n)}. \tag{24}$$

By the definition of k_n, it follows that

$$\begin{aligned}(1+\varepsilon)\widehat{R}_n &= (1+\varepsilon)\left\lceil (1+\varepsilon)^{k_n} \right\rceil \\ &\geq (1+\varepsilon)^{k_n+1} \\ &> R_n^*.\end{aligned}$$

Since R_n^* is an integer, we have $\lfloor (1+\varepsilon)\widehat{R}_n \rfloor \geq R_n^*$. Therefore, using Equation (25) we have

$$(1+\varepsilon) \sum_{i_n=1}^{\widehat{R}_n} a_{i_n}^{(n)} \geq \sum_{i_n=1}^{R_n^*} a_{i_n}^{(n)}. \tag{25}$$

Finally, summing over $n \in [N]$ and using Equation (23) gives us

$$(1+\varepsilon) f(\widehat{\mathbf{r}}) \geq f(\mathbf{r}^*) \implies f(\mathbf{r}^{(\varepsilon)}) \geq (1+\varepsilon)^{-1} f(\mathbf{r}^*).$$

Algorithm. Now we design and analyze a simple algorithm to solve the grid-search problem in (10). First observe that

$$|S_n^{(\varepsilon)}| = 1 + \left\lfloor \log_{1+\varepsilon}(I_n) \right\rfloor = 1 + \left\lfloor \frac{\log_2(I_n)}{\log_2(1+\varepsilon)} \right\rfloor.$$

It follows that the number of feasible solutions for Problem (10) is

$$O\left(\prod_{n=1}^{N} \left(1 + \frac{\log_2(I_n)}{\log_2(1+\varepsilon)} \right) \right) = O\left(\frac{1}{\log_2^N(1+\varepsilon)} \prod_{n=1}^{N} (1 + \log_2(I_n)) \right),$$

since $\log_2(1+\varepsilon) \leq 1$ for $\varepsilon \leq 1$. Further, observing that $\log_2(1+\varepsilon) \geq \varepsilon$ for $0 < \varepsilon \leq 1$ implies a bound of

$$O\left(\frac{1}{\varepsilon^N} \prod_{n=1}^{n} (1 + \log_2(I_n)) \right)$$

on the number of feasible solutions.

After computing the prefix sums for elements of $S_n^{(\varepsilon)}$'s, we can iterate over the elements of $S_1^{(\varepsilon)} \times \cdots \times S_N^{(\varepsilon)}$ in the lexicographical order, check for feasibility, and compute the objective value $f(\mathbf{r})$ for any candidate solution in amortized time $O(1)$. Finally, note that the prefix sums for elements of $S_n^{(\varepsilon)}$ can be computed in $O(I_n)$ time. □

Rationale for two different types of budget splits in Algorithm 2. Instead of considering two different budget splitting methods (i.e., with c_{factor} and c_{core}), one could just consider the budget splits $c_{\text{factor}} \in [\sum_{n=1}^{N} I_n^2]$ over problems of the form:

$$\begin{aligned} \text{maximize} \quad & f(\mathbf{r}) \\ \text{subject to} \quad & \textstyle\prod_{n=1}^{N} R_n \leq c - c_{\text{factor}} \\ & \textstyle\sum_{n=1}^{N} I_n R_n \leq c_{\text{factor}} \\ & R_n \in [I_n] \qquad \forall n \in [N] \end{aligned}$$

This approach also gives a PTAS. The running time, however, is $O((\sum_{n=1}^{N} I_n^2)(\sum_{n=1}^{N} I_n)^{O(1/\varepsilon)})$. In contrast, the running time of Algorithm 2 is $O((\log_{1+\varepsilon}(c) + \lceil 1/\varepsilon \rceil \sum_{n=1}^{N} I_n)(\sum_{n=1}^{N} I_n)^{O(1/\varepsilon)})$. Note that $c \leq \sum_{n=1}^{N} I_n^2 + \prod_{n=1}^{N} I_n$. Therefore, $\log_{1+\varepsilon}(c) = O(\frac{1}{\varepsilon} \sum_{n=1}^{N} \log_2(I_n)) \ll \lceil 1/\varepsilon \rceil \sum_{n=1}^{N} I_n$, where the first equality follows from the concavity of the logarithm function. Thus, our running time is $O((\lceil 1/\varepsilon \rceil \sum_{n=1}^{N} I_n)(\sum_{n=1}^{N} I_n)^{O(1/\varepsilon)})$, which is always smaller than $O((\sum_{n=1}^{N} I_n^2)(\sum_{n=1}^{N} I_n)^{O(1/\varepsilon)})$.

C. Tree tensor networks

Remark C.1. Although the tree tensor network is considered for general Hilbert spaces and has been defined in full generality using the tensor network notations (see, e.g., Hackbusch (2019, Chapter 11) and Krämer (2020, Chapter 3)), we consider finite-dimensional Euclidean spaces to keep our notation simple.

Remark C.2. Theorem 5.6 is implied by Lemma 4.3.

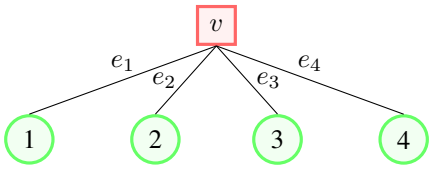

Figure 3. Tree tensor network corresponding to a Tucker decomposition.

Example (Tucker decomposition). Tucker decomposition corresponds to a tree tensor of depth one. For example, the tree tensor in Figure 3 consists of matrices $\mathbf{A}_1 \in \mathbb{R}^{I_1 \times R_{e_1}}$, $\mathbf{A}_2 \in \mathbb{R}^{I_2 \times R_{e_2}}$, $\mathbf{A}_3 \in \mathbb{R}^{I_3 \times R_{e_3}}$, $\mathbf{A}_4 \in \mathbb{R}^{I_4 \times R_{e_4}}$, and tensor $\mathcal{T}_v \in \mathbb{R}^{R_{e_1} \times R_{e_2} \times R_{e_3} \times R_{e_4}}$. The corresponding reconstruction is

$$\widehat{\mathcal{X}} = \mathcal{T}_v \times_{e_4} \mathbf{A}_4 \times_{e_3} \mathbf{A}_3 \times_{e_2} \mathbf{A}_2 \times_{e_1} \mathbf{A}_1.$$

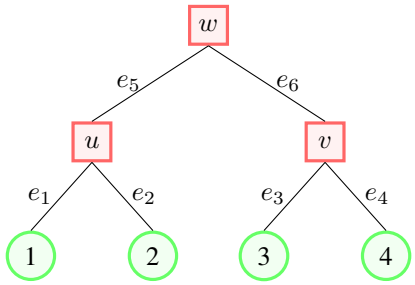

Figure 4. Tree tensor network example corresponding to a hierarchical Tucker decomposition.

Example (Hierarchical Tucker decomposition). To to better understand Definition 5.1, we give an example for a tensor of order 4. Consider the tree illustrated in Figure 4. This tree tensor network corresponds to matrices $\mathbf{A}_1 \in \mathbb{R}^{I_1 \times R_{e_1}}$, $\mathbf{A}_2 \in \mathbb{R}^{I_2 \times R_{e_2}}$, $\mathbf{A}_3 \in \mathbb{R}^{I_3 \times R_{e_3}}$, $\mathbf{A}_4 \in \mathbb{R}^{I_4 \times R_{e_4}}$, and tensors $\mathcal{T}_u \in \mathbb{R}^{R_{e_1} \times R_{e_2} \times R_{e_5}}$, $\mathcal{T}_v \in \mathbb{R}^{R_{e_3} \times R_{e_4} \times R_{e_6}}$, $\mathcal{T}_w \in \mathbb{R}^{R_{e_5} \times R_{e_6}}$. The corresponding reconstruction is

$$\widehat{\mathcal{X}} = \mathcal{T}_w \times_{e_6} \mathcal{T}_v \times_{e_5} \mathcal{T}_u \times_{e_4} \mathbf{A}_4 \times_{e_3} \mathbf{A}_3 \times_{e_2} \mathbf{A}_2 \times_{e_1} \mathbf{A}_1.$$

D. Experiments

We provide short descriptions about each of the tensor datasets used in the core shape experiments in Section 6, which gives some insight into why the algorithms build the core shapes they do.

Cardiac MRI. $256 \times 256 \times 14 \times 20$ tensor whose elements are MRI measurements indexed by (x, y, z, t), where (x, y, z) is a point in space and t corresponds to time.

Hyperspectral. $1024 \times 1344 \times 33$ tensor of time-lapse hyperspectral radiance images of a nature scene that is undergoing illumination changes (Nascimento et al., 2016).

VicRoads. $1084 \times 2033 \times 96$ tensor containing 2033 days of traffic volume data from Melbourne and its surrounding suburbs. This data comes from a network of 1084 road sensors measured in 15 minute intervals (Schimbinschi et al., 2015).

COIL-100. $7200 \times 128 \times 128 \times 3$ tensor containing 7200 colored photos of 100 different objects (72 images per object) taken at 5-degree rotations (Nene et al., 1996). This is a widely-used dataset in the computer vision research community.