KAIST

ICDE 2021 CHANIA CRETE GREECE

IEEE

Continuous CP Decomposition of Sparse Tensor Streams


Taehyung Kwon*


Inkyu Park*


Dongjin Lee


Kijung Shin

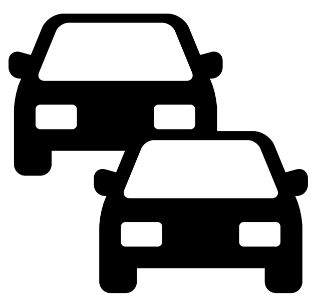
Roadmap

- **Introduction <<**
- Problem Definition
- Proposed Method: **SliceNStitch**
  - Continuous Tensor Model
  - Optimization Algorithms
- Experiment Results
- Conclusions

Tensors are Everywhere

- **Tensors**
  - Multi-dimensional arrays
  - Powerful tools to represent multi-aspect data

- **Tensor Streams**
  - Tensor data are collected incrementally over time



(source, destination, 1)

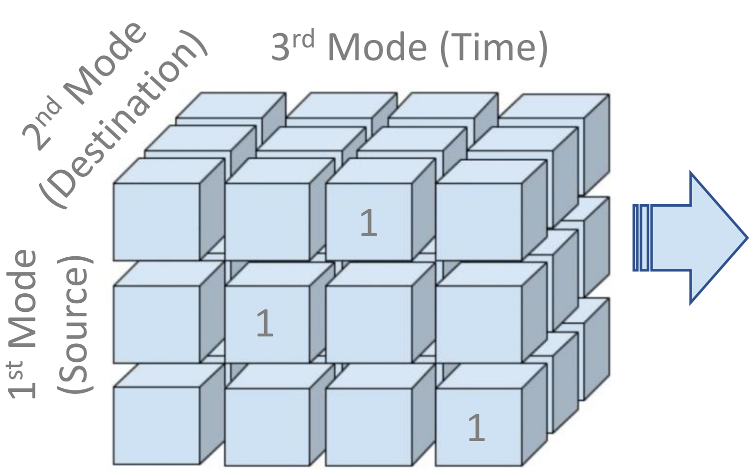
Traffic history data

(user, product, quantity)

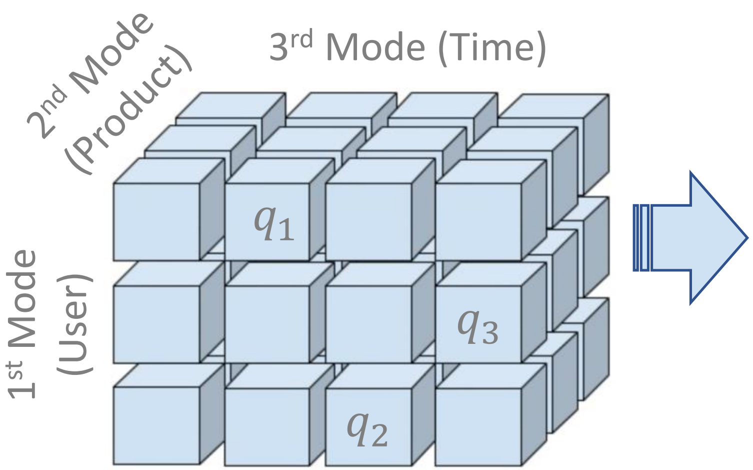
Purchase history data

Tensors are Everywhere

- **Tensors**
  - Multi-dimensional arrays
  - Powerful tools to represent multi-aspect data
- **Tensor Streams**
  - Tensor data are collected incrementally over time



Traffic history data

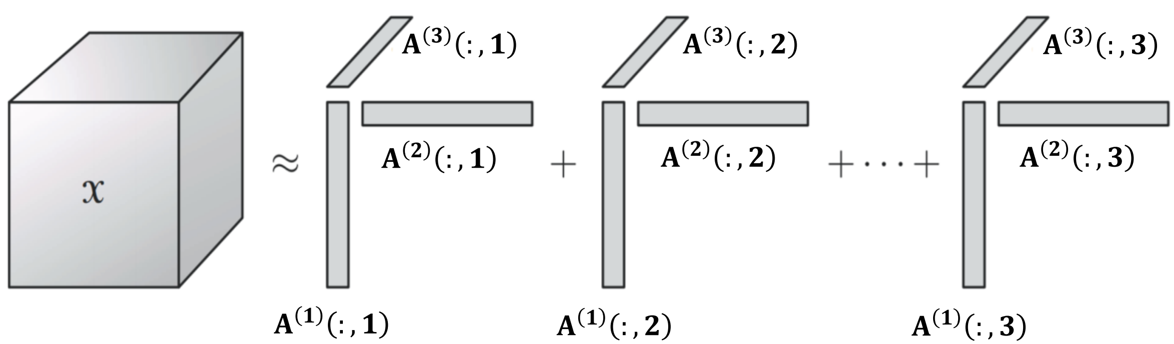


Purchase history data

CP Decomposition

- **CANDECOMP/PARAFAC (CP) decomposition**
  - Low-rank approximation of the tensor

- **Consider:** $\mathcal{X} \in \mathbb{R}^{N_1 \times \cdots \times N_M}$, rank R
- **To Find:** factor matrices $A^{(1)}, \cdots, A^{(M)}$
- **Minimize:** $\|\mathcal{X} - \tilde{\mathcal{X}}\|_F$ where $\tilde{\mathcal{X}} \equiv \sum_{r=1}^{R} A^{(1)}(:,r) \circ \cdots \circ A^{(M)}(:,r)$



CP Decomposition

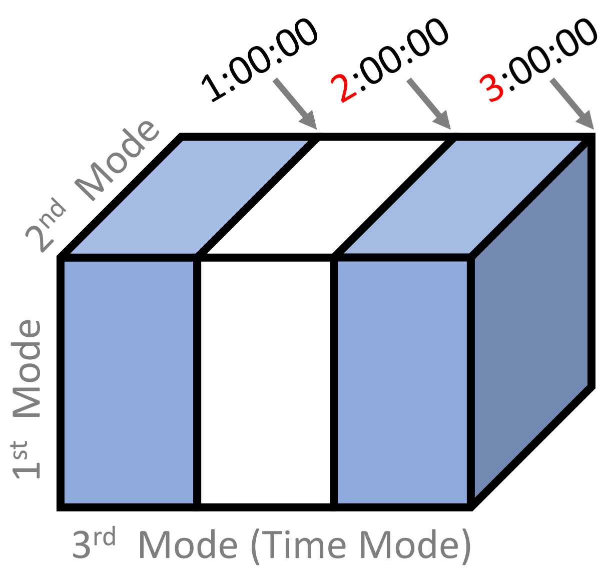
- **Alternating Least Squares (ALS) [2]**
  - Standard algorithm for computing CPD of static tensor

- Input: static tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times \cdots \times N_M}$, rank R
- Initialize factor matrices $\{A^{(m)}\}_{m=1}^{M}$
- While not converge:
  - For $m = 1, \cdots, M$:
    - $A^{(m)} \leftarrow \underset{A^{(m)}}{\operatorname{argmin}} \|\mathcal{X} - \tilde{\mathcal{X}}\|_F$ (least-squares problem)
- Output: factor matrices $\{A^{(m)}\}_{m=1}^{M}$

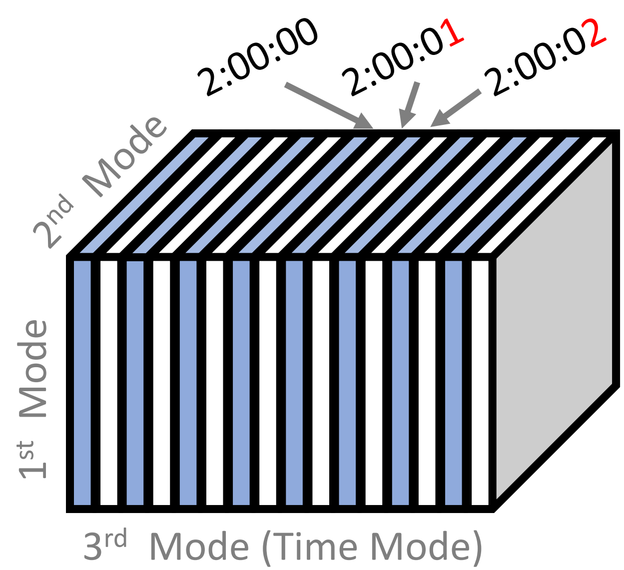
Limitations of Common Tensor Modeling

- Tensor streams grow once per period
  - Outputs of CPD are also updated once per period

- To perform CPD continuously for real-time application,
  - Granularity of the time mode must be extremely fine



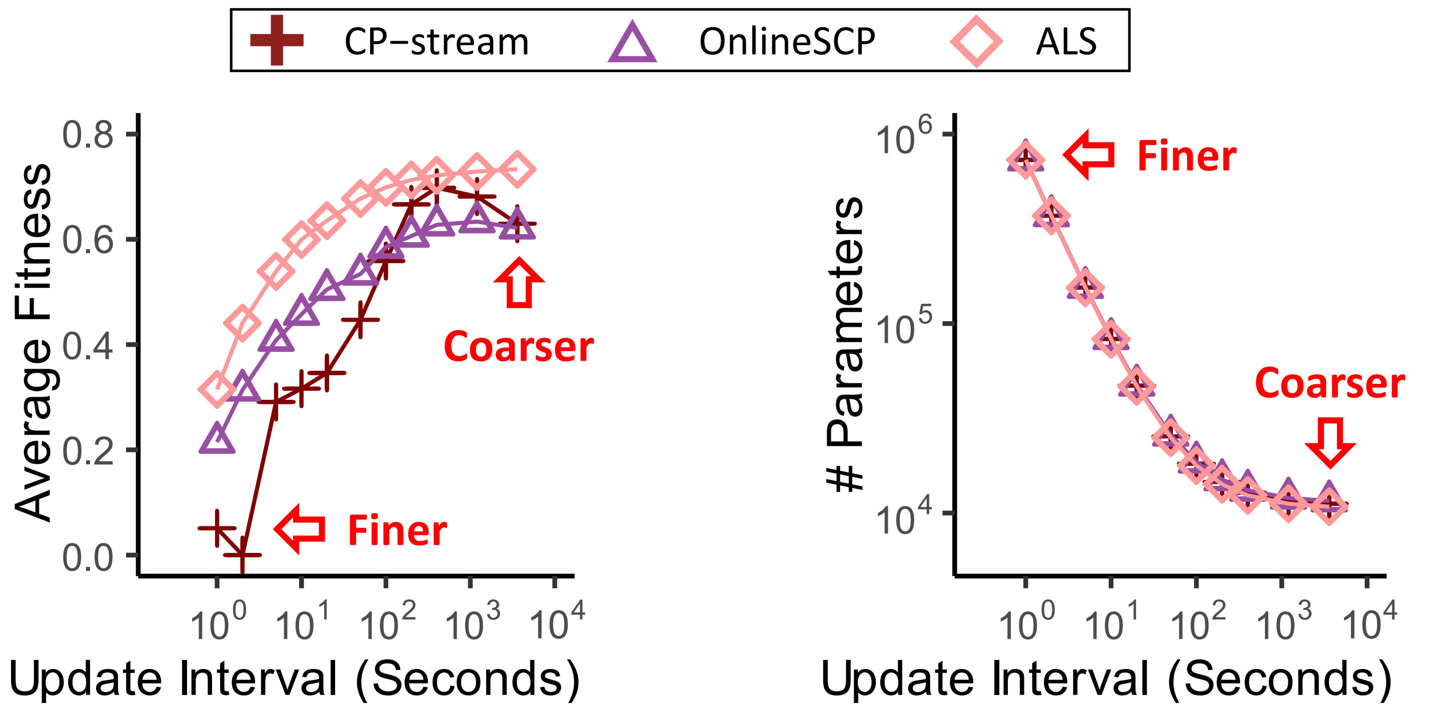
Course-grained Tensor



Fine-grained Tensor

Limitations of Common Tensor Modeling

- Problems of fine-grained tensor modelings
  - Degradation of fitness
  - Increase the number of parameters



In This Work

- We propose **SliceNStitch** for continuous CPD, which is fast, space-efficient, and accurate
  - New data model
  - Fast online algorithms

	Coarse-grained	Fine-grained	SliceNStitch (Proposed)
Update Interval	Long (👎)	Short (👍)	**Short (👍)**
Parameters	Few (👍)	Many (👎)	**Few (👍)**
Fitness	High (👍)	Low (👎)	**High (👍)**

Remark: This work has appeared at ICDE 2021 [1]

Roadmap

- Introduction
- **Problem Definition <<**
- Proposed Method: **SliceNStitch**
    - Continuous Tensor Model
    - Optimization Algorithms
- Experiment Results
- Conclusions

Problem Definition

- **Continuous CP decomposition**

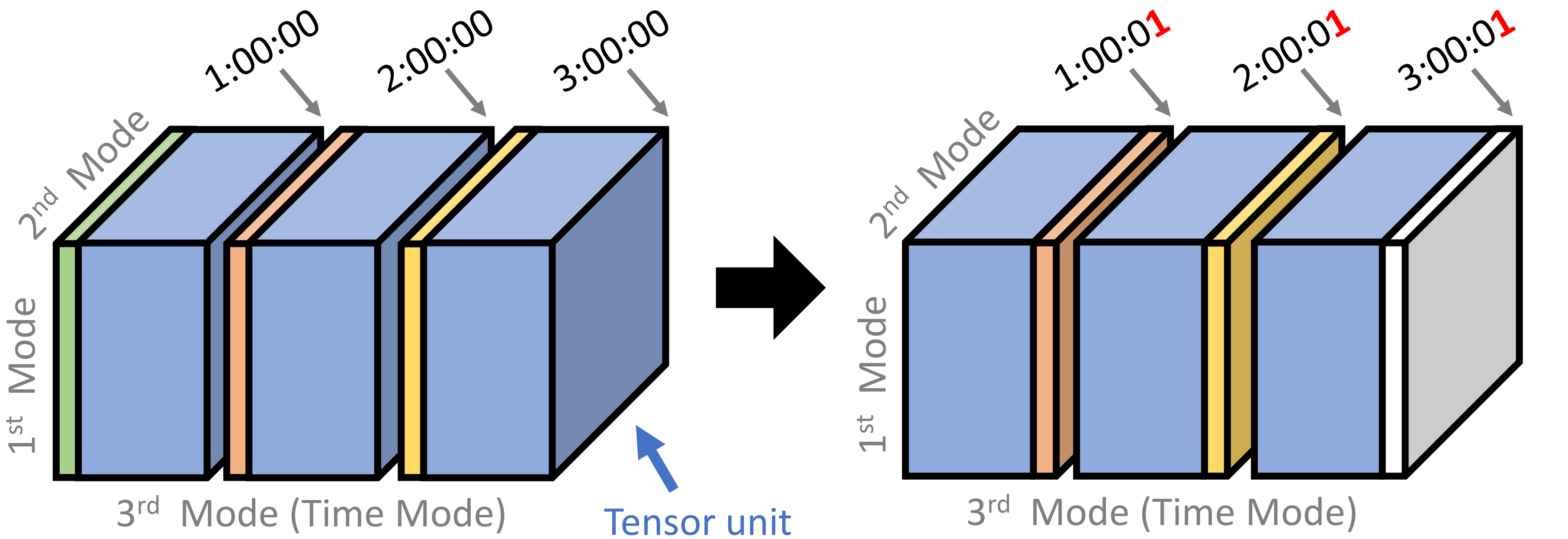
  - **Given:** a multi-aspect data stream
  - **Update:** CP decomposition instantly in response to each new event in the stream
  - **Without:** waiting for the current period to end

Roadmap

- Introduction
- Problem Definition
- Proposed Method: **SliceNStitch**
  - **Continuous Tensor Model <<**
  - Optimization Algorithms
- Experiment Results
- Conclusions

Continuous Tensor Model

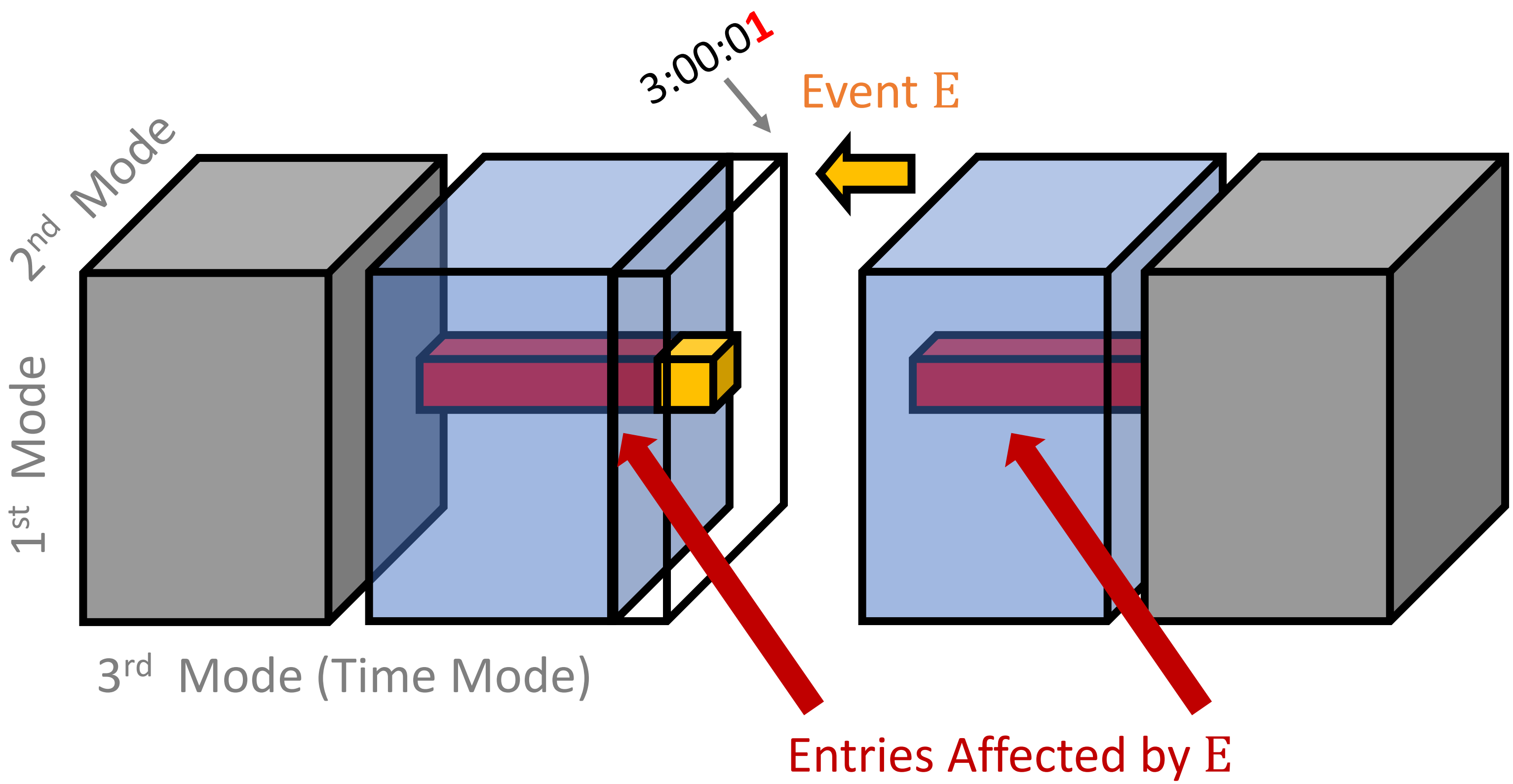
- Tensor window and units evolve at each time



Tensor window at time 3:00:00 **Tensor window at time 3:00:01**

Event-driven Implementation

- Each single data causes an event:
  - Move the quantity to the next tensor unit ($\mathcal{X} \rightarrow \mathcal{X} + \Delta\mathcal{X}$)



Roadmap

- Introduction
- Problem Definition
- Proposed Method: **SliceNStitch**
  - Continuous Tensor Model
  - **Optimization Algorithms <<**
- Experiment Results
- Conclusions

SliceNStitch-Matrix ($\mathbf{SNS_{MAT}}$)

- Factor matrices of previous window are good initial points
- Single iteration of ALS is enough to achieve high accuracy

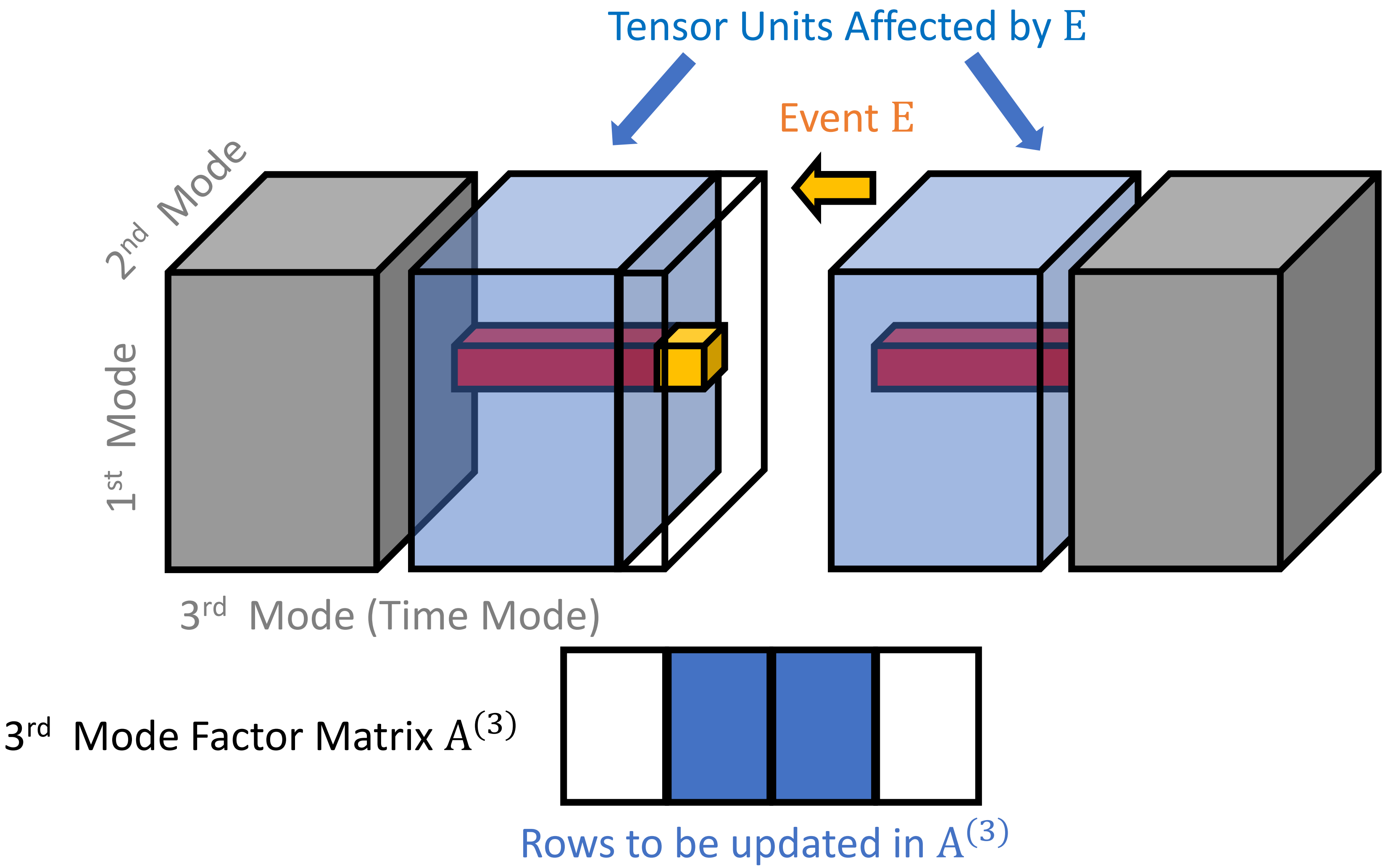
> - Input: (1) tensor $\mathcal{X} + \Delta\mathcal{X} \in \mathbb{R}^{N_1 \times \cdots \times N_M}$
>
>   (2) factor matrices $\left\{A^{(m)}\right\}_{m=1}^{M}$ of previous window $\mathcal{X}$
> - For $m = 1, \cdots, M$:
>   - $A^{(m)} \leftarrow \underset{A^{(m)}}{\operatorname{argmin}} \left\| (\mathcal{X} + \Delta\mathcal{X}) - \tilde{\mathcal{X}} \right\|_F$ (least-squares problem)
> - Output: updated factor matrices $\left\{A^{(m)}\right\}_{m=1}^{M}$

SliceNStitch-Matrix ($\mathbf{SNS_{MAT}}$)

- **Pros:**
  - High-quality solution, since it uses all non-zero entries
- **Cons:**
  - High computational cost, since it uses all non-zero entries **<<**
- **Solution:** Update <u>only the rows of factor matrices</u> which are used for approximating changed entries

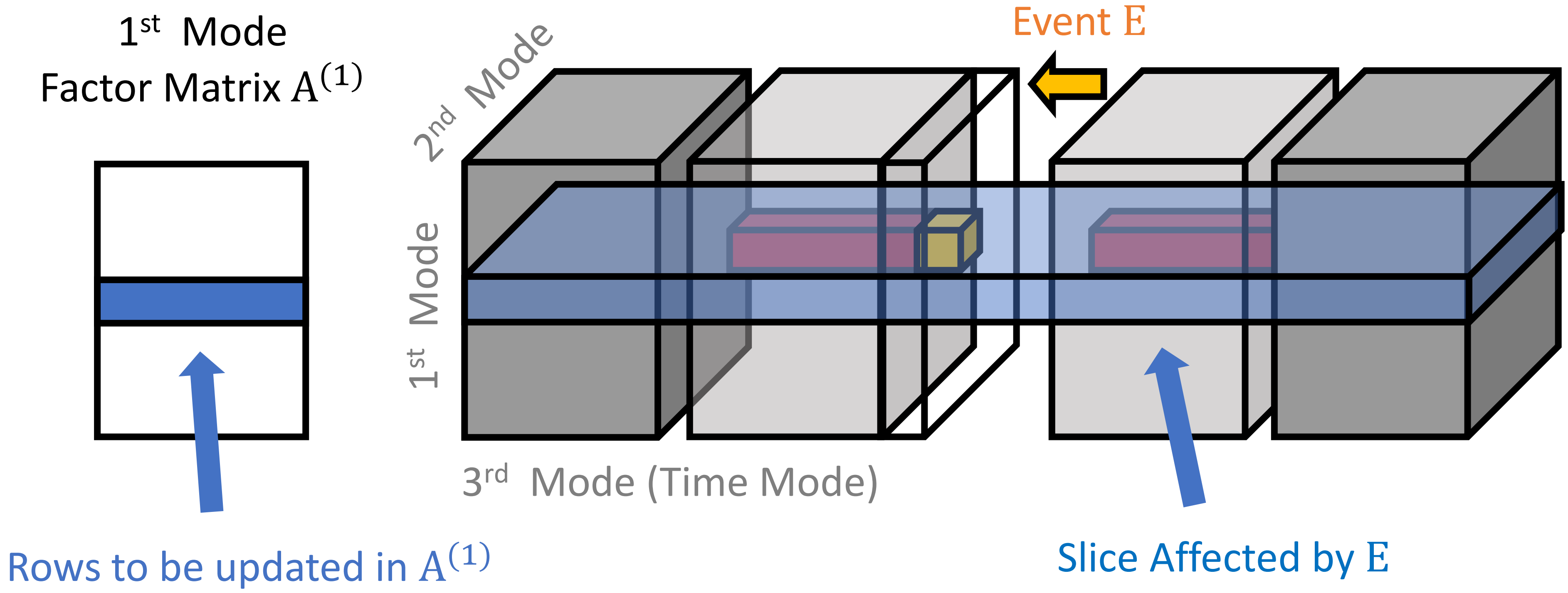
Outline of Improved Algorithms

- Update time mode factor matrix:



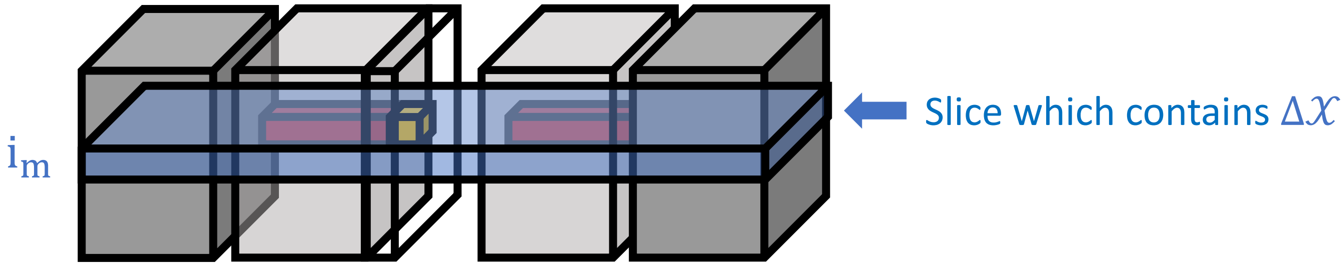
Outline of Improved Algorithms

- Update non-time mode factor matrices:



SliceNStitch-Vector ($\mathbf{SNS_{VEC}}$)

- **Time mode:**
  - Approximate $\mathcal{X}$ as $\tilde{\mathcal{X}}$ in the least squares solution
  - Computational complexity ∝ NNZ($\Delta\mathcal{X}$)
- **Non-time mode:**
  - Approximate the least squares problem
  - $\min_{\mathrm{A}^{(\mathrm{m})}} \left\| (\mathcal{X} + \Delta\mathcal{X}) - \tilde{\mathcal{X}} \right\|_{\mathrm{F}} \rightarrow \min_{\mathrm{A}^{(\mathrm{m})}(:,\mathrm{i_m})} \left\| (\mathcal{X} + \Delta\mathcal{X}) - \tilde{\mathcal{X}} \right\|_{\mathrm{F}}$
  - Computational complexity ∝ NNZ(Slice which contains $\Delta\mathcal{X}$)



SliceNStitch-Vector ($\mathbf{SNS_{VEC}}$)

- **Pros:**
  - Significantly faster than $\mathrm{SNS_{MAT}}$
- **Cons:**
  - Numerically unstable ($\because$ no normalization)
  - Slow if many non-zeros are in the slice **<<**
- **Solution:** Use random sampling with smaller sample size if too many non-zeros are in the slice

SliceNStitch-Random ($\mathbf{SNS_{RND}}$)

- **If NNZ(Slice contains $\boldsymbol{\Delta\mathcal{X}}$) $\leq$ $\boldsymbol{\theta}$:**
  - Use $\mathrm{SNS_{VEC}}$

- **Otherwise:**
  - Randomly select θ indices (= S) from the slice contains $\Delta\mathcal{X}$
  - Define $\bar{\mathcal{X}}_S$ s.t. $\tilde{\mathcal{X}}(\mathrm{J}) + \bar{\mathcal{X}}_\mathrm{S}(\mathrm{J}) = \begin{cases} \mathcal{X}(\mathrm{J}), & \text{if } \mathrm{J} \in \mathrm{S} \\ \tilde{\mathcal{X}}(\mathrm{J}), & \text{otherwise} \end{cases}$: Original tensor : Approximation
  - Approximate $\mathcal{X}$ as $\tilde{\mathcal{X}} + \bar{\mathcal{X}}_S$ in the least squares solution of $\mathrm{SNS_{VEC}}$
  - Computational complexity ∝ θ

SliceNStitch-Random ($\mathbf{SNS_{RND}}$)

- **Pros:**
  - Computational complexity is constant

- **Cons:**
  - Numerically unstable ($\because$ no normalization) **<<**

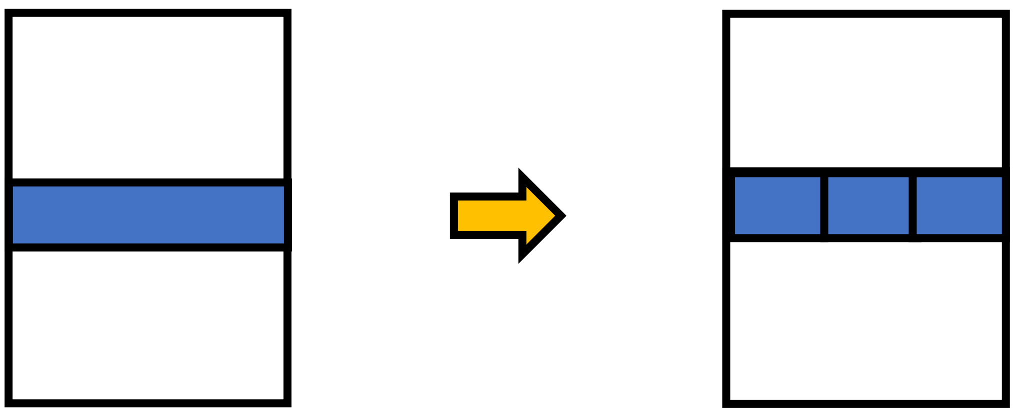
- **Solution:** Use <u>clipping</u> to prevent the extreme value which yields numerical instability

SliceNStitch-Stable ($\mathbf{SNS}^{+}_{\mathbf{VEC}}$, $\mathbf{SNS}^{+}_{\mathbf{RND}}$)

- **Update entries one by one**
  - Clip each value if the absolute value is larger than η

$\mathrm{SNS}_{\mathrm{VEC}}$ and $\mathrm{SNS}_{\mathrm{RND}}$:
Update the row **at once**

$\mathrm{SNS}^{+}_{\mathrm{VEC}}$ and $\mathrm{SNS}^{+}_{\mathrm{RND}}$:
Update row entries **one by one**

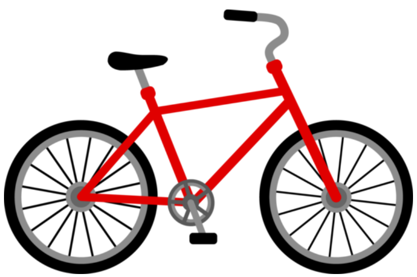


Roadmap

- Introduction
- Problem Definition
- Proposed Method: **SliceNStitch**
  - Continuous Tensor Model
  - Optimization Algorithms
- **Experiment Results <<**
- Conclusions

Experimental Settings

- 4 real-world sparse time series datasets



Divvy Bikes

Chicago Crimes

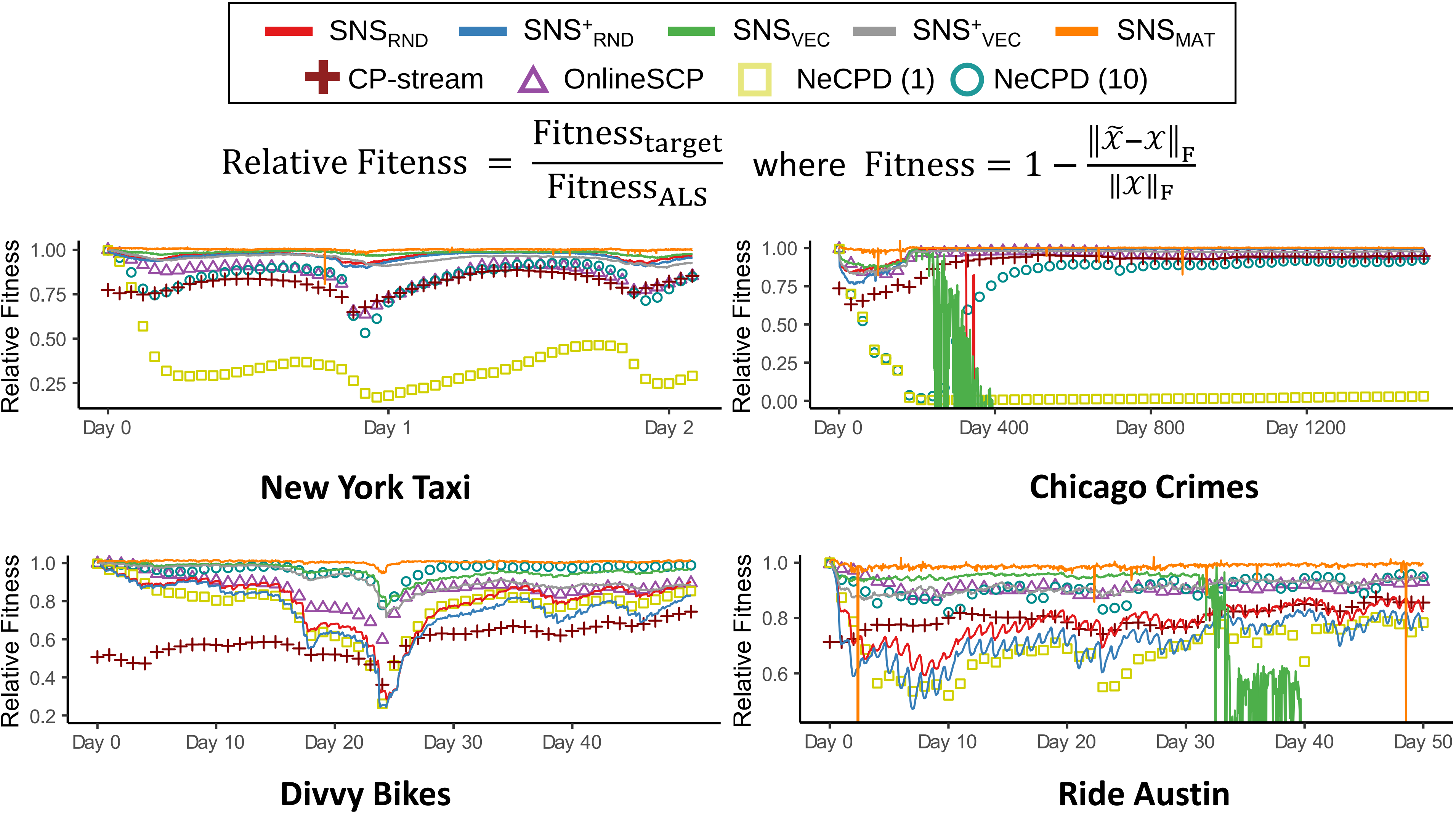
New York Taxi

Ride Austin

- 4 baselines that update CPD periodically
  - **ALS** [2], **onlineSCP** [3], **CP-stream** [4], **NeCPD** [5]

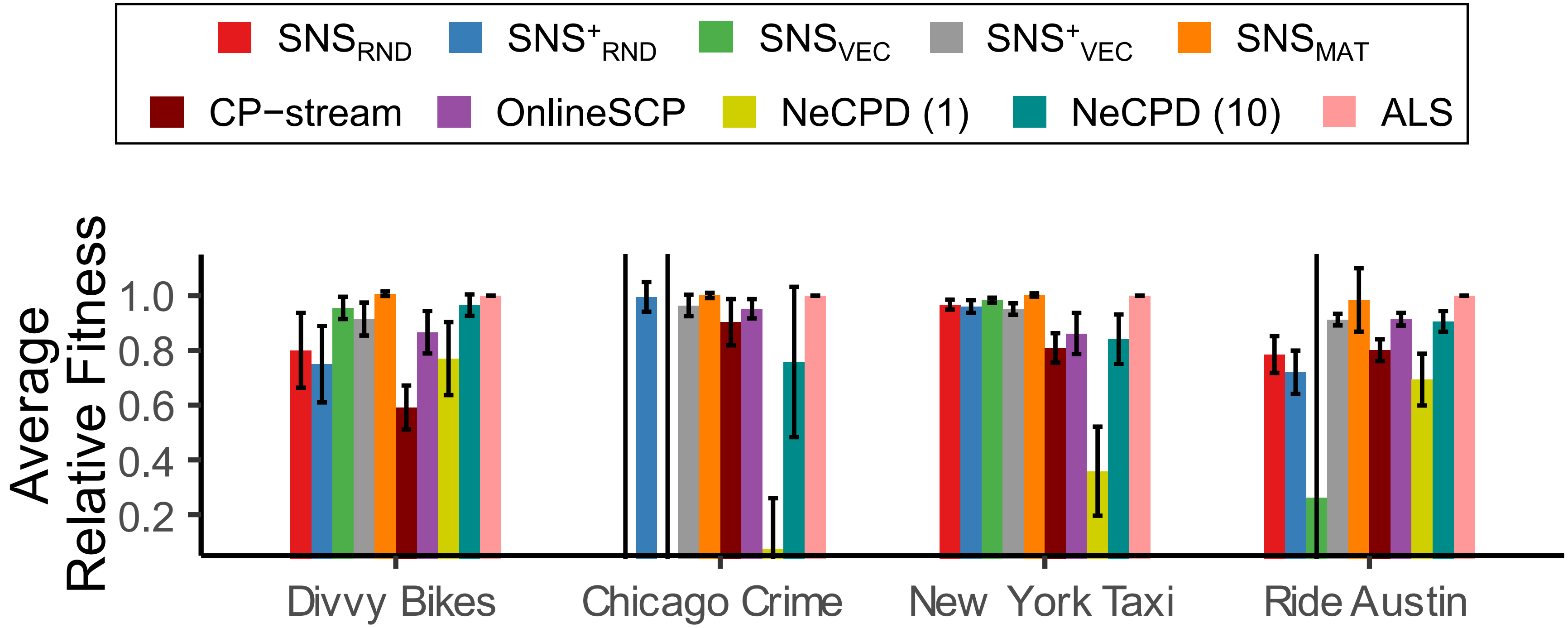
Exp 1. SliceNStitch is Accurate

$$\text{Relative Fitenss} = \frac{\text{Fitness}_{\text{target}}}{\text{Fitness}_{\text{ALS}}} \quad \text{where} \;\; \text{Fitness} = 1 - \frac{\|\tilde{\mathcal{X}} - \mathcal{X}\|_{\mathrm{F}}}{\|\mathcal{X}\|_{\mathrm{F}}}$$



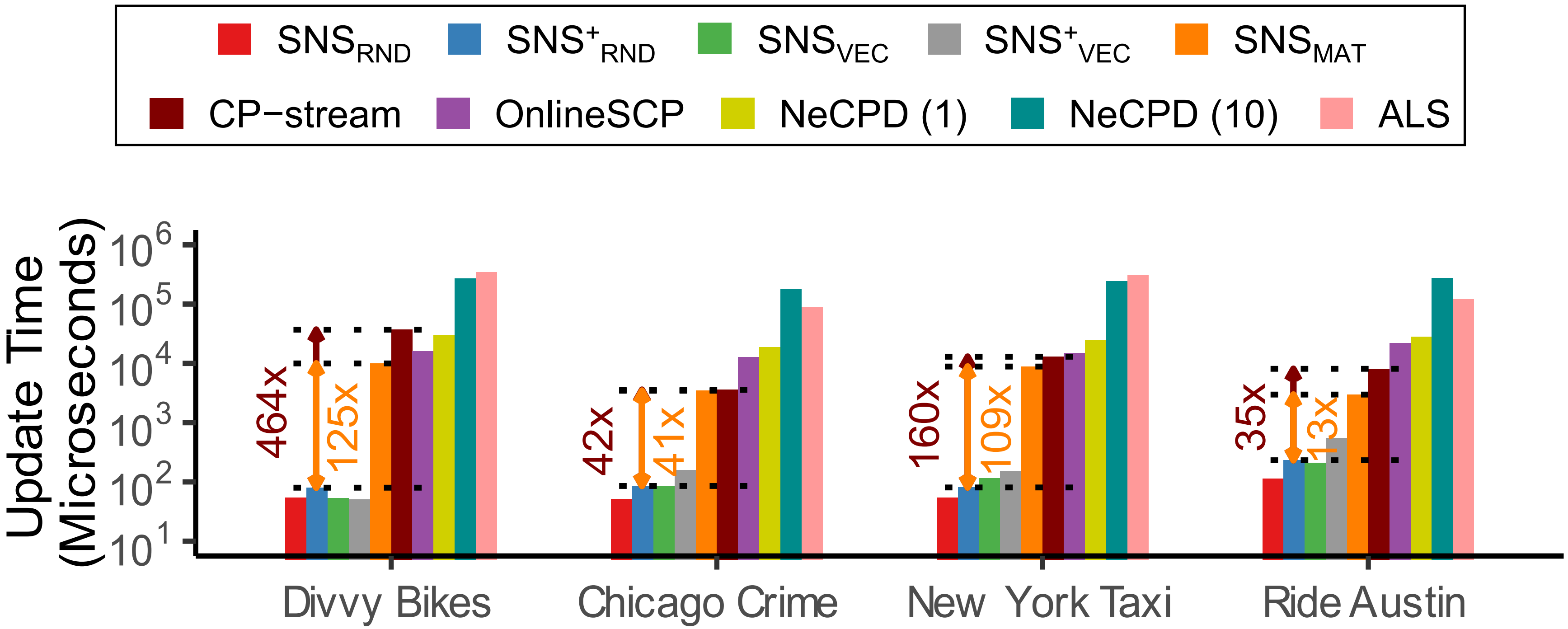
Exp 1. SliceNStitch is Accurate

- SliceNStitch achieve 72 - 100% relative fitness compared to the most accurate baseline



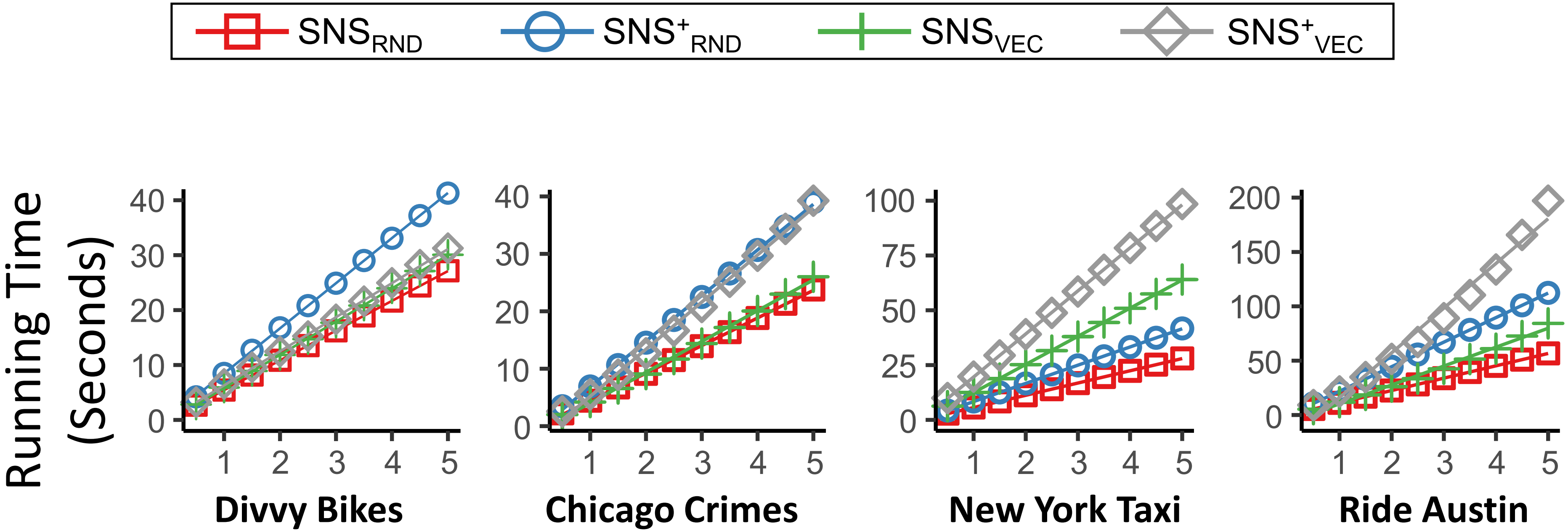
Exp 2. SliceNStitch is Fast

- $\text{SNS}^{+}_{\text{RND}}$ is up to 464 × faster than CP-stream



Exp 3. SliceNStitch is Scalable

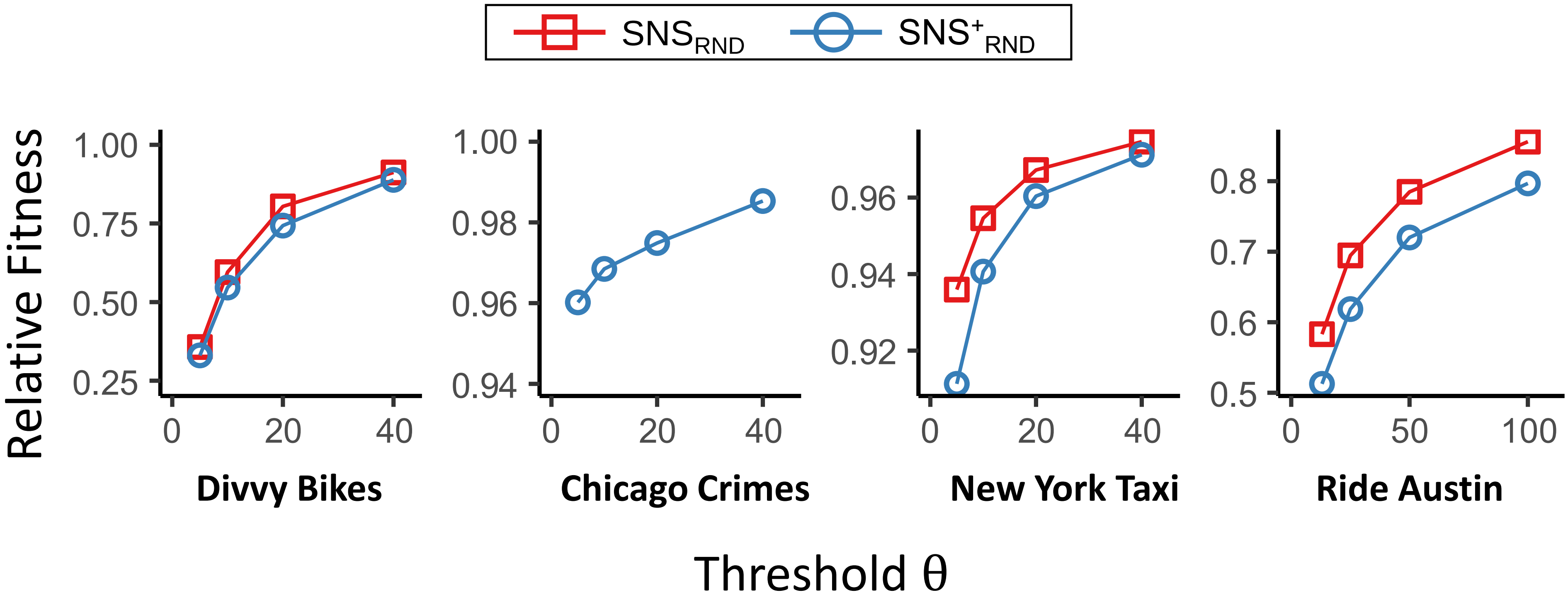
- Total runtime of SliceNStitch is linear in the number of events



The number of events ($\times 10^5$)

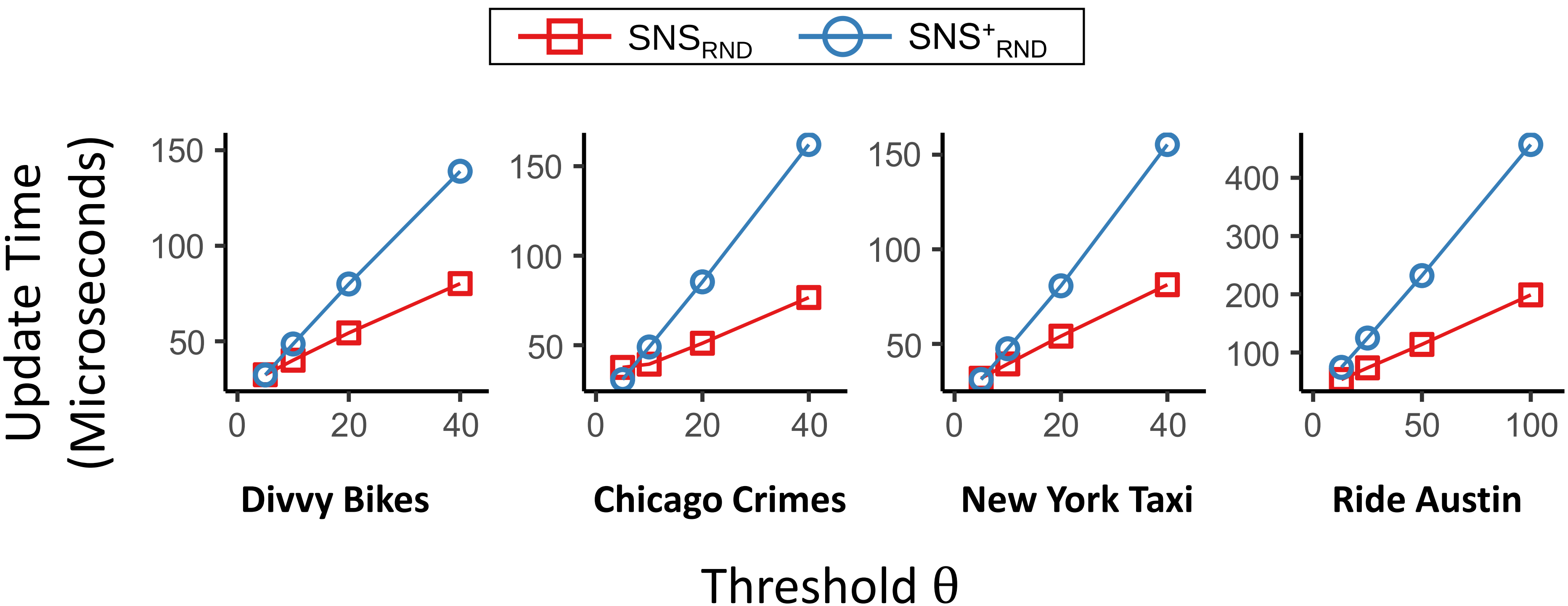
Exp 4. Effect of Sampling Threshold θ

- As θ increases,
  - Fitness increases with diminishing returns
  - Runtime grows linearly



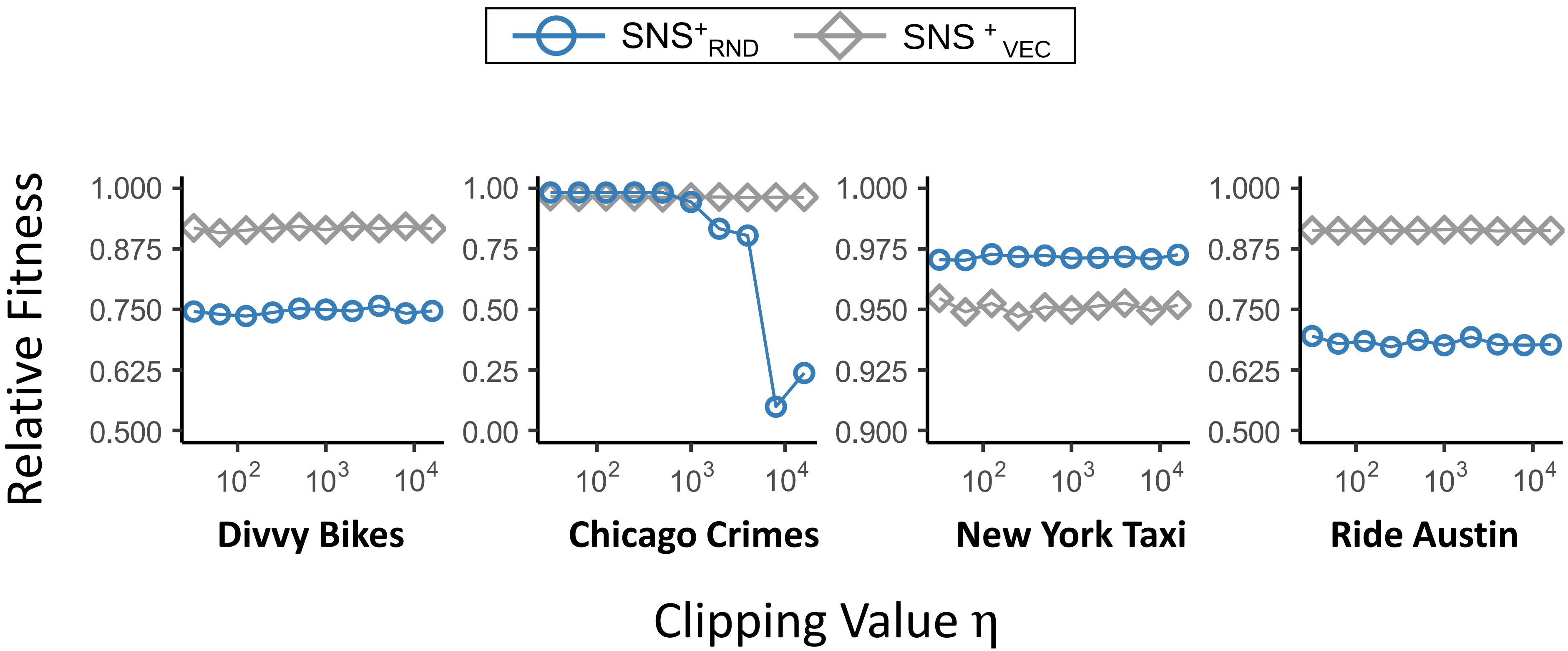
Exp 4. Effect of Sampling Threshold θ

- As θ increases,
  - Fitness increases with diminishing returns
  - Runtime grows linearly



Exp 5. Effect of Clipping Value η

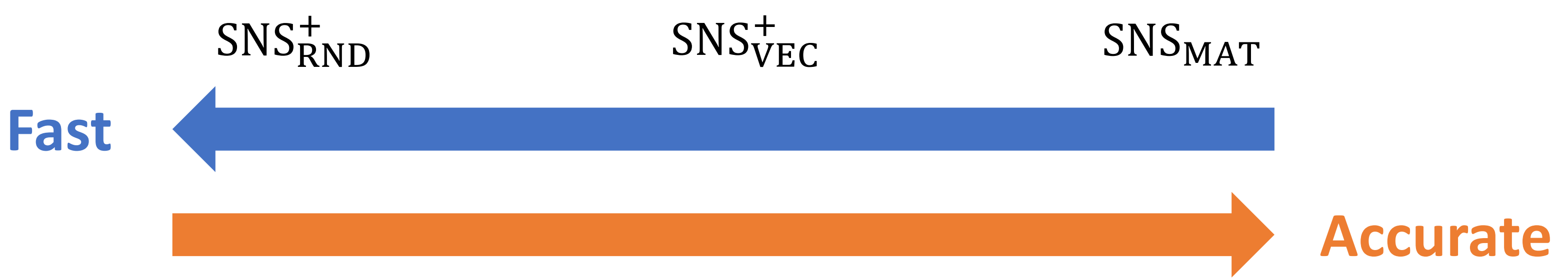
- Fitness is insensitive to η as long as η is small enough



Practitioner's Guide

- We do not recommend $\mathrm{SNS}_{\mathrm{VEC}}$ and $\mathrm{SNS}_{\mathrm{RND}}$ due to instability issues

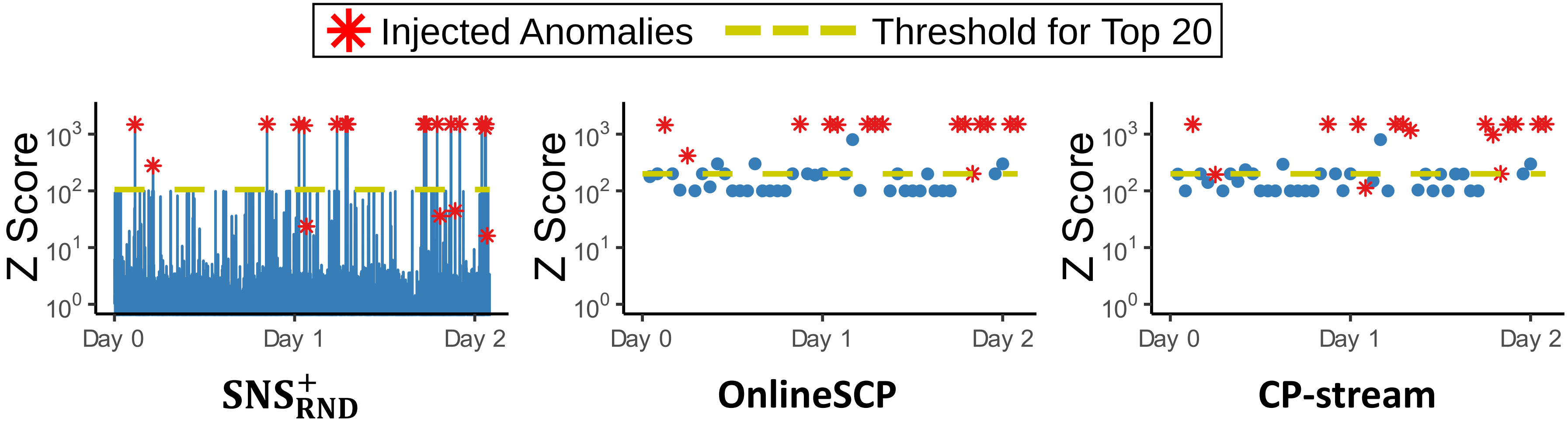
- We recommend using the most accurate version within your runtime budget



- If $\mathrm{SNS}^{+}_{\mathrm{RND}}$ is chosen, increase the sampling threshold θ enough within your runtime budget

Application: Anomaly Detection

- SliceNStitch detects anomalies much faster with comparable accuracy



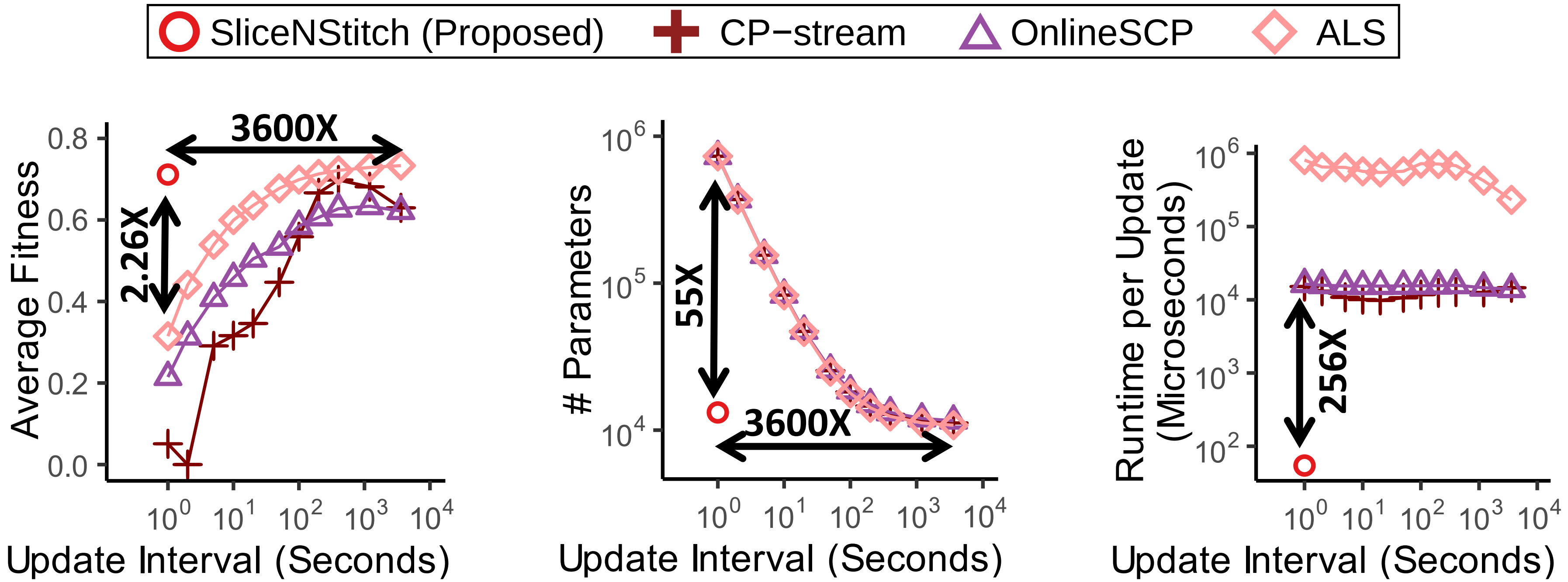
	Precision @ Top 20	Time Gap between Occurrence and Detection
$\text{SNS}^{+}_{\text{RND}}$	**0.80**	**0.0015 sec**
OnlineSCP	**0.80**	1601.00 sec
CP-stream	0.70	1424.57 sec

Roadmap

- Introduction
- Problem Definition
- Proposed Method: **SliceNStitch**
  - Continuous Tensor Model
  - Optimization Algorithms
- Experiment Results
- **Conclusions <<**

Conclusion

- We propose **SliceNStitch** for continuous CPD with
  - **Near-instant updates**
  - **High fitness**
  - **Small number of parameters**



References

[1] Taehyung Kwon*, **Inkyu Park***, Dongjin Lee, and Kijung Shin, "SliceNStitch: Continuous CP Decomposition of Sparse Tensor Streams," in ICDE 2021.

[2] R. A. Harshman, "Parafac2: Mathematical and Technical Notes," UCLA Working Papers in Phonetics, vol. 22, 30-44, 1972.

[3] Shuo Zhou, Sarah M. Erfani, and James Bailey, "Online CP Decomposition for Sparse Tensors," in ICDM, 2018.

[4] Shaden Smith, Kejun Huang, Nicholas D. Sidiropoulos, and George Karypis, "Streaming Tensor Factorization for Infinite Data Sources," in SDM, 2018.

[5] Ali Anaissi, Basem Suleiman, and Seid Miad Zandavi, "NeCPD: An Online Tensor Decomposition with Optimal Stochastic Gradient Descent," arXiv preprint arXiv:2003.08844, 2020.