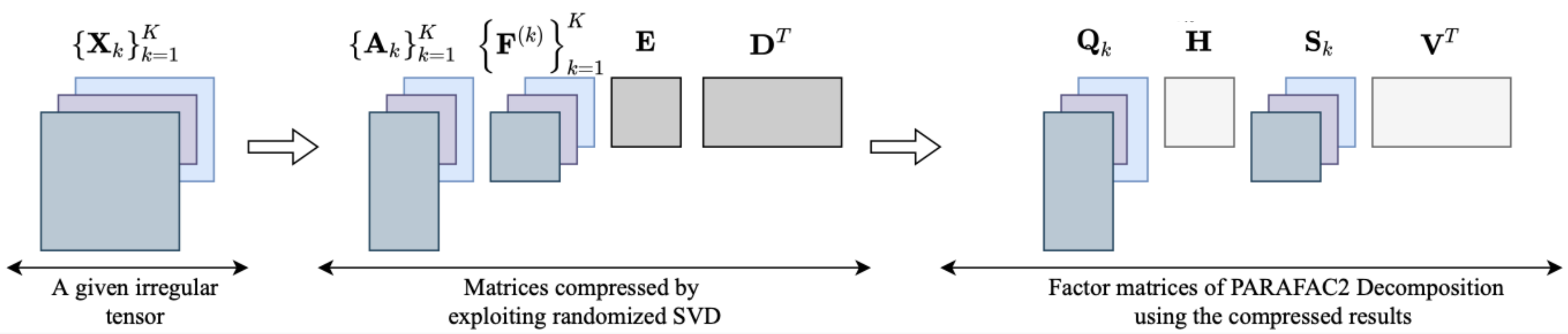
DPar2: Fast and Scalable PARAFAC2 Decomposition for Irregular Dense Tensors

ICDE 2022

Jun-Gi Jang and U Kang
Data Mining Lab
Dept. of CSE
Seoul National University

Overview

- **Q.** Given an irregular dense tensor, how can we efficiently analyze the tensor?
  - Irregular tensor: a collection of matrices whose columns have the same size and rows have different sizes from each other
- **A.** **DPar2,** a fast and scalable tensor decomposition method, efficiently analyzes the irregular tensor

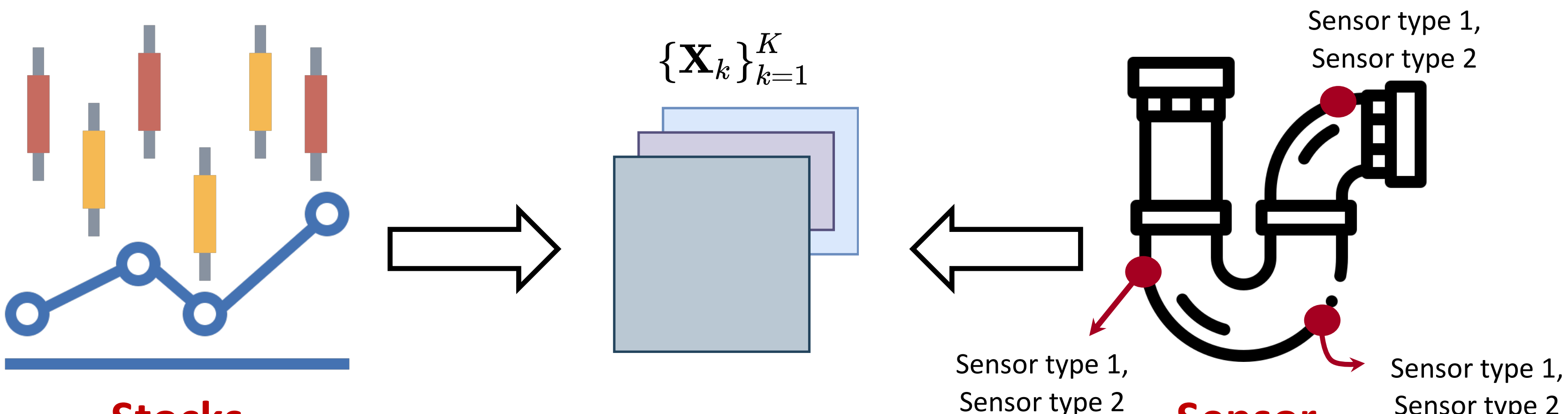


Outline

- **Introduction**
- Proposed Method
- Experiments
- Conclusion

Irregular Dense Tensors

- Several real-world data are represented as **irregular dense tensors**
  - A collection of matrices whose columns have the same size and rows have different sizes from each other



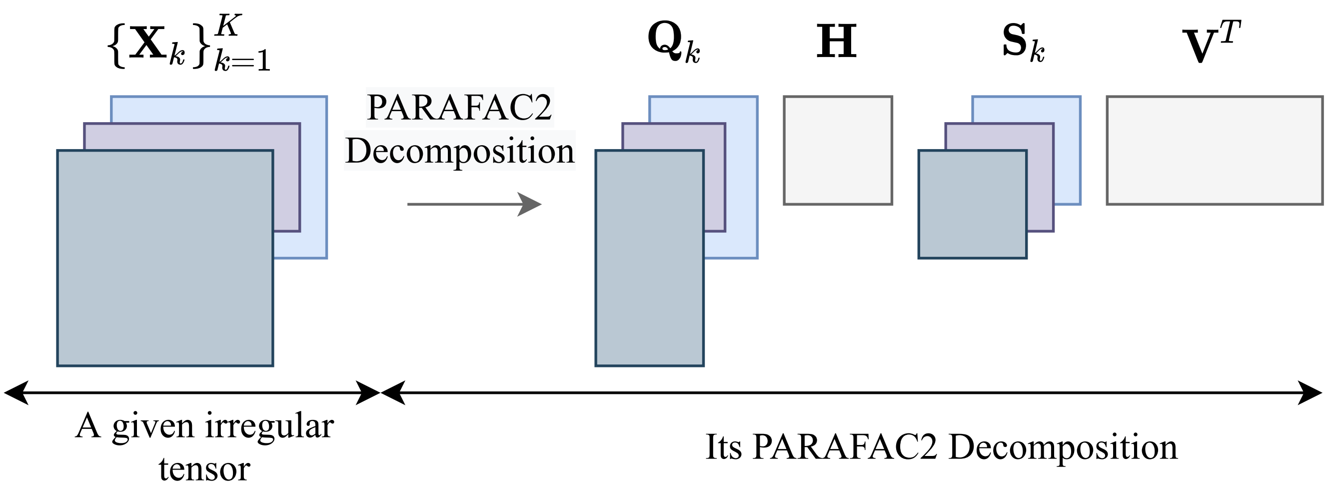
Stocks
Index: (time, feature, stock)
Value: measurement

An irregular dense tensor

Sensor
Index: (time, location, sensor)
Value: measurement

PARAFAC2 Decomposition

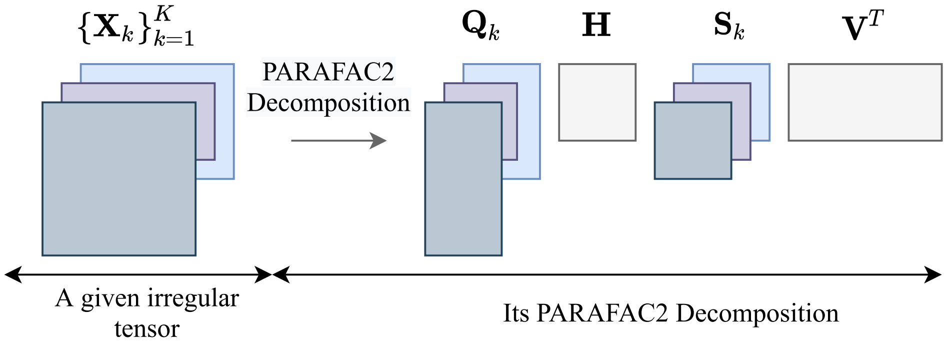
- ***How can we analyze an irregular dense tensor?***
- **PARAFAC2 Decomposition**
  - A fundamental tool to analyze irregular tensors
  - Recently, it has been re-popularized for analysis of electronic health records (EHR) data represented as an irregular tensor



PARAFAC2 Decomposition

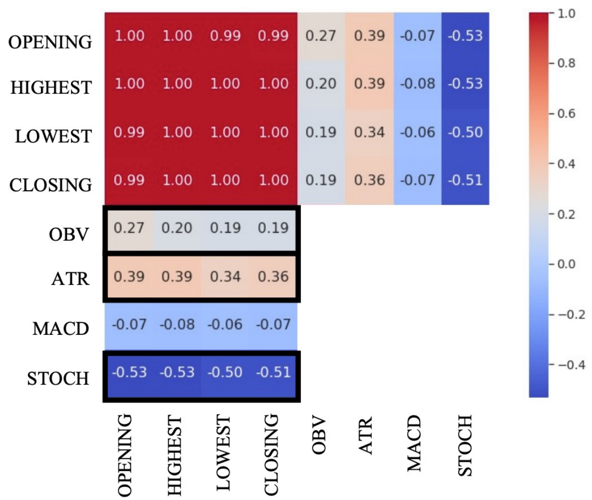
- **Given** an irregular tensor $\{\mathbf{X}_k\}_{k=1}^{K}$, rank R
  - Slice matrix $\mathbf{X}_k \in \mathbb{R}^{I_k \times R}$
- **Obtain** obtain factor matrices $\mathbf{Q}_k \in \mathbb{R}^{I_k \times R}$, $\mathbf{H} \in \mathbb{R}^{I_k \times R}$, $\mathbf{S}_k \in \mathbb{R}^{R \times R}$, $\mathbf{V} \in \mathbb{R}^{J \times R}$ for k=1,…,K
- **Objective function**

$$\min_{\mathbf{Q}_k, \mathbf{H}, \mathbf{S}_k, \mathbf{V}} \sum_{k=1}^{K} \|\mathbf{X}_k - \mathbf{Q}_k \mathbf{H} \mathbf{S}_k \mathbf{V}^T\|_F^2$$



Application

- **Several applications for PARAFAC2 decomposition**
  - Dimensionality reduction, anomaly detection, trend analysis, and phenotype discovery
  - For example, given a stock data (time, feature, stock)



(a) US stock data

Feature analysis

Q: MSFT

(a) Similarity based Result

Rank	Stock Name	Sector
1	Adobe	Technology
2	Amazon.com	Consumer Cyclical
3	Apple	Technology
4	Moody's	Financial Services
5	**Intuit**	Technology
6	ANSYS	Technology
7	Synopsys	Technology
8	**Alphabet**	Communication Services
9	**ServiceNow**	Technology
10	**EPAM Systems**	Technology

Similarity search

Alternating Least Square

- **ALS (Alternating Least Square) is widely used for obtaining factor matrices of PARAFAC2 Decomposition**
  - *Iteratively* updates a factor matrix of a mode while fixing all factor matrices of other modes
  - **(Heavy computational costs)** Require computations with a given tensor at each iteration
    - For example, ALS needs to compute $\mathbf{X}_k\mathbf{V}\mathbf{S}_k\mathbf{H}$ for all k at each iteration

      $\mathbf{X}_k \in \mathbb{R}^{I_k \times J}$ $\quad$ $\mathbf{V}\mathbf{S}_k\mathbf{H} \in \mathbb{R}^{J \times R}$
      - Its computational cost is $O\left(\sum_{k=1}^{K} I_k JR\right)$ proportional to the size of an irregular tensor

Limitation of Previous Works

- **Limitations of previous works**
  - They fail to handle an irregular dense tensor, efficiently
    - Each iteration requires computations involved with an irregular tensor
  - There remains a need for fully employing multicore parallelism

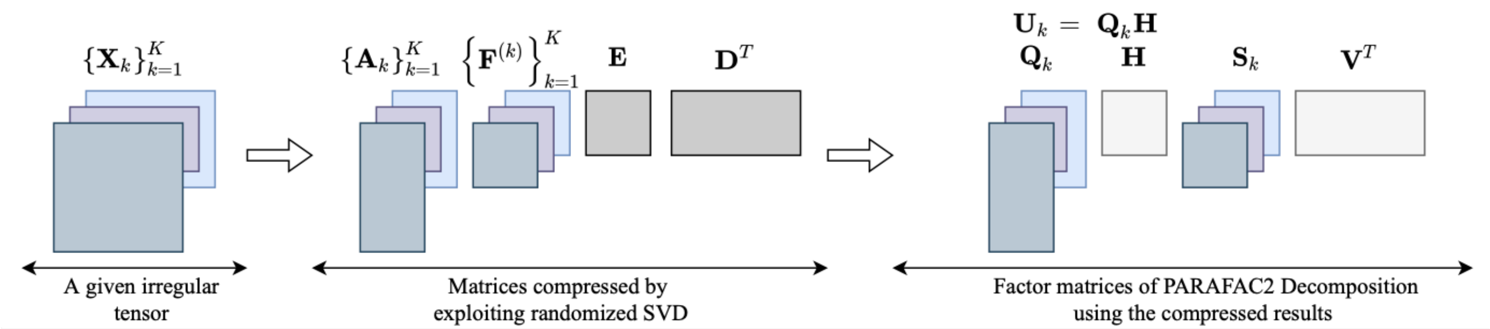
We need to make PARAFAC2 decomposition **faster and more scalable**, to analyze large-scale irregular dense tensors

Outline

- Introduction
- **Proposed Method**
- Experiments
- Conclusion

Proposed Method

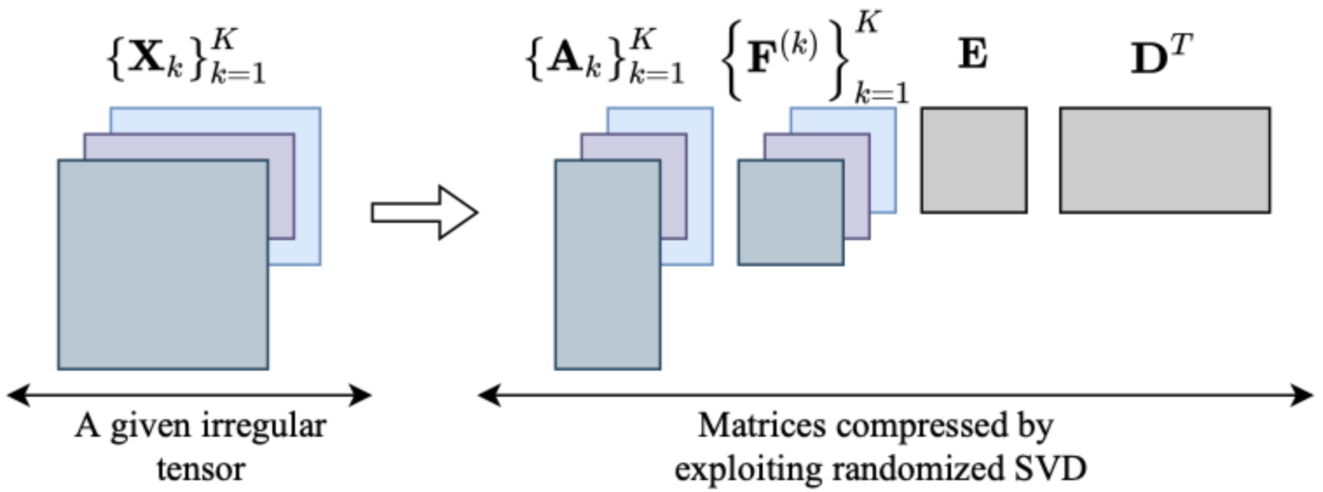
- We propose **DPar2** (**D**ense **PAR**AFAC**2** Decomposition)
  - A **fast** and **scalable** PARAFAC2 decomposition method for irregular dense tensors



- **(Idea 1)** Compressing an irregular tensor using randomized SVD (Singular Value Decomposition)
- **(Idea 2)** Careful reordering of computations with the compression results
  - Exploiting properties of operations and matrices
- **(Idea 3)** Careful distribution of work between threads by considering various lengths of matrices

Compression

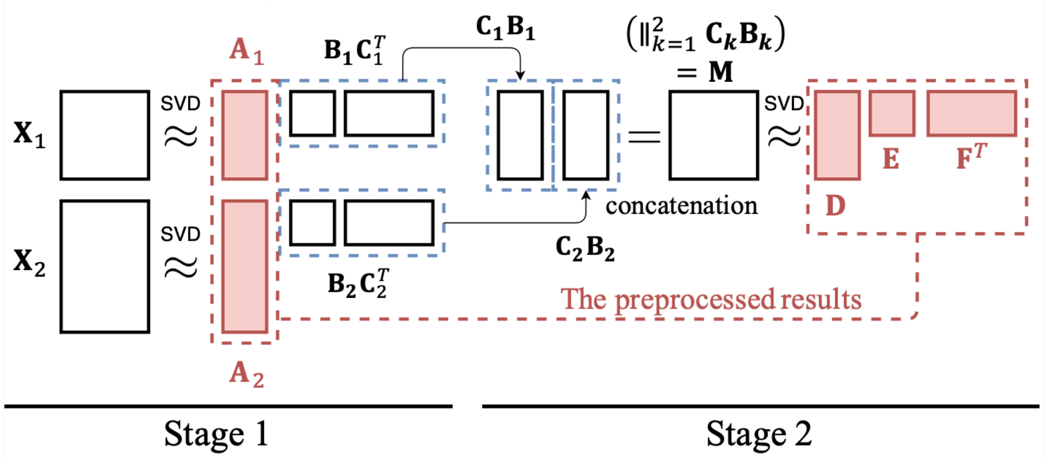
- **Compressing an irregular tensor before iterations**
  - The result is much smaller than an input irregular tensor



- The compression is performed once before iterations, and only the compression results are used at iterations

Compression

- **Compressing an irregular tensor using Randomized SVD**
  - Randomized SVD (Singular Value Decomposition) efficiently compresses matrices with low errors
    - It efficiently computes $\mathbf{X} \approx \mathbf{USV}^T$
- There are two compression stage
  - Stage 1 - compress each slice matrix using randomized SVD
  - Stage 2 - further compress the intermediate data from the first stage



Compression

Details

- **Stage 1** - compress each slice matrix using randomized SVD
  - For all k, compute $\mathbf{X}_k \approx \mathbf{A}_k \mathbf{B}_k \mathbf{C}_k^T$
- **Stage 2** - further compress the intermediate data from the first stage
  - Construct a matrix $\mathbf{M} = \ ||_{k=1}^{K} \mathbf{C}_k \mathbf{B}_k$ by horizontally concatenating $\mathbf{C}_k \mathbf{B}_k$
  - Then, compute $\mathbf{M} \approx \mathbf{D}\mathbf{E}\mathbf{F}^T$

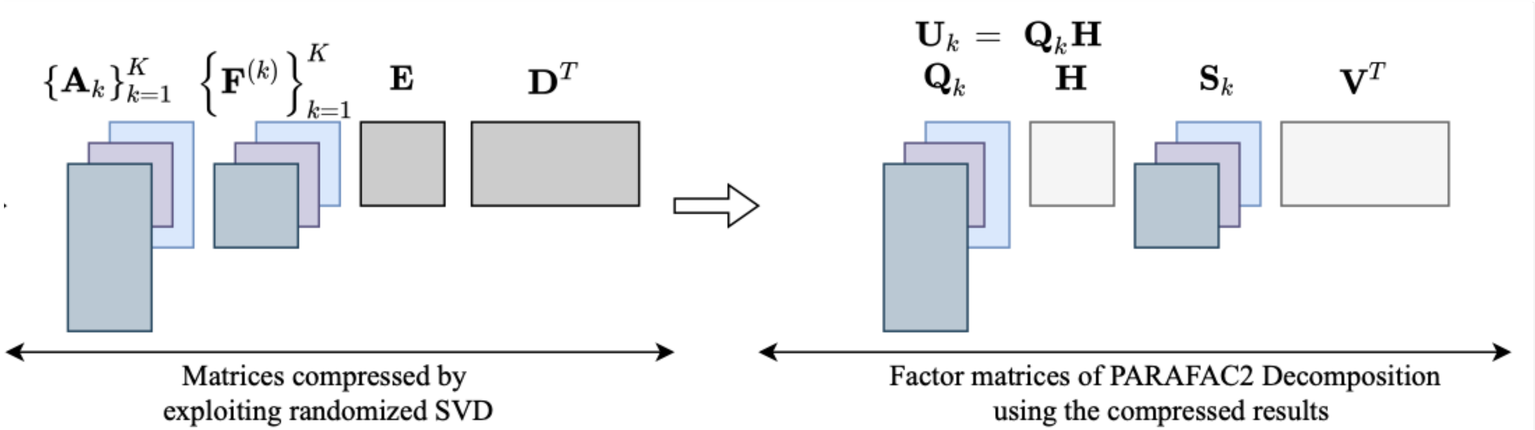
$$\mathbf{F} = \begin{bmatrix} \mathbf{F}^{(1)} \\ \vdots \\ \mathbf{F}^{(K)} \end{bmatrix}$$

- The final output of the compression is $\mathbf{A}_k \mathbf{F}^{(k)} \mathbf{E} \mathbf{D}^T \approx \mathbf{X}_k$
  - $\mathbf{A}_k \in \mathbb{R}^{I_k \times R}$ and $\mathbf{F}^{(k)} \in \mathbb{R}^{R \times R}$ are generated from each slice matrix
  - Only one $\mathbf{E} \in \mathbb{R}^{R \times R}$ and $\mathbf{D} \in \mathbb{R}^{J \times R}$ are generated across all slice matrices

Due to the two-stage compression, we **efficiently** obtain the compression results much **smaller** than an input tensor

Updating Factor Matrices

- **Update factor matrices by exploiting the compression results**
  - **(Naïve approach)** would update factor matrices after reconstruction, but it requires high computational costs and space costs
- **(Idea)** Careful reordering of computations with the compression results



Updating Factor Matrices

Update Procedure of DPar2

Input: $\mathbf{A}_k\mathbf{F}^{(k)}\mathbf{E}\mathbf{D}^T (\approx \mathbf{X}_k)$ for $k = 1, \dots, K$, target rank R

Output: $\mathbf{Q}_k$, $\mathbf{H}$, $\mathbf{S}_k$, $\mathbf{V}$ for $k = 1, \dots, K$

- **Update $\mathbf{Q}_k$** ⟵
- Update $\mathbf{H}$
- Update $\mathbf{S}_k$
- Update $\mathbf{V}$

- **Update $\mathbf{Q}_k$ using the compression results**
- **Naïve Computation (High Cost)**
  - Reconstruct slice matrices from the compression results
  - Compute $\mathbf{Q}_k$ using the reconstructed one
- **Our computation (Low Cost)**
  - Improve efficiency by **avoiding reconstruction** and **redundant** computations for $\mathbf{A}_k$
    - Exploit the property of $\mathbf{A}_k \in \mathbb{R}^{I_k \times J}$
    - $\mathbf{A}_k$ is a column orthogonal matrix, i.e., $\mathbf{A}_k^T\mathbf{A}_k = \mathbf{I}$

Updating Factor Matrices

Update Procedure of DPar2

Input: $\mathbf{A}_k\mathbf{F}^{(k)}\mathbf{E}\mathbf{D}^T(\approx \mathbf{X}_k)$ for $k = 1, \dots, K$, target rank R

Output: $\mathbf{Q}_k$, $\mathbf{H}$, $\mathbf{S}_k$, $\mathbf{V}$ for $k = 1, \dots, K$

- Update $\mathbf{Q}_k$
- **Update $\mathbf{H}$** ⟸
- **Update $\mathbf{S}_k$** ⟸
- **Update $\mathbf{V}$** ⟸

- **Update $\mathbf{H}$, $\mathbf{S}_k$, $\mathbf{V}$**
- Use small factorized matrices (e.g., $\mathbf{A}_k$, $\mathbf{F}^{(k)}$, $\mathbf{E}$, $\mathbf{D}$)
  - They are much smaller than an input tensor
- Carefully reordering of computations with the compression results

With these ideas, we reduce the computational costs and avoid generating large intermediate data

Updating Factor Matrices

Update Procedure of DPar2

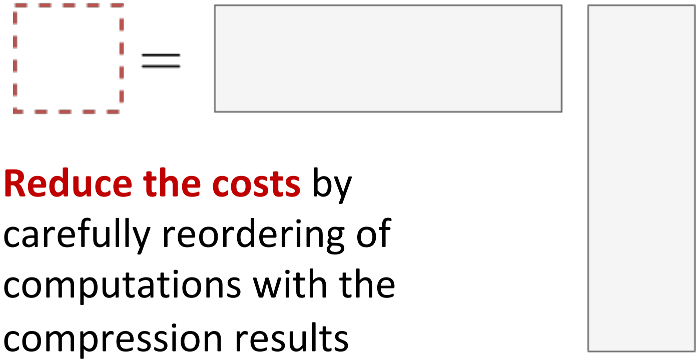
Input: $\mathbf{A}_k\mathbf{F}^{(k)}\mathbf{E}\mathbf{D}^T(\approx \mathbf{X}_k)$ for $k = 1, \ldots, K$, target rank R

Output: $\mathbf{Q}_k$, $\mathbf{H}$, $\mathbf{S}_k$, $\mathbf{V}$ for $k = 1, \ldots, K$

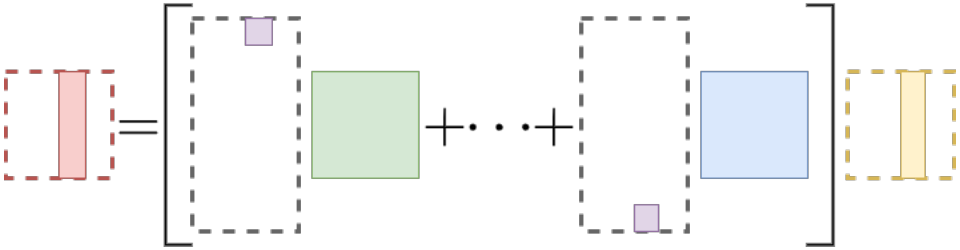
- Update $\mathbf{Q}_k$
- **Update $\mathbf{H}$**
- **Update $\mathbf{S}_k$**
- **Update $\mathbf{V}$**

- **Update $\mathbf{H}$, $\mathbf{S}_k$, $\mathbf{V}$**

Naïve computation with large matrices



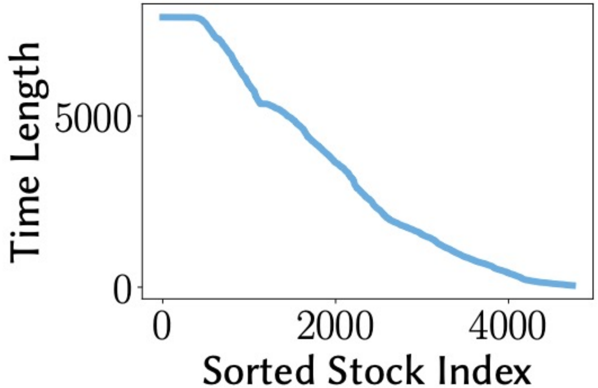
Reduce the costs by carefully reordering of computations with the compression results



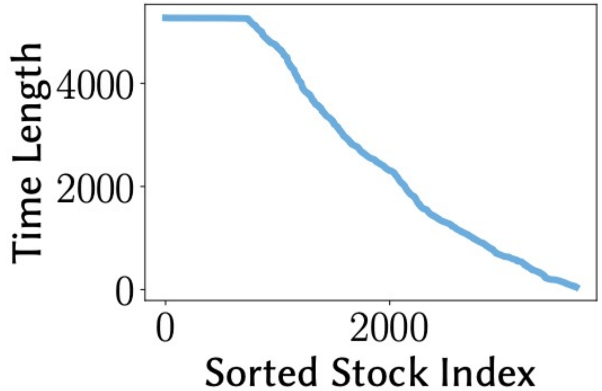
Our computation with small matrices

Multi-core Parallelism

- Given an irregular tensor, the number of rows of slice matrices is different
- For example, stocks have different time lengths due to listing periods



(a) US stock data



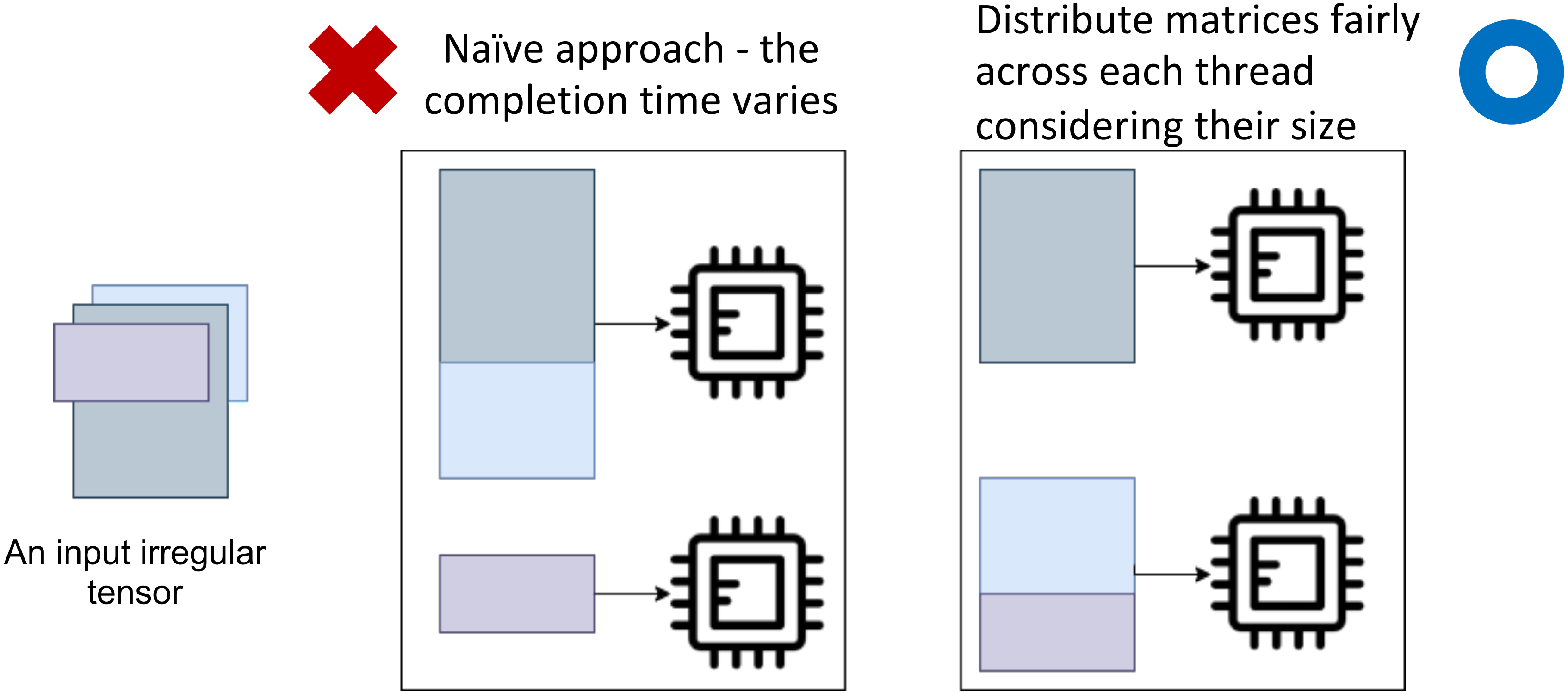
(b) KR stock data

- The **length** of the temporal dimension of input slices
- We sort the lengths in descending order

- No method considers this difference for parallelism

Multi-core Parallelism

- **Careful distribution of work between threads by considering various lengths of matrices**
  - Computational costs of handling a matrix are proportional to its size



Outline

- Introduction
- Proposed Method
- **Experiments**
- Conclusion

Experimental Questions

- **Q1. (Performance)** How quickly and accurately does DPar2 perform PARAFAC2 decomposition compared to other methods?

- **Q2. (Scalability)** How well does DPar2 scale up with respect to tensor size and target rank? How much does the number of threads affect the running time of DPar2?

- **Q3. (Discovery**) What can we discover from real-world tensors using DPar2?

Dataset

- **Dataset**

TABLE II
DESCRIPTION OF REAL-WORLD TENSOR DATASETS.

Dataset	Max Dim. I_k	Dim. J	Dim. K	Summary
FMA[1] [26]	704	2,049	7,997	music
Urban[2] [27]	174	2,049	8,455	urban sound
US Stock[3]	7,883	88	4,742	stock
Korea Stock[4] [3]	5,270	88	3,664	stock
Activity[5] [28], [29]	553	570	320	video feature
Action[5] [28], [29]	936	570	567	video feature
Traffic[6] [30]	2,033	96	1,084	traffic
PEMS-SF[7]	963	144	440	traffic

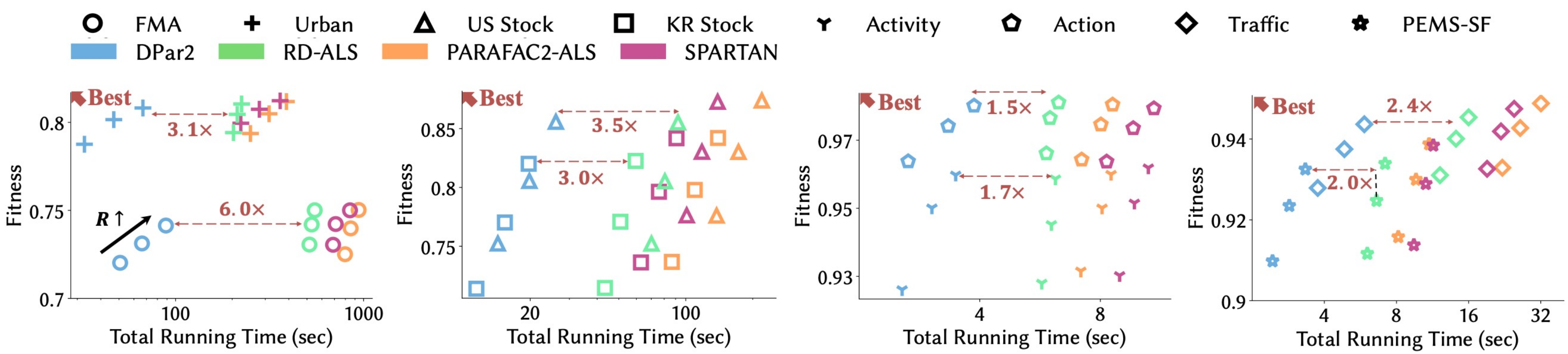
- Each slice matrix of an irregular tensor has different I_k
- J is the size of the common axis
  - The column size of slice matrices
- K is the number of slice matrices in an irregular tensor

Experimental Setting

- **Competitors**
  - 3 existing PARAFAC2 decomposition methods for irregular tensors
    - **PARAFAC2-ALS**: PARAFAC2 decomposition based on ALS approach
    - **RD-ALS**: PARAFAC2 decomposition which preprocesses a given irregular tensor
    - **SPARTAN**: fast and scalable PARAFAC2 decomposition for irregular sparse tensors
- **Metric**
  - **Fitness**: $1 - \left(\frac{\sum_{k=1}^{K} \|\mathbf{X}_k - \tilde{\mathbf{X}}_k\|_F}{\sum_{k=1}^{K} \|\mathbf{X}_k\|_F}\right)$
    - Fitness close to 1 indicates that a model approximates a given input tensor well

Q1. Performance Trade-off

- The upper-left region indicates better performance

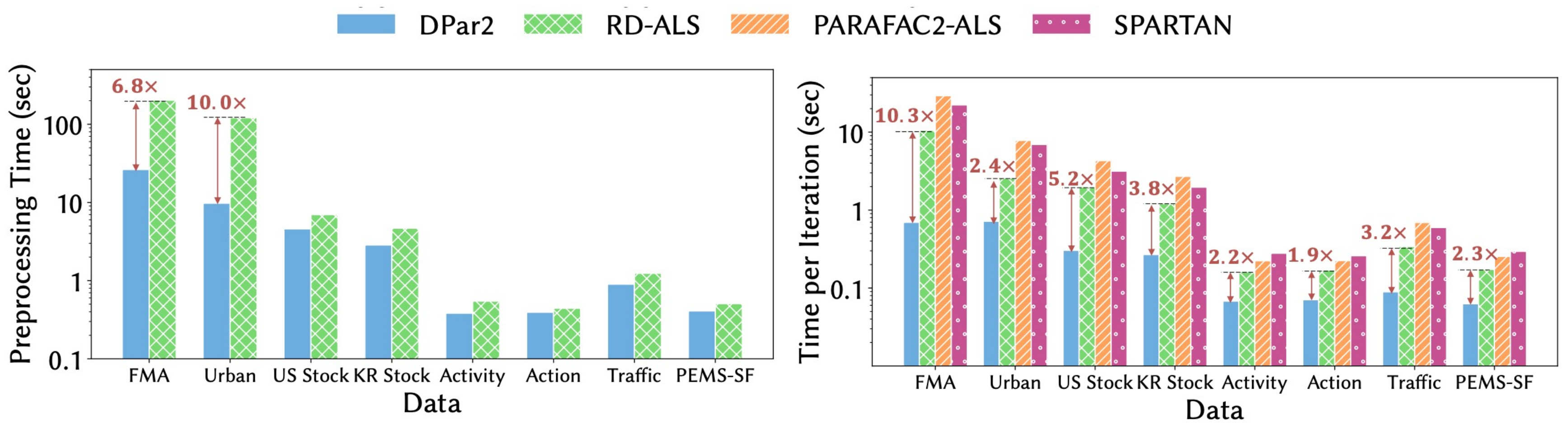


(a) Trade-off (b) Trade-off (c) Trade-off (d) Trade-off

DPar2 **outperforms** the competitors, giving up to **6× faster** than competitors while having comparable fitness

Q1. Performance
Running Time

- Measure preprocessing time and iteration time

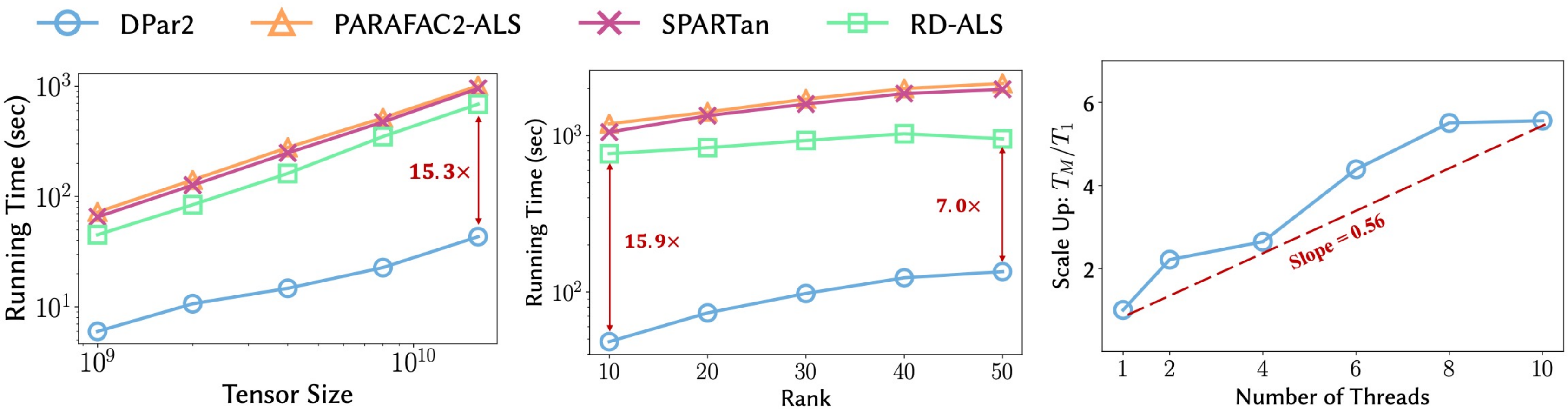


(a) Preprocessing time (b) Iteration time

- Preprocessing time of Dpar2 is faster than RD-ALS which has preprocessing step for an irregular tensor
- Iteration time of DPar2 is **up to 10.3× faster** than competitors due to small compressed data

Q2. Scalability

- Measure scalability on synthetic irregular tensors



(a) Scalability for tensor size (b) Scalability for rank (c) Machine Scalability

- DPar2 is more **scalable** than other PARAFAC2 decomposition methods in terms of both tensor size and rank
- DPar2 gives near-linear machine scalability

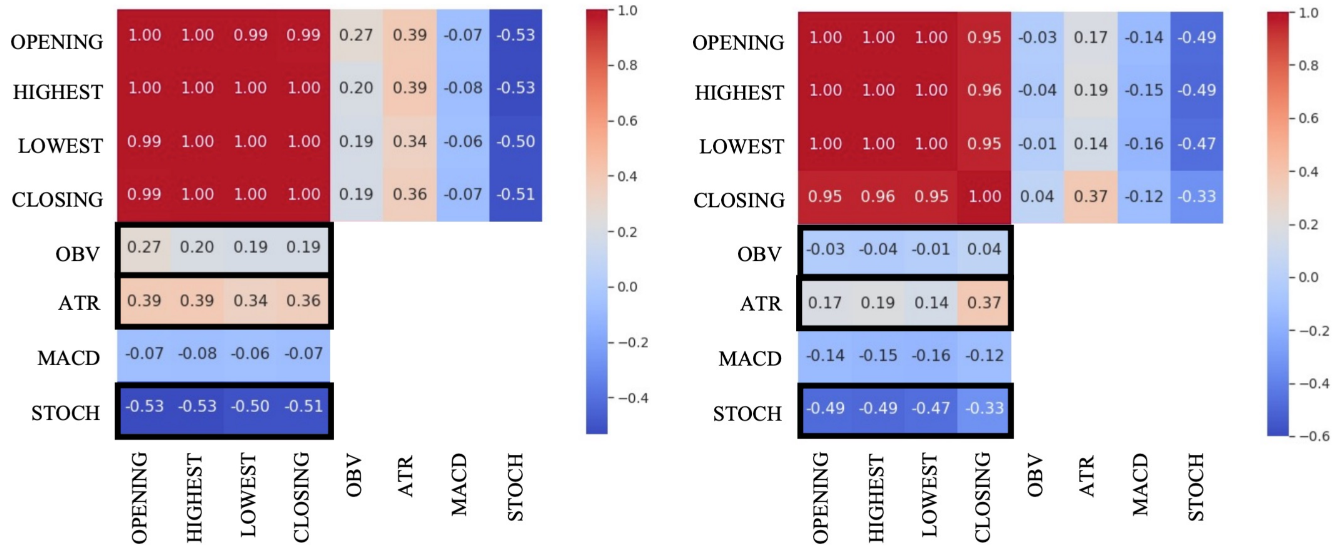
Q3. Discovery

- Given Korean stock and US stock datasets in the form of (time, features, stock), we compare the results between the two datasets

1. Perform DPar2 for Korea stock and US stock datasets, respectively
2. For each dataset, compute Pearson Correlation Coefficient (PCC) between $\mathbf{V}(i,:)$ which are a factor vector of a feature (e.g., opening price, trading volume, and technical indicators)
3. Visualize the correlations
   - For effective visualization, we pick 4 price features and 4 representative technical indicators
     - 4 price features: the opening, the closing, the highest, and the lowest prices
     - 4 representative technical indicators: OBV, ATR, MACD, and STOCH

Q3. Discovery

- Due to the difference between the two markets in terms of market size, market stability, tax, investment behavior, etc., the patterns are different



(a) US stock data (b) Korea stock data

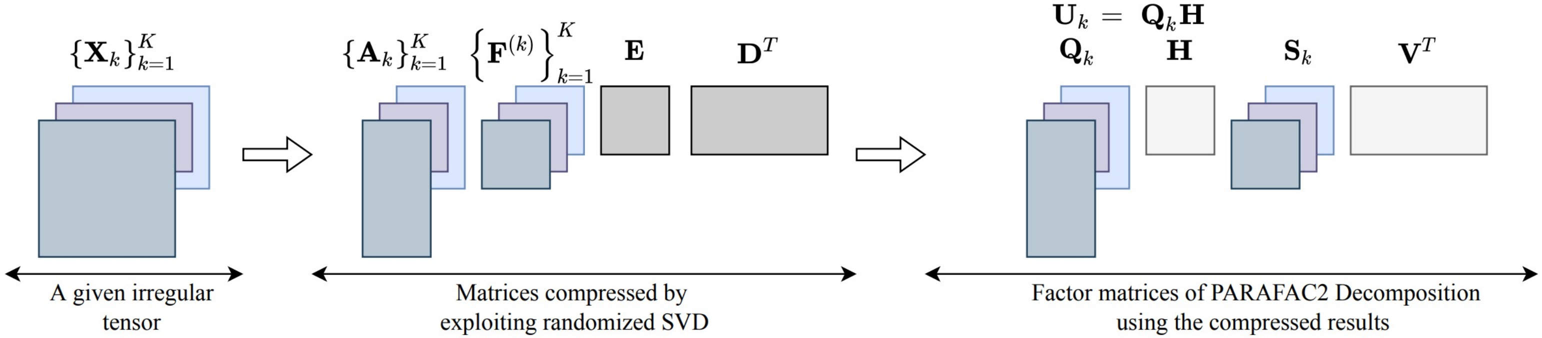
- With DPar2, we efficiently analyze real-world irregular dense tensors

Outline

- Introduction
- Proposed Method
- Experiments
- **Conclusion**

Conclusion

- **(Algorithm)** DPar2 is a **fast** and **scalable** PARAFAC2 decomposition method for irregular dense tensors



- **(Experiment)** DPar2 **outperforms** the previous PARAFAC2 decomposition methods on irregular dense tensors
- **(Discovery)** With DPar2, we find **interesting patterns** in real-world irregular tensors

Thank you !

https://datalab.snu.ac.kr/dpar2

Updating Factor Matrices

Details

Update Procedure of DPar2

Input: $\mathbf{A}_k\mathbf{F}^{(k)}\mathbf{E}\mathbf{D}^T (\approx \mathbf{X}_k)$ for $k = 1, \dots, K$, target rank R

Output: $\mathbf{Q}_k$, $\mathbf{H}$, $\mathbf{S}_k$, $\mathbf{V}$ for $k = 1, \dots, K$

- **Update $\mathbf{Q}_k$** ⬅
- Construct $\boldsymbol{\mathcal{Y}}$
- Update $\mathbf{H}$
- Update $\mathbf{S}_k$
- Update $\mathbf{V}$

- **Update $\mathbf{Q}_k \leftarrow \mathbf{Z}'_k\mathbf{P}'^T_k$ using the compression results**
- **Naïve Computation (High Cost)**
  - Compute $\mathbf{A}_k\mathbf{F}^{(k)}\mathbf{E}\mathbf{D}^T\mathbf{V}\mathbf{S}_k\mathbf{H} \in \mathbb{R}^{I_k \times R}$
  - $\mathbf{Z}'_k\mathbf{\Sigma}'_k\mathbf{P}'^T_k \leftarrow \mathbf{A}_k\mathbf{F}^{(k)}\mathbf{E}\mathbf{D}^T\mathbf{V}\mathbf{S}_k\mathbf{H} \in \mathbb{R}^{I_k \times R}$ by SVD
- **Our computation (Low Cost)**
  - Compute $\mathbf{F}^{(k)}\mathbf{E}\mathbf{D}^T\mathbf{V}\mathbf{S}_k\mathbf{H} \in \mathbb{R}^{R \times R}$
  - $\mathbf{Z}_k\mathbf{\Sigma}_k\mathbf{P}^T_k \leftarrow \mathbf{F}^{(k)}\mathbf{E}\mathbf{D}^T\mathbf{V}\mathbf{S}_k\mathbf{H}$ by SVD
  - $\mathbf{Z}'_k \leftarrow \mathbf{A}_k\mathbf{Z}_k$, $\mathbf{\Sigma}'_k \leftarrow \mathbf{\Sigma}_k$, $\mathbf{P}'_k \leftarrow \mathbf{P}_k$

Since $\mathbf{A}_k \in \mathbb{R}^{I_k \times J}$ is a column orthogonal matrix, we **avoid redundant computations** for $\mathbf{A}_k$ $\Rightarrow$ **reduce computational costs**