BERKELEY LAB

Leverage-Based Sketches for Khatri-Rao Products

Redwood Center for Theoretical Neuroscience

November 1, 2024

Vivek Bharadwaj [1,2], Osman Asif Malik [3], Riley Murray [4], Laura Grigori [5], Aydın Buluç [2,1], James Demmel [1]

[1] EECS Department, UC Berkeley [2] CRD Division, Lawrence Berkeley National Lab
[3] Encube Technologies [4] Sandia National Laboratories
[5] Institute of Mathematics, EPFL & Lab for Simulation and Modelling, Paul Scherrer Institute

The Berkeley BeBOP and PASSION Groups

I work with two groups at UC Berkeley:

- The **B**erkeley **Be**nchmarking and **Op**timization (BeBOP) group led by James Demmel and Katherine Yelick.
- The **P**arallel **A**lgorithms for **S**calable **S**parse computatIONs (PASSION) group led by Aydın Buluç, joint with Lawrence Berkeley National Lab.

Our interests include **linear algebraic computations** and **sparse kernels** that can be parallelized and deployed at **supercomputer scale**.

Figure: Frontier, the first exascale supercomputer in the United States. Credit: OLCF, Wikimedia Commons CC2.0.

Link to Today's Paper

Fast Exact Leverage Score Sampling From Khatri-Rao Products with Applications to Tensor Decomposition [Bha+23]. Link to paper below, more at `https://vivek-bharadwaj.com`.

Introduction

The Khatri-Rao Product

- In this talk, the Khatri-Rao product (KRP, denoted $\odot$) is the column-wise Kronecker product of two matrices:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \odot \begin{bmatrix} w & x \\ y & z \end{bmatrix} = \begin{bmatrix} aw & bx \\ cw & dx \\ ay & bz \\ cy & dz \end{bmatrix}$$

- Output column count is *identical* to inputs. Output row count is the *product* of row counts of inputs.

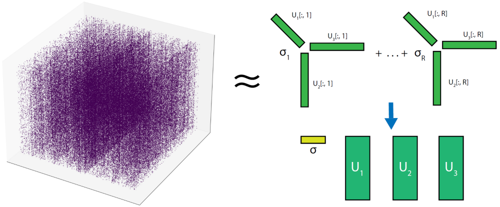
- Appears in signal processing, compressed sensing, PDE inverse problems. Possible use in hyperdimensional computing where $\otimes$, the Kronecker product, used for binding.

Motivating Application

- **Our goal:** efficiently solve an overdetermined linear least-squares problem

$$\min_X \|AX - B\|_F$$

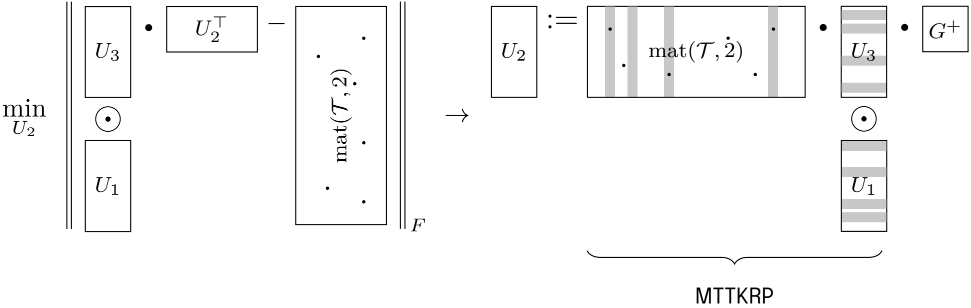
where $A = U_1 \odot \ldots \odot U_N$ with $U_j \in \mathbb{R}^{I \times R}$.



- Key kernel in alternating least-squares Candecomp / PARAFAC (CP) decomposition [LK22].

One Step of ALS Illustrated

$$\min_{U_j} \left\| \left[\bigodot_{k\neq j} U_k \right] \cdot U_j^\top - \mathrm{mat}(\mathcal{T}, j)^\top \right\|_F$$



Randomized Linear Least-Squares

- **Sketch & Solve**: Apply short-wide sketching matrix S to both A and B, solve reduced problem
$$\min_{\tilde{X}} \left\| SA\tilde{X} - SB \right\|_F$$
- Want an (ε, δ) guarantee on solution quality: with high probability $(1-\delta)$,
$$\left\| A\tilde{X} - B \right\|_F \leq (1+\varepsilon) \min_X \|AX - B\|$$
- Could choose S as:
  - An i.i.d. Gaussian / Rademacher random matrix
  - A Countsketch / Sparse Sign embedding (fixed nnz per column)
  - A composition of random diagonal, FFT-like operator, and uniform sparse sampler

Subspace Embedding Matrices

i.i.d. Gaussian

$$\begin{bmatrix} -0.01 & -0.39 & 0.37 \\ -0.47 & 0.74 & -0.10 \end{bmatrix}$$

Countsketch

$$\begin{bmatrix} +1 & 0 & +1 \\ 0 & -1 & 0 \end{bmatrix}$$

- Oblivious subspace embeddings: require no prior information about A; easy to construct and apply when inputs have no structure, but harder for Khatri-Rao products.
- In this talk: we will just **randomly sample** J rows of A and B to form a pair of much shorter matrices. Preserves structure in both input and output.

Our Contributions [Bha+23]

Method	Source	Round Complexity ($\tilde{O}$ notation)
CP-ALS	[KB09]	$N(N+I)I^{N-1}R$
CP-ARLS-LEV	[LK22]	$N(R+I)R^N/(\varepsilon\delta)$
TNS-CP	[Mal22]	$N^3IR^3/(\varepsilon\delta)$
GTNE	[MS22]	$N^2(N^{1.5}R^{3.5}/\varepsilon^3 + IR^2)/\varepsilon^2$
STS-CP	Ours	$N(NR^3 \log I + IR^2)/(\varepsilon\delta)$

- We build a data structure with runtime **logarithmic** in the height of the KRP and quadratic in R to sample from *leverage scores* of A.
- Yields the **STS-CP** algorithm: lower asymptotic runtime for randomized dense CP decomposition than recent SOTA methods (even more advantage for sparse tensors).

Statistical Leverage Scores

Intuition: Do I Need Every Row?

- Consider a univariate regression problem with 100,000 (x, y) points.
- This is a **highly-overdetermined** problem. Can pick a subset of points to perform fitting.

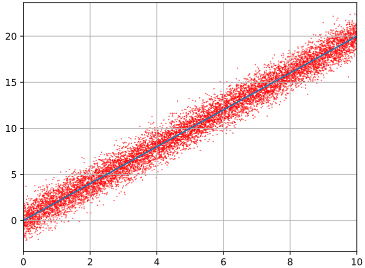


Figure: $y_i \sim x_i + \mathcal{N}(0, 1)$

Leverage Score Sampling

We will sample rows i.i.d. from A according to the *leverage score distribution* on its rows. Given **reduced SVD** $A = U\Sigma V^\top$, the leverage score ℓ_i of row i is

$$\ell_i = \|U[i,:]\|^2 .$$

Theorem (Leverage Score Sampling Guarantees, [Mal22])

Suppose $S \in \mathbb{R}^{J\times I}$ is a leverage-score sampling matrix for $A \in \mathbb{R}^{I\times R}$, and define

$$\tilde{X} := \arg\min_{\tilde{X}} \left\|SA\tilde{X} - SB\right\|_F$$

If $J \gtrsim R\max(\log(R/\delta), 1/(\varepsilon\delta))$, then with probability at least $1-\delta$,

$$\left\|A\tilde{X} - B\right\|_F \leq (1+\varepsilon)\min_X \|AX - B\|_F .$$

Interpretation of Leverage Scores

When A has 1 column, leverage scores are proportional to squared distance from origin.

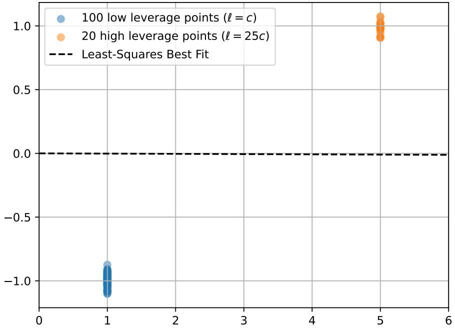
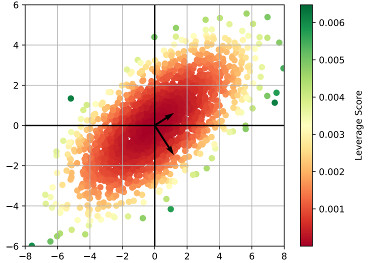


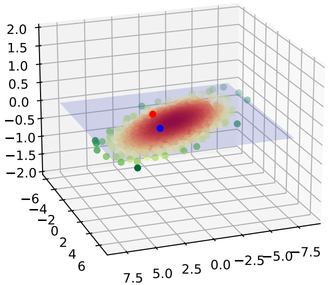
Figure: A univariate regression problem with low and high leverage points (intercept constrained to be 0).

Interpretation of Leverage Scores

In general, leverage scores of A quantify *influence* that each row has on the solution, capture correlation of rows of A with rows of $\Sigma^{-1}V^\top$.



(a) Projection onto xy-plane



(b) (x, y, z) data

Figure: Leverage scores of $(x, y, 0)$ triples from a multivariate normal distribution. Left: components of $\Sigma^{-1}V^\top$ shown. Right: the red point has greater influence than the blue point (both equidistant from $(0, 0, 0)$).

Interpretation of Leverage Scores

- Leverage score sampling captures the geometry of the column space of A.
- Rigorously: sampling i.i.d. with leverage score probabilities leads to an **optimal** [DM20] sample complexity to construct an ℓ_2-subspace embedding matrix S.
- **SE Property:** W.h.p simultaneously for ALL vectors $x \in \mathbb{R}^R$,

$$(1-\tilde{\varepsilon})\,\|Ax\|_2 \leq \|SAx\|_2 \leq (1+\tilde{\varepsilon})\,\|Ax\|_2$$

- In turn, an ℓ_2-S.E. guarantees that our sketched solution has close-to-optimal residual with respect to the original problem.

Prior Work

Problem: Cost to compute all leverage scores exactly is identical to runtime of QR decomposition. Defeats the purpose of sampling!

- (SPALS [Che+16]): Sample rows according to *approximate* leverage scores of A. Worst-case **exponential** in N to achieve (ε, δ) guarantee.

- (CP-ARLS-LEV [LK22]): Similar approximation, hybrid random-deterministic sampling strategy and practical improvements.

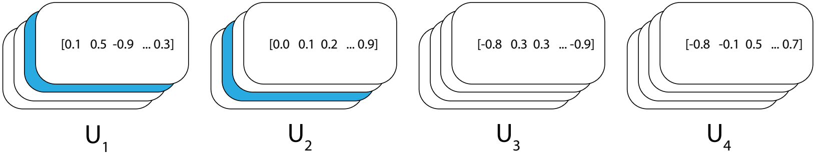
- (TNS-CP [Mal22]): Samples implicitly from exact leverage distribution with **polynomial** complexity to achieve (ε, δ) guarantee, but linear dependence on I for each sample. **We want to accelerate this algorithm.**

Efficient Sampling from Khatri-Rao Products

Implicit Leverage Score Sampling

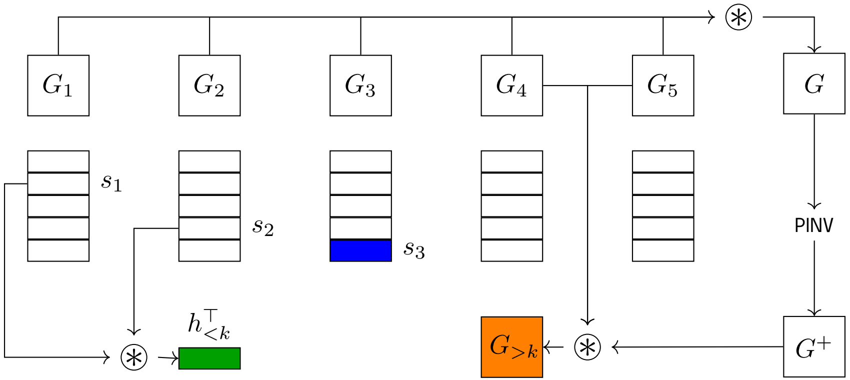
- For $I = 10^7, N = 3$, matrix A has 10^{21} rows. Can't even index rows with 64-bit integers. Instead: use identity $\ell_i = A[i,:]\,(A^\top A)^+ A[i,:]^\top$.

- Draw a row from each of $U_1, ..., U_N$, return their Hadamard product.



- Let $\hat{s}_j$ be a random variable for the row index drawn from U_j. Assume $(\hat{s}_1, ..., \hat{s}_N)$ jointly follows the leverage score distribution on $U_1 \odot ... \odot U_N$.

The Conditional Distribution of $\hat{s}_k$



$$p(\hat{s}_k = s_k \mid \hat{s}_{<k} = s_{<k}) \propto \langle h_{<k} h_{<k}^{\top}, U_k[s_k,:]^{\top} U_k[s_k,:], G_{>k} \rangle$$

A Problem of Variable Dependence

- Each subsequent index depends on the ones that precede it. How to deal with the dependence? Let's look at four approaches:

- Only a finite number of values for $\hat{s}_1, \hat{s}_2$. Precompute and store all possible conditional distributions for $\hat{s}_3$, and similarly for $\hat{s}_4, \hat{s}_5 \ldots$

- Preprocessing time is $\Omega\left(I^N\right)$, not viable for large I.

Preprocessing Time	Time for J Samples	# Samples Required
$\Omega(I^N)$	$O(JN)$	$O(R/(\varepsilon\delta))$

Approach 2: Ignore the Dependence

- Sample **independently** from $U_1, ..., U_N$ based on the leverage scores of each factor matrix [Che+16; LK22].
- No longer sampling from the exact leverage score distribution, so require $O(R^N/(\varepsilon\delta))$ samples to achieve the (ε, δ) guarantee.
- Efficient if R, N low enough. Can easily update if one matrix U_j changes.

Preprocessing Time	Time for J Samples	# Samples Required
$O(NIR^2)$	$O(JN)$	$O(R^N/(\varepsilon\delta))$

Approach 3: Form Full Conditional Distribution

- Compute the full conditional distribution $p(\hat{s}_3 = s_3 \mid \hat{s}_1 = s_1, \hat{s}_2 = s_2)$ for each draw *during* sampling [Mal22].
- Costs $O(IR^2)$ per matrix U_j **per sample**.
- Works well if I is low enough (many dense tensor applications), but performance degrades for $I \geq 10^3$.

Preprocessing Time	Time for J Samples	# Samples Required
$O(NIR^2)$	$O(JNR^2I)$	$O(R/(\varepsilon\delta))$

Approach 4: Segment Tree Sampling (Ours)

- Our approach will require only $O(NR^2 \log I)$ time per sample, after one-time preprocessing costs of $O(IR^2 + R^3)$ per matrix.
- For $R \approx 10^2$, we achieve a sampling time that is practical for sparse tensor decomposition with mode sizes in the tens of millions.

Preprocessing Time	Time for J Samples	# Samples Required
$O(NIR^2)$	$O(NR^3 + JNR^2 \log I)$	$O(R/(\varepsilon\delta))$

Key Sampling Primitive

$$q[i] := C^{-1} \langle h_{<k} h_{<k}^{\top}, U_k[i,:]^{\top} U_k[i,:], G_{>k} \rangle$$

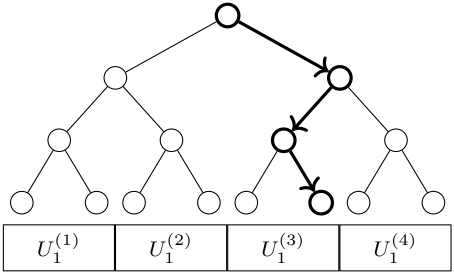
- Imagine we magically had all entries of q - how to sample? Initialize I bins, j-th has width $q[j]$.
- Choose random real r in $[0,1]$, find "containing bin" i such that

$$\sum_{j=0}^{i-1} q[j] < r < \sum_{j=0}^{i} q[j]$$

Binary Tree Inversion Sampling

- Locate bin via binary search (truncated to $\log(I/R)$ levels)
- Root: branch right iff $\sum_{j=0}^{I/2} q\left[j\right] < r$
- Level 2: branch right iff

$$\sum_{j=0}^{I/2} q\left[j\right] + \sum_{j=I/2}^{3I/4} q\left[j\right] < r$$



Key: Can compute summations quickly if we cache information at each node!

Caching Partial Gram Matrices

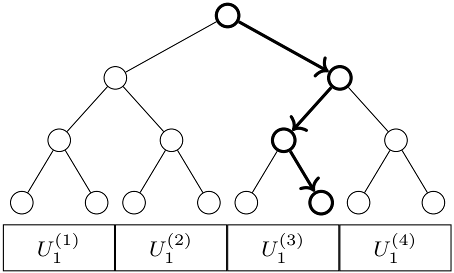
Let an internal node v correspond to an interval of rows $[S(v)...E(v)]$.

$$
\begin{aligned}
\sum_{j=S(v)}^{E(v)} q[j] &= \sum_{j=S(v)}^{E(v)} C^{-1}\langle h_{<k}h_{<k}^\top, U_k[j,:]^\top U_k[j,:], G_{>k}\rangle \\
&= C^{-1}\langle h_{<k}h_{<k}^\top, \sum_{j=S(v)}^{E(v)} U_k[j,:]^\top U_k[j,:], G_{>k}\rangle \\
&= C^{-1}\langle h_{<k}h_{<k}^\top, U_k[S(v):E(v),:]^\top U_k[S(v):E(v),:], G_{>k}\rangle \\
&:= C^{-1}\langle h_{<k}h_{<k}^\top, G^v, G_{>k}\rangle
\end{aligned} \tag{1}
$$

Can compute and store G^v for ALL nodes v in time $O(IR^2)$, storage space $O(IR)$. Use BLAS-3 **syrk** calls in parallel to efficiently construct the tree.

Efficient Sampling after Caching

- At internal nodes, compute $C^{-1}\langle h_{<k} h_{<k}^\top, G^v, G_{>k} \rangle$ in $O(R^2)$ time (read normalization constant from root)
- At leaves, spend $O(R^3)$ time to compute remaining values of q. Can reduce to $O(R^2 \log R)$, see paper.
- Complexity per sample: $O(NR^2 \log I)$ (summed over all tensor modes).



An Empirical Correctness Check

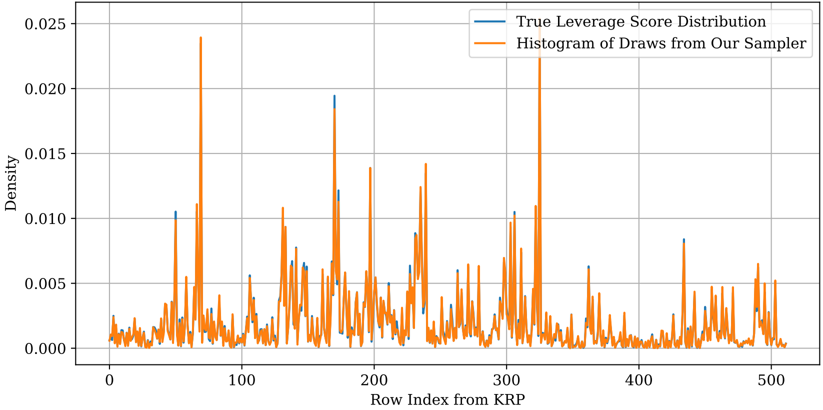


Figure: Distribution Comparison for $U_1 \odot U_2 \odot U_3, U_j \in \mathbb{R}^{8 \times 8}$ initialized i.i.d. Gaussian.

Remarks and Related Work

- The most comparable results to ours appear in work by Woodruff and Zandieh [WZ22], who construct a leverage-score sampler with **input-sparsity runtime**.

- Their sampler can also be used for low-rank approximation, but has $O(N^7)$ worst-case scaling in the KRP dimension.

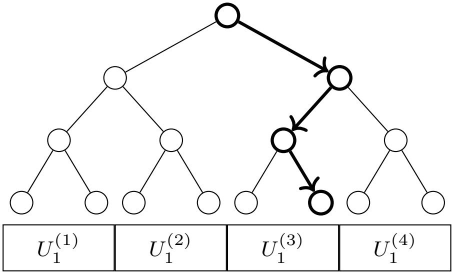
- The best existing **oblivious algorithms** for Khatri-Rao products require either:
  - $\Omega(R^2)$ rows to construct a subspace embedding [Ahl+20].
  - More than input-sparsity runtime.

- **Open Question:** Is LSS **more powerful** than oblivious embedding for Khatri-Rao products, and can we prove a lower bound?

Performance Measurements

High-Performance Parallel Sampling

- We want to execute $J \sim 50,000$ independent random walks down a full, complete tree.
- At each node, execute a matrix-vector multiplication to decide which direction to branch.
- **Solution:** March down the tree one level at a time, computing the branches of ALL random walks using batched GEMV / GEMM.



Runtime Benchmarks (LBNL Perlmutter CPU)

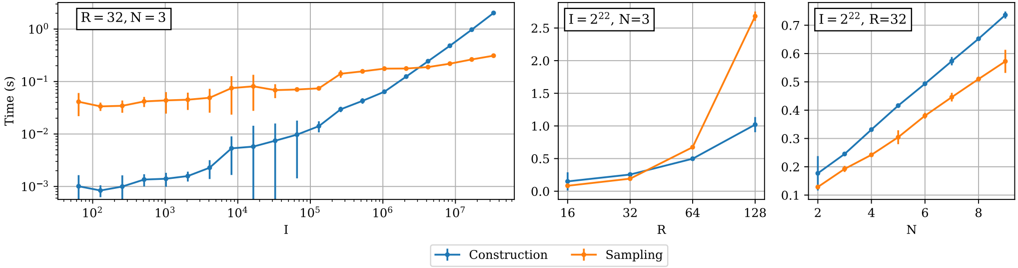


Figure: Time to construct sampler and draw $J = 65,536$ samples. C++ Implementation Linked to OpenBLAS. 1 Node, 128 OpenMP Threads, BLAS3 Construction, BLAS2 Sampling.

Distortion, Ours vs. Approximate Sampling

We define the distortion $D(S, A)$ of sketch S with respect to matrix A by

$$D(S, A) = \kappa(SQ) - 1$$

where Q is any orthonormal basis for the column space of A [Mur+23]. Distortion quantifies the distance preservation property of a sketch.

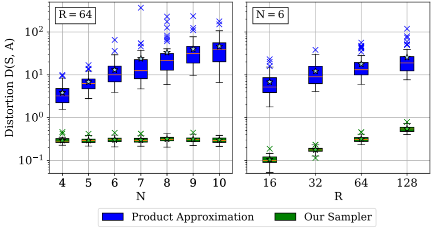


Figure: Sketch distortion as a function of KRP matrix count N and column count R, $J = 65,536$. Green: our sampler. Blue: product approximation by [LK22].

Sparse Tensors from FROSTT

Tensor	Dimensions	NNZ
Uber	$183 \times 24 \times 1.1K \times 1.7K$	$3.3M$
Enron	$6.0K \times 5.6K \times 240K \times 1.2K$	$54M$
NELL-2	$12K \times 9.1K \times 29K$	$77M$
Amazon	$4.8M \times 1.8M \times 1.8M$	$1.7B$
Reddit	$8.2M \times 177K \times 8.1M$	$4.7B$

$$\mathrm{fit}(\tilde{\mathcal{T}}, \mathcal{T}) = 1 - \frac{\left\| \tilde{\mathcal{T}} - \mathcal{T} \right\|_F}{\left\| \mathcal{T} \right\|_F}$$

Accuracy Comparison for Fixed Sample Count

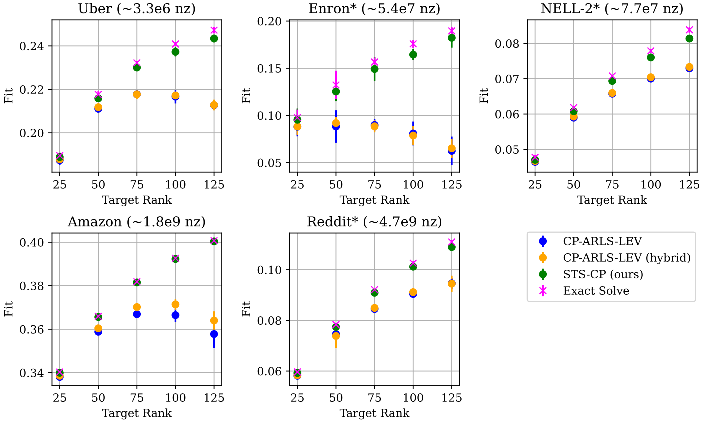


Figure: Sparse tensor ALS accuracy comparison for $J = 2^{16}$ samples, varied target ranks.

STS-CP Makes Faster Progress to Solution

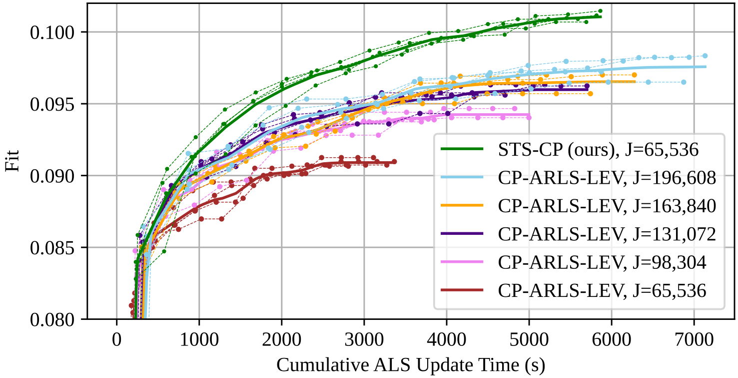
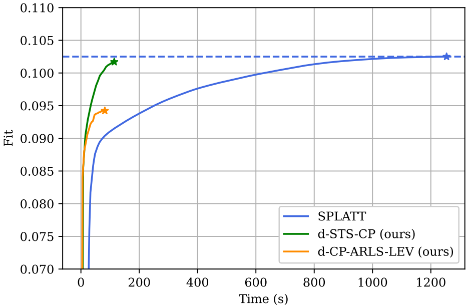


Figure: Fit vs. ALS update time, Reddit tensor, $R = 100$.

Performance at Scale [Bha+24a]



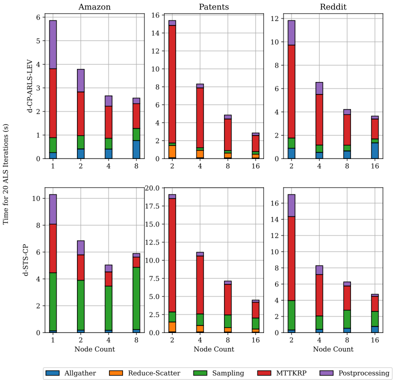


Figure: Fit vs. time, Reddit ($R = 100$) and strong scaling ($R = 25$) for our randomized algorithms.

Conclusions and References

Summary

- We exploited structure in the Khatri-Rao Product to build a leverage score sampler with low polynomial complexity.
- Our implementation relies on well-optimized dense linear algebra primitives, exhibiting strong practical performance in addition to our theoretical guarantees.
- This year, we extended our results to **tensor-train** core chains (find our poster at NeurIPS 2024!) by making key adaptations to our conditional-sampling procedure [Bha+24b].
- Try our code online: `https://github.com/vbharadwaj-bk/fast_tensor_leverage`.

Questions?

References I

[Ahl+20] Thomas D. Ahle, Michael Kapralov, Jakob B. T. Knudsen, Rasmus Pagh, Ameya Velingker, David P. Woodruff, and Amir Zandieh. "Oblivious Sketching of High-Degree Polynomial Kernels". In: *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, 2020, pp. 141–160. DOI: 10.1137/1.9781611975994.9.

[Bha+23] Vivek Bharadwaj, Osman Asif Malik, Riley Murray, Laura Grigori, Aydın Buluç, and James Demmel. "Fast Exact Leverage Score Sampling from Khatri-Rao Products with Applications to Tensor Decomposition". In: *Thirty-seventh Conference on Neural Information Processing Systems*. Dec. 2023.

[Bha+24a] Vivek Bharadwaj, Osman Asif Malik, Riley Murray, Aydın Buluç, and James Demmel. "Distributed-Memory Randomized Algorithms for Sparse Tensor CP Decomposition". In: *Proceedings of the 36th ACM Symposium on Parallelism in Algorithms and Architectures*. SPAA '24. Nantes, France: Association for Computing Machinery, May 2024, pp. 155–168. ISBN: 9798400704161. DOI: 10.1145/3626183.3659980. URL: https://doi.org/10.1145/3626183.3659980.

References II

[Bha+24b] Vivek Bharadwaj, Beheshteh T. Rakhshan, Osman Asif Malik, and Guillaume Rabusseau. "Efficient Leverage Score Sampling for Tensor Train Decomposition". In: *Thirty-eighth Conference on Neural Information Processing Systems*. Dec. 2024. arXiv: 2406.02749 [cs.DS].

[Che+16] Dehua Cheng, Richard Peng, Yan Liu, and Ioakeim Perros. "SPALS: Fast Alternating Least Squares via Implicit Leverage Scores Sampling". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016.

[DM20] Michal Derezinski and Michael W. Mahoney. "Determinantal Point Processes in Randomized Numerical Linear Algebra". In: *CoRR* abs/2005.03185 (2020). arXiv: 2005.03185.

[KB09] Tamara G. Kolda and Brett W. Bader. "Tensor Decompositions and Applications". In: *SIAM Review* 51.3 (Aug. 2009). Publisher: Society for Industrial and Applied Mathematics, pp. 455–500. ISSN: 0036-1445. DOI: 10.1137/07070111X.

[LK22] Brett W. Larsen and Tamara G. Kolda. "Practical Leverage-Based Sampling for Low-Rank Tensor Decomposition". In: *SIAM Journal on Matrix Analysis and Applications* 43.3 (2022), pp. 1488–1517.

References III

[Mal22] Osman Asif Malik. “More Efficient Sampling for Tensor Decomposition With Worst-Case Guarantees”. In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, July 2022, pp. 14887–14917.

[MS22] Linjian Ma and Edgar Solomonik. “Cost-efficient Gaussian tensor network embeddings for tensor-structured inputs”. In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., 2022, pp. 38980–38993.

[Mur+23] Riley Murray, James Demmel, Michael W. Mahoney, N. Benjamin Erichson, Maksim Melnichenko, Osman Asif Malik, Laura Grigori, Piotr Luszczek, Michal Derezinski, Miles E. Lopes, Tianyu Liang, Hengrui Luo, and Jack Dongarra. *Randomized Numerical Linear Algebra: A Perspective on the Field With an Eye to Software*. Tech. rep. UCB/EECS-2023-19. EECS Department, University of California, Berkeley, Feb. 2023.

[WZ22] David Woodruff and Amir Zandieh. “Leverage Score Sampling for Tensor Product Matrices in Input Sparsity Time”. In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, July 2022, pp. 23933–23964.