A Distributed Memory Implementation of CP–POPT–GDGN: An All-at-Once Decomposition Algorithm for Count Tensors

Teresa M. Ranadive

Laboratory for Physical Sciences

February 25th, 2022

Outline

- Introduction
- Background Information
- CP–POPT–GDGN
- Distributed Implementation
- Summary

Introduction

- Tensor decomposition: $\Longrightarrow$ minimize "distance" between tensor $\mathcal{X}$ and decomposition model $\mathcal{M}$:

$$\underset{\text{s.t. constraints}}{\text{minimize}} \quad \sum_{\mathbf{i}} (\underbrace{x_{\mathbf{i}}}_{\text{tensor entry}} - \underbrace{m_{\mathbf{i}}}_{\text{model entry}})^2/2 \qquad \text{vs.} \tag{1}$$

$$\underset{\text{s.t. constraints}}{\text{minimize}} \quad \sum_{\mathbf{i}} (m_{\mathbf{i}} - x_{\mathbf{i}} \log(m_{\mathbf{i}})) + c_{\mathcal{X}}. \tag{2}$$

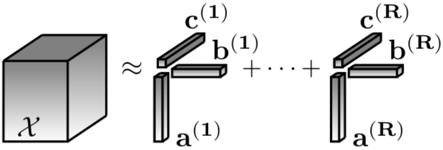
- CP–APR is most commonly used to solve (2).
- All–at–once optimization algorithms often compute more accurate decompositions than alternating algorithms.
- CP–POPT outperforms CP–APR for tensors formed from network traffic data sets, in terms of decomposition accuracy and latent behavior detection.

Introduction

- One iteration of CP–POPT requires $>$ 3x as much time to complete as one iteration of CP–APR (shared memory).
- CP–APR converges in fewer iterations than CP–POPT.
- Implemented distributed memory version of CP–POPT to decrease time required for CP-POPT to compute decompositions:
    - One iteration of CP–POPT often requires less time to complete than one iteration of CP–APR.
    - For larger tensors, CP–POPT currently scales well as the number of processors is increased, provided no tensor dimensions are too large.

Background Information

- A rank-R CP tensor decomposition $\mathcal{M}$ of $\mathcal{X}$:



- Each decomposition model entry is $m_{ijk} = \sum_{r=1}^{R} \mathbf{a}_i^{(r)} \mathbf{b}_j^{(r)} \mathbf{c}_k^{(r)}$.
- To compute a CP count tensor decomposition in practice, solve

$$\begin{aligned} &\text{minimize} \\ \text{s.t.}\quad &\text{all variables} \geq 0 \end{aligned} \quad \sum_{\mathbf{i}} \left(m_{\mathbf{i}} - x_{\mathbf{i}} \log(m_{\mathbf{i}} + \epsilon)\right) + c_{\mathcal{X}}. \tag{3}$$

- Both CP–APR and CP–POPT compute accurate decompositions.

CP–POPT–GDGN Algorithm

Summary of CP–POPT:

(i) Compute a Cauchy point $\mathbf{z^{(k)}}$.

(ii) Use $\mathbf{z^{(k)}}$ to determine which entries of the next iterate will be active (fixed at zero).

(iii) "Scooch" certain entries of $\mathbf{z^{(k)}}$ away from zero.

(iv) Compute a damped Newton based direction to update the free variables and (ideally) obtain a point $\mathbf{x_{GN}^{(k)}}$ such that $f(\mathbf{x_{GN}^{(k)}}) < f(\mathbf{z^{(k)}})$.

(v) Decide whether the next iterate should be given by $\mathbf{x_{GN}^{(k)}}$ or $\mathbf{z^{(k)}}$.

(vi) Adjust the damping parameter for the next iteration.

CP–POPT–GDGN Implementation and Data Sets

- Implemented shared and distributed–memory verison of POPT into ENSIGN.
- Main bottleneck is solving linear system to obtain the generalized damped Gauss–Newton direction.
- System solved using PCG method with Jacobi preconditioner.

Data set	Dimensions	No. of Non-zeros
Chicago	6,186 x 24 x 77 x 32	5,330,673
LANL1	1,433 x 22,077 x 534,687 x 58,389 x 11	40,266,345
LANL2	3,761 x 11,154 x 8,711 x 75,147 x 9	69,082,467
LBNL	1,605 x 4,198 x 1,631 x 4,209 x 868,131	1,698,825
Plant	562 x 124 x 123	8,370
RL1	279 x 248 x 1,757 x 10	46,591
RL2	1,512 x 433 x 4,568 x 4,743	314,276

Table: Tensors generated from seven different data sets.

Decomposition Accuracy

	CP-APR	CP-POPT
Chicago	4.2E-1	4.2E-1
LANL1	2.2E-11	2.2E-11
LANL2	3.5E-9	3.4E-9
LBNL	3.1E-11	2.6E-11
Plant	7.4E-3	7.4E-3
RL1	2.5E-4	2.5E-4
RL2	5.9E-8	5.7E-8

Table: Best value obtained for (3), divided by the number of tensor entries.

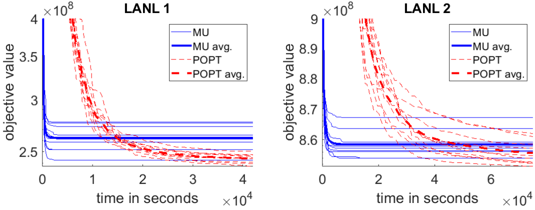
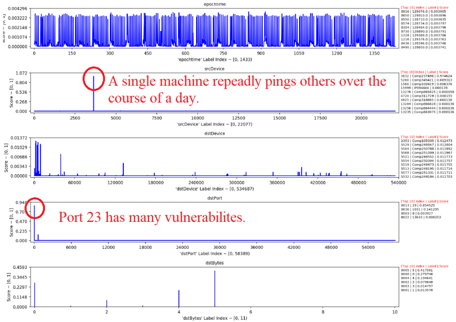


Figure: Objective value vs. time in seconds for each of the two large–scale tensors.

- If $\mathcal{C}$ is a component from a decomposition $\mathcal{M}_1$, the behavior described by $\mathcal{C}$ has also been detected by a second decomposition $\mathcal{M}_2$ if

$$\frac{\langle \mathcal{C}, \mathcal{M}_2 \rangle_F}{\|\mathcal{C}\|_F \|\mathcal{M}_2\|_F} \geq \text{threshold.}$$

- In the LANL1 data set, POPT detected two behaviors that MU did not; MU did not detect any additional behaviors.



Distributed Memory Implementations

- Implemented a distributed memory version of CP–POPT into ENSIGN to compare to existing distributed memory version of CP–APR.
- Data spread across nodes; all factor matrices are stored on all modes.
- Each iteration of CP–POPT still often requires more computation than CP–APR.
- As computing resources are increased, the amount of time required to perform local computation is reduced.
- Nature of CP-POPT algorithm is more conducive to being exploited in the distributed memory implementation.

Decomposition Accuracy Over Time

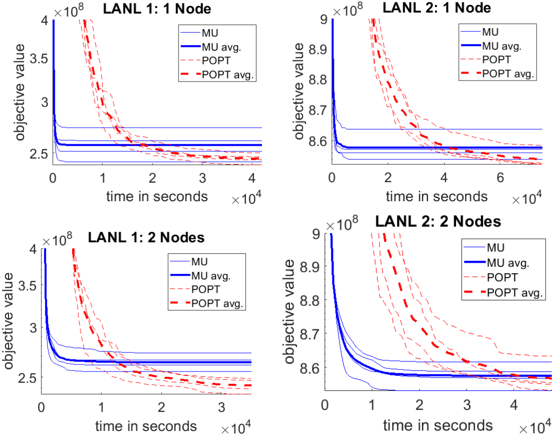


Figure: Objective value vs. time in seconds for each of the two large–scale tensors, using (top) one node and (bottom) two nodes.

Algorithm Scalability

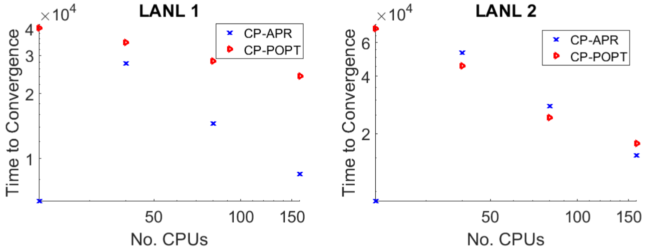


Figure: Time to convergence vs. no. CPUs for each of the two large–scale tensors.

- CP-APR runs for 100 iterations; CP-POPT for 200.
- Long third dimension in LANL1 tensor hinders CP-POPT scalability.
- CP-POPT achieves good scalability on LANL2 tensor; converges in about the same amount of time as CP-APR.

Conclusions

- CP-POPT often achieves more accurate decompositions than CP-APR, but shared memory implementation is slow.
- Distributed memory implementation of CP-POPT is not much slower than distributed memory implementation of CP-APR, and is occasionally faster.
- In future, plan to split data along longest mode to reduce communication for distributed CP-POPT and obtain more scalable implementation.
- Also, want to look into all-at-once optimization methods for other types of tensors (e.g., binary tensors).

References

ENSIGN Tensor Toolbox, Reservoir Labs. Accessed: Jul. 6, 2021. [Online]. Available: https://www.reservoir.com/ensign/

M. Baskaran et al., "Memory-efficient parallel tensor decompositions," in *Proc. IEEE High Perform. Extreme Comput. Conf.*, Waltham, MA, USA, Sept. 12–14, 2017.

E. C. Chi and T. G. Kolda, "On tensors, sparsity, and nonnegative factorizations," *SIAM J. Matrix Anal. and Appl.*, vol. 33, pp. 1272–1299, 2013.

A.D. Kent, "Cybersecurity data sources for dynamic network research," in *Dynamic Networks and Cyber–Security*, World Scientific, 2015, ch. 2, pp. 37-65.

T.M. Ranadive and M.M. Baskaran, "An All–at–Once CP Decomposition Method for Count Tensors" in *Proc. IEEE High Perform. Extreme Comput. Conf.*, Sept. 20–21, 2021.

S. Smith et. al., *FROSTT: The formidable repository of open sparse tensors and tools*, 2017. [Online]. Available: http://frostt.io/

M. Turcotte, A. Kent and C. Hash, "Unified host and network data set," in *Data Science for Cyber-Security*, World Scientific, Nov. 2018, ch. 1, pp. 1-22.