

# A Hybrid Method for Tensor Decompositions that Leverages Stochastic and Deterministic Optimization\*

Jeremy M. Myers<sup>†</sup> and Daniel M. Dunlavy<sup>†</sup>

**Abstract.** In this paper, we propose a hybrid method that uses stochastic and deterministic search to compute the maximum likelihood estimator of a low-rank count tensor with Poisson loss via state-of-the-art local methods. Our approach is inspired by Simulated Annealing for global optimization and allows for fine-grain parameter tuning as well as adaptive updates to algorithm parameters. We present numerical results that indicate our hybrid approach can compute better approximations to the maximum likelihood estimator with less computation than the state-of-the-art methods by themselves.

**Key words.** tensor, canonical polyadic decomposition, GCP, CPAPR

**AMS subject classifications.** 15A69, 65F55

**1. Introduction.** Low-rank tensor decompositions in general, and canonical polyadic (CP) tensor decompositions specifically, are now ubiquitous in the area of multi-way data analysis. Recent research in developing efficient methods for computing CP tensor decompositions via maximum likelihood estimation reflects an emergent dichotomy in numerical linear algebra: deterministic versus randomized algorithms. CP Alternating Poisson Regression (CPAPR) [17] is a deterministic CP tensor decomposition method that alternates over a sequence of convex Poisson loss subproblems iteratively. Generalized CP (GCP) tensor decompositions extend previous work on CP decompositions to incorporate general loss functions and stochastic optimization methods [31, 41]. Previously, in [48], we showed that CPAPR is performant and can compute accurate maximum likelihood estimator approximations with a higher probability than GCP. Here we explore a hybrid method for computing CP tensor decompositions that leverages GCP for scalability and CPAPR for performance and accuracy.

This paper is structured as follows. In [Section 2](#), we introduce notation and formalize several metrics, some used previously in [48], to compare CP decomposition methods. In [Section 3](#), we introduce *Cyclic GCP-CPAPR (CGC)*, a CP decomposition method that alternates between stochastic and deterministic methods to avoid local minima. We also formally define *parameterizations*, *strategies*, and *policies* to optimize CGC for precision and accuracy. In [Section 4](#), we report the results of numerical experiments comparing the hybrid CGC method with the individual GCP and CPAPR methods on real and synthetic sparse tensor data. These experiments offer evidence that our hybrid method can reduce maximum likelihood estimator approximation error and computational cost versus current methods. In [Section 5](#), we discuss our conclusions and proposed future work including:

---

\*Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SANDYYYY-NNNNR.

<sup>†</sup>Sandia National Laboratories ([jermyer@sandia.gov](mailto:jermyer@sandia.gov), [dmdunla@sandia.gov](mailto:dmdunla@sandia.gov))

- potential improvements in CGC computational efficiency via the choices of method parameterizations, strategies, and policies (subsection 5.1); and
- comparisons of CGC with generic optimization methods (subsection 5.2).

## 2. Background and Related Work.

**2.1. Notation and conventions.** The field of real numbers and the ring of integers are denoted as  $\mathbb{R}$  and  $\mathbb{Z}$ , respectively. The real numbers and integers restricted to non-negative values are denoted as  $\mathbb{R}_+$  and  $\mathbb{Z}_+$ , respectively. The *order* of a tensor is the number of *dimensions* or *ways*. Each tensor dimension is called a *mode*. A scalar (tensor of order zero) is represented by a lowercase letter, e.g.,  $x$ . A bold lowercase letter denotes a vector (tensor of order one), e.g.,  $\mathbf{v}$ . A matrix (tensor of order two) is denoted by a bold capital letter, e.g.,  $\mathbf{A} \in \mathbb{R}^{m \times n}$ . Tensors of order three and higher are expressed with a bold capital script letter, e.g.,  $\mathcal{X} \in \mathbb{R}^{m \times n \times p}$ . Values *derived*, *computed*, *approximated*, or *estimated* are typically written with a hat or a tilde—e.g.,  $\widehat{\mathbf{M}} \in \mathbb{R}^{m \times n \times p}$  may be a tensor model of parameters fit to data and  $\widetilde{\Sigma} \in \mathbb{R}^{n \times n}$  may be a diagonal matrix containing approximate singular values of a matrix.

The  $i$ -th entry of a vector  $\mathbf{v}$  is denoted  $v_i$ , the  $(i, j)$  entry of a matrix  $\mathbf{M}$  is denoted  $m_{ij}$ , and the  $(i, j, k)$  entry of a three-way tensor  $\mathcal{T}$  is denoted  $t_{ijk}$ . Indices are scalar values that can range from 1 to a value denoted by the capitalized version of the index variable, e.g.,  $i = 1, \dots, I$ . We use MATLAB-style notation for subarrays formed from a subset of indices of a vector, matrix, or tensor mode. We use the shorthand  $i_j : i_k$  when the subset of indices forming a subarray is the range  $i_j, \dots, i_k$ . The special case of a colon  $:$  by itself indicates all elements of a mode, e.g., the  $j$ -th column of the matrix  $\mathbf{A}$  is  $\mathbf{A}(:, j) = \mathbf{A}(i_1 : i_I, j)$ . We use the *multi-index*

$$(2.1) \quad \mathbf{i} := (i_1, i_2, \dots, i_d) \quad \text{with} \quad i_j \in \{1, 2, \dots, I_j\} \quad \text{for} \quad j = 1, \dots, d,$$

as a convenient shorthand for the  $(i_1, i_2, \dots, i_d)$  entry of a  $d$ -way tensor.

Superscript  $T$  denotes non-conjugate matrix transpose. We assume vectors  $\mathbf{u}$  and  $\mathbf{v}$  are column vectors so that  $\mathbf{u}^T \mathbf{v}$  is an inner product of vectors and  $\mathbf{u} \mathbf{v}^T$  is an outer product of vectors. We also denote outer products of vectors as  $\mathbf{u} \circ \mathbf{v} = \mathbf{u} \mathbf{v}^T$ . The number of matrix or tensor non-zero elements is denoted  $\text{nnz}(\cdot)$ ; conversely, the number of zeros in a matrix or tensor is denoted  $\text{nz}(\cdot)$ .

**2.2. Canonical polyadic decomposition.** There is growing interest to extend low-rank matrix decompositions to multi-way arrays, or *tensors*. One fundamental low-rank tensor decomposition is the *canonical polyadic (CP)* decomposition. The CP tensor decomposition represents a tensor as a finite sum of rank-one outer products, a generalization of the matrix singular value decomposition (SVD) to tensors. For example, the rank- $k$  SVD of a matrix  $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$  is

$$(2.2) \quad \mathbf{A} \approx \mathbf{U}_1 \Sigma \mathbf{U}_2^T = \sum_{r=1}^k \sigma_r \mathbf{u}_1(:, r) \circ \mathbf{u}_2(:, r),$$

subject to orthogonality constraints  $\mathbf{U}_1^T \mathbf{U}_1 = \mathbf{U}_2^T \mathbf{U}_2 = \mathbf{I}_k$ , where  $\mathbf{I}_k$  is the identity matrix with  $k$  rows and columns. Note that with  $\mathbf{U}_1 = \mathbf{U}$  and  $\mathbf{U}_2 = \mathbf{V}$  we recover the standard

form of the matrix SVD:  $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ . In the parlance of CP,  $\mathbf{U}_1$  and  $\mathbf{U}_2$  are *factor matrices* corresponding to the first and second modes of  $\mathbf{A}$ , respectively. The column vectors of the factor matrices are referred to as *components*; for example,  $\mathbf{U}_1(:, 1)$  refers to the first component of the factor matrix corresponding to the first mode of  $\mathbf{A}$ . We will fix this language in the next section. In contrast to the matrix SVD, there are no orthogonality constraints on the columns of the factor matrices  $\mathbf{U}_1$  and  $\mathbf{U}_2$  of the CP decomposition; thus we treat the matrix SVD as a special case of CP decomposition. Nonetheless, low-rank CP decompositions are appealing for reasons similar to those of the low-rank SVD, including dimensionality reduction, compression, de-noising, and more. Interpretability of CP decompositions on real problems is well-documented, with applications including exploratory temporal data analysis and link prediction [18], chemometrics [47], neuroscience [4], and social network and web link analysis [39, 40].

One particular application of interest is when the tensor contains non-negative entries that are assumed to follow a Poisson distribution. In this case, the low-rank CP tensor model of Poisson parameters must satisfy certain nonnegativity and stochasticity constraints. In the next few sections we cover the details of the low-rank CP tensor models of Poisson parameters and decompositions which are the focus of this work.

**2.3. Low-rank CP tensor model.** Assume  $\mathcal{X}$  is an  $d$ -way tensor of size  $I_1 \times \cdots \times I_d$ . The tensor  $\mathcal{X}$  is rank-one if it can be expressed as the outer product of  $d$  vectors, each corresponding to a mode in  $\mathcal{X}$ , i.e.,

$$(2.3) \quad \mathcal{X} = \mathbf{a}_1 \circ \mathbf{a}_2 \circ \cdots \circ \mathbf{a}_d.$$

More broadly, the *rank* of a tensor  $\mathcal{X}$  is the smallest number of rank-one tensors that generate  $\mathcal{X}$  as their sum [40] and is the generalization of matrix rank to tensors. We concentrate on the problem of approximating a tensor of data with a low-rank CP tensor model, i.e., the sum of relatively few rank-one tensors.

Let  $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \dots, \lambda_d] \in \mathbb{R}^d$  be a vector of scalars and let  $\mathbf{A}_1 \in \mathbb{R}^{I_1 \times R}$ ,  $\mathbf{A}_2 \in \mathbb{R}^{I_2 \times R}$ ,  $\dots$ ,  $\mathbf{A}_d \in \mathbb{R}^{I_d \times R}$  be matrices. The *rank- $R$  canonical polyadic (CP) tensor model* of  $\mathcal{X}$  [15, 25] is:

$$(2.4) \quad \mathcal{X} \approx \mathcal{M} = \llbracket \boldsymbol{\lambda}; \mathbf{A}_1, \dots, \mathbf{A}_d \rrbracket = \sum_{r=1}^R \lambda_r \mathbf{A}_1(:, r) \circ \cdots \circ \mathbf{A}_d(:, r).$$

Each  $\mathbf{A}_k \in \mathbb{R}^{I_k \times R}$  is a *factor matrix* with  $I_k$  rows and  $R$  columns, which we refer to as *factors*. For example, the  $j$ -th component of the mode- $k$  factor matrix is the column vector  $\mathbf{A}_k(:, j)$ . We refer to the form  $\mathcal{M} = \llbracket \boldsymbol{\lambda}; \mathbf{A}_1, \dots, \mathbf{A}_d \rrbracket$  as a *Kruskal tensor*.

**2.4. Computing the CP decomposition for non-negative tensors.** We focus on an application where all of the entries in a data tensor are non-negative integers or *counts*. For the remainder of this work, let  $\mathcal{X} \in \mathbb{Z}_+^{I_1 \times \cdots \times I_d}$  be a  $d$ -way tensor of non-negative integers, let  $\mathcal{M}$  be a CP tensor model of the form (2.4), and assume the following about  $\mathcal{X}$ :

1. the data in  $\mathcal{X}$  are sampled from a fixed Poisson distribution,
2. the relationships of entries in  $\mathcal{X}$  can be represented by a multilinear form,
3. the tensor  $\mathcal{X}$  has low-rank structure, and

4. the rank of  $\mathbf{X}$  is known *a priori*.

Chi and Kolda showed in [17] that under these assumptions a *Poisson CP tensor model* is an effective low-rank approximation of  $\mathbf{X}$ . The Poisson CP tensor model has shown to be effective in analyzing latent patterns and relationships in count data across many application areas, including food production [13], network analysis [11, 19], term-document analysis [16, 28], email analysis [14], link prediction [18], geospatial analysis [21, 27], web page analysis [38], and phenotyping from electronic health records [26, 29, 30]

One numerical approach to fit low-rank Poisson CP tensor models to data, *tensor maximum likelihood estimation*, has proven to be effective. Computing a Poisson CP tensor model via tensor maximum likelihood estimation involves minimizing the following non-linear, non-convex optimization problem:

$$(2.5) \quad \min_{\mathbf{M}} f(\mathbf{X}, \mathbf{M}) = \min \sum_{\mathbf{i}} m_{\mathbf{i}} - x_{\mathbf{i}} \log m_{\mathbf{i}},$$

where  $\mathbf{i}$  is the multi-index (2.1),  $x_{\mathbf{i}} \geq 0$  is an entry in  $\mathbf{X}$ , and  $m_{\mathbf{i}} > 0$  is a parameter in the Poisson CP tensor model  $\mathbf{M}$ . The function  $f(\mathbf{X}, \mathbf{M})$  in (2.5) is the negative of the log-likelihood of the Poisson distribution (omitting the constant  $\sum_{\mathbf{i}} \log(x_{\mathbf{i}}!)$  term). We will refer to it simply as *negative log-likelihood (NLL)*.

In contrast to linear maximum likelihood estimation [50], where a single parameter is estimated using multiple data instances, tensor maximum likelihood estimation fits a single parameter in an approximate low-rank model to a single data instance. Within this context, low-rank structure means that multiple instances in the data are linked via the low-rank structure to a single model parameter, a sort of multilinear maximum likelihood estimation. This distinction is not made anywhere else in the literature, to the best of our knowledge. One additional clarification is that we estimate the Poisson parameters that most likely generate the data by minimizing (2.5), rather than the Poisson parameters used to sample the data. To see this, consider the univariate case of the Poisson distribution, whose probability mass function is

$$(2.6) \quad P_{\mu}(X = x) = \frac{\mu^x e^{-\mu}}{x!}.$$

If  $\mu = 1.5$  then there is a high likelihood of drawing a sample that is 1 or 2. In either case, maximum likelihood estimation tends to find the parameter that generates the sample (say  $\mu = 1.95$  if the observation is 2) than of computing the natural parameter that generated the underlying distribution of the sampled model (namely  $\mu = 1.5$ ).

Much of the research associated with computing low-rank Poisson CP tensor models via tensor maximum likelihood estimation has focused on local methods [17, 24, 31, 41], particularly with respect to computational performance [7, 8, 10, 44, 49, 52, 57]. Many of the current methods for Poisson CP tensor decomposition can be broadly classified as either an *alternating* [15, 25] or an *all-at-once* optimization method [1, 2, 51]. Alternating local methods iteratively solve a series of subproblems by fitting each factor matrix sequentially, with the remaining factor matrices held fixed. Alternating tensor decomposition methods are a form of coordinate descent (CD) [61], where each factor matrix is a block of components that are fit sequentially while the remaining factor matrices (i.e., component blocks) are held fixed. Since

each block corresponds to a lower-dimensional problem, alternating tensor methods employ block CD iteratively to solve a series of easier problems. CP Alternating Poisson Regression (CPAPR) was introduced by Chi and Kolda in [17] as a non-linear Gauss-Seidel approach to block CD that uses a fixed-point majorization-minimization algorithm called *Multiplicative Updates* (CPAPR-MU). Hansen et al. in [24] presented two Newton-based, active set gradient projection methods using up to second-order information, *Projected Damped Newton* (CPAPR-PDN) and *Projected Quasi-Newton* (CPAPR-PQN). Moreover, they provided extensions to these methods where each component block of the CPAPR minimization can be further separated into independent, highly-parallelizable row-wise subproblems; these methods are *Projected Damped Newton for the Row subproblem* (CPAPR-PDNR) and *Projected Quasi-Newton for the Row subproblem* (CPAPR-PQNR).

All-at-once optimization methods update all optimization variables simultaneously. Generalized Canonical Polyadic decomposition (GCP) [31] is a meta-method: an all-at-once optimization approach where an approximate CP tensor model is fit with arbitrary loss function via tensor maximum likelihood estimation. The original GCP method has two variants: 1) deterministic, which uses quasi-Newton optimization L-BFGS; and 2) stochastic, which uses either gradient descent (SGD) or Adam [36] optimization. We focus here on GCP-Adam [41], which applies Adam for scalability.

More generally, we focus on the GCP and CPAPR families of tensor maximum likelihood-based local methods for Poisson CP tensor decomposition for the following reasons:

1. *Theory*: Method convergence, computational costs, and memory demands are well-understood.
2. *Software*: High-level MATLAB code implementing both families is available in MATLAB Tensor Toolbox<sup>12</sup> [5, 6]. High-performance C++ code that leverages the Kokkos hardware abstraction library [20] to provide parallel computation on diverse computer architectures (e.g., x86-multicore, ARM, GPU, etc.) is available with SparTen<sup>3</sup> for CPAPR [57] and Genten<sup>4</sup> for GCP [52]. Additional open-source software for MATLAB includes N-Way Toolbox [3] and Tensorlab [60]. Commercial software includes ENSIGN Tensor Toolbox [9].

There are other approaches in the literature that seek to fit models with other distributions in the exponential family or that use other algorithms to estimate parameters. Alternating least squares methods are relatively easy to implement and effective when used with LASSO-type regularization [12, 22]. The method of Ranadive et al. [54], CP-POPT-DGN, is an all-at-once active set trust-region gradient-projection method. CP-POPT-DGN is functionally very similar to CPAPR-PDN. Whereas CP-POPT-DGN computes the search direction via preconditioned conjugate gradient (PCG), CPAPR-PDNR computes the search direction via Cholesky factorization. The most significant differences are: 1) CP-POPT-DGN is all-at-once whereas all CPAPR methods are alternating and 2) CPAPR can take advantage of the separable row subproblem formulation to achieve more fine-grained parallelism. The Generalized

---

<sup>12</sup>[https://gitlab.com/tensors/tensor\\_toolbox](https://gitlab.com/tensors/tensor_toolbox).

<sup>2</sup>See functions `cp_apr` for CPAPR-MU, PQNR, and PDNR and `gcp_opt` for GCP-SGD and Adam.

<sup>3</sup><https://gitlab.com/tensors/sparten>.

<sup>4</sup><https://gitlab.com/tensors/genten>.

Gauss-Newton method of Vandecastelle et al. [59] follows the GCP framework to fit arbitrary non-least squares loss via an all-at-once optimization and trust-region-based Gauss-Newton approach. Hu et al. [32, 33] re-parameterized the Poisson regression problem to leverage Gibbs sampling and variational Bayesian inference to account for the inability of CPAPR to handle missing data. Other problem transformations include probabilistic likelihood extensions via Expectation Maximization [34, 53] and a Legendre decomposition [56] instead of a CP decomposition.

Our final consideration is how we run CPAPR and GCP. These local methods tend to converge to local minima. We mitigate this by using a multi-start strategy [23, 46] to compute a set of approximations from many random starting points in the feasible domain of the optimization problem. From this set, we choose the “best” local minimizer—i.e., the approximation that minimizes (2.5)—as the approximation to the global optimizer. In turn, the effectiveness of a given method is determined in part by the probability it will converge to a solution approximating the global optimizer.

**2.5. Assessing approximation error.** We define several tools that we will use to compare the effectiveness of a given method in computing a model that minimizes (2.5). Let  $\mathcal{X}$  be a  $d$ -way data tensor with dimensions  $I_1, \dots, I_d$ . Let  $\mathcal{K}_N = \{\widehat{\mathcal{M}}_1, \dots, \widehat{\mathcal{M}}_N\}$  be a set of rank- $R$  Poisson CP tensor approximations computed from  $N$  random starting points by some process, e.g., CPAPR. Let  $\mathcal{M}^*$  denote the *maximum likelihood estimator (MLE)*, i.e., the global minimizer of (2.5). In general, the MLE is not known. As a result, we compute approximations of  $\mathcal{M}^*$ . Let  $\widehat{\mathcal{M}}_{\mathcal{K}_N}^*$  denote the current *approximate MLE*, i.e. the rank- $R$  Poisson CP tensor model that is the best approximation to  $\mathcal{M}^*$  from  $\mathcal{K}_N$  computed using one of the local numerical optimization methods described above:

$$(2.7) \quad \widehat{\mathcal{M}}_{\mathcal{K}_N}^* = \{\widehat{\mathcal{M}}_j \mid f(\mathcal{X}, \widehat{\mathcal{M}}_j) \leq f(\mathcal{X}, \widehat{\mathcal{M}}_k), k = 1, \dots, N, j < k\}.$$

The condition that  $j < k$  guarantees that the set is nonempty in the case of a tie. If a Poisson CP tensor model is found that is a better approximation to  $\mathcal{M}^*$  than  $\widehat{\mathcal{M}}_{\mathcal{K}_N}^*$ , then we denote it as  $\widehat{\mathcal{M}}_+^*$ .

1. The *signed relative difference in NLL* between  $\widehat{\mathcal{M}}_n$  and  $\widehat{\mathcal{M}}_{\mathcal{K}_N}^*$  is

$$(2.8) \quad \Delta_r^{(n)} := \frac{f(\mathcal{X}, \widehat{\mathcal{M}}_n) - f(\mathcal{X}, \widehat{\mathcal{M}}_{\mathcal{K}_N}^*)}{|f(\mathcal{X}, \widehat{\mathcal{M}}_{\mathcal{K}_N}^*)|}.$$

In our experiments in Section 4, along with our specific choices for the data tensor  $\mathcal{X}$ ,  $\Delta_r^{(n)} < 0$  means  $\widehat{\mathcal{M}}_n$  has lower NLL than  $\widehat{\mathcal{M}}_{\mathcal{K}_N}^*$  and that  $\widehat{\mathcal{M}}_n = \widehat{\mathcal{M}}_+^*$ . That is, it is a better minimizer of (2.5) than  $\widehat{\mathcal{M}}_{\mathcal{K}_N}^*$ . The opposite holds when  $\Delta_r^{(n)} > 0$ .

2. The probability that a given Poisson CP tensor method  $M$  converges to  $\widehat{\mathcal{M}}_n \in \mathcal{K}_N$  such that  $f(\mathcal{X}, \widehat{\mathcal{M}}_n)$  is within a ball of radius  $\epsilon > 0$  of  $f(\mathcal{X}, \widehat{\mathcal{M}}_{\mathcal{K}_N}^*)$  from any starting point in the feasible region of (2.5) is defined as

$$P_M \left( |f(\mathcal{X}, \widehat{\mathcal{M}}_n) - f(\mathcal{X}, \widehat{\mathcal{M}}_{\mathcal{K}_N}^*)| < \epsilon \right), \quad n = 1, \dots, N.$$



We estimate  $P_M$  over the set of models  $\mathcal{K}_N = \{\widehat{\mathcal{M}}_1, \dots, \widehat{\mathcal{M}}_N\}$  as

$$(2.9) \quad \widehat{P}_M(\mathcal{K}_N, \epsilon) = \frac{\#\widehat{\mathcal{M}}_n \in \mathcal{K}_N \text{ for which } |f(\mathcal{X}, \widehat{\mathcal{M}}_n) - f(\mathcal{X}, \widehat{\mathcal{M}}_{\mathcal{K}_N}^*)| < \epsilon}{N}.$$

3. We will frequently use a measure describing similarity between two Kruskal tensors based on their algebraic properties called *factor match score (FMS)* [17, 41–43]. FMS is the maximum sum of cosine similarities over all permutations of the column vectors of all the factor matrices between two Kruskal tensors,  $\widehat{\mathcal{M}} = \llbracket \widehat{\lambda}; \widehat{\mathbf{A}}_1, \dots, \widehat{\mathbf{A}}_d \rrbracket$  and  $\widetilde{\mathcal{M}} = \llbracket \widetilde{\lambda}; \widetilde{\mathbf{A}}_1, \dots, \widetilde{\mathbf{A}}_d \rrbracket$ :

$$(2.10) \quad \text{FMS}(\widehat{\mathcal{M}}, \widetilde{\mathcal{M}}) = \max_{\hat{\pi}(\cdot), \tilde{\pi}(\cdot)} \frac{1}{R} \sum_{r=1}^R \left( 1 - \frac{|\hat{\xi}_r - \tilde{\xi}_r|}{\max\{\hat{\xi}_r, \tilde{\xi}_r\}} \right) \prod_{n=1}^d \frac{\widehat{\mathbf{A}}_n(:, \hat{\pi}(j))^T \widetilde{\mathbf{A}}_n(:, \tilde{\pi}(j))}{\|\widehat{\mathbf{A}}_n(:, \hat{\pi}(j))\| \|\widetilde{\mathbf{A}}_n(:, \tilde{\pi}(j))\|},$$

where  $\hat{\xi}_r = \widehat{\lambda}_r \prod_{n=1}^d \|\widehat{\mathbf{A}}_n(:, r)\|$  and  $\tilde{\xi}_r = \widetilde{\lambda}_r \prod_{n=1}^d \|\widetilde{\mathbf{A}}_n(:, r)\|$ .

An FMS score of 1 indicates collinearity among the columns of all factor matrices and thus a perfect match between the two Kruskal tensors. As in [45], we say  $\widehat{\mathcal{M}}$  and  $\widetilde{\mathcal{M}}$  are *similar* if  $\text{FMS}(\widehat{\mathcal{M}}, \widetilde{\mathcal{M}}) \geq 0.85$  and *equal* if  $\text{FMS}(\widehat{\mathcal{M}}, \widetilde{\mathcal{M}}) \geq 0.95$ , which are common values used to define acceptable matches in recent work [17, 24, 41]. FMS is a particularly useful measure of the effectiveness of a method in relating the low-rank structure of an approximation to that of a known model. One approach that we take in Section 4 is to take the current approximate MLE from a very large set of computed models. Using FMS, we estimate the probability that a method computes models that have the same algebraic structure as the approximate MLE. We formalize this now. For each computed solution  $\widehat{\mathcal{M}}_n \in \mathcal{K}_N$ ,  $n = 1, \dots, N$ , define an indicator function  $\psi_n(\widehat{\mathcal{M}}_{\mathcal{K}_N}^*, \widehat{\mathcal{M}}_n, t)$  that is 1 when the  $n$ -th model has  $\text{FMS}(\widehat{\mathcal{M}}_{\mathcal{K}_N}^*, \widehat{\mathcal{M}}_n) \geq t$  and 0 otherwise; i.e.,

$$(2.11) \quad \psi_n(\widehat{\mathcal{M}}_{\mathcal{K}_N}^*, \widehat{\mathcal{M}}_n, t) = \begin{cases} 1, & \text{if } \text{FMS}(\widehat{\mathcal{M}}_{\mathcal{K}_N}^*, \widehat{\mathcal{M}}_n) \geq t \\ 0, & \text{otherwise.} \end{cases}$$

We use (2.11) in our discussions below to quantify the *fraction over  $N$  solves with FMS greater than  $t$* ,

$$(2.12) \quad \Psi(\widehat{\mathcal{M}}_{\mathcal{K}_N}^*, \mathcal{K}_N, t) = \frac{1}{N} \sum_{n=1}^N \psi_n(\widehat{\mathcal{M}}_{\mathcal{K}_N}^*, \widehat{\mathcal{M}}_n, t), \quad \widehat{\mathcal{M}}_n \in \mathcal{K}_N, \forall n \in \{1, \dots, N\}.$$

The *area under the curve (AUC)* associated with this proportion, i.e., FMS when  $t$  is

greater than a threshold  $\tau \in [0, 1]$ , is

$$\begin{aligned}
 \text{AUC}_{\text{FMS}}(\widehat{\mathcal{M}}_{\mathcal{K}_N}^*, \mathcal{K}_N, \tau) &= \int_{t=\tau}^{t=1} \Psi(\widehat{\mathcal{M}}_{\mathcal{K}_N}^*, \mathcal{K}_N, t) dt \\
 (2.13) \qquad &= \frac{1}{N} \int_{t=\tau}^{t=1} \sum_{n=1}^N \psi_n(\widehat{\mathcal{M}}_{\mathcal{K}_N}^*, \widehat{\mathcal{M}}_n, t) dt, \\
 &\quad \widehat{\mathcal{M}}_n \in \mathcal{K}_N, \forall n \in \{1, \dots, N\}.
 \end{aligned}$$

AUC takes real values in  $[0, 1]$ . It can be visualized as the unit square with area 0 when the fraction of  $N$  solves with FMS greater than  $t = 0$  is 0. We define the edge case to be AUC with area 1, which includes the fraction of  $N$  solves with FMS *equal* to 1.

**3. GCP-CPAPR Hybrid Method.** We develop Cyclic GCP-CPAPR (CGC), a hybrid Poisson CP tensor method that *cycles* between a stochastic method to compute a model approximation and a deterministic method to resolve the model to the best accuracy possible at scale. We define parameterizations and cycle *strategies* in [Section 3](#), which prescribe how CGC iterates in each cycle. In [Subsection 3.1](#) we introduce the concept of *policies*, automated updates to strategies.

For simplicity of exposition, we assume access to a wrapper function `cp_poisson()` as an interface to different Poisson CP solvers with the following usage:

- `M = cp_poisson( X, R, Minit, METHOD, OPTS )` is a Poisson CP tensor model `M` with `R` components,
- `Minit` is an initial guess to `M`,
- `METHOD` specifies the Poisson CP tensor algorithm,
- `OPTS` specifies parameters used by `METHOD`.

Let  $L \in \mathbb{N}$  be a number of *cycles*. Define *strategy* to be the  $L$ -length array of structures, `strat`, specifying the following for each cycle  $l \in \{1, \dots, L\}$ :

- `S_method`: stochastic search method, e.g. GCP-Adam.
- `D_method`: deterministic search method, e.g. CPAPR-MU.
- `S_opts`: stochastic search parameterization, including stochastic search budget,  $j$ , measured in *epochs*.
- `D_opts`: deterministic search parameterization, including deterministic search budget,  $k$ , measured in *iterations*.

CGC iterates from an initial guess  $\widehat{\mathcal{M}}^{(0)}$  via a two-stage alternation between stochastic and deterministic search for  $L$  cycles to return a Poisson CP tensor approximation  $\widehat{\mathcal{M}}^{(L)}$ . In the first stage of the  $l$ -th cycle, stochastic search method `strat(l).S_method` is run starting from  $\widehat{\mathcal{M}}^{(l-1)}$  for  $j$  epochs, parameterized by `strat(l).S_opts` to return an intermediate solution,  $\widetilde{\mathcal{M}}^{(l)}$ . In the second stage, deterministic search method `strat(l).D_method` is run to refine  $\widetilde{\mathcal{M}}^{(l)}$  for  $k$  iterations, parameterized by `strat(l).D_opts`, to return the  $l$ -th iterate,  $\widehat{\mathcal{M}}^{(l)}$ . The details of CGC are given below in [Algorithm 3.1](#). We only consider stochastic search followed by deterministic search in each cycle and not the opposite. Since stochastic search directions are found using estimates of the objective function from sample points, it is possible that



the algorithm converges to a minimum yet remains marked as *not converged* if the objective function value is only coarsely estimated. Subsequently, it is likely that stochastic search will move away from the optimum.

---

**Algorithm 3.1** Cyclic GCP-CPAPR (CGC)

---

```

1: function CGC(Sparse tensor of count data  $\mathcal{X} \in \mathbb{R}^{I_1 \times \dots \times I_d}$ , number of components  $R$ ,
   number of cycles  $L$ ,  $L$ -array of structures strat defining  $L$  strategies, initial guess for
    $R$ -component Kruskal tensor model  $\widehat{\mathcal{M}}^{(0)}$ .)
2:   for  $l = 1, \dots, L$  do
3:      $\widetilde{\mathcal{M}}^{(l)} \leftarrow \text{cp\_poisson}(\mathcal{X}, R, \widehat{\mathcal{M}}^{(l-1)}, \text{strat}(l).\text{S\_method}, \text{strat}(l).\text{S\_opts});$ 
4:      $\widehat{\mathcal{M}}^{(l)} \leftarrow \text{cp\_poisson}(\mathcal{X}, R, \widetilde{\mathcal{M}}^{(l)}, \text{strat}(l).\text{D\_method}, \text{strat}(l).\text{D\_opts});$ 
5:   return Kruskal tensor model  $\widehat{\mathcal{M}}^{(L)}$ .

```

---

**3.1. Cycle policies.** A *static policy* prescribes the cycle strategies before runtime. However, we also are interested in developing a strategy that changes dynamically based on convergence metrics collected at runtime, which we call an *adaptive policy*. We leave a comprehensive investigation of their impact on CGC performance to future work.

**4. Numerical Experiments.** In this section we present the results of preliminary numerical experiments with CGC. [Subsection 4.1](#) compares GCP-Adam and CPAPR-MU, both using out-of-the-box parameter settings on a real-world dataset, which motivates our inquiry into CGC. The experiments in [Subsection 4.2](#) use parameters for GCP-Adam and CPAPR-MU taken from prior work [48], also on real-data. The experiments in [Subsection 4.3](#) reflect controlled experiments of CGC on synthetic data.

Each experiment assumes access to either the CGC solver (i.e. [Algorithm 3.1](#)) or `cp_poisson` mentioned in [Section 3](#). Additionally, each experiment assumes access to a function `K = create_guess([I1, ..., Id], R)` that generates a random initial guess Kruskal tensor according to the process described in [17, §6.1] taking inputs: 1) the row dimensions of the factor matrices,  $[I_1, \dots, I_d]$ , and 2) number of components,  $R$ , i.e., the column dimension for all factor matrices in the Kruskal tensor. A reference implementation can be found in MATLAB Tensor Toolbox [6].<sup>5</sup> We provide the experimental setups in MATLAB-style syntax throughout.

**4.1. Default parameterization.** Our first experiment establishes baseline results for GCP-Adam and CPAPR-MU using only default parameters. Briefly, both methods are called to compute a rank-10 CP decomposition provided the same initial guess. The stochastic search method is set to “GCP-Adam” and the deterministic search method is set to “CPAPR-MU”; no other method options are specified. The results are reported in [Table 1](#) as 1) the fraction of solves with FMS greater than 0.95 and 2) runtime. We note that CPAPR-MU converges to the approximate MLE a higher fraction of times than GCP-Adam. We caution that the dismal performance of GCP-Adam compared to CPAPR-MU is directly attributable to the use of default software parameters. In private correspondence,<sup>6</sup> the authors of GCP-Adam

---

<sup>5</sup>See [https://gitlab.com/tensors/tensor\\_toolbox/-/blob/master/create\\_guess.m](https://gitlab.com/tensors/tensor_toolbox/-/blob/master/create_guess.m).

<sup>6</sup>T.G. Kolda, email to author, March 8, 2021 .

recommended against the use of default parameters.

Table 1: Baseline results using GCP-Adam and CPAPR-MU on *Chicago Crime* data tensor [55].  $\Psi(\widehat{\mathcal{M}}_{\mathcal{K}_N}^*, \mathcal{K}_{20}, 0.95)$  is the fraction of solves “equal” to the MLE from 20 random initial guesses. *Time (sec)* is the median time in seconds to solution across all random starts.

	GCP-Adam	CPAPR-MU
$\Psi(\widehat{\mathcal{M}}_{\mathcal{K}_N}^*, \mathcal{K}_{20}, 0.95)$	0.00	0.35
Time (sec)	1837	842

**4.2. GCP→CPAPR.** In this section, we show that computing an initial approximation with GCP-Adam and refining it with CPAPR-MU, which we refer to simply as GCP→CPAPR, produces good approximations more efficiently than GCP-Adam alone. We compute rank-10 Poisson tensor approximations of the *Chicago Crime* [55] sparse data tensor from 20 random starting points. The parameterizations of stochastic and deterministic search are specified as fields in the structure `strat`, following the conventions provided in Section 3. The parameterizations for both stochastic and deterministic search methods were taken from [48]. In the first stage, GCP-Adam is run with initial learning rate  $\alpha_0 = 10^{-3}$  rate until final learning rate  $\alpha_f = 10^{-15}$ . Solutions are checkpointed after the last iteration using learning rates  $\alpha_0$  and  $\alpha_f$ , as well as the midpoint learning rate,  $\alpha_m = 10^{-9}$ .

The fraction of solves equal to the approximate MLE,  $\Psi(\widehat{\mathcal{M}}_{\mathcal{K}_N}^*, \mathcal{K}_{20}, 0.95)$  column in Table 2, increases as GCP performs more work. This is expected behavior; see the *Stage 1* column. Since the  $\alpha_0$  is fixed across all experiments and since the stochastic learning rate is reduced monotonically, *smaller* final learning rate  $\alpha_f$  means *more* GCP-Adam epochs, i.e., *more GCP-Adam work*. In the second stage, we pass each approximation from the first stage to CPAPR-MU as an initial guess and iterate until KKT-based convergence, which was set to the default tolerance  $10^{-4}$ .

The fraction of solves equal to the approximate MLE after the second stage are reported in the *Stage 2* column. Beginning with a small amount of first stage computation (learning =  $10^{-3}$ ), second stage CPAPR-MU computes additional equal approximations, from 0.25 to 0.30 of solves. To achieve the 0.30 level, first stage GCP-Adam must be run to learning rate  $10^{-9}$ , which requires *more wall clock time* (see: Stage 1 with learning rate  $10^{-9}$  versus Stage 2 with learning rate  $10^{-3}$ ). Second stage CPAPR-MU computes additional equal approximations with first stage initial guesses computed to this learning rate, increasing from 0.30 to 0.45 fraction of solves. Similarly, to achieve the 0.45 level, first stage GCP-Adam must be run to learning rate  $10^{-15}$ , which again requires more wall clock time than the second stage with learning rate  $10^{-9}$ . Second stage CPAPR-MU finds that all first stage approximations computed to learning rate  $10^{-15}$  have converged in its termination criterion.

The *Time Ratio Stage 2/Stage 1* column reports the ratio of median wall clock times of Stage 2 over Stage 1. As the learning rate becomes smaller, the second stage requires less work. This scaling is promising because it is indicative of a computational cost-approximation error trade-off that can be optimized for with a less naive parameterization. At the lowest

Table 2: Quality of GCP→CPAPR as a function of stochastic learning rate on *Chicago Crime* data tensor [55] from 20 random starting points. Stage 1 refers to GCP-Adam computations; Stage 2 refers to CPAPR-MU computations.  $\Psi(\widehat{\mathcal{M}}_{\mathcal{K}_N}^*, \mathcal{K}_{20}, 0.95)$  is the fraction of random starts converging to solutions “equal” to the numerical Kruskal tensor with the lowest NLL. *Time (sec.)* is the median time to solution across all random starts. We note that in these experiments the GCP-Adam solver used tuned stochastic sampling values to compute better model approximations than those in Table 1, hence the discrepancy.

learning rate	$\Psi(\widehat{\mathcal{M}}_{\mathcal{K}_N}^*, \mathcal{K}_{20}, 0.95)$		Time (sec.)		Time ratio Stage 2/Stage 1
	Stage 1	Stage 2	Stage 1	Stage 2	
$\alpha_0 = 10^{-03}$	0.25	0.30	556	1342	2.4
$\alpha_m = 10^{-09}$	0.30	0.45	1841	2643	1.4
$\alpha_f = 10^{-15}$	0.45	0.45	2850	3649	1.3

learning rate,  $10^{-15}$ , the second stage incurs computational overhead to verify the first stage approximations converged in its criterion, even if no CPAPR-MU iterations take place. This, coupled with the discussion above regarding wall clock time, demonstrates a diminishing return of performing too much GCP work in the first stage followed by second stage CPAPR-MU. We conclude from the above discussion that *two-stage GCP→CPAPR can achieve the same quality of solution with less work than GCP-Adam alone on real-world data with proper parameterization*.

**4.3. Single-cycle, constant-work unit parameterization.** The experiments in Subsection 4.2 provide cursory evidence that GCP→CPAPR can produce better results than CPAPR-MU or GCP-Adam alone. However, we must view those experiments as purely anecdotal since real-world data can be only hypothesized to follow the assumptions in subsection 2.4. In this section, we report on preliminary controlled experiments with the full CGC method described in section 3 using synthetic low-rank Poisson multilinear data.

The experiments that follow are restricted to Algorithm 3.1 with  $L = 1$  cycles and a static strategy, detailed below. The study of Algorithm 3.1 with  $L > 1$  cycles and adaptive strategies is left to future work. We first describe the data in Subsection 4.3.1 and methodology in Subsection 4.3.2. We report numerical results in Subsection 4.3.3.

**4.3.1. Data.** We limit our numerical experiments to a single synthetic dataset that is sufficiently challenging for our methods—in terms of size, sparsity, and low-rank structure—yet small enough to support reasonable solution times. We generated a  $1000 \times 1000 \times 1000$  rank-20 tensor of non-negative integers drawn from a Poisson distribution with 98,026 non-zero entries (approximately 0.01% dense) using the function `create_problem`<sup>7</sup> in MATLAB Tensor Toolbox. The function `create_problem` generates a “true” solution Poisson CP tensor model, which is sampled to generate the synthetic sparse count tensor. In our experiments below, we do not attempt to recover this model. See subsection 2.4 for further discussion.

<sup>7</sup>See [https://gitlab.com/tensors/tensor\\_toolbox/-/blob/master/create\\_problem.m](https://gitlab.com/tensors/tensor_toolbox/-/blob/master/create_problem.m).

#### 4.3.2. Methodology.

*Out-of-sample test set.* For our experiments, we first generated 10,000 random Poisson CP tensor models as initial guesses using `create_problem`. Starting from each random initial guess, we computed a rank- $R$  Poisson CP tensor approximation using GCP-Adam and CPAPR-MU, which were both run to the smallest reasonable tolerance of each method: final learning rate  $\alpha_f = 10^{-15}$  for GCP-Adam and KKT violation  $\tau = 10^{-15}$  for CPAPR-MU. We refer to this collection of  $S = 20,000$  Poisson CP tensor approximations as the *out-of-sample test set*,  $\mathcal{T}_S$ . From these 20,000 Poisson CP tensor approximations, we chose the model with the lowest NLL as the current approximate MLE, denoted  $\widehat{\mathbf{M}}_{\mathcal{T}_S}^*$ , with objective function value  $f^*$ .

*CGC run.* In these experiments, we fix  $W = 100$  so that the number of GCP-Adam epochs and CPAPR-MU iterations sum to  $W$  in the first (and only) cycle. Starting from a random initial guess, we run [Algorithm 3.1](#) for  $j$  GCP-Adam epochs and  $k$  CPAPR-MU iterations, with  $j \in \{0, \dots, W\}$  and  $k = W - j$ . We refer to the combination of GCP-Adam epochs and CPAPR-MU iterations as the CGC  $(j, k)$  pair for convenience. Note that when  $j = 0$ , CGC with  $L = 1$  cycles is equivalent to a single run of 100 CPAPR iterations; conversely, when  $j = 100$  so that  $k = 0$ , CGC with  $L = 1$  cycles is equivalent to a single run of 100 GCP epochs. We generated  $N = 100$  random initial guesses, which were different from the random initial guesses chosen to compute the 20,000 models in  $\mathcal{T}_S$ . We believe this aids the statistical interpretability of our results. The set of these 10,100 CGC model approximations is denoted  $\mathcal{K}_N$ .

**4.3.3. Numerical results.** We contrast CGC with GCP-Adam and CPAPR-MU alone using the tools in [Subsection 2.5](#). [Figure 1](#) plots the minimum signed relative difference (2.8) of the approximations for each CGC  $(j, k)$ . Negative values (in blue) are models from  $\mathcal{K}_N$  that are closer to  $\mathbf{M}^*$  than  $\widehat{\mathbf{M}}_{\mathcal{T}_S}^*$ ; positive values (in red) are the opposite. We see three regimes. In the first, the best CGC approximation is worse than the approximate MLE for CGC with little to no work budget allocation to CPAPR. The second regime demonstrates that some amount of stochastic search followed by deterministic search can compute Poisson CP tensor approximations that are better minimizers of (2.5) than the approximate MLE. This holds for stochastic work budget allocations between 4–36%. There is no clear pattern in the third regime, where the stochastic work budget is above 36%.

Our second result considers whether CGC finds a good approximation to the global optimizer with the same accuracy or better than GCP-Adam or CPAPR-MU. In [Table 3](#) we report the fraction of model approximations near to the approximate global optimizer, computed using (2.9). For CGC results, we compute  $\widehat{P}_{CGC}(\widehat{\mathbf{M}}, \epsilon)$  for each combination of GCP-Adam epochs  $j$  and CPAPR-MU iterations  $k$  across  $N = 100$  random initializations, but report only the maximum of (2.9) among all combinations. The rightmost column lists the corresponding best  $(j, k)$  pair. Within radius  $\epsilon = 10^{-2}$ , CPAPR-MU has the highest probability of computing solutions with NLL close to the  $f(\mathcal{X}, \widehat{\mathbf{M}}_{\mathcal{T}_S}^*)$ . Note that the best CGC result for this level of  $\epsilon$  is equivalent to CPAPR-MU since the GCP-Adam work allocation is zero, i.e.,  $(j, k) = (0, 100)$ . The lower CGC proportion in [Table 3](#) (0.65 versus 0.69) is likely attributed to either 1) smaller computational budget of the CGC run versus CPAPR-MU run in the out-of-sample test (100 vs. 1000 iterations) or 2) higher variance from fewer multi-starts in the

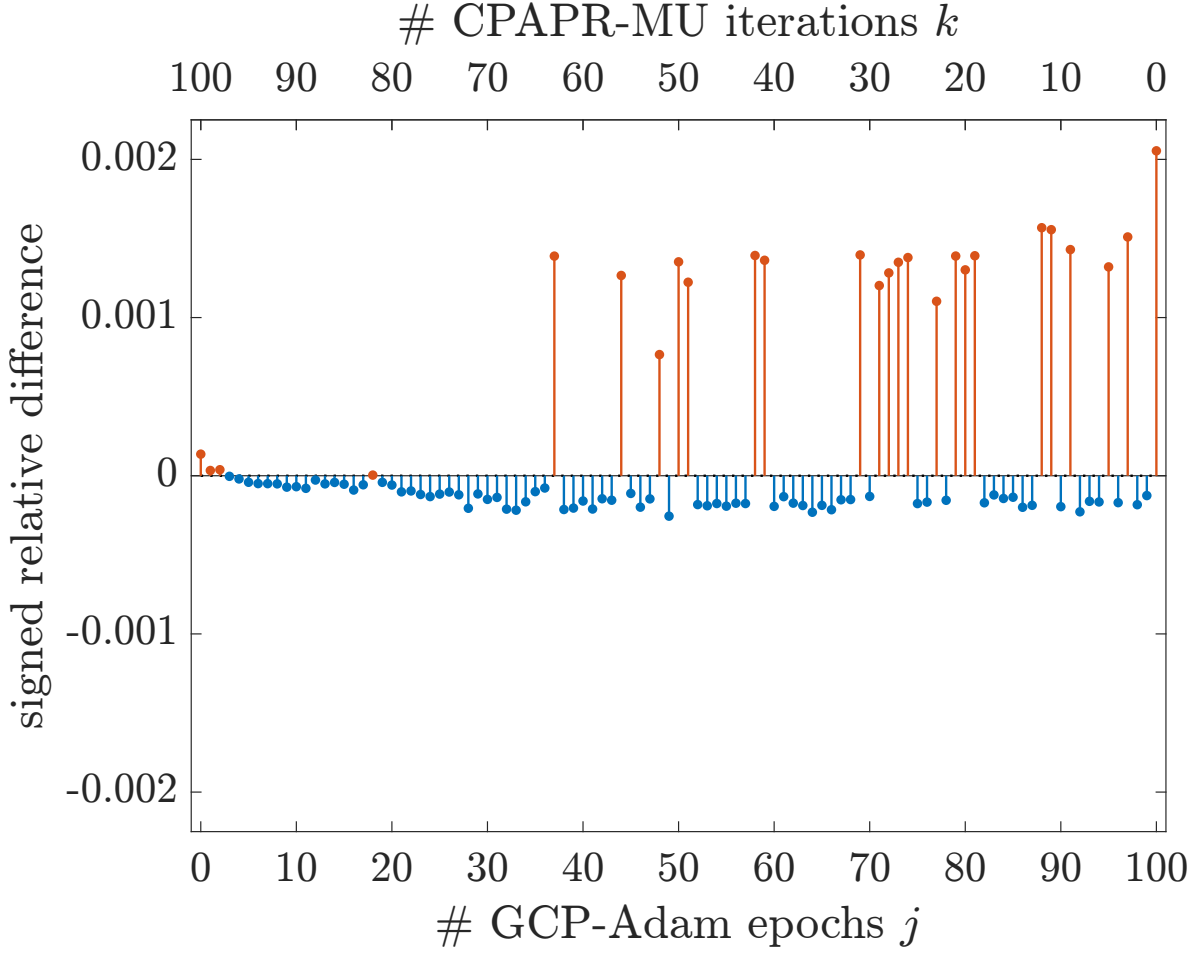


Figure 1: Best signed relative difference in NLL (2.8) of CGC model approximations  $\widehat{\mathcal{M}}_n \in \mathcal{K}_N$  for various  $(j, k)$  versus the approximate MLE,  $\widehat{\mathcal{M}}_{\mathcal{T}_S}^*$ . The data points along the  $x$ -axis correspond to the minimum signed relative difference in NLL  $\Delta_r^{(n)}$  among all  $n = 1, \dots, 100$  random initial guesses for each CGC  $(j, k)$  pair. Negative (blue) values indicate the best  $\widehat{\mathcal{M}}_n$  for a given  $(j, k)$  pair is a better minimizer than  $\widehat{\mathcal{M}}_{\mathcal{T}_S}^*$ ; positive (red) values indicate the opposite.

CGC run versus the out-of-sample test (100 vs. 10000 multi-starts). With radius  $\epsilon \leq 10^{-3}$ , CGC computes Poisson CP tensor approximations with NLL closest to  $f(\mathcal{X}, \widehat{\mathcal{M}}_{\mathcal{T}_S}^*)$  with the highest probability. An interesting trend is that as the radius  $\epsilon$  gets smaller, the best CGC parameterization corresponds to increasing the allocation to GCP steps. We will see similar patterns in the analyses that follow.

Lastly, we evaluate CGC as a method to approximate the global optimizer in terms of the algebraic fit of CGC models to the MLE using measures defined by equations (2.11)–(2.13). For each constant work unit allocation  $(j, k)$ , we compute the fraction over  $N$  solves with FMS

Table 3: Estimate of probability each method computes a solution within  $\epsilon$ -radius of approximate global optimizer. *CPAPR* is short for CPAPR-MU; *GCP* is short for GCP-Adam.

$\epsilon$	$\hat{P}_{CPAPR}(\mathcal{T}_S, \epsilon)$	$\hat{P}_{GCP}(\mathcal{T}_S, \epsilon)$	$\hat{P}_{CGC}(\mathcal{K}_N, \epsilon)$	best CGC $(j, k)$ pair
$10^{-1}$	1.00	1.00	1.00	all
$10^{-2}$	0.69	0.27	0.65	(0, 100)
$10^{-3}$	0.05	0	0.16	(1, 99)
$10^{-4}$	< 0.01	0	0.13	(4, 96)
$10^{-5}$	0	0	0.03	(8, 92)
$10^{-6}$	0	0	0.01	(8, 92)

greater than  $t$ ,  $\Psi(\widehat{\mathcal{M}}_{\mathcal{T}_S}^*, \mathcal{K}_N, t)$ , with  $t \in [0, 1]$ . Since all curves showed the same behavior for  $t < 0.5$ , we plot values for  $t \in [0.5, 1]$  in Figure 2, together with the same values computed for GCP-Adam and CPAPR-MU. A detailed view of the same data is presented in Figure 3, which partitions the data by the first 11  $(j, k)$  pairs (Figure 3a), the 79 middle pairs (Figure 3b), and the last 11 pairs (Figure 3c). As described earlier, the differences in the plots Figure 2–Figure 3 between CGC with  $(j, k) = (0, 100)$  and CPAPR-MU (blue triangles) and CGC with  $(j, k) = (100, 0)$  and GCP-Adam (green circles) is mainly a function of the work constraint of CGC. CGC was constrained to 100 work units in both cases, whereas in the out-of-sample test that generated the GCP-Adam and CPAPR-MU results, GCP-Adam ran for a maximum of 10,000 epochs and CPAPR-MU ran for a maximum of 1000 iterations. We call out this distinction to highlight the results in these figures. Especially in Figure 3a and Figure 3b, we see a higher proportion of CGC solutions equal to the approximate MLE than GCP-Adam or CPAPR-MU despite the computational constraint. Thus, these figures provide evidence that some amount of stochastic search followed by deterministic search can find models with superior fit and higher accuracy than GCP-Adam and CPAPR-MU by themselves. This conclusion is particularly clear for scores in the range  $\geq 0.95$ .

We also compute the area under each curve (2.13) in Figure 2 as a metric that estimates the probability of finding a model with good algebraic fit to the MLE from the out-of-sample test set. Figure 4 summarizes model accuracy as the area under the curves in Figure 2–Figure 3 for “similar” and “equal” fits. We can conclude that CGC has a higher probability of computing approximations that are equal to the MLE for a range of GCP-CPAPR work budget allocations than GCP-Adam or CPAPR-MU alone.

**5. Conclusions and Future Work.** The results in section 4 beg for further inquiry into CGC. Our primary conclusion is *CGC can minimize low-rank approximation error while reducing computational costs*. We saw that CGC can reduce approximation error with high accuracy relative to GCP-Adam and CPAPR-MU. Additionally, since CGC was run in our experiments with a far stricter computational budget than GCP-Adam and CPAPR-MU—CGC was constrained to 100 total CGC work units whereas GCP-Adam and CPAPR-MU runs were allowed for far longer to run to very low convergence tolerance—we argue that CGC is more computationally efficient. The implication is that the performance gain allows



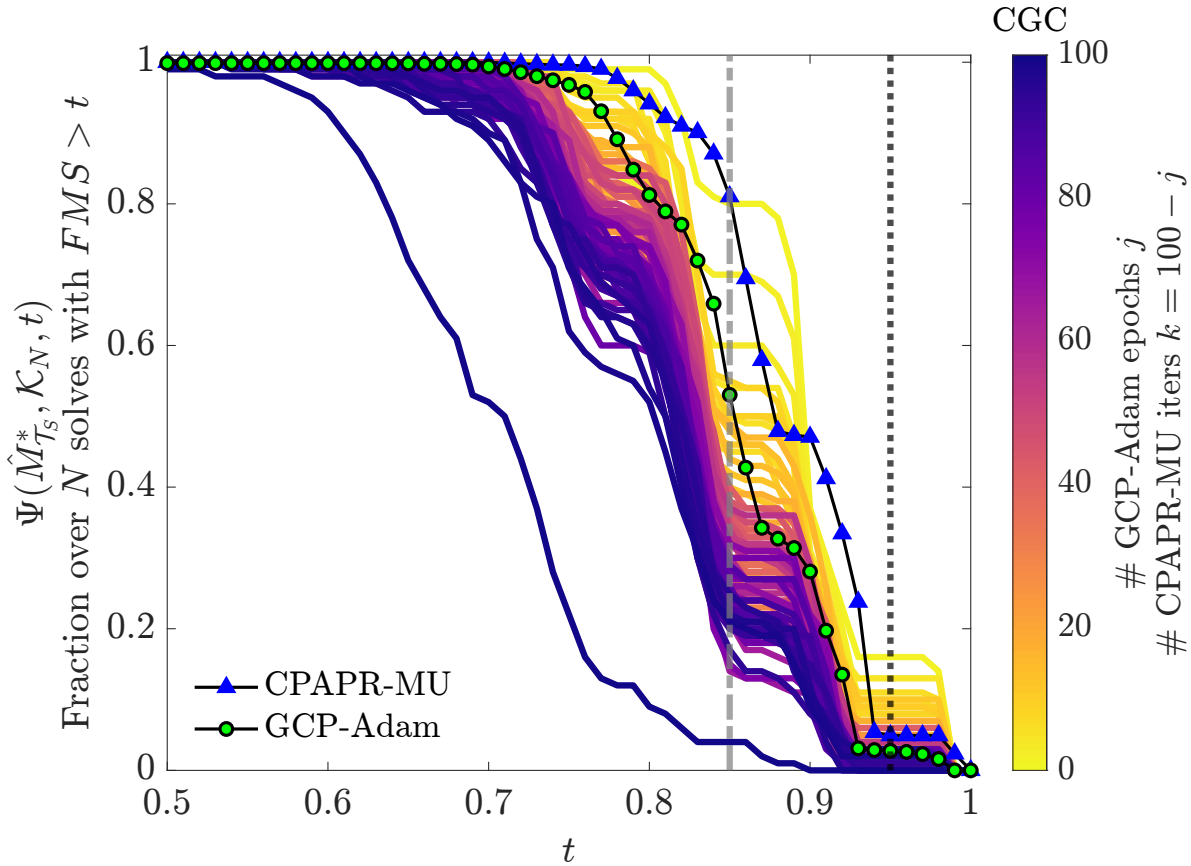


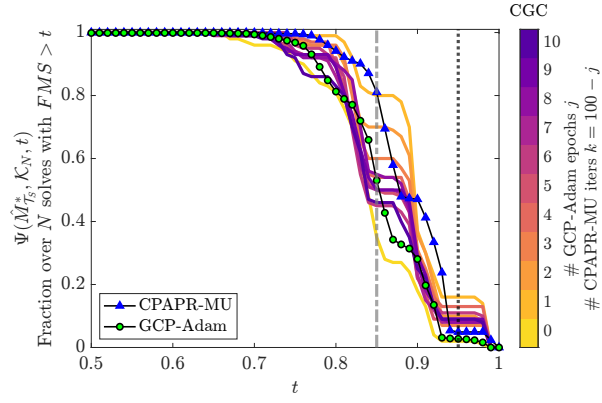
Figure 2: Factor match scores between CP models computed with CGC, CPAPR-MU, and GCP-Adam and the approximate global optimizer,  $\widehat{\mathcal{M}}_{\mathcal{T}_S}^*$ . The dash-dot gray vertical lines and dotted black vertical lines denote the levels of “similar” and “equal” described in [45].

even more multi-starts, and subsequently, a greater number of high accuracy approximations. Motivated by these observations, we propose to explore the following ideas.

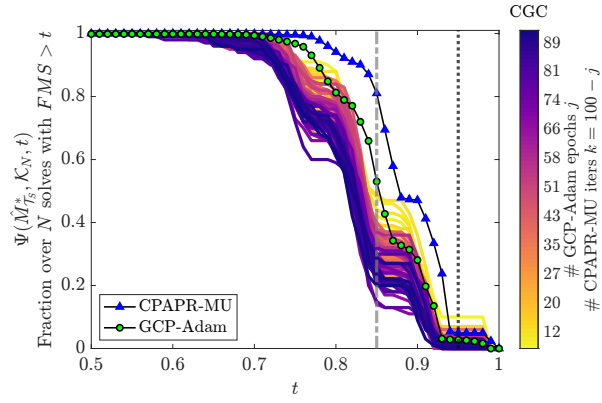
**5.1. Parameterizing CGC.** How to parameterize CGC in such a way that model approximation error be reduced and computational cost be optimized further? Our numerical experiments showed promising results despite a naive strategy. We will extend our ideas to running CGC with  $L > 1$  cycles to study the effect of adaptive versus static per-cycle strategies. A goal is to condition the strategy updates on convergence metrics, which will require deeper understanding of both stochastic search and deterministic refinement.

**5.2. Comparison with other black-box methods.** CGC is similar in spirit to Simulated Annealing [37]: stochastic search “heats” the iterative path away from local minima and deterministic search “cools” the iterative path towards the global optimum. In addition to comparisons with GCP-Adam and CPAPR-MU, we believe a direct comparison to Simulated Annealing, and potentially other global optimization methods, is necessary. This can be done

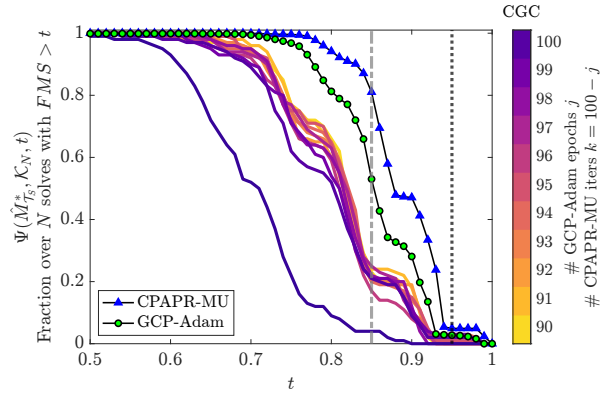




(a) First 11 pairs.



(b) Middle 79 pairs.



(c) Last 11 pairs.

Figure 3: Detail of curves in Figure 2. Colormaps scaled for clarity.

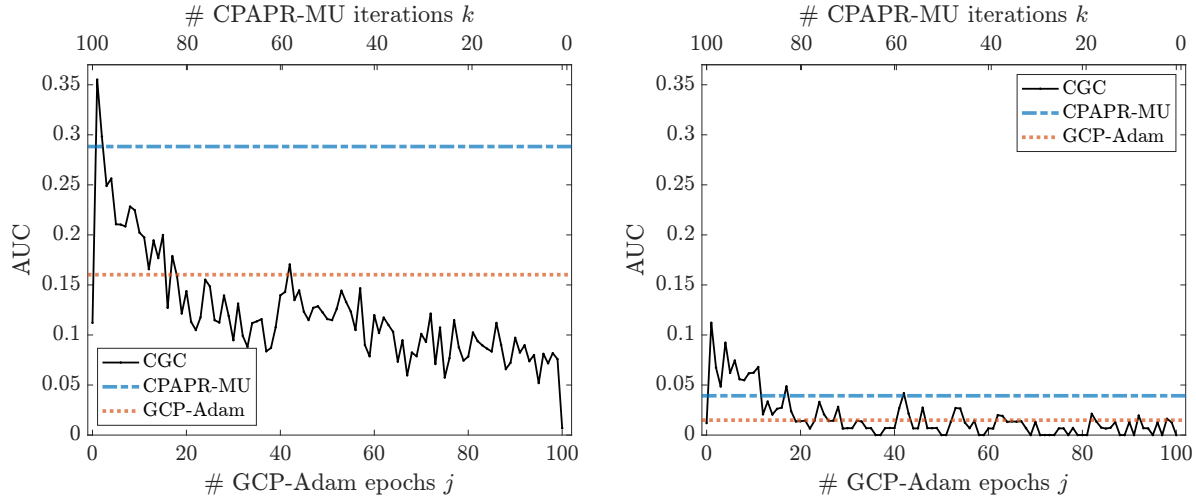


Figure 4: For each CGC  $(j, k)$  pair, areas under the curve  $\text{AUC}_{\text{FMS}}(\widehat{\mathcal{M}}_{\mathcal{T}_S}^*, \mathcal{K}_N, 0.85)$  (left) and  $\text{AUC}_{\text{FMS}}(\widehat{\mathcal{M}}_{\mathcal{T}_S}^*, \mathcal{K}_N, 0.95)$  (right). The area under the CPAPR and GCP curves across all 10,000 random starts are displayed as horizontal lines. All curves in both plots are normalized by the area of the fit level (0.15 and 0.05, respectively).

easily using the standard Simulated Annealing method from MATLAB Global Optimization Toolbox [58] which is an implementation of Adaptive Simulated Annealing [35].

**Acknowledgments.** We thank Eric Phipps of Sandia National Laboratories for assistance with Genten, a high-performance GCP solver.

## REFERENCES

- [1] E. ACAR, D. M. DUNLAVY, AND T. G. KOLDA, *A scalable optimization approach for fitting canonical tensor decompositions*, Journal of Chemometrics, 25 (2011), pp. 67–86.
- [2] E. ACAR, T. G. KOLDA, AND D. M. DUNLAVY, *All-at-once Optimization for Coupled Matrix and Tensor Factorizations*, May 2011, <https://arxiv.org/abs/1105.3422v1>.
- [3] C. ANDERSEN AND R. BRO, *The N-way Toolbox for Matlab*, Chemometrics and Intelligent Laboratory Systems, 52 (2000), pp. 1–4, [https://doi.org/10.1016/S0169-7439\(00\)00071-X](https://doi.org/10.1016/S0169-7439(00)00071-X).
- [4] C. ANDERSEN AND R. BRO, *Practical aspects of PARAFAC modeling of fluorescence excitation-emission data*, Journal of Chemometrics, 17 (2003), pp. 200–215, <https://doi.org/10.1002/cem.790>.
- [5] B. BADER AND T. KOLDA, *Efficient MATLAB Computations with Sparse and Factored Tensors*, SIAM Journal on Scientific Computing, 30 (2008), pp. 205–231.
- [6] B. W. BADER, T. G. KOLDA, ET AL., *MATLAB Tensor Toolbox Version 3.0-dev*, Aug. 2017.
- [7] M. BASKARAN, T. HENRETTY, AND J. EZICK, *Fast and Scalable Distributed Tensor Decompositions*, in 2019 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, Sept. 2019, IEEE, pp. 1–7, <https://doi.org/10.1109/HPEC.2019.8916319>.
- [8] M. BASKARAN, T. HENRETTY, B. PRADELLE, M. H. LANGSTON, D. BRUNS-SMITH, J. EZICK, AND R. LETHIN, *Memory-efficient parallel tensor decompositions*, in 2017 IEEE High Performance Extreme Computing Conference (HPEC), 2017, pp. 1–7.
- [9] M. BASKARAN, D. LEGGAS, B. VON HOFER, M. H. LANGSTON, J. EZICK, AND P.-D. LETOURNEAU, *ENSGN*. [Computer Software] <https://doi.org/10.11578/dc.20220120.1>, Jan. 2022, <https://doi.org/>

- 10.11578/dc.20220120.1.
- [10] M. BASKARAN, B. MEISTER, AND R. LETHIN, *Low-overhead load-balanced scheduling for sparse tensor computations*, in 2014 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, Sept. 2014, IEEE, pp. 1–6, <https://doi.org/10.1109/HPEC.2014.7041006>.
  - [11] M. M. BASKARAN, T. HENRETTY, J. EZICK, R. LETHIN, AND D. BRUNS-SMITH, *Enhancing Network Visibility and Security through Tensor Analysis*, *Future Generation Computer Systems*, 96 (2019), pp. 207–215.
  - [12] J. A. BAZERQUE, G. MATEOS, AND G. B. GIANNAKIS, *Inference of Poisson count processes using low-rank tensor data*, in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 5989–5993, <https://doi.org/10.1109/ICASSP.2013.6638814>.
  - [13] R. BRO, *Multiway Analysis in the Food Industry. Models, Algorithms and Applications*, PhD thesis, University of Amsterdam, 1998.
  - [14] B. W. B. B.W, M. W. BERRY, AND M. BROWNE, *Survey of Text Mining II*, Springer, London, 2008, ch. Discussion Tracking in Enron Email Using PARAFAC.
  - [15] J. D. CARROLL AND J.-J. CHANG, *Analysis of individual differences in multidimensional scaling via an  $n$ -way generalization of “Eckart-Young” decomposition*, *Psychometrika*, 35 (1970), pp. 283–319.
  - [16] P. A. CHEW, B. W. BADER, T. G. KOLDA, AND A. ABDELALI, *Cross-language Information Retrieval using PARAFAC2*, in KDD ’07: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2007, pp. 143–152.
  - [17] E. C. CHI AND T. G. KOLDA, *On Tensors, Sparsity, and Nonnegative Factorizations*, *SIAM Journal on Matrix Analysis and Applications*, 33 (2012), pp. 1272–1299.
  - [18] D. M. DUNLAVY, T. G. KOLDA, AND E. ACAR, *Temporal Link Prediction using Matrix and Tensor Factorizations*, *ACM Transactions on Knowledge Discovery from Data*, 5 (2011), p. 10 (27 pages).
  - [19] D. M. DUNLAVY, T. G. KOLDA, AND W. P. KEGELMEYER, *Multilinear Algebra for Analyzing Data with Multiple Linkages*, in *Graph Algorithms in the Language of Linear Algebra*, J. Kepner and J. Gilbert, eds., *Fundamentals of Algorithms*, SIAM, 2011, pp. 85–114.
  - [20] H. C. EDWARDS, C. R. TROTT, AND D. SUNDERLAND, *Kokkos: Enabling manycore performance portability through polymorphic memory access patterns*, *Journal of Parallel and Distributed Computing*, 74 (2014), pp. 3202–3216.
  - [21] J. EZICK, T. HENRETTY, M. BASKARAN, R. LETHIN, J. FEO, T.-C. TUAN, C. COLEY, L. LEONARD, R. AGRAWAL, B. PARSONS, AND W. GLODEK, *Combining Tensor Decompositions and Graph Analytics to Provide Cyber Situational Awareness at HPC Scale*, in 2019 IEEE High Performance Extreme Computing Conference (HPEC), 2019, pp. 1–7.
  - [22] M. P. FRIEDLANDER AND K. HATZ, *Computing non-negative tensor factorizations*, *Optimization Methods and Software*, 23 (2008), pp. 631–647, <https://doi.org/10.1080/10556780801996244>.
  - [23] A. GY\`{ORGY AND L. KOC SIS, *Efficient Multi-Start Strategies for Local Search Algorithms*, *Journal of Artificial Intelligence Research*, 41 (2011), pp. 705–720.
  - [24] S. HANSEN, T. PLANTENGA, AND T. G. KOLDA, *Newton-based optimization for Kullback–Leibler non-negative tensor factorizations*, *Optimization Methods and Software*, 30 (2015), pp. 1002–1029.
  - [25] R. A. HARSHMAN, *Foundations of the PARAFAC procedure: Models and conditions for an “explanatory” multi-modal factor analysis*, *UCLA Working Papers in Phonetics*, 16 (1970), pp. 1–84.
  - [26] J. HENDERSON, J. C. HO, A. N. KHO, J. C. DENNY, B. A. MALIN, J. SUN, AND J. GHOSH, *Granite: Diversified, Sparse Tensor Factorization for Electronic Health Record-Based Phenotyping*, in 2017 IEEE International Conference on Healthcare Informatics (ICHI), 2017, pp. 214–223, <https://doi.org/10.1109/ICHI.2017.61>.
  - [27] T. HENRETTY, M. BASKARAN, J. EZICK, D. BRUNS-SMITH, AND T. A. SIMON, *A quantitative and qualitative analysis of tensor decompositions on spatiotemporal data*, in 2017 IEEE High Performance Extreme Computing Conference (HPEC), 2017, pp. 1–7.
  - [28] T. S. HENRETTY, M. H. LANGSTON, M. BASKARAN, J. EZICK, AND R. LETHIN, *Topic modeling for analysis of big data tensor decompositions*, in *Disruptive Technologies in Information Sciences*, vol. 10652, 2018, pp. 52–64.
  - [29] J. C. HO, J. GHOSH, S. R. STEINHUBL, W. F. STEWART, J. C. DENNY, B. A. MALIN, AND J. SUN, *Limestone: High-throughput candidate phenotype generation via tensor factorization*, *Special Section: Methods in Clinical Research Informatics*, 52 (2014), pp. 199–211, <https://doi.org/10.1016/j.jbi.2014>.

- 07.001.
- [30] J. C. HO, J. GHOSH, AND J. SUN, *Marble: High-Throughput Phenotyping from Electronic Health Records via Sparse Nonnegative Tensor Factorization*, in Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA, 2014, Association for Computing Machinery, pp. 115–124, <https://doi.org/10.1145/2623330.2623658>.
  - [31] D. HONG, T. G. KOLDA, AND J. A. DUERSCH, *Generalized Canonical Polyadic Tensor Decomposition*, SIAM Review, 62 (2020), pp. 133–163.
  - [32] C. HU, P. RAI, AND L. CARIN, *Zero-Truncated Poisson Tensor Factorization of Massive Binary Tensors*, Aug. 2015.
  - [33] C. HU, P. RAI, C. CHEN, M. HARDING, AND L. CARIN, *Scalable Bayesian Non-negative Tensor Factorization for Massive Count Data*, in Machine Learning and Knowledge Discovery in Databases, A. Appice, P. P. Rodrigues, V. Santos Costa, J. Gama, A. Jorge, and C. Soares, eds., Cham, 2015, Springer International Publishing, pp. 53–70.
  - [34] K. HUANG AND N. D. SIDIROPOULOS, *Kullback-Leibler principal component for tensors is not NP-hard*, in 2017 51st Asilomar Conference on Signals, Systems, and Computers, 2017, pp. 693–697, <https://doi.org/10.1109/ACSSC.2017.8335432>.
  - [35] L. INGBER, *Adaptive simulated annealing (ASA): Lessons learned*, Control Cybernetics, 25 (1996), pp. 33–54.
  - [36] D. P. KINGMA AND J. BA, *Adam: A Method for Stochastic Optimization*, in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, Y. Bengio and Y. LeCun, eds., 2015.
  - [37] S. KIRKPATRICK, C. D. GELATT, AND M. P. VECCHI, *Optimization by simulated annealing*, SCIENCE, 220 (1983), pp. 671–680.
  - [38] T. KOLDA AND B. BADER, *The TOPHITS Model for Higher-order Web Link Analysis*, in Proceedings of Link Analysis, Counterterrorism and Security 2006, 2006.
  - [39] T. KOLDA, B. BADER, AND J. KENNY, *Higher-order Web link analysis using multilinear algebra*, in Fifth IEEE International Conference on Data Mining (ICDM'05), 2005, pp. 8 pp.–, <https://doi.org/10.1109/ICDM.2005.77>.
  - [40] T. G. KOLDA AND B. W. BADER, *Tensor Decompositions and Applications*, SIAM Review, 51 (2009), pp. 455–500.
  - [41] T. G. KOLDA AND D. HONG, *Stochastic Gradients for Large-Scale Tensor Decomposition*, SIAM Journal on Mathematics of Data Science, 2 (2020), pp. 1066–1095.
  - [42] B. KORTH AND L. R. TUCKER, *The distribution of chance congruence coefficients from simulated data*, Psychometrika, 40 (1975), pp. 361–372.
  - [43] B. KORTH AND L. R. TUCKER, *Procrustes matching by congruence coefficients*, Psychometrika, 41 (1976), pp. 531–535.
  - [44] P.-D. LETOURNEAU, M. BASKARAN, T. HENRETTY, J. EZICK, AND R. LETHIN, *Computationally Efficient CP Tensor Decomposition Update Framework for Emerging Component Discovery in Streaming Data*, in 2018 IEEE High Performance Extreme Computing Conference (HPEC), 2018, pp. 1–8.
  - [45] U. LORENZO-SEVA AND J. M. F. TEN BERGE, *Tucker's Congruence Coefficient as a Meaningful Index of Factor Similarity*, Methodology, 2 (2006), pp. 57–64, <https://doi.org/10.1027/1614-2241.2.2.57>.
  - [46] R. MARTÍ, P. PARDALOS, AND M. RESENDE, *Handbook of Heuristics*, Springer International Publishing, Aug. 2018, <https://doi.org/10.1007/978-3-319-07124-4>.
  - [47] J. MOCKS, *Topographic components model for event-related potentials and some biophysical considerations*, IEEE Transactions on Biomedical Engineering, 35 (1988), pp. 482–484, <https://doi.org/10.1109/10.2119>.
  - [48] J. M. MYERS AND D. M. DUNLAVY, *Using Computation Effectively for Scalable Poisson Tensor Factorization: Comparing Methods Beyond Computational Efficiency*, in 2021 IEEE High Performance Extreme Computing Conference, HPEC 2020, Waltham, MA, USA, September 21-25, 2020, IEEE, 2021, pp. 1–7, <https://doi.org/10.1109/HPEC49654.2021.9622795>.
  - [49] J. M. MYERS, D. M. DUNLAVY, K. TERANISHI, AND D. S. HOLLMAN, *Parameter Sensitivity Analysis of the SparTen High Performance Sparse Tensor Decomposition Software*, in 2020 IEEE High Performance Extreme Computing Conference, HPEC 2020, Waltham, MA, USA, September 21-25, 2020, IEEE, 2020, pp. 1–7, <https://doi.org/10.1109/HPEC43674.2020.9286210>.

- [50] I. J. MYUNG, *Tutorial on maximum likelihood estimation*, Journal of Mathematical Psychology, 47 (2003), pp. 90–100, [https://doi.org/10.1016/S0022-2496\(02\)00028-7](https://doi.org/10.1016/S0022-2496(02)00028-7).
- [51] A.-H. PHAN, P. TICHAVSKÝ, AND A. CICHOCKI, *Low Complexity Damped Gauss–Newton Algorithms for CANDECOMP/PARAFAC*, SIAM Journal on Matrix Analysis and Applications, 34 (2013), pp. 126–147, <https://doi.org/10.1137/100808034>.
- [52] E. T. PHIPPS AND T. G. KOLDA, *Software for Sparse Tensor Decomposition on Emerging Computing Architectures*, SIAM Journal on Scientific Computing, 41 (2019), pp. C269–C290.
- [53] P. RAI, C. HU, M. HARDING, AND L. CARIN, *Scalable Probabilistic Tensor Factorization for Binary and Count Data*, in Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI’15, AAAI Press, 2015, pp. 3770–3776.
- [54] T. M. RANADIVE AND M. M. BASKARAN, *An All-at-Once CP decomposition method for count tensors*, 2021 IEEE High Performance Extreme Computing Conference (HPEC), (2021), pp. 1–8.
- [55] S. SMITH, J. W. CHOI, J. LI, R. VUDUC, J. PARK, X. LIU, AND G. KARYPIS, *FROSTT: The Formidable Repository of Open Sparse Tensors and Tools*, 2017.
- [56] M. SUGIYAMA, H. NAKAHARA, AND K. TSUDA, *Legendre Decomposition for Tensors*, in Advances in Neural Information Processing Systems, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds., vol. 31, Curran Associates, Inc., 2018, <https://proceedings.neurips.cc/paper/2018/file/56a3107cad6611c8337ee36d178ca129-Paper.pdf>.
- [57] K. TERANISHI, D. M. DUNLAVY, J. M. MYERS, AND R. F. BARRETT, *SparTen: Leveraging Kokkos for On-node Parallelism in a Second-Order Method for Fitting Canonical Polyadic Tensor Models to Poisson Data*, in 2020 IEEE High Performance Extreme Computing Conference (HPEC), 2020, pp. 1–7, <https://doi.org/10.1109/HPEC43674.2020.9286251>.
- [58] I. THE MATHWORKS, *Symbolic Math Toolbox*, Natick, Massachusetts, United State, 2019, <https://www.mathworks.com/help/symbolic/>.
- [59] M. VANDECAPPELLE, N. VERVLIT, AND L. D. LATHAUWER, *A second-order method for fitting the canonical polyadic decomposition with non-least-squares cost*, IEEE Transactions on Signal Processing, 68 (2020), pp. 4454–4465, <https://doi.org/10.1109/TSP.2020.3010719>.
- [60] N. VERVLIT, O. DEBALS, AND L. DE LATHAUWER, *Tensorlab 3.0 — Numerical optimization strategies for large-scale constrained and coupled matrix/tensor factorization*, in 2016 50th Asilomar Conference on Signals, Systems and Computers, 2016, pp. 1733–1738, <https://doi.org/10.1109/ACSSC.2016.7869679>.
- [61] S. J. WRIGHT, *Coordinate descent algorithms*, Mathematical Programming, 151 (2015), pp. 3–34, <https://doi.org/10.1007/s10107-015-0892-3>.