Minimizing Low-Rank Models of High-Order Tensors: Hardness, Span, Tight Relaxation, and Applications

Nicholas D. Sidiropoulos, *Fellow, IEEE*, Paris A. Karakasis, *Student Member, IEEE,* and Aritra Konar, *Member, IEEE*

***Abstract*—We consider the problem of finding the smallest or largest entry of a tensor of order N that is specified via its rank decomposition. Stated in a different way, we are given N sets of R-dimensional vectors and we wish to select one vector from each set such that the sum of the Hadamard product of the selected vectors is minimized or maximized. We show that this fundamental tensor problem is NP-hard for any tensor rank higher than one, and polynomial-time solvable in the rank-one case. We also propose a continuous relaxation and prove that it is tight for any rank. For low-enough ranks, the proposed continuous reformulation is amenable to low-complexity gradient-based optimization, and we propose a suite of gradient-based optimization algorithms drawing from projected gradient descent, Frank-Wolfe, or explicit parametrization of the relaxed constraints. We also show that our core results remain valid no matter what kind of polyadic tensor model is used to represent the tensor of interest, including Tucker, HOSVD/MLSVD, tensor train, or tensor ring. Next, we consider the class of problems that can be posed as special instances of the problem of interest. We show that this class includes the partition problem (and thus all NP-complete problems via polynomial-time transformation), integer least squares, integer linear programming, integer quadratic programming, sign retrieval (a special kind of mixed integer programming / restricted version of phase retrieval), and maximum likelihood decoding of parity check codes. We demonstrate promising experimental results on a number of hard problems, including state-of-art performance in decoding low density parity check codes and general parity check codes.**

***Index Terms*—Tensors, rank decomposition, inner product, algorithms, complexity theory, NP-hard problems, error control coding, maximum likelihood decoding, parity check codes, belief propagation, under/over-determined linear equations, Galois fields, sign retrieval.**

Manuscript received 29 May 2023; revised 19 October 2023; accepted 21 November 2023. Date of publication 4 December 2023; date of current version 14 December 2023. This research was not supported by any sponsor. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Yuxin Chen. *(Corresponding author: Nicholas D. Sidiropoulos.)*

Nicholas D. Sidiropoulos and Paris A. Karakasis are with the Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22904 USA (e-mail: nikos@virginia.edu; karakasis@virginia.edu).

Aritra Konar is with the Department of Electrical Engineering, KU Leuven, B-3001 Leuven, Belgium (e-mail: aritra.konar@kuleuven.be).

This article has supplementary downloadable material available at https://doi.org/TSP.2023.3338062, provided by the authors.

Digital Object Identifier 10.1109/TSP.2023.3338062

I. Introduction

FINDING the smallest or largest entry of a matrix or tensor specified via its low-rank factors is a fundamental problem with numerous applications in science and engineering. The matrix version of the problem has received some attention [1], [2], motivated by applications in graph mining (e.g., most significant missing link prediction, nodes that share many neighbors), text/speech/audio similarity search and retrieval (e.g., using text embeddings), and recommender systems (e.g., finding the best item-context combination to recommend to a given user).

The tensor version of the problem is considerably more powerful, as it allows going beyond bipartite matching / prediction which can be broadly useful in knowledge discovery. As an example, given embeddings of patients, conditions, drugs, clinical trials, therapies, we may be interested in finding the best match, as measured by a function of the pointwise product of the respective embeddings. Tensors can also be used to model high-dimensional probability distributions, wherein low-rank tensor models are used to break the curse of dimensionality while allowing easy marginalization and computation of conditional probabilities, which are crucial for prediction [3]. In this context, finding the largest element corresponds to finding the mode/maximum likelihood or maximum *a-posteriori* estimate of missing variables, and it is not unusual to encounter very high-order probability tensors indexed by hundreds of categorical input variables. Despite the obvious importance of the tensor version of this problem, there is scant literature about it - we found only [4], which essentially extends the approach in [2] to the low-order tensor case.

A. Prior Work

The matrix case was considered in [1], which proposed using a power-method type algorithm that works directly with the low-rank factors. In [2], the authors developed a randomized "diamond sampling" approach for computing the maximum element of the product of two matrices (which could be, e.g., two low-rank factors) in what they called the MAD (Maximum All-pairs Dot-product) search. Their algorithm comes with probabilistic performance guarantees and was demonstrated to work well in practice using a variety of datasets. As already mentioned above, [4] extends the randomized diamond sampling approach in [2] to "star sampling" for the tensor case. These randomized algorithms do not scale well to high-order tensors, owing to the curse of dimensionality.

A very different approach for the higher-order tensor version of the problem has been proposed in the computational physics/numerical algebra literature [5]. The basic idea of [5] (see also references therein) is as follows. By vectorizing the tensor and putting the resulting (very long) vector on the diagonal of a matrix, the tensor elements become eigenvalues corresponding to coordinate basis eigenvectors. This suggests that the maximum element of the tensor can be computed through a power iteration involving this very large matrix. Of course power iterations implemented naively will have prohibitive complexity (as tensor vectorization produces a very long vector and thus a huge matrix). The idea is therefore to employ a tensor factorization model to ease the matrix-vector multiplication of the diagonal matrix and the interim solution vector. This multiplication can be computed by summing the elements of the pointwise (Hadamard) product of the two vectors – the vectorized tensor on the matrix diagonal and the interim solution vector. This product can be efficiently computed for various tensor models, but the pointwise multiplication of two tensors of rank R generally has rank up to R^2, necessitating conversion back to a rank-R tensor. The natural way to do this is via rank-R least-squares approximation of the higher-rank product. This is a hard and generally ill-posed computational problem, because the best approximation may not even exist. Thus the method cannot be used with a tensor rank decomposition model (known as *Canonical Polyadic Decomposition* or *CPD*). The pointwise multiplication and least-squares approximation are easier with a so-called *tensor train* (TT) model. The Hadamard product of two TT models is another TT model whose wagon (factor) matrices are the Kronecker products of the respective factor matrices of the two TT models. This implies that pointwise multiplication of two TT models has complexity of order NIR^4 for an I^N tensor of TT rank R for each wagon. Moreover, rank reduction for TT models is SVD-based and has complexity order NIR^3. Hence, reducing the rank of the pointwise multiplication (which is R^2 due to the Kronecker product) induces a complexity of order $NI(R^2)^3 = NIR^6$, and as a result, only small TT ranks can be supported in practice. Another limitation of [5] is that it is hard to pin down if and when the iteration will converge, because of the rank reduction step. The numerical experiments in [5] are interesting but limited in terms of exploring the accuracy-speedup trade-offs that can be achieved.

Unlike [5], our emphasis is on the tensor rank decomposition (CPD) model, in good part because many of the problems that we consider herein admit *closed-form* tensor rank decomposition formulations, thus bypassing the need for computational model fitting and/or rank reduction. Another issue is that, as we show for the first time in this paper, the core problem is NP-hard even with a TT model, so there is no intrinsic computational benefit to using one tensor decomposition model over the other.

B. Contributions

Our contributions relative to the prior art are as follows:

- We focus on the higher-order tensor version of the problem, and we analyze its computational complexity. We show that the problem is easy when the tensor rank is equal to one, but NP-hard otherwise – even if the rank is as small as two.
- We consider optimization-based (as opposed to algebraic or randomized) algorithms that can be used to compute good approximate solutions in the high-order tensor case. We provide an equivalent "fluid" (continuous-variable) reformulation of what is inherently a multi-way selection problem, and a suite of relatively lightweight gradient-based approximation algorithms. The continuous reformulation is derived by showing that a certain continuous relaxation involving a probability distribution instead of hard selection for each mode is actually tight. This is true for any rank, i.e., even for full-rank (unstructured) tensor models. The proposed algorithms take advantage of various optimization frameworks, from projected gradient descent to Franke-Wolfe and various ways of explicitly parametrizing the probability simplex constraints. For low-enough ranks, the associated gradient computations are computationally lightweight, even for very high-order tensors.
- We show that our main results remain valid no matter what kind of polyadic tensor model is used to represent the tensor of interest, including Tucker, HOSVD/MLSVD, tensor train, or tensor ring.
- We explore the "span" of the core problem considered, i.e., the class of problems that can be posed as special instances of computing the minimum or maximum element of a tensor from its rank decomposition. We show that this class includes the partition problem (and thus all NP-complete problems via polynomial-time transformation), as well as integer least squares, integer linear programming, integer quadratic programming, sign retrieval (a special kind of mixed integer programming / restricted version of phase retrieval), maximum likelihood decoding of parity check codes, and finding the least inconsistent solution of an overdetermined system of linear equations over a Galois field.
- Finally, we demonstrate promising experimental results on a number of hard problems, including better than state-of-art performance in decoding both low density parity check codes and general parity check codes, and sign retrieval.

II. Preliminaries

Consider a matrix $\mathbf{M} = \mathbf{A}_1\mathbf{A}_2^T$, where $\mathbf{M}$ is $I_1 \times I_2$, $\mathbf{A}_1$ is $I_1 \times R$ and $\mathbf{A}_2$ is $I_2 \times R$, with $R \leq \min(I_1, I_2)$ ($R < \min(I_1, I_2)$ if $\mathbf{M}$ is not full-rank). Three other ways to write the same relation are $\mathbf{M} = \sum_{r=1}^{R} \mathbf{A}_1(:,r)\mathbf{A}_2(:,r)^T$, where $\mathbf{A}(:,r)$ stands for the r-th column of $\mathbf{A}$; $\mathbf{M} = \sum_{r=1}^{R} \mathbf{A}_1(:,r) \circ \mathbf{A}_2(:,r)$, where $\circ$ stands for the vector outer product; and $\mathbf{M}(i_1, i_2) = \sum_{r=1}^{R} \mathbf{A}_1(i_1, r)\mathbf{A}_2(i_2, r)$, where $\mathbf{M}(i_1, i_2)$ denotes an element of matrix $\mathbf{M}$, with obvious notation.

From the latter, notice that element (i_1, i_2) of matrix $\mathbf{M}$ is the inner product of two row vectors, namely row $\mathbf{A}_1(i_1,:)$ of matrix $\mathbf{A}_1$, and row $\mathbf{A}_2(i_2,:)$ of matrix $\mathbf{A}_2$.

We are interested in finding the smallest or largest element of matrix $\mathbf{M}$ from the factors $\mathbf{A}_1$ and $\mathbf{A}_2$. From the latter relation, we see that seeking the smallest (or largest) element of $\mathbf{M}$ corresponds to seeking a pair of row vectors, one drawn from $\mathbf{A}_1$ and the other from $\mathbf{A}_2$, which have the smallest (or largest) possible inner product. One obvious way to do this is to generate all inner products, i.e., the elements of the product $\mathbf{A}_1\mathbf{A}_2^T$, and find the smallest or largest. This entails complexity $I_1 I_2 R$, which can be high, even when R is small – e.g., consider the case where I_1 and I_2 are in the order of 10^5 or more. Is there a way to avoid performing order $I_1 I_2$ computations? When $R = \min(I_1, I_2)$, such complexity is unavoidable, because then the elements of $\mathbf{M}$ can be freely chosen independently of each other (e.g., from an i.i.d. Gaussian distribution). When R is small however, there seems to be hope.

Generalizing, consider a tensor $\mathcal{T}$ of order N and size $I_1 \times I_2 \times \cdots \times I_N$, for which we are given a low-rank factorization, namely

$$\mathcal{T} = \sum_{r=1}^{R} \mathbf{A}_1(:,r) \circ \mathbf{A}_2(:,r) \circ \cdots \circ \mathbf{A}_N(:,r),$$

where $\mathbf{A}_n$ is $I_n \times R$, and R (when minimal for this decomposition to hold) is the rank of $\mathcal{T}$, in which case the above is known as the *canonical polyadic decomposition* (CPD) of $\mathcal{T}$ [6]. When the elements of $\mathcal{T}$ are non-negative and the elements of the factor matrices $\mathbf{A}_n$ are constrained to be non-negative, then the smallest R for which such a decomposition exists is called the non-negative rank of $\mathcal{T}$, which can be higher than the (unconstrained) rank of $\mathcal{T}$. In the sequel, we do not assume that R is minimal; any polyadic decomposition will do for our purposes. In scalar form, we get that

$$\mathcal{T}(i_1, i_2, \cdots, i_N) = \sum_{r=1}^{R} \mathbf{A}_1(i_1, r)\mathbf{A}_2(i_2, r) \cdots \mathbf{A}_N(i_N, r),$$

which reveals that every element of $\mathcal{T}$ is an inner product of N row vectors, one drawn from each of the matrices $\mathbf{A}_1, \cdots, \mathbf{A}_N$. An alternative way to write this is using the Hadamard (element-wise) product, denoted by $*$, as

$$\mathcal{T}(i_1, i_2, \cdots, i_N) = (\mathbf{A}_1(i_1,:) * \mathbf{A}_2(i_2,:) * \cdots * \mathbf{A}_N(i_N,:))\,\mathbf{1},$$

where $\mathbf{1}$ is the $R \times 1$ vector of all 1's, used to sum up the R components of the Hadamard product. We see that finding the smallest or largest element of $\mathcal{T}$ is tantamount to finding the smallest or largest inner product of N row vectors, one drawn from each $\mathbf{A}_n$ matrix. There is an obvious way to do this, by generating all $I_1 \times I_2 \times \cdots \times I_N$ elements of $\mathcal{T}$, at a complexity cost of $R(N-1)$ flops each, but this exhaustive search is much worse than it is in the matrix case. It quickly becomes intractable even for moderate N and I_N, e.g., $N = 20$ with $I_n = 10$, $\forall n \in \{1, \cdots, N\}$. Is there a more efficient way to do this?

III. Theory

When $R = 1$ and all the $\mathbf{A}_n$ matrices (column vectors $\mathbf{a}_n$ in this case) only have non-negative entries, it is easy to see that the smallest element of $\mathcal{T}$ is $\mathcal{T}(\check{i}_1, \cdots, \check{i}_N)$ with $\check{i}_n \in \arg\min_{i_n} \mathbf{a}_n(i_n)$, and likewise the largest is $\mathcal{T}(\hat{i}_1, \cdots, \hat{i}_N)$ with $\hat{i}_n \in \arg\max_{i_n} \mathbf{a}_n(i_n)$. For $R > 1$ however, the answer is no, in the worst case – even if $R = 2$ and the elements of all the $\mathbf{A}_n$ matrices are non-negative. We have the following result.

Theorem 1: Finding the smallest element of a tensor from its rank factorization is NP-hard, even when the tensor rank is as small as $R = 2$ and the rank-one factors are all non-negative. This means that for large N, the worst-case complexity is exponential in N.

Proof: We show that an arbitrary instance of the *partition problem*, which is known to be NP-complete [7], can be converted to a specific instance of the decision version of the problem of interest. This means that if we could efficiently solve any instance of the problem of interest, we would be in position to efficiently solve an arbitrary instance of the partition problem, which is not possible according to the current scientific consensus, unless P=NP.

Recall the *partition problem*, which is as follows. We are given N positive integers $w_1, \cdots, w_N$ (repetitions are allowed), and we wish to know whether there is a subset of indices $\mathcal{S}$, such that

$$\sum_{n \in \mathcal{S}} w_n = \sum_{n \notin \mathcal{S}} w_n.$$

Consider binary variables $\{i_n \in \{0,1\}\}_{n=1}^N$, with i_n designating whether $n \in \mathcal{S}$ or not, i.e., $i_n = 1$ if $n \in \mathcal{S}$, else $i_n = 0$. Deciding whether a suitable $\mathcal{S}$ exists is equivalent to deciding whether

$$\min_{\{i_n \in \{0,1\}\}_{n=1}^N} \left(\sum_{n=1}^{N} w_n i_n - \sum_{n=1}^{N} w_n (1 - i_n) \right)^2 \overset{?}{>} 0.$$

We will instead use a reformulation of the above which is more conducive for our purposes, namely

$$\min_{\{i_n \in \{0,1\}\}_{n=1}^N} \left(e^{\sum_{n=1}^N w_n i_n} - e^{\sum_{n=1}^N w_n (1-i_n)} \right)^2 \overset{?}{>} 0.$$

Expanding the square and noting that the product term is a constant, we obtain

$$\min_{\{i_n \in \{0,1\}\}_{n=1}^N} \left(e^{2\sum_{n=1}^N w_n i_n} + e^{2\sum_{n=1}^N w_n (1-i_n)} \right) \overset{?}{>} 2e^{\sum_{n=1}^N w_n}.$$

Each of the exponential terms is separable, i.e., a rank-one tensor comprising non-negative factors. For example, the first term is

$$e^{\sum_{n=1}^N 2w_n i_n} = e^{2w_1 i_1} e^{2w_2 i_2} \cdots e^{2w_N i_N}.$$

It follows that the function to be minimized is a tensor of size $2 \times \cdots \times 2 = 2^N$ and rank $R = 2$, with the following 2×2 matrix factors

$$\mathbf{A}_n = \begin{bmatrix} 1 & e^{2w_n} \\ e^{2w_n} & 1 \end{bmatrix}, \quad \forall n \in \{1, \cdots, N\}.$$

Hence, the decision version of our problem with $R = 2$ and non-negative factors contains every instance of the partition problem as a special case. It follows that the decision version of our problem is NP-hard, and thus its optimization version is NP-hard as well. □

In fact, the same NP-hardness result carries over to other popular tensor models. These include the so-called Tucker model [8], Multilinear SVD (MLSVD) [9], the Tensor Train (TT) decomposition [10], and the Tensor Ring (TR) decomposition [11].

Theorem 2: Finding the smallest element of a tensor from its Tucker, MLSVD, TT, or TR factorization is NP-hard.

Proof: All these models are outer product decompositions that are related. In particular, a rank two CPD is equivalent to a Tucker model with a diagonal $2 \times 2 \times \cdots \times 2$ core matrix. Thus the closed-form rank-two model of the partition problem is also a low multilinear rank $(= 2)$ Tucker model. We may orthonormalize the loading matrices $\mathbf{A}_n$ if we want to obtain an MLSVD model that features a dense core, obtained by absorbing the inverse transformations into the core. Thus if we could find the minimum element of every Tucker or MLSVD model of multilinear rank $(2, 2, \cdots, 2)$ efficiently, we would be in position to solve all instances of the partition problem.

For the TT decomposition, [12] has shown that for low CPD ranks (smaller than or equal to any of the tensor outer size dimensions), it is possible to explicitly construct an equivalent TT model whose lower-order cores exhibit the same low rank as the original high-order CPD model. It follows that we can express our CPD model of the partition problem as a TT model with core ranks equal to two. Finally, the TR model is a generalization of the TT model. In all cases, if we could find the corresponding minimum element efficiently, we would be in position to efficiently solve every instance of the partition problem. □

Remark 1: The decision version of the partition problem is NP-complete, which means that every other NP-complete problem can be transformed in polynomial time to an instance of our problem of interest, i.e., computing the minimum element of a tensor of non-negative rank two. Theorem 1 thus speaks not only for the hardness, but also for the span of problems that can be considered within our present framework.

Remark 2: Theorem 1 adds a new problem to a list of tensor problems that are known to be NP-hard [13]. Notwithstanding, it is interesting that the problem of finding the smallest or largest entry is NP-hard even for tensors of rank as low as $R = 2$.

Remark 3: Multi-way partitioning [14] is a generalization that seeks to split N numbers to M optimally balanced groups. This can also be cast as finding the minimum element of a tensor from its low-rank factors. Let $S = \sum_{n=1}^{N} w_n$, and $i_n \in \{1, \cdots, M\}$ be a variable indicating to which of the M groups w_n is assigned to. Consider minimizing the loss function

$$\sum_{m=1}^{M} \left(e^{\sum_{n=1}^{N} w_n 1(i_n = m)} - e^{S/M} \right)^2.$$

Expanding the square we obtain two rank-one terms and a constant that is irrelevant to optimization. The overall loss is therefore a tensor of rank $R = 2M$, and it is easy to write out the associated factor matrices, similar to what we did for the basic partition problem.

When the low-rank factors comprise positive and negative values, then even the $R = 1$ case can be challenging, as the minimum or the maximum element of each $\mathbf{a}_n$ can be involved in generating the overall minimum (or maximum), due to the presence of signs. In principle, this seemingly entails explicit or implicit enumeration over 2^N possibilities. (If a zero exists in any $\mathbf{a}_n$, then zero should also be considered as a candidate at the very end.) The good news is that there is structure to this problem: one can invoke the principle of optimality of dynamic programming (DP). The key observation is that

$$\min_{i,j} \boldsymbol{\alpha}_i \boldsymbol{\beta}_j = \min \begin{pmatrix} \min_i \boldsymbol{\alpha}_i \min_j \boldsymbol{\beta}_j \\ \min_i \boldsymbol{\alpha}_i \max_j \boldsymbol{\beta}_j \\ \max_i \boldsymbol{\alpha}_i \min_j \boldsymbol{\beta}_j \\ \max_i \boldsymbol{\alpha}_i \max_j \boldsymbol{\beta}_j \end{pmatrix}$$

and likewise

$$\max_{i,j} \boldsymbol{\alpha}_i \boldsymbol{\beta}_j = \max \begin{pmatrix} \min_i \boldsymbol{\alpha}_i \min_j \boldsymbol{\beta}_j \\ \min_i \boldsymbol{\alpha}_i \max_j \boldsymbol{\beta}_j \\ \max_i \boldsymbol{\alpha}_i \min_j \boldsymbol{\beta}_j \\ \max_i \boldsymbol{\alpha}_i \max_j \boldsymbol{\beta}_j \end{pmatrix}.$$

Thinking of $\boldsymbol{\alpha}$ as the vector having as elements the minimum and the maximum element products over the first k modes and $\boldsymbol{\beta}$ as $\mathbf{a}_{k+1}$, we can compute the minimum and maximum up to the $k + 1$-th mode using the formulas above. It follows that

Proposition 1: Finding the smallest or largest element of a rank-one tensor from $\left\{\mathbf{a}_n \in \mathbb{R}^{I_n \times 1}\right\}_{n=1}^{N}$ can be accomplished via DP at complexity that is linear in N.

When $R > 1$, the problem of finding the minimum (or maximum) element of a CPD-factored tensor can be described as follows. One has N buckets of R-dimensional vectors, with the n-th bucket having I_n vectors – the rows of $\mathbf{A}_n$. Finding the minimum element of the CPD-factored tensor is equivalent to selecting a single row vector from each bucket $\mathbf{A}_n$ such that the inner product of the N resulting vectors is minimized. This is inherently a discrete optimization problem that is NP-hard per Theorem 1. A possible solution strategy is to employ coordinate descent: fixing all indices except i_n, we are looking to minimize or maximize over i_n the inner product

$$\mathbf{A}_n(i_n, :)\mathbf{d}_{-n}^T, \quad \text{for} \quad \mathbf{d}_{-n} := \underset{m \neq n}{*} \mathbf{A}_m(i_m, :),$$

which only requires computing the matrix-vector product $\mathbf{A}\mathbf{d}_{-n}^T$ and finding its smallest or largest element. Such discrete coordinate descent can be extended to optimizing over small (and possibly randomly chosen) blocks of variables at a time. This is akin to what is known as alternating (block coordinate) optimization in the context of tensor factorization, and it can work quite well in practice for small to moderate N. However, such an approach does not seem to scale well with higher N, and it is not particularly elegant. We would like to have the option of updating all variables at once, as we do in continuous optimization – but the problem at hand is discrete and does not appear amenable to tools from continuous optimization. Thankfully, appearances can be deceiving. We have the following result.

Theorem 3: Finding the smallest element of a N-way tensor in CPD form is equivalent to the following continuous relaxation involving probability distributions $\{\mathbf{p}_n\}_{n=1}^N$

$$\min_{\left\{\mathbf{p}_n \geq 0,\ \mathbf{1}_{I_n}^T \mathbf{p}_n = 1\right\}_{n=1}^N} \left(\left(\mathbf{p}_1^T \mathbf{A}_1\right) * \cdots * \left(\mathbf{p}_N^T \mathbf{A}_N\right)\right) \mathbf{1}_R. \quad (1)$$

Proof: Let $\{\check{\mathbf{p}}_n\}_{n=1}^N$ denote an optimal solution to (1), and let $\check{\mathbf{q}}_{-n} := \mathbf{A}_n \left(\overset{*}{_{m\neq n}}\mathbf{A}_m^T \check{\mathbf{p}}_m\right)$. Note that the minimum of (1) is then equal to $\check{\mathbf{p}}_n^T \check{\mathbf{q}}_{-n}$. The minimum of the inner product $\check{\mathbf{p}}_n^T \check{\mathbf{q}}_{-n}$ is clearly attained when $\check{\mathbf{p}}_n$ is a Kronecker delta that selects a minimum element of $\check{\mathbf{q}}_{-n}$ (there might be multiple minimal elements in $\check{\mathbf{q}}_{-n}$ that happen to be exactly equal). By virtue of optimality, $\check{\mathbf{p}}_n$ cannot combine non-minimal elements of $\check{\mathbf{q}}_{-n}$, because that would clearly increase the cost, contradicting optimality. Thus $\check{\mathbf{p}}_n$ generates a convex combination of the possibly multiple equivalent minimal elements of $\check{\mathbf{q}}_{-n}$. But because the latter are exactly equal, the same cost is produced by putting all the mass in only one of them.

Thus, given an optimal solution $\{\check{\mathbf{p}}_n\}_{n=1}^N$ of (1), we can always round $\check{\mathbf{p}}_1$ so that it is integral, without loss of optimality. This yields another optimal solution of (1) to which we can apply the same argument to round $\check{\mathbf{p}}_2$ this time, and so on. The proof is therefore complete. □

Theorem 4: Finding the smallest element of a N-way tensor in Tucker, MLSVD, TT, or TR form is likewise equivalent to the corresponding continuous relaxation.

Proof: All these decomposition models are fundamentally sums of outer products, i.e., rank-one tensors. Thus they can always be put in the form of multilinear (polyadic) decomposition, albeit such decomposition will not necessarily be *canonical*, i.e., of minimal rank, nor will it be unique. Notice however that our proof of the previous Theorem does not assume anything about uniqueness or the number of components in the decomposition. Hence it applies to all these models. As a special case, it applies to full-rank tensors; but then there is no way to avoid the curse of dimensionality, i.e., exponential complexity in gradient computations, see below. Thus it is low-rankness that saves the day. □

IV. Methods

Theorem 3 opens the door to derivative-based optimization. The gradient of the cost function,

$$\begin{aligned} f(\mathbf{p}_1, \cdots, \mathbf{p}_N) &= \left(\left(\mathbf{p}_1^T \mathbf{A}_1\right) * \cdots * \left(\mathbf{p}_N^T \mathbf{A}_N\right)\right) \mathbf{1}_R \\ &= \left(\overset{*}{_n}\ \mathbf{p}_n^T \mathbf{A}_n\right) \mathbf{1}_R, \end{aligned}$$

with respect to $\mathbf{p}_n$ is given by

$$\nabla_{\mathbf{p}_n} f = \mathbf{q}_{-n} := \mathbf{A}_n \left(\overset{*}{_{m\neq n}}\mathbf{p}_m^T \mathbf{A}_m\right)^T.$$

This is very easy to compute. When $\mathbf{p}_n^T \mathbf{A}_n$ contains no zeros, we only have to compute $\overset{*}{_m}\mathbf{p}_m^T \mathbf{A}_m$ once, at a cost of $\sum_{n=1}^N I_n R + (N-1)R$ flops, and then divide each time by the leave-one-out factor to compute all the Hadamard products needed. This is followed by a final matrix-vector multiplication for each n. The total cost is thus $2\sum_{n=1}^N I_n R + (2N-1)R$ flops to compute all gradients. To understand what happens when zeros appear, it suffices to consider each column (latent dimension) separately, as multiplications and division happen at the column (rank-one factor) level. If the first element of one and only one of the $\mathbf{p}_n^T \mathbf{A}_n$ is zero, then we should also compute the nonzero leave-one-out product of all other first elements and use that only for the gradient update of the $\mathbf{p}_n$ which generated the said zero element. If the first element of more than one $\mathbf{p}_n^T \mathbf{A}_n$ is zero, then all leave-one-out products are zero, hence there is no need to even consider the first dimension in the gradient update, because the corresponding component of the gradient is zero. This implies that for each of the latent dimensions, $r \in \{1, \cdots, R\}$, we only need to detect if there is a single zero and compute the product of the nonzero elements. Hence, even in the presence of zeros, the worst-case complexity is linear in N, as stated above.

We need to enforce the probability simplex constraints, and projected gradient descent (PGD) is a natural choice for this purpose; but we have several other choices. Among them, the Frank-Wolfe algorithm has certain advantages. In our particular context, computing the minimum inner product of the gradient over the feasible set separates across modes, and for each mode it boils down to finding the minimum element of the corresponding part of the gradient vector. Thus Frank-Wolfe bypasses projection onto the simplex, which requires an iterative algorithm. Frank-Wolfe applied to nonconvex cost functions with convex constraints enjoys nice convergence guarantees [15] – note that our multilinear cost function is nonconvex, but the probability simplex constraints are convex.

Another way to bypass the projection onto the simplex is to introduce what is sometimes referred to as the *Hadamard parametrization* of a probability distribution $\mathbf{p}(i) = (\mathbf{s}(i))^2$, where $\mathbf{p}$ is the Hadamard product of $\mathbf{s}$ with itself [16]; or the *amplitude parametrization* $\mathbf{p}(i) = |\mathbf{s}(i)|^2$ of the quantum literature, where $\mathbf{s}$ is complex and $\mathbf{p}$ is the Hadamard product of $\mathbf{s}$ and the conjugate of $\mathbf{s}$. With these parametrizations $\mathbf{p}$ always has non-negative elements and sums up to one if and only if $||\mathbf{s}||_2 = 1$. Thus the simplex constraint is transformed to a unit sphere constraint. One advantage of this is that projection of any vector on the unit sphere only involves normalization, i.e., $\mathbf{s} = \frac{\mathbf{s}}{||\mathbf{s}||_2}$. A drawback is that the unit sphere is not a convex set, which complicates convergence analysis. The third option when it comes to parametrizing the simplex constraint is to use $\mathbf{p}(i) = \frac{e^{\mathbf{v}(i)}}{\sum_{j=1}^I e^{\mathbf{v}(j)}}$, where vector $\mathbf{v}$ is unconstrained. This parametrization emerges from mirror descent using negative entropy, but it can also be motivated from the viewpoint of turning simplex-constrained optimization into an unconstrained problem on which gradient descent can be applied. Following the latter viewpoint and applying the chain rule

$$\frac{\partial f}{\partial \mathbf{v}_n(i)} = \mathbf{p}_n(i)\mathbf{g}_n(i) - \mathbf{p}_n(i)\mathbf{p}_n^T \mathbf{g}_n,$$

where $\mathbf{g}_n := \nabla_{\mathbf{p}_n} f$.

The aforementioned simplex parametrization approaches can model any finite distribution. In our particular context, we know

Algorithm 1 Min CPD via Frank-Wolfe w/ adaptive stepsize

```
Input: $\{\mathbf{A}_n\}_{n=1}^N$, curvature parameter C.
  1. Initialize mode distributions $\mathbf{p}_n$ randomly, $\forall n \in \{1, \cdots, N\}$.
  2. repeat
  3. Compute mode gradients, update directions, adaptive stepsize:
       Set $g_t = 0$
       for n=1 to N
         Compute $\nabla_{\mathbf{p}_n} f = \mathbf{A}_n \left( \underset{m \neq n}{*} \mathbf{p}_m^T \mathbf{A}_m \right)^T$
         Find $v_n^* = \min_{i_n=1}^{I_n} \nabla_{\mathbf{p}_n} f(i_n)$
         Set $\mathcal{M}_n = \{ i_n \mid \nabla_{\mathbf{p}_n} f(i_n) = v_n^* \}$.
         Set $\mathbf{d}_n(i_n) = 1/|\mathcal{M}_n|, \forall i_n \in \mathcal{M}_n$
         Accumulate $g_t = g_t + (\mathbf{p}_n - \mathbf{d}_n)^T \nabla_{\mathbf{p}_n} f$
       end for
       Set $\lambda_t = \min(\frac{g_t}{C}, 1)$
  4. Update mode distributions:
       for n=1 to N
         $\mathbf{p}_n = (1 - \lambda_t)\mathbf{p}_n + \lambda_t \mathbf{d}_n$
       end for
  5. until convergence criterion met
 Output: $i_n^* \in \arg\max_{i \in I_n} \mathbf{p}_n(i)$.
```

Algorithm 2 Min CPD via simplex PGD with momentum

```
Input: $\{\mathbf{A}_n\}_{n=1}^N$, step size, momentum parameters $\lambda, \beta$, resp.
  1. Init. mode distributions $\mathbf{p}_n$ randomly, $\forall n \in \{1, \cdots, N\}$.
  2. Initialize gradients $\mathbf{g}_n = \mathbf{0}_{I_n \times 1}$, $\forall n \in \{1, \cdots, N\}$.
  3. repeat
  4. Compute mode gradients, update mode distributions:
       for n=1 to N
         Compute $\nabla_{\mathbf{p}_n} f = \mathbf{A}_n \left( \underset{m \neq n}{*} \mathbf{p}_m^T \mathbf{A}_m \right)^T$
         Accumulate momentum $\mathbf{g}_n = (1 - \beta)\mathbf{g}_n + \beta \nabla_{\mathbf{p}_n} f$
         Update and project onto simplex $\mathbf{p}_n = \mathcal{P}_\Omega (\mathbf{p}_n - \lambda \mathbf{g}_n)$
       end for
  5. until convergence criterion met
Output: $i_n^* \in \arg\max_{i \in I_n} \mathbf{p}_n(i)$.
```

Algorithm 3 Min CPD via Exponential parametrization

```
Input: $\{\mathbf{A}_n\}_{n=1}^N$, step size parameter $\lambda$.
  1. Init. $\mathbf{v}_n$ randomly from i.i.d. normal distribution.
  2. repeat
  3. Compute mode gradients, update mode distributions:
       for n=1 to N
         Compute $\mathbf{g}_n := \nabla_{\mathbf{p}_n} f = \mathbf{A}_n \left( \underset{m \neq n}{*} \mathbf{p}_m^T \mathbf{A}_m \right)^T$
         Compute $\frac{\partial f}{\partial \mathbf{v}_n(i)} = \mathbf{p}_n(i)\mathbf{g}_n(i) - \mathbf{p}_n(i)\mathbf{p}_n^T \mathbf{g}_n, \forall i$
         Update $\mathbf{v}_n(i) = \mathbf{v}_n(i) - \lambda \frac{\partial f}{\partial \mathbf{v}_n(i)}, \forall i$
         Update $\mathbf{p}_n(i) = \frac{e^{\mathbf{v}_n(i)}}{\sum_{j=1}^I e^{\mathbf{v}_n(j)}}, \forall i$
       end for
  4. until convergence criterion met
 Output: $i_n^* \in \arg\max_{i \in I_n} \mathbf{p}_n(i)$.
```

Algorithm 4 Min CPD via “discrete Gaussian” parametrization

```
Input: $\{\mathbf{A}_n\}_{n=1}^N$, step size parameter $\lambda$.
  1. Init. $\mathbf{v}_n$ randomly from i.i.d. normal distribution.
  2. repeat
  3. Compute mode gradients, update mode distributions:
       for n=1 to N
         Compute $\mathbf{g}_n := \nabla_{\mathbf{p}_n} f = \mathbf{A}_n \left( \underset{m \neq n}{*} \mathbf{p}_m^T \mathbf{A}_m \right)^T$
         Compute $\frac{\partial f}{\partial \mathbf{v}_n(i)} = \mathbf{p}_n(i)\mathbf{g}_n(i) - \mathbf{p}_n(i)\mathbf{p}_n^T \mathbf{g}_n, \forall i$
         Compute $\frac{\partial f}{\partial \mu_n} = \sum_{i=1}^I \frac{\partial f}{\partial \mathbf{v}_n(i)} \frac{2(i-\mu_n)}{\sigma_n^2}$
         Compute $\frac{\partial f}{\partial \sigma_n} = \sum_{i=1}^I \frac{\partial f}{\partial \mathbf{v}_n(i)} \frac{2(i-\mu_n)^2}{\sigma_n^3}$
         Update $\mu_n = \mu_n - \lambda \frac{\partial f}{\partial \mu_n}$
         Update $\sigma_n = \sigma_n - \lambda \frac{\partial f}{\partial \sigma_n}$
         Update $\mathbf{v}_n(i) = -\left( \frac{i-\mu_n}{\sigma_n} \right)^2, \forall i$
         Update $\mathbf{p}_n(i) = \frac{e^{\mathbf{v}_n(i)}}{\sum_{j=1}^I e^{\mathbf{v}_n(j)}}, \forall i$
       end for
  4. until convergence criterion met
 Output: $i_n^* \in \arg\max_{i \in I_n} \mathbf{p}_n(i)$.
```

that the sought distribution can be restricted to be unimodal – after all, the final solution can be “rounded” to a Kronecker delta without loss of optimality per the proof of Theorem 3. Towards this end, we can use as potential map $\mathbf{v}(i)$ a discrete analog of the exponent of a Gaussian, namely

$$\mathbf{v}(i) = -\left(\frac{i - \mu}{\sigma} \right)^2,$$

where $\mu \in \mathbb{R}$ is a location parameter and $\sigma \in \mathbb{R}$ controls the spread of the distribution. Using the chain rule again, the derivatives for updating these two “root” parameters are

$$\frac{\partial f}{\partial \mu} = \sum_{i=1}^I \frac{\partial f}{\partial \mathbf{v}_n(i)} \frac{2(i - \mu)}{\sigma^2}$$

and

$$\frac{\partial f}{\partial \sigma} = \sum_{i=1}^I \frac{\partial f}{\partial \mathbf{v}_n(i)} \frac{2(i - \mu)^2}{\sigma^3}.$$

Frank-Wolfe, PGD, and the exponential parametrization work well for random instances of the partition problem, subject to suitable tuning of the step-size related parameters. Pseudo-code listings of these four algorithms are provided as Algorithms 1, 2, 3, 4 for the Frank-Wolfe, PGD, exponential, and “discrete Gaussian” parametrization, respectively. For the partition problem, the Frank-Wolfe Algorithm 1 appears to offer the best performance and the lowest complexity in our proof-of-concept experiments. Each of these algorithms is useful in different application contexts, as we will see. Algorithm 4 uses the most compact parametrization of all, and is often the most efficient in terms of iterations needed for convergence.

For the Frank-Wolfe method, for which there is proof that the algorithm attains a stationary point at a rate of $\mathcal{O}(\frac{1}{\sqrt{t}})$, where t is the number of iterations, computing the associated curvature constant is non-trivial. In the matrix case ($N = 2$), the Lipschitz constant of the gradient of the cost function for our problem, which can be used to bound the curvature constant in Frank-Wolfe [15] is determined by the principal singular value of $\mathbf{A}_1 \mathbf{A}_2^T$. To see this, note that for $N = 2$ the cost function can be written as

$$f(\mathbf{p}_1, \mathbf{p}_2) = \mathbf{p}_1^T \mathbf{A}_1 \mathbf{A}_2^T \mathbf{p}_2,$$

the gradient of which is given by

$$\nabla_{[\mathbf{p}_1^T \mathbf{p}_2^T]^T} f = \begin{bmatrix} \mathbf{A}_1 \mathbf{A}_2^T \mathbf{p}_2 \\ \mathbf{A}_2 \mathbf{A}_1^T \mathbf{p}_1 \end{bmatrix}.$$

It follows that

$$\left|\left|\nabla_{\left[\mathbf{p}_1^T\mathbf{p}_2^T\right]^T} f - \nabla_{\left[\mathbf{q}_1^T\mathbf{q}_2^T\right]^T} f\right|\right|_2^2 = \left|\left|\begin{bmatrix}\mathbf{A}_1\mathbf{A}_2^T(\mathbf{p}_2-\mathbf{q}_2)\\ \mathbf{A}_2\mathbf{A}_1^T(\mathbf{p}_1-\mathbf{q}_1)\end{bmatrix}\right|\right|_2^2,$$
$$\leq \left(\sigma_{\max}\left(\mathbf{A}_1\mathbf{A}_2^T\right)\right)^2 \left|\left|\begin{bmatrix}\mathbf{p}_2-\mathbf{q}_2\\ \mathbf{p}_1-\mathbf{q}_1\end{bmatrix}\right|\right|_2^2,$$

hence

$$\left|\left|\nabla_{\left[\mathbf{p}_1^T\mathbf{p}_2^T\right]^T} f - \nabla_{\left[\mathbf{q}_1^T\mathbf{q}_2^T\right]^T} f\right|\right|_2 \leq \sigma_{\max}\left(\mathbf{A}_1\mathbf{A}_2^T\right)\left|\left|\begin{bmatrix}\mathbf{p}_2-\mathbf{q}_2\\ \mathbf{p}_1-\mathbf{q}_1\end{bmatrix}\right|\right|_2.$$

This is unfortunate, for computing $\sigma_{\max}\left(\mathbf{A}_1\mathbf{A}_2^T\right)$ is more costly than finding the smallest element of the product $\mathbf{A}_1\mathbf{A}_2^T$ by brute-force. In the N-way case, the situation is more complicated. Note that for $N=2$ the difference of the gradients is linear in the difference of the mode distributions. This is not true when $N>2$.

A. Complexity

Assuming $I_n = I$, $\forall n$ for brevity, the cost of computing all gradients with respect to the mode distributions $\mathbf{p}_n$ is of order NIR, as we have seen. The Frank-Wolfe algorithm maintains this complexity order per iteration. When employing the exponential parametrization of the mode distributions or the "discrete Gaussian" parametrization, the extra gradient back-propagation steps have complexity NI, thus again maintaining overall per-iteration complexity of order NIR. The actual complexity of the overall algorithm depends on a number of critical parameters, including the choice of gradient stepsize, the maximum number of iterations allowed ("hard-stop"), and the initialization used – intelligent / application-specific or random. We are using these algorithms to tackle NP-hard problems, so initialization does matter. In certain applications, such as parity check decoding, there is a natural initialization that we can use (the channel output bits), but in others, like the sign retrieval application that we will consider in some detail, there is no "natural" initialization that we can use. Through experimentation, we have found that one initialization that works well in many cases is to use the DP-based algorithm to compute an optimal solution for each rank-one factor separately, and then pick among those the one that is best for the higher-rank tensor minimization problem. This often gives a good "universal" (application-agnostic) initialization the complexity of which is of order $NIR + NR^2$.

Tuning the gradient stepsize is not difficult in our experience, but it is application-dependent. The maximum number of gradient iterations is set between 300 and $3,000$ in all our experiments, even for high-order problems (high N). Thus complexity is always polynomial of order NIR, but there is no guarantee that the optimal solution will be found. Notwithstanding, as we show in our experiments, the optimization performance attained is often state-of-art.

V. How Expressive Is the Class of Problems Considered?

We have seen that the class of problems that can be viewed as special instances of finding the minimum element of a tensor from its rank-one factors is broad – it contains the partition problem, and thus any NP-complete problem can be transformed in polynomial time to our problem of interest. Still, such transformation may not be obvious, and one wonders whether there exist broadly useful classes of NP-hard problems that are directly amenable, or easily transformable, to instances of the problem of interest. As we will see next, the answer is affirmative for various important optimization models that are frequently used in engineering.

A. Integer Linear Programming

Consider the Integer Linear Programming (ILP) problem

$$\min_{\mathbf{x}\in\mathbb{S}^N} \mathbf{c}^T\mathbf{x} \quad \text{s.t. } \mathbf{H}\mathbf{x} \leq \mathbf{b}, \tag{2}$$

where $\mathbf{H}\in\mathbb{R}^{M\times N}$, $\mathbf{b}\in\mathbb{R}^M$, $\mathbf{c}\in\mathbb{R}^N$, and $\mathbb{S}^N$ is a finite subset of $\mathbb{R}^N$. $\mathbb{S}^N$ will typically be a finite lattice, i.e., the Cartesian product of finite subsets of $\mathbb{R}$. In the supplementary material, we show that it is possible to transform this ILP problem to minimizing

$$\min_{\mathbf{x}\in\mathbb{S}^N}\left\{e^{t\mathbf{c}^T\mathbf{x}} + \sum_{m=1}^{M}\lambda_m e^{t(\mathbf{c}+\rho\mathbf{h}_m)^T\mathbf{x}}\right\},$$

where $\lambda_m := e^{-t\rho\mathbf{b}(m)}$, for sufficiently large (problem-specific) ρ and t. Every exponential inside the brackets is a rank-one tensor (a separable function of the variables in $\mathbf{x}$), and thus the above is a tensor of order N and rank at most $M+1$, which is very low compared to the maximal possible rank (I^{N-1} when $\mathbb{S}^N = \mathcal{I}^N$). Here $\mathcal{I}^N$ is the Cartesian product of N copies of $\mathcal{I}$ (note that $\mathbb{S}^N$ need not be a Cartesian product in general – we are using slightly overloaded notation).

B. Integer Least Squares

Consider the integer least squares (ILS) problem

$$\min_{\mathbf{x}\in\mathbb{S}^N} ||\mathbf{H}\mathbf{x}-\mathbf{b}||_2^2, \tag{3}$$

where $\mathbf{H}\in\mathbb{R}^{M\times N}$, $\mathbf{b}\in\mathbb{R}^M$. Let $\mathbf{G} := \mathbf{H}^T\mathbf{H}$ and $\mathbf{c} := 2\mathbf{H}^T\mathbf{b}$. Then, it is easy to see that

$$\min_{\mathbf{x}\in\mathbb{S}^N} ||\mathbf{H}\mathbf{x}-\mathbf{b}||_2^2 \equiv \min_{\mathbf{x}\in\mathbb{S}^N} \mathbf{x}^T\mathbf{G}\mathbf{x} - \mathbf{c}^T\mathbf{x}$$
$$\equiv \min_{\mathbf{x}\in\mathbb{S}^N}\sum_{n=1}^{N}\sum_{m=1}^{N}\mathbf{G}(n,m)\mathbf{x}(n)\mathbf{x}(m) - \sum_{n=1}^{N}\mathbf{c}(n)\mathbf{x}(n). \tag{4}$$

Note that a term of the form[1]

$$\gamma\mathbf{x}(n)\mathbf{x}(m) = \gamma\mathbf{x}^0(1)\cdots\mathbf{x}^0(n-1)\mathbf{x}(n)\mathbf{x}^0(n+1)\cdots \mathbf{x}^0(m-1)\mathbf{x}(m)\mathbf{x}^0(m+1)\cdots\mathbf{x}^0(N)$$

is separable (rank-one), and so is $\gamma(\mathbf{x}(n))^2$. It follows that the cost function in (4) has rank at most $\frac{N(N-1)}{2}+N$, which is again very low compared to the maximal possible rank (I^{N-1} when $\mathbb{S}^N = \mathcal{I}^N$). Note that here we have used the symmetry of $\mathbf{G} := \mathbf{H}^T\mathbf{H}$ to reduce the required number of rank-one terms.

[1]We may assume without loss of generality that $m \geq n$. Here, $\mathbf{x}^0$ is element-wise exponentiation, i.e., a vector of all ones.

C. Integer Quadratic Programming

Consider the integer indefinite quadratic problem $\min_{\mathbf{x}\in\mathbb{S}^N} \mathbf{x}^T\mathbf{Q}\mathbf{x}$, where $\mathbf{Q}$ is not necessarily positive semidefinite, or even symmetric. Note that we can always symmetrize without loss of generality, as $\mathbf{x}^T\mathbf{Q}\mathbf{x} = (\mathbf{x}^T\mathbf{Q}\mathbf{x})^T = \mathbf{x}^T\mathbf{Q}^T\mathbf{x}$, and thus $\mathbf{x}^T\mathbf{Q}\mathbf{x} = \mathbf{x}^T\frac{(\mathbf{Q}+\mathbf{Q}^T)}{2}\mathbf{x} = \mathbf{x}^T\mathbf{G}\mathbf{x}$, with $\mathbf{G} := \frac{(\mathbf{Q}+\mathbf{Q}^T)}{2}$. It follows that integer quadratic programming corresponds to finding a minimum element of a CPD model of rank $\frac{N(N-1)}{2}$.

D. Mixed Integer Programming: Sign Retrieval

So far we have considered optimization problems with purely categorical ("integer") variables. As an example of a mixed integer problem that falls under our framework, we next consider *sign retrieval* (e.g., see [17], [18]). This is a special case of *phase retrieval* [19], [20], and both have important applications in a broad range of disciplines, from optical imaging [19] to wireless communication – where it can be used for channel estimation from coarse channel quality measurements [21]. The starting point of the sign retrieval problem is the measurement model

$$\mathbf{y} = |\mathbf{A}\mathbf{x} + \mathbf{v}|,$$

where $\mathbf{x} \in \mathbb{R}^{N\times 1}$ is a vector of unknowns to be estimated, $\mathbf{A} \in \mathbb{R}^{M\times N}$ with $M > N$ (typically $M >> N$) is known, $\mathbf{v}$ is additive white Gaussian noise, $|\cdot|$ takes the absolute value of its argument, and $\mathbf{y} \in \mathbb{R}_+^{M\times 1}$ is the sign-less vector of measurements. Treating $\mathbf{s} = \text{sign}(\mathbf{y})$ and $\mathbf{x}$ as deterministic unknowns, maximum likelihood estimation amounts to

$$\min_{\mathbf{s}\in\{\pm1\}^{M\times 1},\ \mathbf{x}\in\mathbb{R}^{N\times 1}} ||\mathbf{D}(\mathbf{s})\mathbf{y} - \mathbf{A}\mathbf{x}||_2^2,$$

where $\mathbf{D}(\mathbf{s})$ is a diagonal matrix holding the elements of $\mathbf{s}$ on its diagonal. The problem is separable with respect to the continuous parameters $\mathbf{x}$. That is, solving for $\mathbf{x}$ as a function of $\mathbf{s}$, $\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{D}(\mathbf{y})\mathbf{s}$ and substituting the result back into the cost function, we obtain

$$\min_{\mathbf{s}\in\{\pm1\}^{M\times 1}} ||(\mathbf{I} - \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T)\mathbf{D}(\mathbf{y})\mathbf{s})||_2^2,$$

where we have used $\mathbf{D}(\mathbf{s})\mathbf{y} = \mathbf{D}(\mathbf{y})\mathbf{s}$. Expanding and using the idempotence of $(\mathbf{I} - \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T)$, we can further rewrite the problem as

$$\min_{\mathbf{s}\in\{\pm1\}^{M\times 1}} \mathbf{s}^T\mathbf{Q}\mathbf{s},$$

$$\text{with } \mathbf{Q} := \mathbf{D}(\mathbf{y})(\mathbf{I} - \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T)\mathbf{D}(\mathbf{y}).$$

It follows from the preceding subsection on integer quadratic programming that the sign retrieval problem corresponds to finding a minimum element of a CPD model of rank $\frac{M(M-1)}{2}$.

E. Maximum Likelihood/Minimum Distance Decoding of Parity Check Codes Over Galois Fields

Consider a parity check code [22] over GF(2) with $M \times N$ parity check matrix $\mathbf{C}$, where M is the number of parity checks, N is the codeword length, and the code rate is $\frac{K}{N}$, where $K := N - M$. A certain $N \times 1$ binary vector $\mathbf{x} \in \{0,1\}^N$ is a valid codeword if and only if $\mathbf{C}\mathbf{x} = \mathbf{0}_{M\times 1}$ in GF(2) modulo 2 arithmetic, i.e., $\text{mod}(\mathbf{C}\mathbf{x}, 2) = \mathbf{0}_{M\times 1}$ in real arithmetic. Assuming that the coded bits are transmitted over a memoryless binary symmetric channel (BSC) with cross-over probability $p < 0.5$, or over an additive white Gaussian noise (AWGN) channel, maximum likelihood decoding reduces to minimum Hamming or Euclidean distance decoding, respectively. Noting that for $\{0,1\}$-encoding of the channel input $\mathbf{x}$ and output $\mathbf{y}$, Euclidean distance squared is equal to Hamming distance, we can write both using real arithmetic as

$$\min_{\mathbf{x}\in\{0,1\}^N|\text{mod}(\mathbf{C}\mathbf{x},2)=\mathbf{0}} ||\mathbf{y} - \mathbf{x}||_2^2. \tag{5}$$

Optimal decoding is an NP-hard problem for most "irregular" codes of current interest[2], including low-density parity check (LDPC) codes which can come close to attaining the Shannon limit. These are decoded using an iterative message passing technique known as belief propagation, which performs well for code graphs that are free of short loops [23], [24]. We will next show that the same optimal decoding problem can be approached in a very different way: as the problem of computing the minimum element of a low-rank tensor. Consider the following problem

$$\min_{\mathbf{x}\in\{0,1\}^N} -\sum_{m=1}^{M}(-1)^{\mathbf{C}(m,:)\mathbf{x}} + \frac{1}{e}\left(1 + \frac{1}{N}\right)^{\sum_{n=1}^N(\mathbf{y}(n)-\mathbf{x}(n))^2}. \tag{6}$$

Note that $(-1)^{\mathbf{C}(m,:)\mathbf{x}} = (-1)^{\sum_{n=1}^N \mathbf{C}(m,n)\mathbf{x}(n)} = \prod_{n=1}^N(-1)^{\mathbf{C}(m,n)\mathbf{x}(n)}$, which is a rank-one tensor. Likewise, $\left(1+\frac{1}{N}\right)^{\sum_{n=1}^N(\mathbf{y}(n)-\mathbf{x}(n))^2} = \prod_{n=1}^N\left(1+\frac{1}{N}\right)^{(\mathbf{y}(n)-\mathbf{x}(n))^2}$ is separable, and thus a rank-one tensor. The overall cost function in (6) is therefore a tensor of rank (at most) $M+1$. We have the following result.

Proposition 2: Solving (6) is equivalent to solving (5).

Proof: Note that each term of the sum on the left is equal to 1 when the corresponding parity equation is satisfied, or -1 otherwise. With the minus sign up front of the sum, minimization of this term will produce a valid codeword that satisfies all parity checks and yield an overall value $-M$. Any constraint that is violated will increase this term by 2. The term on the right is monotonically increasing with the squared loss $||\mathbf{y}-\mathbf{x}||_2^2 = \sum_{n=1}^N(\mathbf{y}(n)-\mathbf{x}(n))^2$. The maximum value of the latter is N, while $\left(1+\frac{1}{N}\right)^N$ is upper bounded by its limit as $N\to\infty$, which is the base of the natural logarithm, e. It follows that the term on the right is strictly less than 1 (and lower bounded by $\frac{1}{e}$). Since the all-zero codeword satisfies all parity checks, it follows that the optimum solution to (6) should have cost less than or equal to $-M+1$. Moreover, this value cannot be attained by any $\mathbf{x}$ which does not satisfy all parity checks, because such a $\mathbf{x}$ would incur a cost of at least $-M+2$, even discarding the second term in (6). Hence, any solution of (6) must satisfy all parity checks. Among all $\mathbf{x}$ that satisfy all parity checks, those that minimize $\sum_{n=1}^N(\mathbf{y}(n)-\mathbf{x}(n))^2$ yield the least overall cost in (6), and thus the proof is complete. □

[2]E.g., Convolutional codes with short memory can be optimally and efficiently decoded using DP.

Remark 4: When $\mathbf{y}$ is real-valued (AWGN channel) the only difference is the scaling of the second term, which should be $\frac{1}{c}$ instead of $\frac{1}{e}$, with $c := (1+\frac{1}{N})^{||\mathbf{z}||_2^2}$ and

$$\mathbf{z}(n) := \begin{cases} \mathbf{y}(n), & \mathbf{y}(n) > 0.5, \\ \mathbf{y}(n)-1, & \mathbf{y}(n) \le 0.5. \end{cases}$$

F. Codes Over Higher-Order Galois Fields

Our framework can also handle the decoding of codes over higher-order Galois fields. For example, let $L = 2^\ell$ and consider a system of parity equations over GF(L). Such systems are of form $\mathbf{Cx} = \mathbf{q}$, where both $\mathbf{C}$ and $\mathbf{q}$ are given, and the equality is modulo $L = 2^\ell$, i.e., $\mathbf{Cx} - \mathbf{q}$ is a vector of integer multiples of L. We can handle this type of equation by bringing in a familiar signal processing tool, namely, the complex roots of unity. That is, we seek to minimize over $\mathbf{x} \in \{0, \cdots, L-1\}^N$ the following cost function

$$-\sum_{m=1}^{M} \text{Re}\left\{e^{j\frac{2\pi}{L}(\mathbf{C}(m,:)\mathbf{x}-\mathbf{q}(m))}\right\} + \frac{1}{\tilde{c}}\left(1+\frac{1}{N}\right)^{\sum_{n=1}^{N}(\mathbf{y}(n)-\mathbf{x}(n))^2}, \tag{7}$$

where $j := \sqrt{-1}$. The same logic applies for appropriately choosing $\tilde{c}$.

G. Least Inconsistent Solution of Overdetermined Linear System of Equations Over GF(2)

So far, we have been dealing with underdetermined linear equations over a GF, and looking at minimum distance solutions relative to an "anchor". This is similar to the minimum norm solution of underdetermined linear equations in the real or complex field. The overdetermined version of the problem is also of interest, and has many applications – e.g., in cryptanalysis [25], [26], [27]. It turns out that our framework can deal with this problem as well. We are given a set of $M > N$ linear equations in N variables $\mathbf{Gx} = \mathbf{y} \Leftrightarrow \mathbf{Gx} + \mathbf{y} = \mathbf{0}$ over GF(2). The system is usually inconsistent, so we seek a $\mathbf{x}$ that minimizes the number of violated constraints, i.e., a solution that is least inconsistent. This can be simply posed as

$$\min_{\mathbf{x}\in\{0,1\}^N} -\sum_{m=1}^{M}(-1)^{\mathbf{G}(m,:)\mathbf{x}+\mathbf{y}(m)}. \tag{8}$$

H. Underdetermined or Overdetermined?

Every linear code can be generated as a linear combination of the columns of a code generating matrix $\mathbf{G}$. When $\mathbf{G}$ is tall ($M \times N$ with $M > N$) and full column rank, the valid code words live in a subspace of dimension N and they are orthogonal to the rows of a parity check matrix $\mathbf{C}$ of size $(M-N) \times M$. Given a noisy code word, we may pose the optimal decoding problem in two equivalent ways:

- One is in code space, i.e., find a valid code word that is closest to the given noisy code word, and this is the formulation in (6), wherein the unknown vector $\mathbf{x}$ is the sought clean code word. After solving (6), we need to solve a consistent system of linear equations over GF(2) (via Gaussian elimination) to obtain the latent information sequence – unless the code is systematic / in standard form wherein the information sequence appears as the prefix of the code sequence.
- The other option is to pose the problem in the lower-dimensional space of the information sequence, i.e., find an information sequence that produces a valid code word that is closest to the given noisy code word; this is the formulation in (8), wherein the unknown vector $\mathbf{x}$ is the sought information sequence[3]. When we take this route, there is no need for the additional Gaussian elimination step at the end, as we directly recover the information sequence. Note that the number of optimization variables is smaller and the rank of the CPD model is higher this way, but the complexity of our algorithms is linear in the number of unknowns and the rank, so this does not really affect the complexity of our approach.
- The difference between the two ways of approaching the problem lies in the initialization. If we go via (6) there is a natural initialization for $\mathbf{x}$ – the received noisy code word. Notice that this works for any parity check matrix. If we choose to solve (8) on the other hand, there is no obvious way to initialize the information sequence $\mathbf{x}$, unless the code is systematic – in which case we read out a noisy version of $\mathbf{x}$ from the noisy code word itself. It is therefore preferable to use (6) unless the code is systematic. The code is not systematic in cryptography applications for example.

I. Reprise

As we conclude this section, it is useful to reflect on what we learned. The take-home point is that there are many important problems which can be posed as instances of low-rank tensor minimization, for which it is not even necessary to perform tensor factorization – the low-rank factors can be readily derived analytically, in closed-form. These clean-cut mappings of classic hard problems to instances of low-rank tensor minimization reinforce our hopes that, even in cases where the problem is more complicated and the cost function is not fully known (i.e., only examples / samples of the cost function are given), it may be possible to model those "blind" optimization problems using low-rank tensor factorization and minimization.

Another important observation which stems from uniqueness of tensor completion, is that under certain conditions it is not even necessary to completely specify an instance of ILS or ILP in the traditional sense of providing its input parameters $\mathbf{H}, \mathbf{b}, \mathbf{c}$ in order to solve it. It is enough to specify the cost at certain (randomly chosen or systematic) points, and let tensor completion fill out the "rest of the problem". This possibility is certainly intriguing, and the direct result of uniqueness of low-rank tensor completion and our problem reformulation.

[3]In the problem statements, $\mathbf{x}$ is always taken to be an $N \times 1$ vector for consistency, but what is N (length of code word or length of information sequence) changes depending on the context.

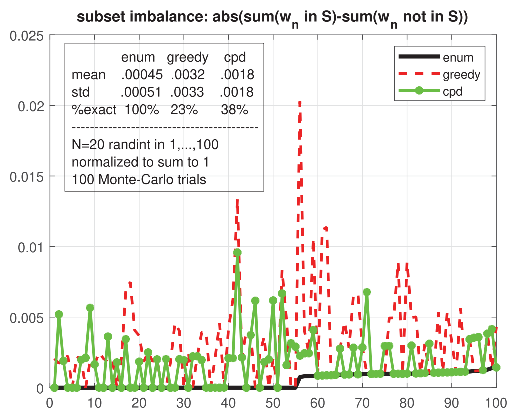

Fig. 1. Partition gap for 100 problem instances with $N = 20$ random integers in $\{1, \cdots, 100\}$, normalized to sum to 1. Greedy, optimal (enumeration-based), and proposed minimum CPD algorithm using Frank-Wolfe. Each point on the x axis corresponds to a problem instance, and the instances are sorted in order of increasing partition gap for the optimal enumeration-based solution.

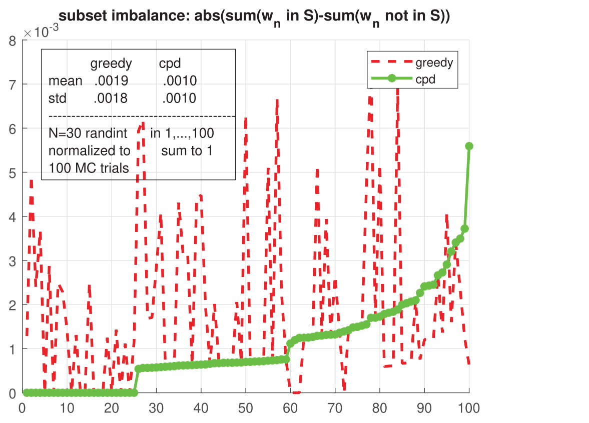

Fig. 2. Partition gap for 100 problem instances with $N = 30$ random integers in $\{1, \cdots, 100\}$, normalized to sum to 1. Greedy and proposed minimum CPD algorithm using Frank-Wolfe. Enumeration is too costly at $N = 30$ (2^{30} possible subsets), so the instances are sorted in order of increasing CPD partition gap.

VI. Experiments

A. Partition

Since we used the partition problem to establish the hardness of our problem in the worst case, let us compare one of the proposed continuous optimization algorithms for minCPD to an established approximation algorithm for the partition problem. For this purpose, we will use *greedy partitioning algorithm* [14] which first sorts the given numbers and then parses the sorted list from largest to smallest, assigning each to the bucket with the smallest running sum. This algorithm comes with a 7/6 approximation guarantee in terms of the larger sum it outputs divided by the larger sum of an optimal partition. For smaller N we also use enumeration to compute the optimal partition as another baseline.

The results obtained using the greedy algorithm and minCPD via Frank-Wolfe with adaptive stepsize (Algorithm 1) are summarized in Fig. 1 for $N = 20$ (with enumeration as another baseline) and Fig. 2 for $N = 30$ (without enumeration), for 100 Monte-Carlo trials each. In each trial, N random integers in $\{1, \cdots, 100\}$ are first drawn and then normalized to sum to 1. For Frank-Wolfe, we used $C = 5$, a hard-stop at a maximum of 1000 iterations, and 5 random initializations.

It is clear that Algorithm 1 outperforms the well-established greedy algorithm, and in many cases attains the optimal solution (zero subset imbalance, or the optimal subset imbalance obtained via enumeration, see Figs. 1 and 2).

B. Sign Retrieval

Our second set of experiments considers the application of our framework to the problem of sign retrieval. Towards this end, we use Algorithm 4 with stepsize parameter fixed to 0.1 and a hard limit of 10^3 gradient iterations. For initialization, we use the rank-one DP algorithm of Proposition 1, which is used to efficiently determine the minimum of each rank-one factor. The best of these rank-one factor minima (the one that minimizes the full-rank cost) is then used to initialize Algorithm 4.

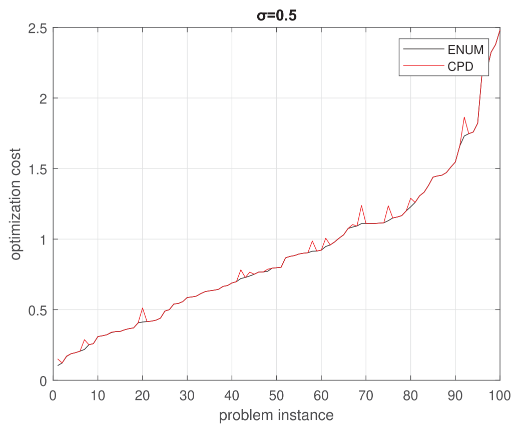

Fig. 3. Sign retrieval: cost attained by the enumeration-based solution and the proposed CPD-based approach, for $M = 12$, $N = 6$, $\sigma = 0.5$. See text for details. Each point on the x axis corresponds to a problem instance, and the instances are sorted in order of increasing cost for the optimal enumeration-based solution.

Ten additional random initializations are also used in case the DP initialization is not good enough. As a baseline, we use enumeration over all possible vectors of sign variables.

For each setting of the sign retrieval problem parameters ($N = \text{length}(\mathbf{x})$, $M = \text{length}(\mathbf{y})$, $\sigma = \text{std}(\mathbf{v}(m))$), we conduct 100 Monte-Carlo trials. For each trial, we draw random i.i.d. standard Gaussian $\mathbf{x}$ and $\mathbf{A}$, and i.i.d. zero-mean Gaussian noise $\mathbf{v}$ of standard deviation σ_v.

For our first experiment, we choose $N = 6$, $M = 12$, and $\sigma_v = 0.5$. Note that for this application the order of the tensor used in the CPD model is M and its rank is $\frac{M(M-1)}{2}$. Fig. 3 shows the cost function value attained for 100 randomly drawn problem instances, constructed as specified above. Note that CPD comes very close to the optimal cost of enumeration, with a few minor spikes, for all instances considered. For this application, what is perhaps more important than the value of the cost function is the squared error between the ground-truth $\mathbf{x}$ and the $\hat{\mathbf{x}}$ estimated by a given algorithm. The values of squared error

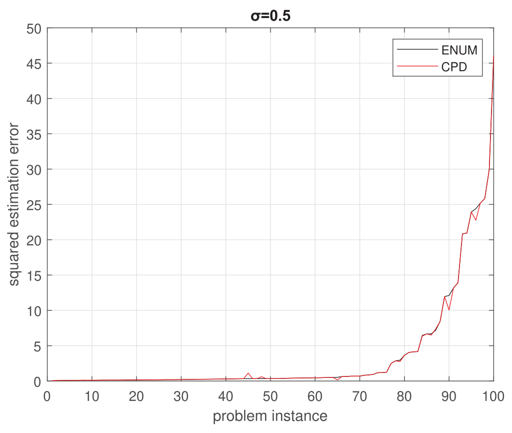

Fig. 4. Sign retrieval: squared $\mathbf{x}$-estimation error attained by the enumeration-based solution and the proposed CPD-based approach, for $M = 12$, $N = 6$, $\sigma = 0.5$. Each point on the x axis corresponds to a problem instance, and the instances are sorted in order of increasing squared error of the enumeration-based solution.

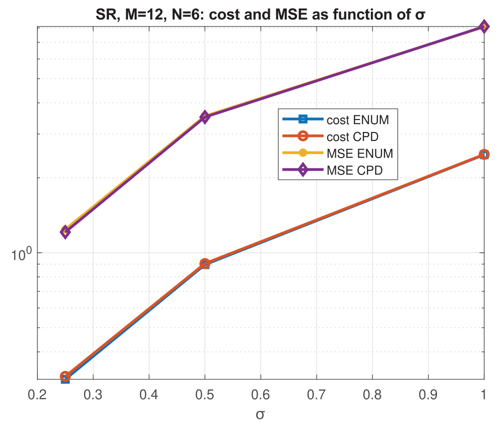

Fig. 5. Sign retrieval: mean optimization cost and MSE attained by the enumeration-based solution and the proposed CPD-based approach as a function of σ, for $M = 12$, $N = 6$.

attained by CPD and enumeration are shown in Fig. 4. Notice that enumeration is by definition optimal in terms of the cost function, but not necessarily optimal in terms of instantaneous or even mean squared error (MSE) – as the cost function is only a surrogate for MSE. Indeed, there are a few instances where CPD is better than enumeration in terms of squared error, and vice-versa. Overall though, the two approaches are very close in this experiment.

Monte-Carlo averages of the optimization cost and the squared error are depicted in Figs. 5 and 6 as a function of σ and N (with $M = 2N$), respectively. We again observe the excellent performance of the proposed CPD approach in this set of experiments.

C. Decoding Parity Check Codes

In our last set of experiments, we consider decoding five different rate-$\frac{1}{2}$ parity check codes, for different code densities and lengths N. For the first three scenarios $N = 32$, while for the last two $N = 96$. In the first scenario we use a custom LDPC code design[4] [28]. In the remaining four scenarios, we use a systematic parity check matrix whose non-identity block is randomly generated from an i.i.d. $\{0, 1\}$-Bernoulli distribution of density either 0.2 or 0.8. Each code is used for encoding i.i.d. sequences of $N/2$ information bits, and the coded sequences are transmitted over BSCs with cross-over probability $p \in \{10^{-2.5}, 10^{-2}, 10^{-1.5}, 10^{-1}, 10^{-0.5}\}$. For each scenario and cross-over probability, we conduct 10,000 Monte-Carlo runs and compare the decoding performance of our method to two baselines: i) enumeration (applicable only for $N = 32$ due to its exponential complexity) and ii) belief propagation (BP) based decoding. Regarding our method, we use Algorithm 4 with step size equal to 0.05. The initialization of Algorithm 4 in terms of $\{\sigma_n\}_{n=1}^N$ and $\{\mu_n\}_{n=1}^N$ is $\sigma_n = 0.5$, $\forall n$, while μ_n is set to the received noisy (possibly flipped) value of the corresponding code bit. Algorithm 4 is terminated when the number of iterations exceeds the limit of 2,000 iterations or when the relative change of the objective drops below 10^{-9}. As for the BP baseline, we use the *ldpcDecode* function of

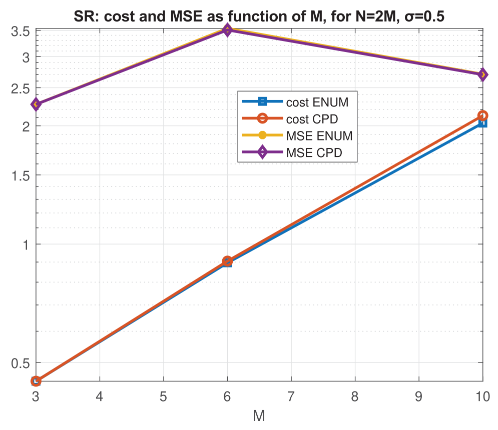

Fig. 6. Sign retrieval: mean optimization cost and MSE attained by the enumeration-based solution and the proposed CPD-based approach as a function of N, for $M = 2N$ and $\sigma = 0.5$.

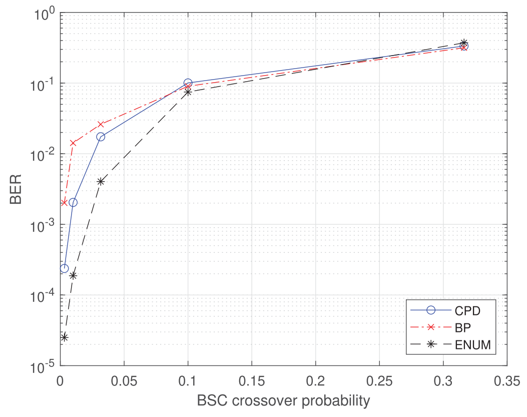

Fig. 7. Parity check decoding: BER vs BSC cross-over probability for the first scenario: LDPC code of $N = 32$ and $M = 16$.

[4]https://rptu.de/en/channel-codes/channel-codes-database/more-ldpc-codes

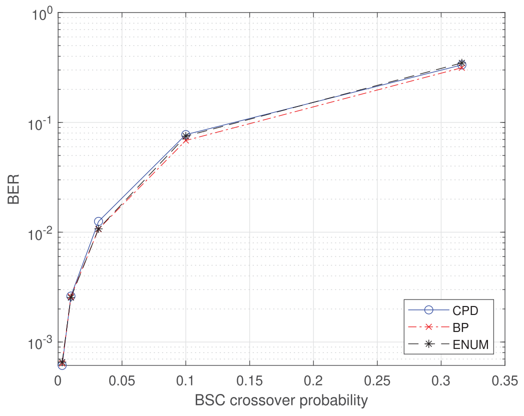

Fig. 8. Parity check decoding: BER vs BSC cross-over probability for the second scenario: random code of $N = 32$, $M = 16$, and 20% density.

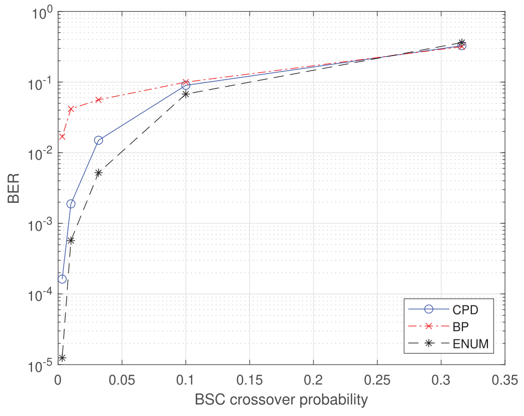

Fig. 9. Parity check decoding: BER vs BSC cross-over probability for the third scenario: random code with $N = 32$, $M = 16$ and 80% density.

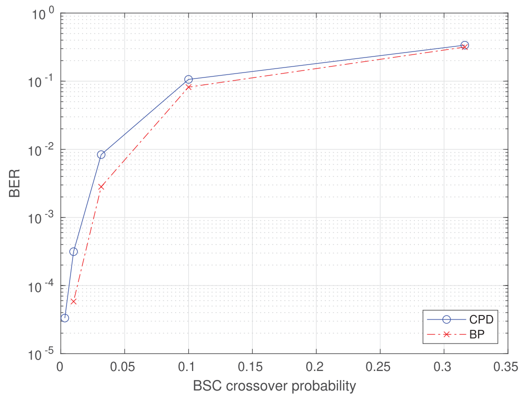

Fig. 10. Parity check decoding: BER vs BSC cross-over probability for the fourth scenario: random code with $N = 96$, $M = 48$ and 20% density.

MATLAB, to which the log-likelihood ratios based on the corresponding cross-over probabilities are provided as initialization, while the upper limit of iterations is set to 100 (we did not observe any improvement beyond that).

In Figs. 7–11, we report the average Bit Error Rate (BER) for all the methods. We can observe that, in general, Algorithm 4

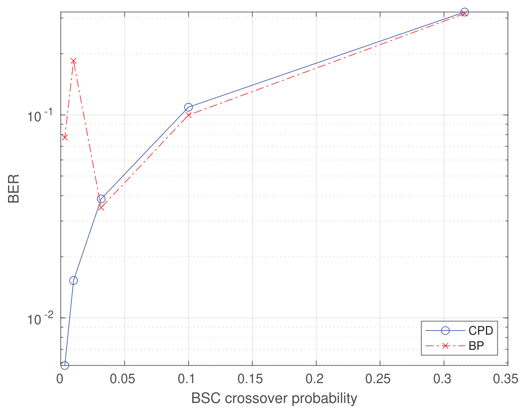

Fig. 11. Parity check decoding: BER vs BSC cross-over probability for the fifth scenario: random code with $N = 96$, $M = 48$ and 80% density.

achieves better or comparable performance than BP. At low BSC cross-over probabilities, we can see that Algorithm 4 outperforms BP for the custom LDPC code design of [28], and the high density parity check codes. In the rest of the cases the two methods attain comparable performance. BP has a small advantage for the longer low-density code in Fig. 10, as expected; BP works best with low-density long codes. We note that, unlike BP, Algorithm 4 does not (need to) use the BSC cross-over probability, which is non-trivial to estimate for time-varying channels. We also note that BP exhibits a degradation in performance for the longer and denser code at low BSC error rates, see Fig. 11. This is repeatable (not an artifact of insufficient Monte-Carlo averaging) and likely due to the existence of many short loops in this case. On the other hand, BP is significantly faster than Algorithm 4. Overall though, given that Algorithm 4 is a completely new and application-agnostic take on a well-studied problem, the fact that it outperforms in terms of BER a proven MATLAB implementation of a custom-designed and widely used algorithm is satisfying. Matlab programs that can be used to reproduce the results in this subsection can be found in the companion supplementary material in IEEE Xplore.

VII. Conclusion

We have considered a fundamental tensor problem and showed that it is NP-hard. While most tensor problems are NP-hard [13], it is surprising to see that our particular problem is NP-hard for rank as small as two, but not for rank equal to one. We note here that the best rank-one least squares approximation problem for tensors is already NP-hard.

Given the discrete / combinatorial nature of the problem considered, it is also unexpected to see that it admits an equivalent continuous reformulation. While this reformulation is itself NP-hard by virtue of equivalence, it opens the door for gradient-based approaches from the nonconvex continuous optimization literature.

We have shown that an impressive variety of hard optimization problems that are widely used in engineering can be posed as special instances of our problem of interest. These include integer least squares, integer linear and quadratic programming, certain mixed integer programming problems, and solving systems of underdetermined and overdetermined linear equations over Galois fields. For all these problems, the low-rank factorization needed to set up the optimization problem is available analytically, in simple closed-form. There is no need for tensor factorization.

As tangible signal processing and communications engineering applications, we delved into sign retrieval and the decoding of linear parity check codes. We have shown that the performance of the proposed suite of gradient-based approaches is surprisingly good in many of these applications, and under certain conditions it can beat tried-and-proven application-specific algorithms that come with certain performance guarantees. This success is sometimes dependent on using a suitable initialization, either application-specific ("dirty" code bits in the case of parity decoding) or "universal" (the DP algorithm used to initialize the gradient iterations for sign retrieval) in other cases. For other problems, like the partition, a few random initializations seem to work well.

Our main results (hardness, equivalence of continuous reformulation) are also applicable to all popular tensor models beyond CPD, including Tucker/HOSVD, TT, and TR.

Note that for the families of problems and applications considered in this paper, R is a small constant or linear / very low-order polynomial function of N. For so-called *black box* optimization problems out in the wild, the worst-case R can be exponential in N, and we would have to use low-rank approximation to keep complexity at bay. This is not an issue though for the various problems considered in this paper.

We are currently working on further improving scalability and speed, establishing convergence for some of the algorithmic variants, and considering applications in a variety of settings. Performance analysis is naturally of interest, but is likely to be application-specific. We are also considering top-k/bottom-k extensions which are appropriate for top-k recommendation and other applications. We hope to report on these directions in forthcoming work.

Acknowledgment

The authors thank Prof. Farzad Farnoud-Hassanzadeh, who provided useful insights on linear GF coding and decoding, and the reviewers for appreciating this work and offering useful suggestions.

References

[1] N. Higham and S. Relton, "Estimating the largest elements of a matrix," *SIAM J. Sci. Comput.*, vol. 38, no. 6, pp. C584–C601, 2016.

[2] G. Ballard, T. G. Kolda, A. Pinar, and C. Seshadhri, "Diamond sampling for approximate maximum all-pairs dot-product (MAD) search," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Los Alamitos, CA, USA: IEEE Comput. Soc. Press, Nov. 2015, pp. 11–20, doi: 10.1109/ICDM.2015.46.

[3] N. Kargas, N. D. Sidiropoulos, and X. Fu, "Tensors, learning, and 'Kolmogorov extension' for finite-alphabet random vectors," *IEEE Trans. Signal Process.*, vol. 66, no. 18, pp. 4854–4868, Sep. 2018, doi: 10.1109/TSP.2018.2862383.

[4] Z. Lu, Y. Hu, and B. Zeng, "Sampling for approximate maximum search in factorized tensor," in *Proc. 26th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2017, pp. 2400–2406.

[5] M. Espig, W. Hackbusch, A. Litvinenko, H. Matthies, and E. Zander, "Iterative algorithms for the post-processing of high-dimensional data," *J. Comput. Phys.*, vol. 410, Mar. 2020, Art. no. 109396.

[6] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, Jul. 2017.

[7] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco, CA, USA: Freeman, 1979.

[8] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[9] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1253–1278, 2000, doi: 10.1137/S0895479896305696.

[10] I. V. Oseledets, "Tensor-train decomposition," *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2295–2317, 2011, doi: 10.1137/090752286.

[11] Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki, "Tensor ring decomposition," 2016, *arXiv:1606.05535*.

[12] Y. Zniyed, R. Boyer, A. L. de Almeida, and G. Favier, "High-order CPD estimation with dimensionality reduction using a tensor train model," in *Proc. 26th Eur. Signal Process. Conf. (EUSIPCO)*, 2018, pp. 2613–2617.

[13] C. J. Hillar and L.-H. Lim, "Most tensor problems are NP-hard," *J. ACM*, vol. 60, no. 6, pp. 1–39, Nov. 2013, doi: 10.1145/2512329.

[14] E. L. Schreiber, R. E. Korf, and M. D. Moffitt, "Optimal multi-way number partitioning," *J. ACM*, vol. 65, no. 4, pp. 1–61, Jul. 2018, doi: 10.1145/3184400.

[15] S. Lacoste-Julien, "Convergence rate of Frank-Wolfe for non-convex objectives," 2016, *arXiv:1607.00345*.

[16] Q. Li, D. McKenzie, and W. Yin, "From the simplex to the sphere: Faster constrained optimization using the Hadamard parametrization," 2021, *arXiv:2112.05273*.

[17] B. Leshem, O. Raz, A. Jaffe, and B. Nadler, "The discrete sign problem: Uniqueness, recovery algorithms and phase retrieval applications," *Appl. Comput. Harmon. Anal.*, vol. 45, no. 3, pp. 463–485, 2018.

[18] K. Suzuki, C. Tsutake, K. Takahashi, and T. Fujii, "Compressing sign information in DCT-based image coding via deep sign retrieval," 2022. [Online]. Available: https://arxiv.org/abs/2209.10712

[19] Y. Shechtman, Y. C. Eldar, O. Cohen, H. N. Chapman, J. Miao, and M. Segev, "Phase retrieval with application to optical imaging: A contemporary overview," *IEEE Signal Process. Mag.*, vol. 32, no. 3, pp. 87–109, May 2015.

[20] Y. C. Eldar, N. Hammen, and D. G. Mixon, "Recent advances in phase retrieval [lecture notes]," *IEEE Signal Process. Mag.*, vol. 33, no. 5, pp. 158–162, Sep. 2016.

[21] T. Qiu, X. Fu, N. D. Sidiropoulos, and D. P. Palomar, "MISO channel estimation and tracking from received signal strength feedback," *IEEE Trans. Signal Process.*, vol. 66, no. 7, pp. 1691–1704, Apr. 2018.

[22] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2004.

[23] A. Shokrollahi, "LDPC codes: An introduction," in *Coding, Cryptography and Combinatorics*, K. Feng, H. Niederreiter, and C. Xing, Eds., Basel, Switzerland: Birkhäuser, 2004, pp. 85–110.

[24] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[25] J. Håstad, "Some optimal inapproximability results," *J. ACM*, vol. 48, no. 4, pp. 798–859, Jul. 2001, doi: 10.1145/502090.502098.

[26] J. Håstad, S. Huang, R. Manokaran, R. O'Donnell, and J. Wright, "Improved NP-inapproximability for 2-variable linear equations," *Theory Comput.*, vol. 13, pp. 1–15, Dec. 2017.

[27] J. Håstad, "Solving systems of linear equations over finite fields." KTH. Accessed: May 17, 2023. [Online]. Available: https://www.csc.kth.se/johanh/sms07.pdf

[28] S. Abu-Surra, D. DeClercq, D. Divsalar, and W. E. Ryan, "Trapping set enumerators for specific LDPC codes," in *Proc. Inf. Theory Appl. Workshop (ITA)*, Piscataway, NJ, USA: IEEE Press, 2010, pp. 1–5.

Nicholas D. Sidiropoulos (Fellow, IEEE) received the Diploma in electrical engineering from Aristotle University of Thessaloniki, Thessaloniki, Greece, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland at College Park, College Park, MD, USA, in 1988, 1990, and 1992, respectively. He is the Louis T. Rader Professor with the Department of ECE, University of Virginia. He has previously served as a Faculty with the University of Minnesota and the Technical University of Crete, Greece. His research interests are in signal processing, communications, optimization, tensor decomposition, and machine learning. He received the NSF/CAREER award in 1998, IEEE Signal Processing Society (SPS) Best Paper Award in 2001, 2007, 2011, and 2023, and the IEEE SPS Donald G. Fink Overview Paper Award in 2023. He served as a IEEE SPS Distinguished Lecturer (2008–2009), the Vice President—Membership of the IEEE SPS (2017–2019), and the Chair of the SPS Fellow Evaluation Committee (2020–2021). He received the 2010 IEEE SPS Meritorious Service Award, the 2013 Distinguished Alumni Award of the ECE Department, University of Maryland, the 2022 EURASIP Technical Achievement Award, and the 2022 IEEE SPS Claude Shannon–Harry Nyquist Technical Achievement Award. He is a fellow of EURASIP (2014).

Paris A. Karakasis (Student Member, IEEE) received the Diploma and M.Sc. degree in electrical and computer engineering from the Technical University of Crete, Chania, Greece, in 2017, and 2019, respectively. Currently, he is working toward the Ph.D. degree with the Electrical and Computer Engineering Department, University of Virginia, Charlottesville, VA, USA. His research interests include signal processing, optimization, machine learning, tensor decomposition, and graph mining.

Aritra Konar (Member, IEEE) received the B.Tech. degree in electronics and communications engineering from West Bengal University of Technology, West Bengal, India, and the M.S. and Ph.D. degrees in electrical engineering from the University of Minnesota, Minneapolis, MN, USA, in 2011, 2014, and 2017, respectively. He is an Assistant Professor with the Department of Electrical Engineering, KU Leuven, Leuven, Belgium. From 2017–2022, he was a Postdoctoral Researcher with the Department of ECE, University of Virginia, Charlottesville, VA, USA. His research interests include signal processing, graph mining, nonlinear optimization, and data analytics.