# Research and Creative Experience for Undergraduates Program (RCEU) Summer 2021

# Optimizing Sparse Tensor Computations via Orderings and Multilayered Data-Structures

*Trevor Garnett, Dr. Joshua Dennis Booth*

*Department of Computer Science*

## Introduction

The purpose of this research was to study the effect of different reordering schemes on the performance of CPD which is a sparse tensor computation used in training neural networks.

**What is a tensor?**
A tensor is a generalization of a matrix or a vector; in other words, a tensor is an n-dimensional structure containing data.

**What does it mean for a tensor to be sparse?**
This means that the tensor has a lot of 0's. Due to the relative size of the tensor to the few non-zero elements, we would like to form a data structure that does not do computations on the zero elements and moves the non-zero elements close to each other in memory

**What is CPD?**
CPD is a complicated tensor computation that "attempts to decompose a tensor into a set of rank-one tensors"[2] known by two names:
1. Canonical Polyadic Decomposition
2. CANDECOMP/PARAFAC Decomposition.

**How could a reordering scheme affect the speed of CPD?**
Reordering schemes can theoretically improve both spatial locality and temporal locality.
- Temporal locality is the reuse of data within a short period of time.
- Spatial Locality is the use of data with relatively close storage locations.

*Example: A 3-dimensional tensor reordered to show greater spatial locality*

$$\left|\begin{array}{cccc|cccc|cccc} 0&3&0&3&0&0&0&0&2&0&0&2\\ 0&0&0&0&1&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&1&0&0&0&0&0\\ 0&3&0&3&0&0&0&0&2&0&2&0 \end{array}\right| \longrightarrow \left|\begin{array}{cccc|cccc|cccc} 3&3&0&0&0&0&0&0&0&0&0&0\\ 3&3&0&0&0&2&2&0&0&0&0&0\\ 0&0&0&0&0&2&2&0&0&0&1&1\\ 0&0&0&0&0&0&0&0&0&0&1&1 \end{array}\right|$$

## Background

Sparse tensor computations are ubiquitous in both scientific and commonplace applications; they are in used by recommender systems – such as the one used by Netflix, in network traffic analysis, and in data mining. Note that this is not an all-inclusive list. As pervasive as sparse tensor computations are, it makes sense to attempt to make them faster.

It is known that orderings "can lead to significant performance gains... [by] exploiting spatial and temporal locality"[2]. However; the problem of finding a perfect reordering is NP-hard and thus does not have a solution. However, there are several heuristics that attempt to find an approximate solution. There are two main papers that introduce orderings, and both introduce more than one ordering. The orderings that produced the most speedup in each paper, which would be SPLATT's mode-independent ordering[2] and HiParTI's Lexi-Order[3], will be evaluated. The ordering schemes will be evaluated in terms of **efficiency** (the amount of time required to order a tensor) and **effectiveness** (the amount of speedup achieved over the non-ordered tensor with an equivalent number of cores).

Briefly, SPLATT's ordering algorithm works by doing a k-way partition of a tensor's indices and then permuting them accordingly. Thus, SPLATT's ordering algorithm with both a 3-way partition and a 6-way partition will be observed.

### References

1. Karypis, George et al. - http://frostt.io/tensors/
2. Karypis, George, et al. "SPLATT: Efficient and PARALLEL Sparse Tensor-Matrix Multiplication." IEEE Xplore, IEEE, 20 July 2015, ieeexplore.ieee.org/document/7161496/.
3. Karypis, George et al. - https://github.com/ShadenSmith/splatt
4. Li, Jiajia, et al. "HiCOO: Hierarchical Storage of SPARSE TENSORS." SC18: International Conference for High Performance Computing, Networking, Storage and Analysis, 2018, doi:10.1109/sc.2018.00022.
5. Li, JiaJia et al. - https://github.com/pnnl/HiParTI

## Results

**Tensor Suite**

| Tensor | Dimensions | Non-Zeros |
|---|---|---|
| nell-1 | 2.902M x 2.143M x 25.50M | 143.6M |
| nell-2 | 12.09K x 9.184K x 28.82K | 76.88M |
| nips | 2.482K x 2.862K x 14.04K x 17 | 3.101M |
| crime | 6.186K x 24 x 77x 32 | 5.330M |
| enron | 6.066K x 5.699K x 244.3K x 1.176K | 54.20M |
| flickr | 319.7K x 28.15M x 1.607M x 731 | 112.9M |
| delicious | 532.9K x 17.26M x 2.480M x 1.443K | 140.1M |

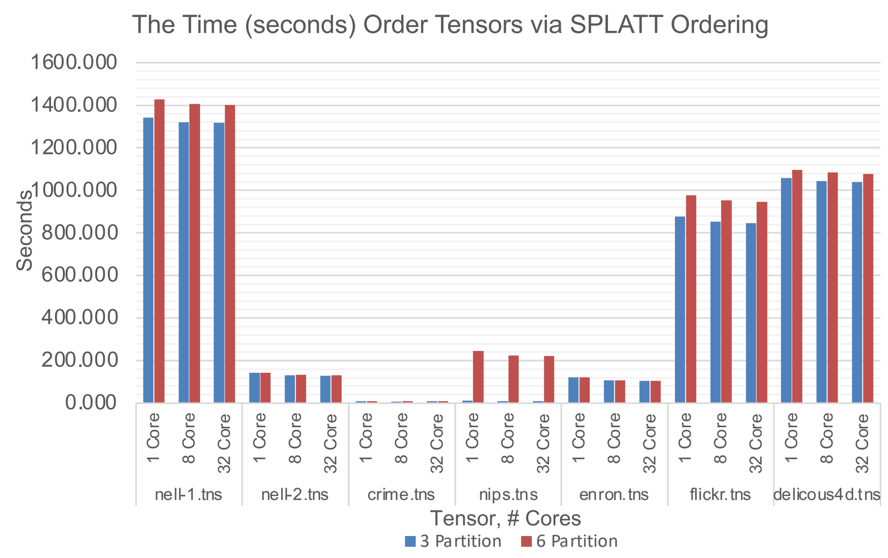


Figure 1A: Time to order tensors via the SPLATT orderings.

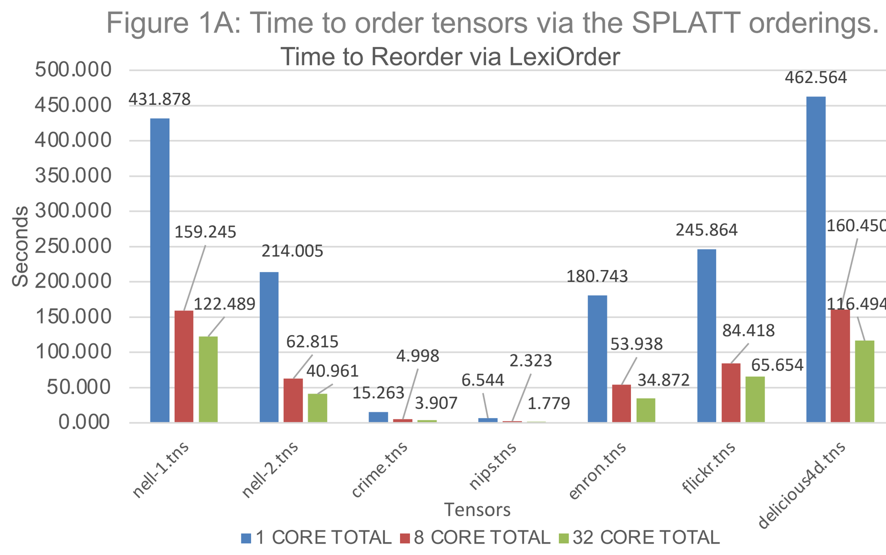


Figure 1B: Time to order tensors via Lexi-Order.

*Figure 1A* demonstrates that a 3-way partition ordering is always faster than a 6-way partition ordering. Excluding nips, which was ordered 22x faster via a 3-way partition than a 6-way partition, a 3-way partition averages to be over 3% faster than a 6-way partition ordering. *Figure 1B* displays the scalability of Lexi-Order, since the time to order the tensors decreased with more cores on each tensor.

We observe that SPLATT, both 3-way and 6-way partitioning, is faster sequentially on nell-2, crime, and enron; Lexi-Order is faster sequentially on nell-1, nips, flickr, and delicious. Thus, the performance in serial is dependent on the tensor. However, when parallelism is introduced, since Lexi-Order scales well and SPLATT does not, Lexi-Order is consistently faster with the higher core counts. Thus, with both 8 and 32 cores, Lexi-Order is much faster than either of SPLATT's orderings. Overall, we observe that **Lexi-Order is more efficient.**

*Figure 2* displays a measure of speedup acquired by an ordering on each tensor with different core counts. This number is acquired by dividing the time taken by CPD on a tensor with no reordering by the time taken by CPD on a tensor with an ordering. In other words:

$$\text{Speedup} = \frac{\text{time with no ordering}}{\text{time with ordering}}$$

Therefore, points above the line (> 1) indicate a speedup caused by the reordering; points below the line indicate a slowdown. Note that not all points lie above the line, since these are just heuristics. However, averaged across all seven tensors, there tends to be a speedup. The results made it is difficult to access which ordering is the most effective.

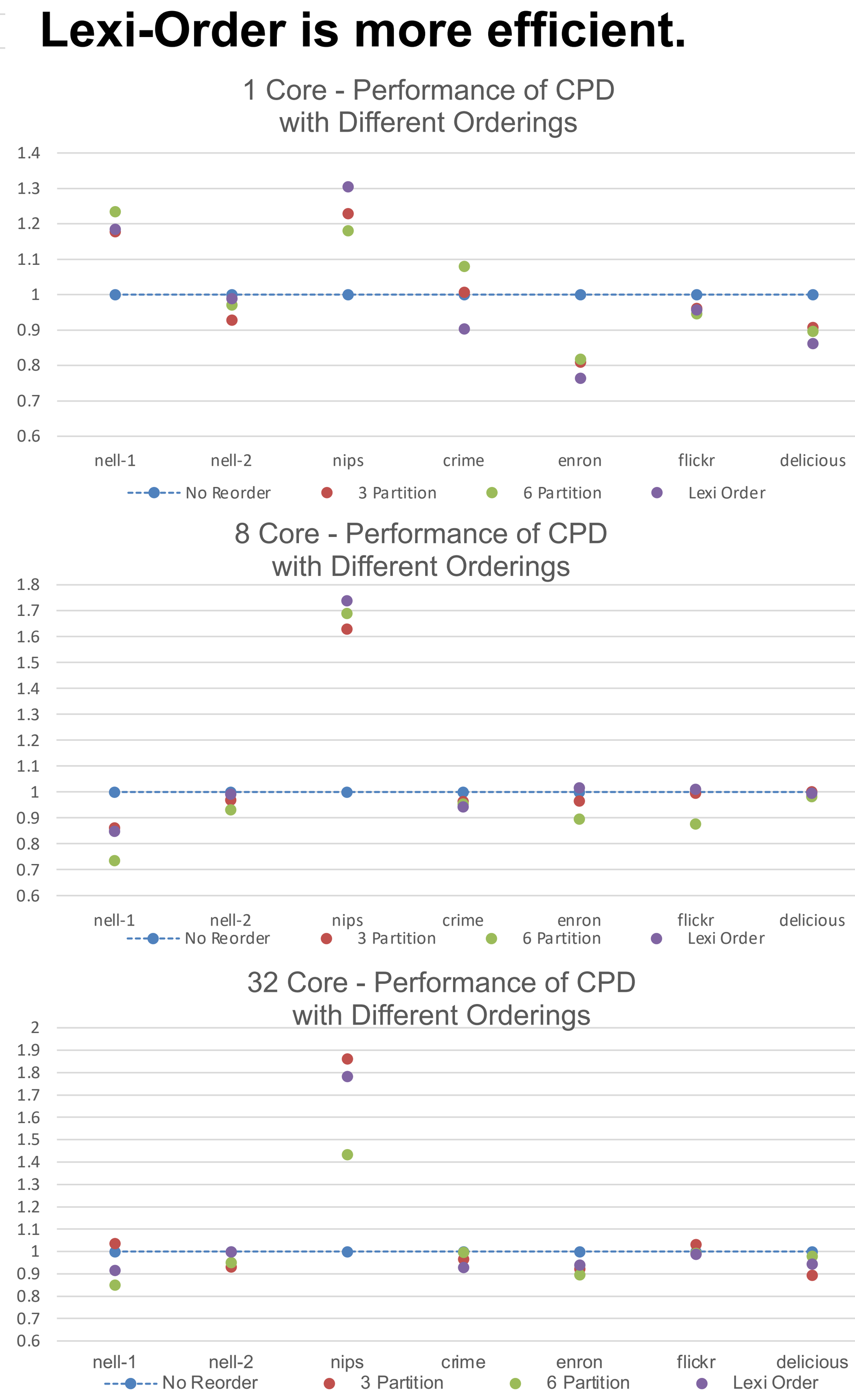


Figure 2: Measures the speedup (i.e. $\frac{\text{time with no ordering}}{\text{time with ordering}}$)

## Impact

This research is the
- First work to compare different ordering schemes, besides the paper that introduced Lexi-Order – it will thus be a guideline for future research in the field.
- It was found that Lexi Order is much more efficient, and that Lexi-Order and 3-Partition SPLATT are similarly effective, since they see similar average speedup.

## Acknowledgements

We would like to thank the UAH Office of the President, Office of the Provost, Office of the Vice President for Research and Economic Development, The Dean of the College of Science, the Dean of the College of Engineering, the Alabama Louis Stokes Alliances for Minority Participation, and the Alabama Space Grant Consortium for funding during the RCEU program. This research will continue under funding by NSF Grant 2044633 "CAREER: Fast, Energy Efficient Irregular Kernels via Neural Acceleration"

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE