Using Computation Effectively for Scalable Poisson Tensor Factorization: Comparing Methods Beyond Computational Efficiency

Jeremy M. Myers
College of William and Mary
Williamsburg, VA
jmmyers@cs.wm.edu

Daniel M. Dunlavy
Sandia National Laboratories
Albuquerque, NM
dmdunla@sandia.gov

***Abstract*—Poisson Tensor Factorization (PTF) is an important data analysis method for analyzing patterns and relationships in multiway count data. In this work, we consider several algorithms for computing a low-rank PTF of tensors with sparse count data values via maximum likelihood estimation. Such an approach reduces to solving a nonlinear, non-convex optimization problem, which can leverage considerable parallel computation due to the structure of the problem. However, since the maximum likelihood estimator corresponds to the global minimizer of this optimization problem, it is important to consider how effective methods are at both leveraging this inherent parallelism as well as computing a good approximation to the global minimizer. In this work we present comparisons of multiple methods for PTF that illustrate the tradeoffs in computational efficiency and accurately computing the maximum likelihood estimator. We present results using synthetic and real-world data tensors to demonstrate some of the challenges when choosing a method for a given tensor.**

***Index Terms*—Poisson Tensor Factorization, CP-APR, GCP, maximum likelihood estimation**

I. Introduction

The task of computing a low-rank canonical polyadic (CP) tensor model via Poisson maximum likelihood estimation—also known as the *Poisson Tensor Factorization (PTF)* problem [1]—has proven useful in analyzing latent patterns and relationships in count data across many application areas (e.g., network analysis [2]–[5], term-document analysis [6], [7], email analysis [8], link prediction [9], geosptial analysis [5], [10], and web page analysis [11]). Much of the research associated with computing these low-rank PTF models has focused on algorithms [1], [12]–[14] and scalability [15]–[21]. However, in many data analysis applications, decision makers, analysts, and data scientists are often constrained by time and computational resources available. Computational approaches must take such constraints into account when providing PTF solutions for data analysis.

The low-rank PTF modeling problem is a nonlinear, non-convex optimization problem [1]. Existing approaches for solving this problem consist of iterative methods that solve sequences of local optimization subproblems [1], [12]–[14] and thus produce and approximate a solution that is a local optimizer and not the maximum likelihood estimator (i.e., the global optimizer). In this work, we illustrate that existing methods find the global optimizer only a fraction of solves using a multi-start strategy [22], [23], i.e., generating approximate solutions from multiple starting points in the feasible domain of the optimization problem. Thus, it is critical to examine both computational efficiency and algorithmic effectiveness in finding the correct solution when comparing methods used for solving the low-rank PTF problem. To our knowledge, this is the first work to compare methods in terms of both computational efficiency of scalable methods and computing the maximum likelihood estimator versus local optimizers for the low-rank PTF problem.

State-of-the-art algorithms for the low-rank PTF problem employ a variety of approaches with different convergence rates and memory usage patterns, leading to various trade-offs in terms of computational efficiency. CP Alternating Poisson Regression (CP-APR) [1], [12] solves a bound-constrained, nonlinear, non-convex optimization problem deterministically to minimize the negative log-likelihood of the Poisson distribution. Currently, there are three variants in the CP-APR family of algorithms, each using a different amount of derivative information in solving the optimization problem. On the other hand, Generalized CP (GCP) [13] solves an unconstrained optimization problem for any loss function using first-order information only. The GCP-Adam [14] variant uses random sampling to improve computational efficiency.

We present a study of these methods that considers how the complex interplay of rates of convergence, computational costs, and memory demands impact their ability to meet the challenges posed by the PTF problem at scale. To our knowledge, the only previous comparison of these methods was conducted by Kolda and Hong in [14], where the focus was on computational efficiency. Our study here differs in that we emphasize the likelihood of algorithm convergence to the maximum likelihood estimator versus convergence to local minimizers in addition to computational efficiency.

The main contributions of our work are as follows:

- Comparisons of several scalable methods for Poisson Tensor Factorization.
- Illustration of the trade-offs between computational efficiency and approximate solution quality of PTF methods applied to synthetic and real-world tensor examples.

TABLE I
SUMMARY OF PTF ALGORITHM CHARACTERISTICS

Algorithm	Optimization	Convergence	Termination Criterion	Cost Per Outer Iteration	HPC Code
MU	fixed-point iteration	sublinear	KKT point	$(d+K)\cdot\mathcal{O}(r\cdot \text{nnz}(\mathcal{X}))$	SparTen
PQNR	quasi-Newton	superlinear	KKT point	$d\cdot\mathcal{O}(\text{nnz}(\mathcal{X}))+K\cdot\mathcal{O}(r)\cdot\sum_{k=1}^{d} I_k$	SparTen
PDNR	damped Newton	quadratic	KKT point	$d\cdot\mathcal{O}(\text{nnz}(\mathcal{X}))+K\cdot\mathcal{O}(r^3)\cdot\sum_{k=1}^{d} I_k$	SparTen
GCP-Adam	stochastic gradient descent	linear	Fail to decrease loss	$\mathcal{O}(rs)$	Genten

d: # modes. K: Maximum allowable # inner iterations. I_k: # rows in the kth mode. s: # stochastic samples.

II. METHODS

A d-way tensor $\mathcal{X}$ has size $n_1 \times n_2 \times \cdots \times n_d$. The low-rank CP tensor decomposition for a sparse tensor $\mathcal{X}$ of count data can be computed by solving the PTF problem:

$$\min_{\mathbf{A}_1,\ldots,\mathbf{A}_d} \sum_{\mathbf{i}} \mathcal{M}_{\mathbf{i}} - \mathcal{X}_{\mathbf{i}} \log \mathcal{M}_{\mathbf{i}}, \tag{1}$$

where multi-index $\mathbf{i}$ is a tuple over the tensor entries, $\mathcal{M}$ is a rank-r CP model the size of $\mathcal{X}$, and $\mathbf{A}_k$, $k \in \{1,\ldots,d\}$, are factor matrices of the CP model defined as:

$$\mathcal{M} = \sum_{j=1}^{r} \lambda_j \mathbf{a}_1(:,j) \circ \cdots \circ \mathbf{a}_d(:,j). \tag{2}$$

In (2), each λ_j is a scalar weight, $\mathbf{a}_k(:,j)$ is the j^{th} column in the k^{th} factor matrix, and $\circ$ is the generalized vector outer product. The solution of the low-rank PTF problem in (1) is the maximum likelihood estimator, $\widehat{\mathcal{M}}$, of $\mathcal{X}$, assuming the entries of $\mathcal{X}$ are drawn from a Poisson distribution.

A. CP Alternating Poisson Regression (CP-APR)

The CP-APR algorithms are alternating, block-coordinate descent methods that solve a bound-constrained nonlinear, non-convex formulation of the PTF problem with KKT-based optimality convergence criteria. The current CP-APR variants are: 1) Multiplicative Updates (MU) [1], 2) Projected Quasi-Newton for Row subproblem (PQNR) [12], and 3) Projected Damped Newton for Row subproblem (PDNR) [12]. All CP-APR methods are implemented in the MATLAB Tensor Toolbox [24] (as the function `cp_apr`) and SparTen [15], a C++ library that uses the Kokkos [25] performance-portability library for parallel execution on heterogeneous computer architectures. A single minimization over each mode individually is an *outer iteration*, and in each outer iteration subproblems are solved iteratively using up to a maximum number of *inner iterations*. Convergence is measured using the typical Karush-Kuhn-Tucker (KKT) optimality conditions employed for constrained optimization problems [26].

Brief descriptions of the CP-APR methods are as follows:

a) MU: A fixed-point iteration optimization that converges at a sublinear rate in most cases and linearly at best. Despite the slow rate of convergence, MU has the lowest computational cost per outer iteration of any CP-APR method. Note that MU is a generalization of the the Nonnegative Matrix Factorization method from Lee and Seung [27].

b) PQNR: A 1st-order quasi-Newton optimization for the row subproblem formulation of the PTF problem. The particular implementation used is a limited memory BFGS (L-BFGS) scheme [26], thereby reducing the theoretical convergence rate (from a superlinear rate as for full quasi-Newton methods). However, in practice a near-superlinear rate has been observed when solving PTF problems [12].

c) PDNR: A 2nd-order damped Newton optimization for the row subproblem formulation of the PTF problem. Although attractive for its quadratic convergence, PDNR is the most costly CP-APR method per outer iteration since the Hessian matrix of the function in the summand of (1) is formulated explicitly and used to compute a search direction for each row subproblem.

B. Generalized CP (GCP)

GCP [13] is an all-at-once, gradient descent method derived for arbitrary loss functions (including the Poisson likelihood function) associated with CP models. In this work, we use the GCP-Adam variant [14], which employs a stochastic gradient descent (SGD) optimization sampling nonzero and zero entries for objective function value estimation and gradient computations. As such, GCP-Adam can achieve a linear convergence rate and enjoys the cheapest cost per outer iteration of any PTF method considered in this work, which, for fixed rank r, is dependent only on the number of the optimization variables sampled. Presently, GCP-Adam terminates when the method fails to reduce the loss function after a user-defined number of outer iterations. GCP and its variants are available in the MATLAB Tensor Toolbox (as the function `gcp_opt`) and Genten [17], a C++ library for tensor decompositions. Like SparTen, Genten uses Kokkos for parallel computations. The GCP naming convention follows from the machine learning community: GCP optimizes using gradient information for a preset number of *iterations*; a pre-determined number of iterations form an *epoch*; and, progress is tracked by evaluating—estimating, in the case of GCP-Adam—the objective function.

The CP-APR and GCP-Adam methods described above are summarized along with their computational complexity in Table I. Note that the "HPC Code" column in the table denotes the name of the C++ Kokkos-based high performance computing (HPC) implementation. The table illustrates the computational trade-offs across the methods along with some of the challenges in choosing the most effective method for a given problem.

C. Factor Match Score

The typical approach for determining the quality of an approximate solution to a minimization problem such as in (1) is to measure the absolute or relative difference between the function values at the solution and the approximate solution. Due to the rich algebraic structure of CP models, an alternate measure of the quality of an approximate solution has been developed called the *factor match score (FMS)*. The FMS between two CP tensors, $\mathcal{M}$ and $\widehat{\mathcal{M}}$, is defined as:

$$\begin{aligned}&\mathrm{FMS}(\mathcal{M}, \widehat{\mathcal{M}}) = \\ &\max_{\pi(\cdot),\, \hat{\pi}(\cdot)} \frac{1}{r} \sum_{j=1}^{r} \prod_{k=1}^{d} \cos\left(\mathbf{a}_k(:\,, \pi(j)), \hat{\mathbf{a}}_k(:\,, \hat{\pi}(j))\right) \\ &\text{s.t.} \quad \|\mathbf{a}_k(:\,, i)\|_2 = 1;\ i = 1, \ldots, r;\ k = 1, \ldots, d. \end{aligned} \tag{3}$$

The permutations $\pi(\cdot)$, $\hat{\pi}(\cdot)$, re-order the columns of $\mathbf{A}_k$ and $\widehat{\mathbf{A}}_k$ to maximize the cosine similarity among all possible pairwise combinations of the r components between mode-k factors. A score of 1 indicates collinearity among the columns of all factor matrices and thus a perfect match between the two CP models. The values $[0.85, 0.94]$ of "fair similarity" and ≥ 0.95 as "equal" due to Lorenzo-Seva and ten Berge [28] are common values used to define acceptable matches in recent work [1], [12], [14].

III. Experimental Design

A. Data

In our numerical experiments comparing methods for PTF, we use the following tensors:

1) *Synthetic*: A three-way tensor of counts of size $500 \times 400 \times 300$, rank $r = 10$, and 866K nonzero entries (1.44% dense), 10% of which are noise. The size, dimensionality, tensor rank, and sparsity are similar to those seen in numerical experiments in [1], [12], [14], [29]. This tensor was generated using the `create_problem` method from the MATLAB Tensor Toolbox.
2) *Chicago Crime*: A sparse tensor dataset available as part of the FROSTT tensor benchmark suite [30]. The four-way tensor has dimensions $6168 \times 24 \times 77 \times 32$ and 5.3M nonzeros (1.46% dense). This tensor is an exemplar dataset from the FROSTT collection: challenging, yet small enough to fit in memory and support multi-start computations with reasonable solution times. The *Chicago Crime* dataset is used in numerical experiments in [14]–[16], [31].

B. Method Parameters

Each of the PTF methods described above includes algorithm and runtime parameters that need to be defined by users. In an effort to provide fair comparisons between methods, we conducted a grid search with 20 initial guesses over each method's parameters to identify the parameters that most impacted accuracy, in terms of minimizing (1). The best parameter settings for each method were chosen as those which most reliably minimized the Poisson negative log-likelihood loss; that is, the lowest sample mean and standard deviation values. Where there were ties, we chose the settings with the highest proportion of solutions nearest the maximum likelihood estimator, as measured by FMS. In the case of a synthetic tensor, the tie-break required eight digits of precision. In the case of a real-world dataset, only a threshold of FMS ≥ 0.9 was necessary.

The parameter selection process for CP-APR methods was based on results from previous work; see [15], [16] for details, including parameter descriptions. The main parameter for each of the CP-APR methods having the most impact on accuracy and performance as measured above are listed in Table II, along with the parameter values used in the grid search.

For GCP-Adam, we identified two parameters as most impactful: (1) the number of stochastic samples used per iteration (i.e., how many times the data is observed per epoch) and (2) the maximum allowable number of failures, which determines the final step size. We tuned Genten first over the number of samples while holding the final step size fixed at double machine precision. We then increased the final step size until sensitivities in the final result were observed. This had the benefit of reducing computational cost by avoiding unneeded smaller step sizes while maintaining consistent accuracy.

In our experiments on synthetic data, the CP-APR inverse Hessian is approximated with 3 *L-BFGS update pairs* in PQNR. *KKT violation* was set to 10^{-5} for MU and 10^{-4} for PQNR and PDNR. We run the CP-APR damped Newton method with the *divide-by-zero offset* for gradient and Hessian computations set to 10^{-5}. We use $s = 86,602$ samples (equivalent to observing the nonzero and zero entries 10 times per epoch) using stratified sampling and terminate when Genten fails to decrease the objective function three times (equivalent to 10^{-6} final step size). We note that the default choice by Genten—100,000 samples for objective function evaluations and 25,981 samples for gradient computations—was included in our grid search as well; all other preliminary experiments used the same value for both objective function and gradient calculations (i.e., some percentage of the tensor nonzeros).

In our experiments on *Chicago Crime*, the CP-APR *KKT violation* was set to 10^{-4} for MU and 10^{-5} for PQNR and PDNR. Again, 3 *L-BFGS update pairs* were used for the PQNR inverse Hessian approximation. The *divide-by-zero offset* for gradient and Hessian computations was set to 10^{-10} for PDNR. Additionally, PDNR used the multi-kernel optimization presented in [15] to address the known load imbalance issues with widely ranging numbers of nonzeros across the row subproblems. We use $s = 266,534$ samples (equivalent to viewing the nonzero and zero entries 5 times per epoch) using stratified sampling and terminate when Genten fails to decrease the objective function four times (equivalent to 10^{-7} final step size). Similarly, our grid search included the default Genten choice—100,000 samples for objective function evaluations and 159,921 samples for gradient computations—with the remaining experiments using a single value for both

objective function and gradient calculations. Additionally, we tested the value used on *Chicago Crime* in [14], the sum of the sizes of the dimensions $s = 6,319$.

TABLE II
PARAMETER TUNING GRID SEARCH.

Method	Parameter	Values
MU	KKT Violation	10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}
PQNR	KKT Violation	10^{-2}, 10^{-4}, 10^{-5}, 10^{-10}, 10^{-15}
PQNR	L-BFGS pairs	3, 5, 8
PDNR	KKT Violation	10^{-2}, 10^{-4}, 10^{-5}, 10^{-10}, 10^{-15}
PDNR	divide-by-zero	10^{-5}, 10^{-10}, 10^{-15}
GCP-Adam	samples	*Synthetic*: 867, 4331, 8661, 17321, 25981, 43301, 86602, 100000, 866011
		Chicago Crime: 5331, 6319, 26654, 53307, 100000, 106614, 159921, 266534, 533068

C. Multi-start Details

For the multi-start experiments, we use a simple random sampling scheme to choose 40 different initial guesses (which are different than those used in the grid search in Section III-B). We run each method starting from each of the initial guesses and compute rank $r = 10$ PTF approximate solutions for both datasets. In the case of the *Chicago Crime* these settings were chosen to support direct comparison of results with previous PTF results [14].

D. Maximum Likelihood Estimators

We identify the rank-10 maximum likelihood estimator (MLE) as the best approximate solution, in terms of the lowest negative Poisson log-likelihood function in (1), using all methods over the 40 runs for both tensors. Although this is not an analytic solution, we believe that it is a sufficient estimator for the purposes of this study.

E. Computational Environment

All experiments were run on a Dual Socket Intel Xeon Gold 6140 2.30 GHz CPU with 192 GB memory, 18 cores per socket, and each core supporting two hyperthreads. SparTen and Genten both used Kokkos version 3.4 built with GCC 8.3.0 and configured for OpenMP parallelization. All runs use all available hardware threads only (i.e., no hyperthreading).

IV. RESULTS

In this section, we present the results and analysis of our numerical experiments on synthetic and real-world tensor data. We extended SparTen's `Multiplicative-Update` and Genten's `gcp-sgd` (GCP-Adam) to report exact Poisson loss values without impacting performance profiling (i.e., after timing the outer iteration in CP-APR MU and each epoch in GCP-Adam).

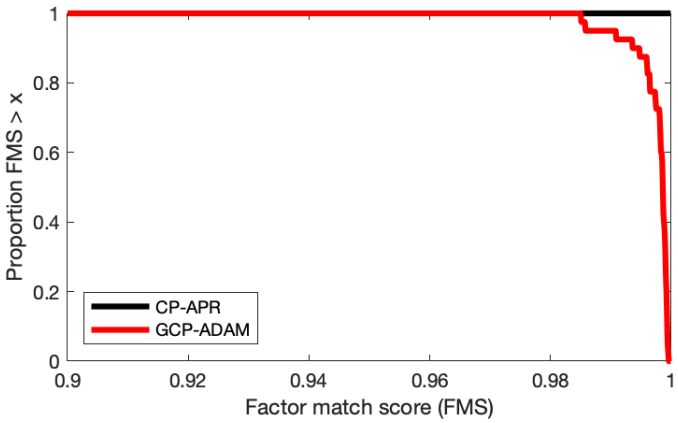


Fig. 1. The proportion of computed solutions that are algebraically similar to the MLE for a range of factor match scores. Each algorithm solved the synthetic $500 \times 400 \times 300$ rank $r = 10$ dataset for 40 different initial guesses. All CP-APR methods were arbitrarily close to the MLE in FMS.

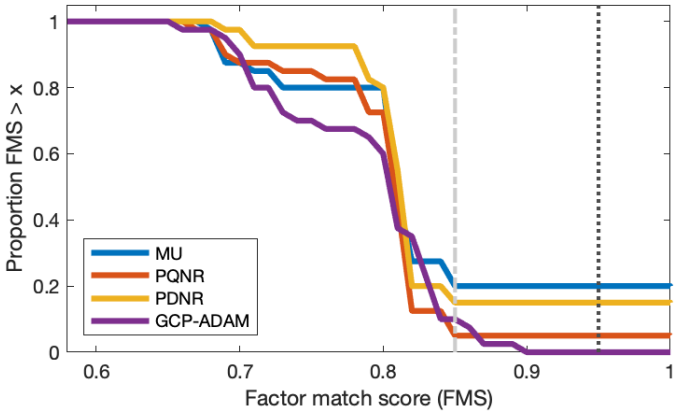


Fig. 2. The factor match score between computed CP models of the real-world *Chicago Crime* tensor and the MLE versus the proportion of solutions with scoring at a given level or higher. The dash-dot gray vertical line marks the 0.85 lower bound of "fair similarity" and the dotted black line marks the 0.95 "equal" level seen in prior art.

A. Results on Synthetic Data

FMS is used to measure the quality of the approximate solutions computed using the various PTF methods. Figure 1 illustrates the fraction of the 40 approximate solutions for each method that were above a particular FMS threshold as defined on the x-axis. Note that for the synthetic data in our study, FMS values are consistent across all methods, so we report CP-APR collectively here. Roughly 95% of GCP results are accurate within two digits; less than 50% are accurate within three. In all experiments, we see that the approximate solutions can be considered as equal to the MLE in terms of algebraic structure, as all FMS values are above 0.95 (see Section II-C).

Timing and accuracy results, in terms of negative Poisson log-likelihood function values, are shown in Figure 3(a). The traces of the objective function values for all runs are shown as dotted lines, the final values marked with circles, and the median trace across all iterations denoted by a solid line. All CP-APR approximate solutions are numerically equivalent

to the MLE and PQNR is the most efficient in terms of computational time required to converge to a solution. Both the low relative error and high cost in terms of computational times among GCP runs are indicative of the large number of samples used. Predictably, performance improvements gained using fewer samples come at the expense of increased error and variability in accuracy.

B. Results on Real-World Data

Results from comparable experiments to those presented above but on the *Chicago Crime* data are presented in Figures 2 and 3(b). Figure 2 shows the proportion of each method's approximate solutions that are similar to the maximum likelihood estimator for a given FMS value. MU is the best CP-APR method using this measure in both the range $[0.85, 0.94]$ (fair similarity) and ≥ 0.95 (equal). MU is roughly $2\times$ the proportion of GCP-Adam in the first interval.

Figure 3(b) shows the traces of the objective function values (estimated for GCP) and the final and median trace loss values. Note the consistency in reaching relatively low negative log-likelihood function values across CP-APR methods. We note that MU has the best performance in terms of runtime and consistently low error. Interestingly, MU never converged within the KKT-based tolerance on the *Chicago Crime* dataset, indicating that some type of stagnation may be occurring.

The high runtime of PDNR, in spite of the multi-kernel optimization for addressing nonzero load imbalances across row subproblems, is particularly noteworthy. Furthermore, PDNR failed to converge in 2 of the 40 runs. The traces of these runs suggest that PDNR is more likely than other CP-APR methods to stagnate very far from a KKT point. Surprisingly, the final approximate solutions from these two runs are not the least accurate in terms of negative log-likelihood; one solution even has a factor match score with the MLE above 0.98 (the other is below 0.74). In contrast, PQNR is quite reliable: converging in every instance, matching the MLE in terms of FMS almost as well as PDNR but nearly an order of magnitude faster, and with roughly the same variability in runtime as GCP-Adam.

GCP-Adam exhibits high variability in accuracy but with runtime results comparable to PQNR. Only 10% of GCP-Adam solutions satisfy the 0.85 FMS threshold of "fair similarity" to the MLE; no GCP-Adam solutions are ≥ 0.9 FMS. We highlight the importance of the number of samples used by GCP-Adam with respect to its effectiveness in recovering the maximum likelihood estimator. Our experiments demonstrate the discrepancy between the estimated and exact objective function value, which may adversely impact the iterative path taken by GCP-Adam. We leave the opportunity for solutions to this problem—e.g., correction steps through variance reduction or improved bounds—to future work.

Interestingly, the three horizontal bands of CP-APR solutions is suggestive of convergence to local minimizers. However, we leave the assessment of these bands as true local minimizers to future work. If these truly are local minimizers, and only apply to the CP-APR methods, then this indicates that GCP-Adam may be more robust in not stagnating in the basins of attraction of local minimizers.

V. Conclusion and Discussion

We presented results from numerical experiments to compare methods for the Poisson Tensor Factorization problem that consider algorithm effectiveness in addition to computational efficiency, the latter of which is the focus of prior work. Our experiments on synthetic and real-world tensor data emphasize the need to consider convergence to the maximum likelihood estimator (as opposed to local optimizers) to complement analysis based solely on scalability and/or computational efficiency. In particular, our results indicate that the steps required to improve the effectiveness of a scalable, stochastic algorithm for the PTF problem may render an otherwise computationally efficient method less effective than its more costly counterparts, all the while producing less accurate solutions with high variability.

The effectiveness of MU in our experiments is particularly notable. This method reliably computed the maximum likelihood estimator for both datasets and was an order of magnitude faster than at least one of its competitors in both instances, despite the fact that it never converged to the user-defined tolerance in any experiment run on *Chicago Crime*. Similarly, during the grid search and final experiments, PQNR and PDNR continue to try to satisfy the KKT conditions despite marginal improvement in objective function minimization. These results, taken together with the discussion of PDNR in Section IV-B, indicate that there is more research needed to understand the relationships between convergence of the methods presented here and the multilinear structure of the low-rank CP Poisson tensor models.

Acknowledgement

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.

We thank Keita Teranishi and Eric Phipps (both at Sandia National Laboratories) for assistance in extending SparTen and Genten to report exact Poisson loss values, respectively.

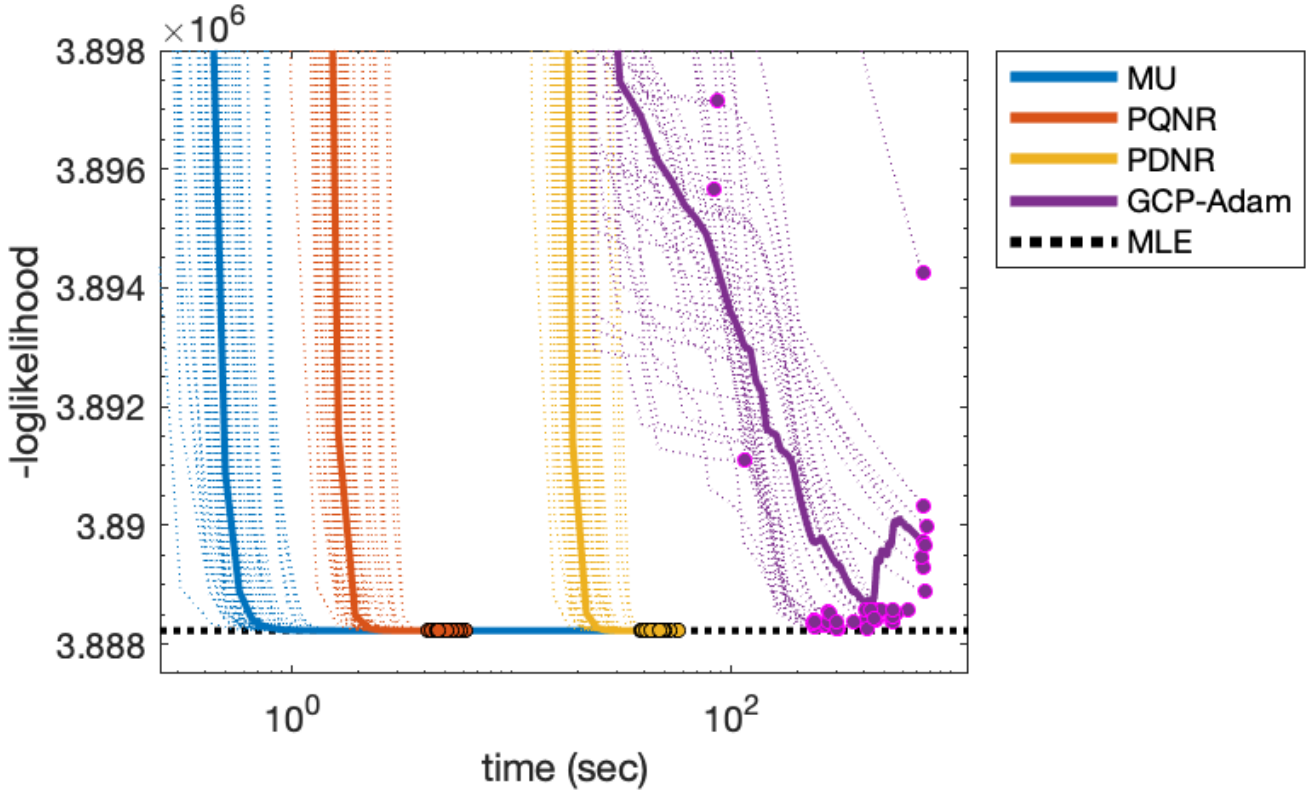
References

[1] E. C. Chi and T. G. Kolda, "On tensors, sparsity, and nonnegative factorizations," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, pp. 1272–1299, jan 2012.

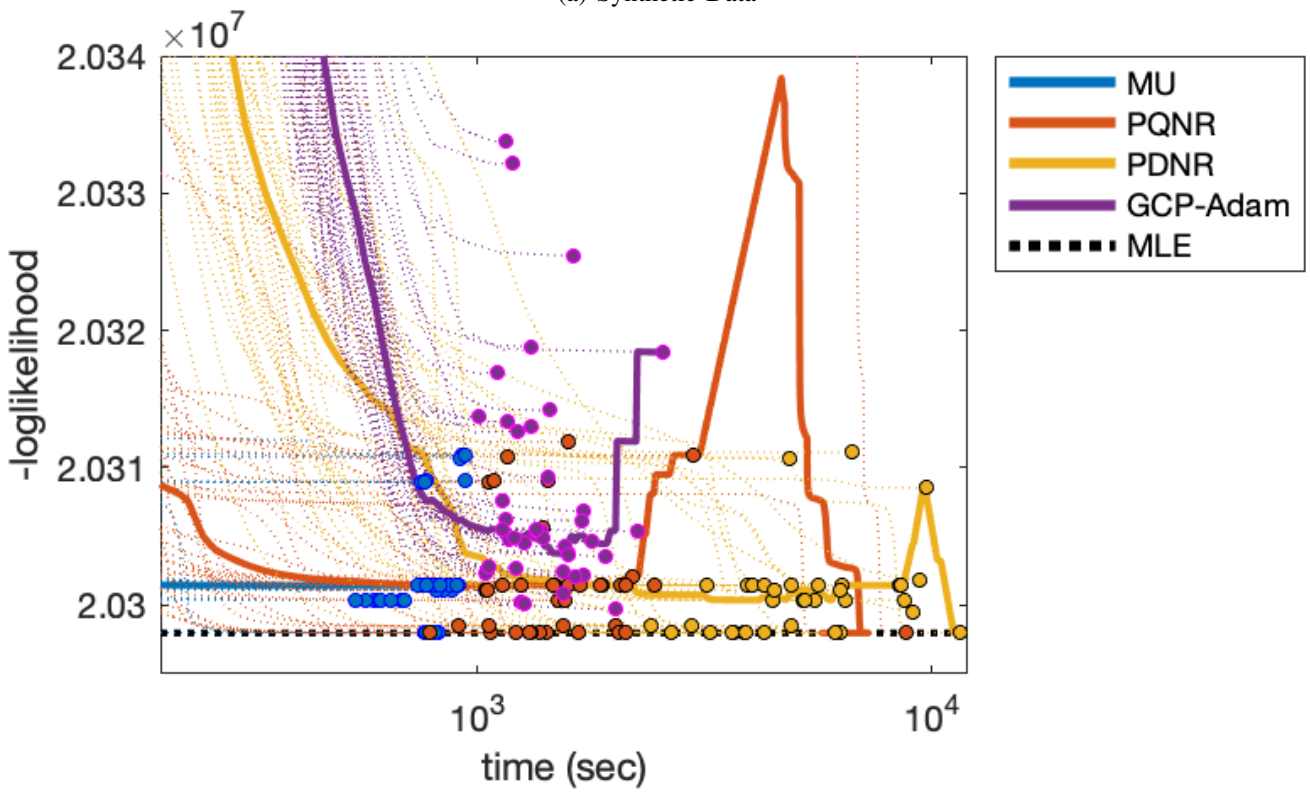
[2] D. M. Dunlavy, T. G. Kolda, and W. P. Kegelmeyer, "Multilinear algebra for analyzing data with multiple linkages," in *Graph Algorithms in the Language of Linear Algebra*, ser. Fundamentals of Algorithms, J. Kepner and J. Gilbert, Eds. SIAM, 2011, pp. 85–114.

[3] 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.

[4] 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*, vol. 96, pp. 207–215, 2019.



(a) Synthetic Data



(b) *Chicago Crime* Data

Fig. 3. Comparing performance and accuracy between CP-APR variants and GCP-Adam on (a) the synthetic $500 \times 400 \times 300$ rank $r = 10$ tensor and (b) the *Chicago Crime* tensor. Dotted lines show the traces of the Poisson loss function evaluations after each outer iteration for individual runs. Solid lines show the median function values at the end of each outer iteration across all traces. Filled circles show the exact negative log-likelihood values of the approximate solution. The black dotted line is the negative log-likelihood of the maximum likelihood estimator.

[5] D. Leggas, T. S. Henretty, J. Ezick, M. Baskaran, B. von Hofe, G. Cimaszewski, M. H. Langston, and R. Lethin, "Multiscale data analysis using binning, tensor decompositions, and backtracking," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, 2020, pp. 1–7.

[6] 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.

[7] 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.

[8] B. B. B.W., M. Berry, and M. Browne, *Survey of Text Mining II*. Springer, London, 2008, ch. Discussion Tracking in Enron Email Using PARAFAC.

[9] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations," *ACM Transactions on Knowledge Discovery from Data*, vol. 5, no. 2, p. 10 (27 pages), February 2011.

[10] 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.

[11] T. Kolda and B. Bader, "The TOPHITS model for higher-order web link analysis," in *Proceedings of Link Analysis, Counterterrorism and Security 2006*, 2006.

[12] S. Hansen, T. Plantenga, and T. G. Kolda, "Newton-based optimization for kullback–leibler nonnegative tensor factorizations," *Optimization Methods and Software*, vol. 30, no. 5, pp. 1002–1029, apr 2015.

[13] D. Hong, T. G. Kolda, and J. A. Duersch, "Generalized canonical polyadic tensor decomposition," *SIAM Review*, vol. 62, no. 1, pp. 133–163, jan 2020.

[14] T. G. Kolda and D. Hong, "Stochastic gradients for large-scale tensor decomposition," *SIAM Journal on Mathematics of Data Science*, vol. 2, no. 4, pp. 1066–1095, jan 2020.

[15] 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)*. IEEE, sep 2020.

[16] J. M. Myers, D. M. Dunlavy, K. Teranishi, and D. Hollman, "Parameter sensitivity analysis of the SparTen high performance sparse tensor decomposition software," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, sep 2020.

[17] E. T. Phipps and T. G. Kolda, "Software for sparse tensor decomposition on emerging computing architectures," *SIAM Journal on Scientific Computing*, vol. 41, no. 3, pp. C269–C290, jan 2019.

[18] M. Baskaran, T. Henretty, and J. Ezick, "Fast and scalable distributed tensor decompositions," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, 2019, pp. 1–7.

[19] 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.

[20] 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.

[21] 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)*, 2014, pp. 1–6.

[22] A. Gy orgy and L. Kocsis, "Efficient multi-start strategies for local search algorithms," *Journal of Artificial Intelligence Research*, vol. 41, pp. 705–720, 2011.

[23] R. Martí, J. A. Lozano, A. Mendiburu, and L. Hernando, *Multi-start Methods*. Cham: Springer International Publishing, 2018, pp. 155–175.

[24] B. W. Bader, T. G. Kolda *et al.*, "Matlab tensor toolbox version 3.2.1," Available online, April 2021. [Online]. Available: http://www.sandia.gov/~tgkolda/TensorToolbox/

[25] 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*, vol. 74, no. 12, pp. 3202–3216, dec 2014.

[26] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York: Springer, 2006.

[27] D. Lee and H. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.

[28] U. Lorenzo-Seva and J. M. F. ten Berge, "Tucker's congruence coefficient as a meaningful index of factor similarity," *Methodology*, vol. 2, no. 2, pp. 57–64, jan 2006.

[29] J. Cai, M. Baskaran, B. Meister, and R. Lethin, "Optimization of symmetric tensor computations," in *2015 IEEE High Performance Extreme Computing Conference (HPEC)*, 2015, pp. 1–7.

[30] S. Smith, J. W. Choi, J. Li, R. Vuduc, J. Park, X. Liu, and G. Karypis. (2017) FROSTT: The formidable repository of open sparse tensors and tools. [Online]. Available: http://frostt.io/

[31] T. M. Ranadive and M. M. Baskaran, "Large-scale sparse tensor decomposition using a damped gauss-newton method," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, 2020, pp. 1–8.