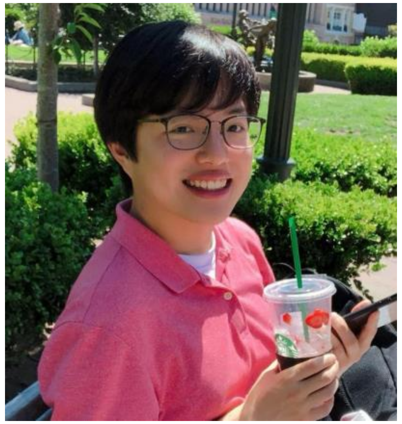
SliceNStitch: Continuous CP Decomposition of Sparse Tensor Streams


Taehyung Kwon*


Inkyu Park*

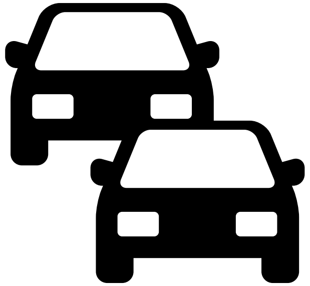

Dongjin Lee


Kijung Shin

Tensors are Everywhere

- Tensors are powerful tools for representing time-evolving multi-aspect data

- Multi-aspect data stream:
  a sequence of timestamped M-tuples $\{(e_n = (i_1, \cdots, i_{M-1}, v_n), t_n)\}_{n \in \mathbb{R}}$
  - $i_1, \cdots, i_{M-1}$: non-time mode coordinates
  - v_n: value of the event
  - t_n: time when e_n occurs



(source, destination, 1)

Traffic history

(location, type, 1)

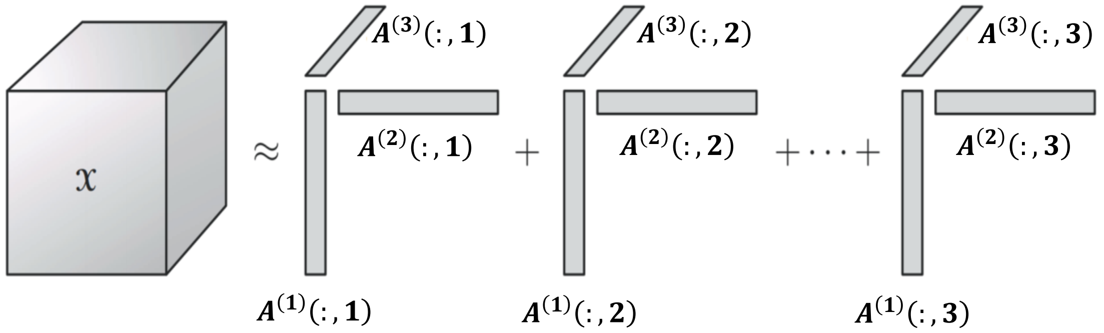
Crime history

(user, product, color, quantity)

Purchase history

CANDECOMP/PARAFAC Decomposition (CPD)

- CPD gives a low-rank approximation $\widetilde{\mathcal{X}}$ of the tensor $\mathcal{X}$
- Given an M-mode tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times \cdots \times N_M}$ and rank $R \in \mathbb{N}$,
  - $\mathcal{X} \approx \widetilde{\mathcal{X}} \equiv \sum_{r=1}^{R} A^{(1)}(:,r) \circ \cdots \circ A^{(M)}(:,r)$
  - where $A^{(1)}, \cdots, A^{(M)}$: Factor matrices



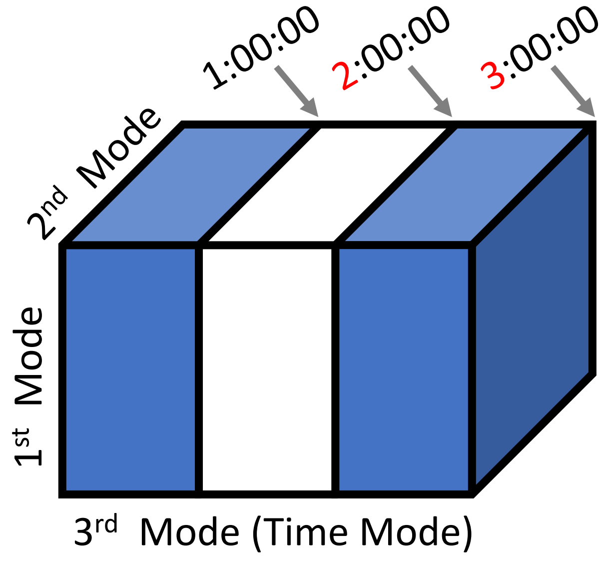
- Goal of CPD: factor matrices that minimize the difference between $\mathcal{X}$ and $\widetilde{\mathcal{X}}$

$$\min_{A^{(1)},\cdots,A^{(M)}} \|\mathcal{X} - \widetilde{\mathcal{X}}\|_F$$

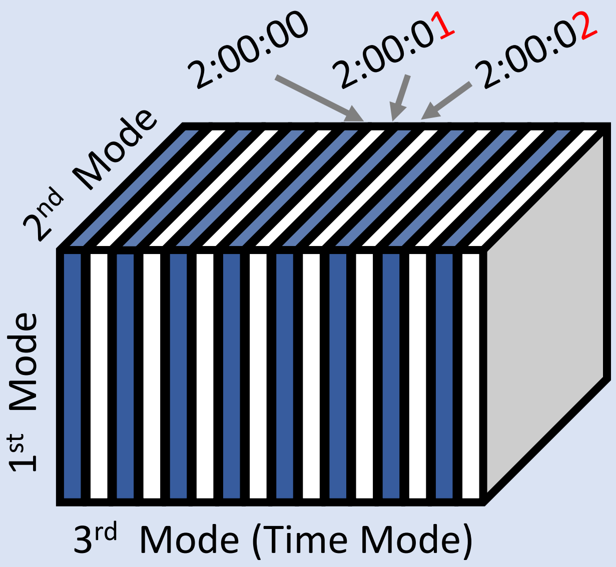
Limitation of Common Tensor Modeling

- Dynamic tensors grow once per period
  → Outputs of CPD are also updated once per period

- To perform CPD continuously for real-time application,
  - One can make the granularity of the time mode extremely fine



Course-grained Tensor



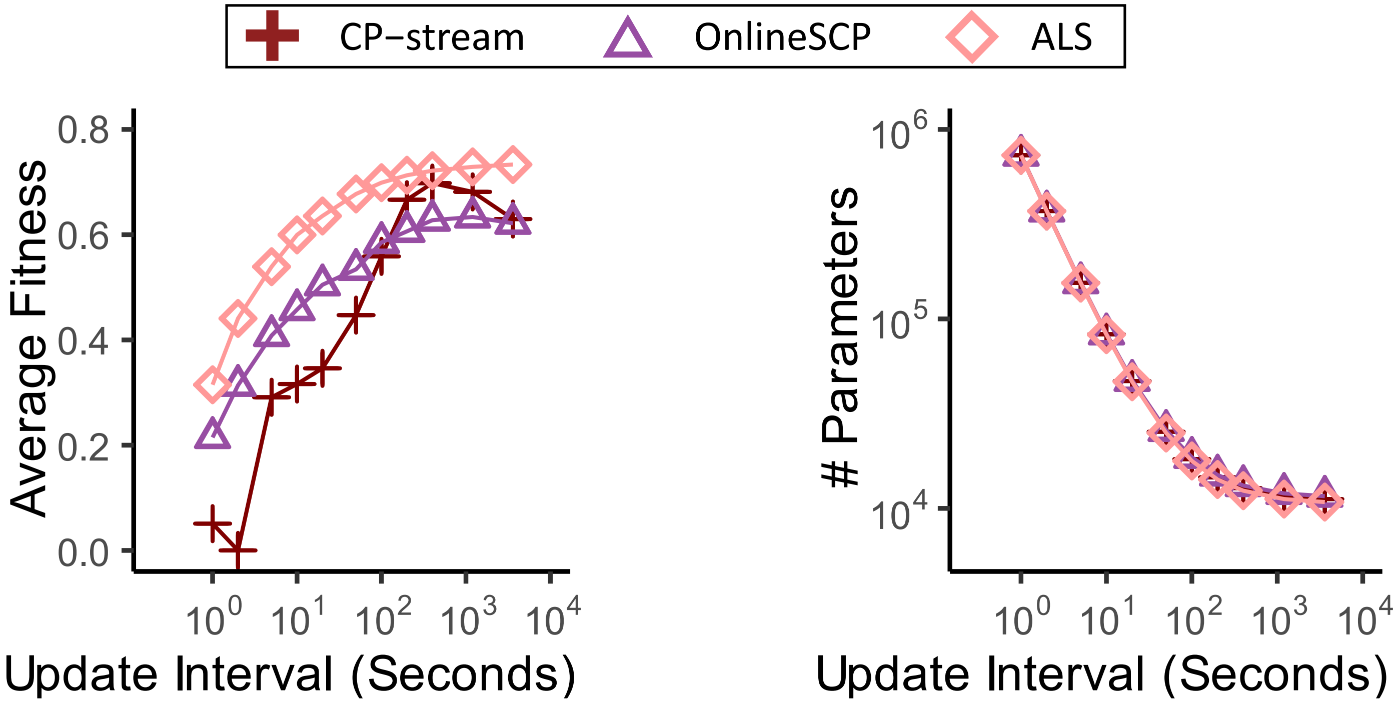
Fine-grained Tensor

Limitation of Common Tensor Modeling

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

∴ **Not suitable for real-time application!**

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



Our Problem: Continuous CP Decomposition

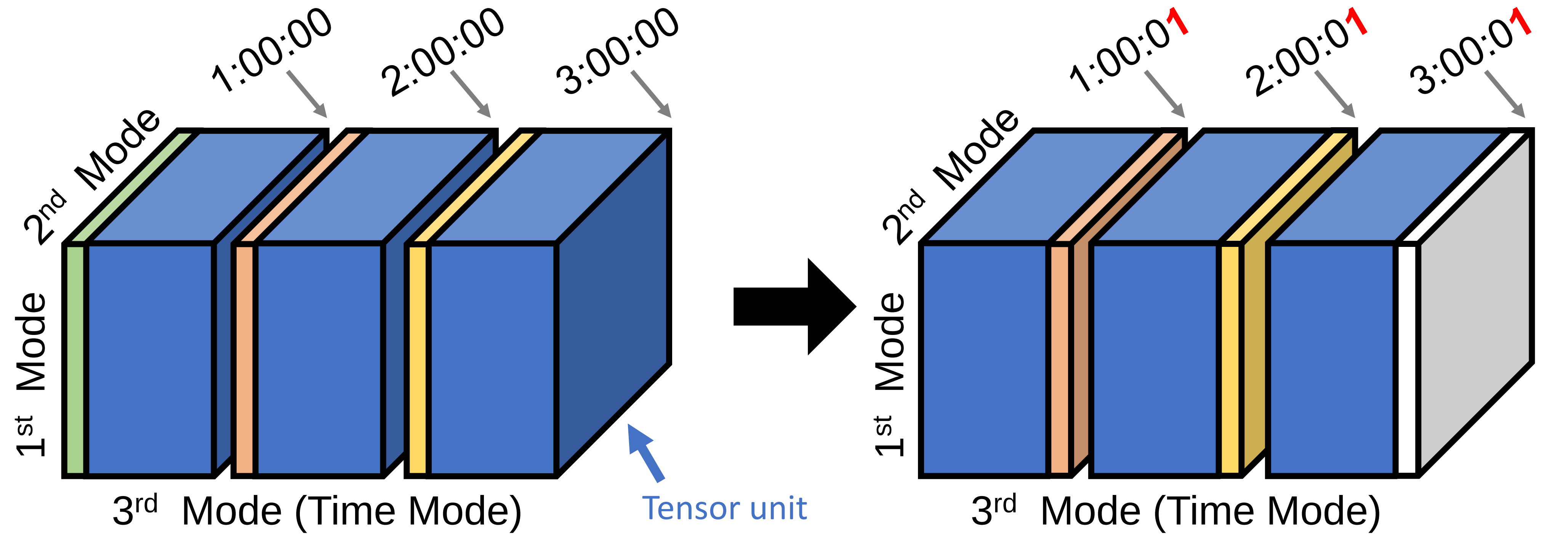
- How can we continuously analyze multi-aspect data streams using CPD?
  - Given a multi-aspect data stream
  - Update its CP decomposition instantly in response to each new tuple in the stream
  - Without having to wait for the current period to end

Road Map

- Introduction
- Problem Definition
- **Data Model: Continuous Tensor Model**
- Optimization Algorithms: SliceNStitch
- Experiment Results
- Conclusions

Proposed Tensor Model

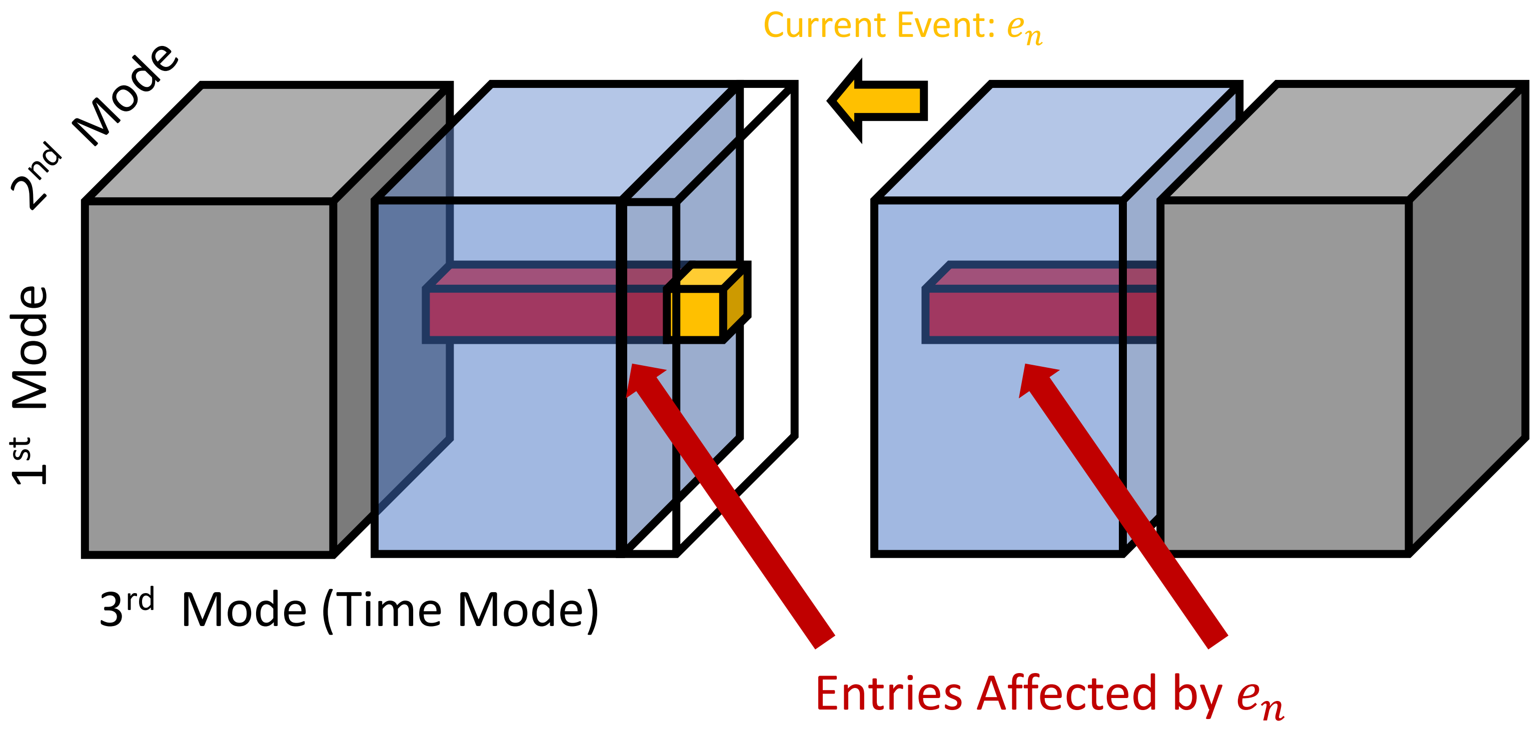
- Continuous Tensor Model
  - The modeled tensor window and units evolve at each time



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

Event-driven Implementation

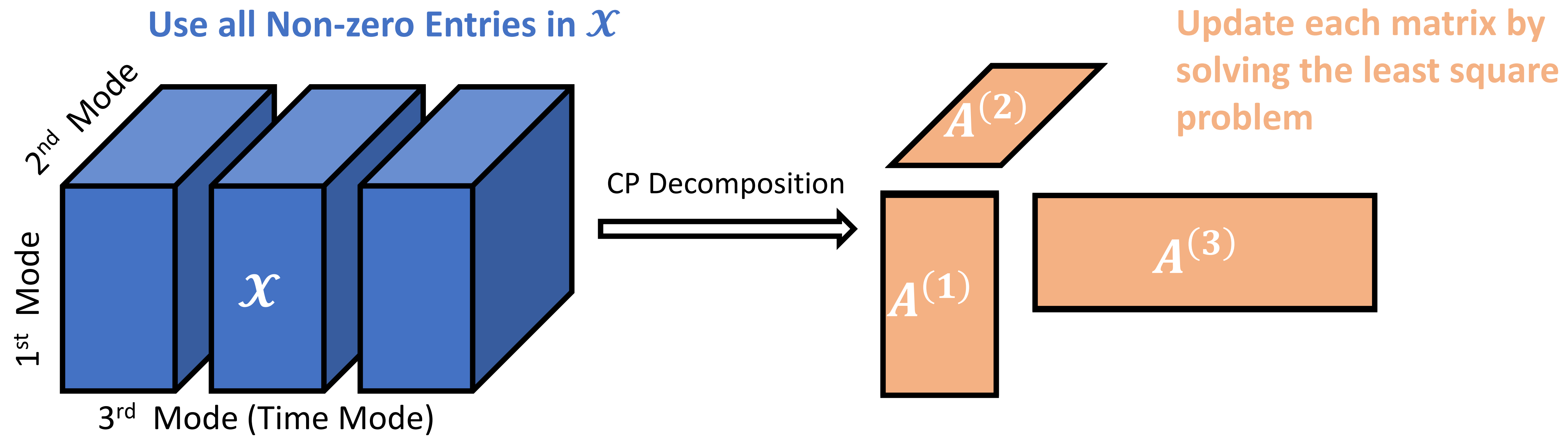
- For the tensor window $\mathcal{X}$,
- a tuple $(e_n = (i_1, \cdots, i_{M-1}, v), t_n)$ causes an event:



Road Map

- Introduction
- Problem Definition
- Data Model: Continuous Tensor Model
- **Optimization Algorithms: SliceNStitch**
- Experiment Results
- Conclusions

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



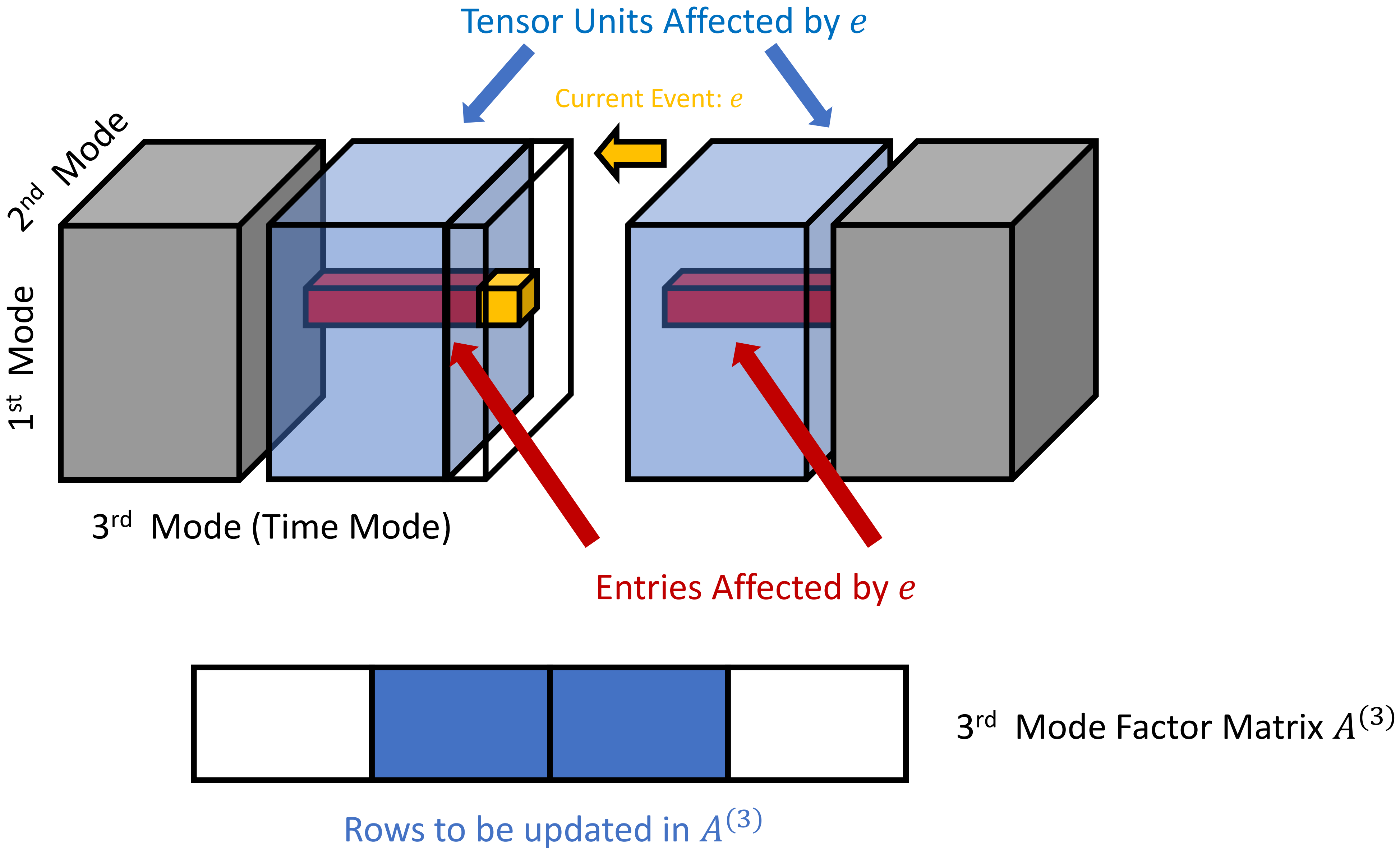
Pros

High-quality solution 😊

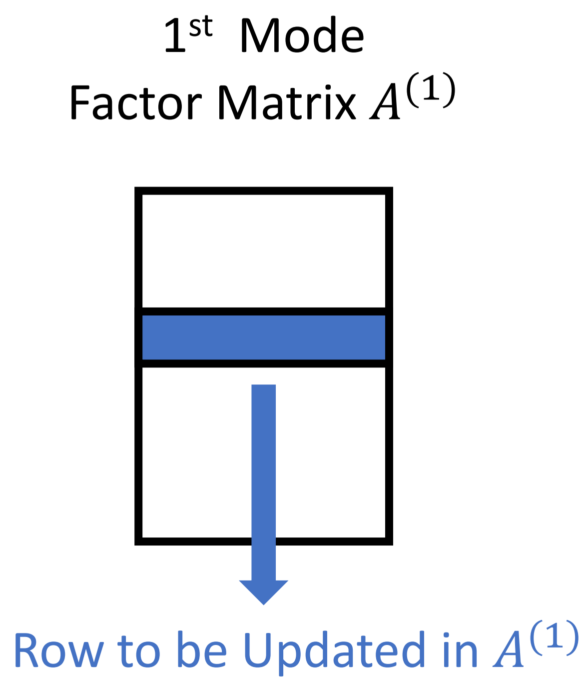
Cons

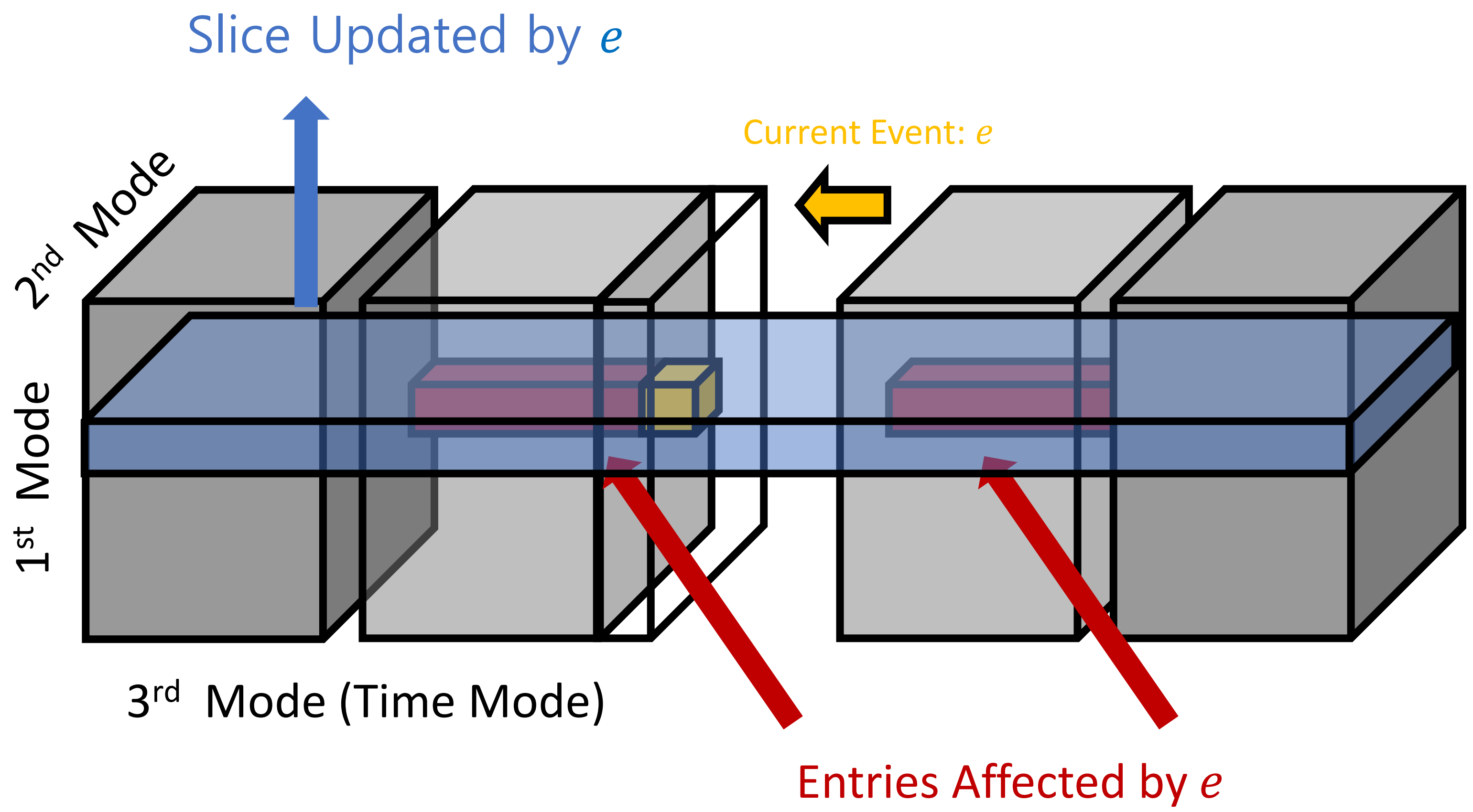
High computational cost 😢

Common Outline of the Other Algorithms: Time Mode



Common Outline of the Other Algorithms: Non-Time Mode



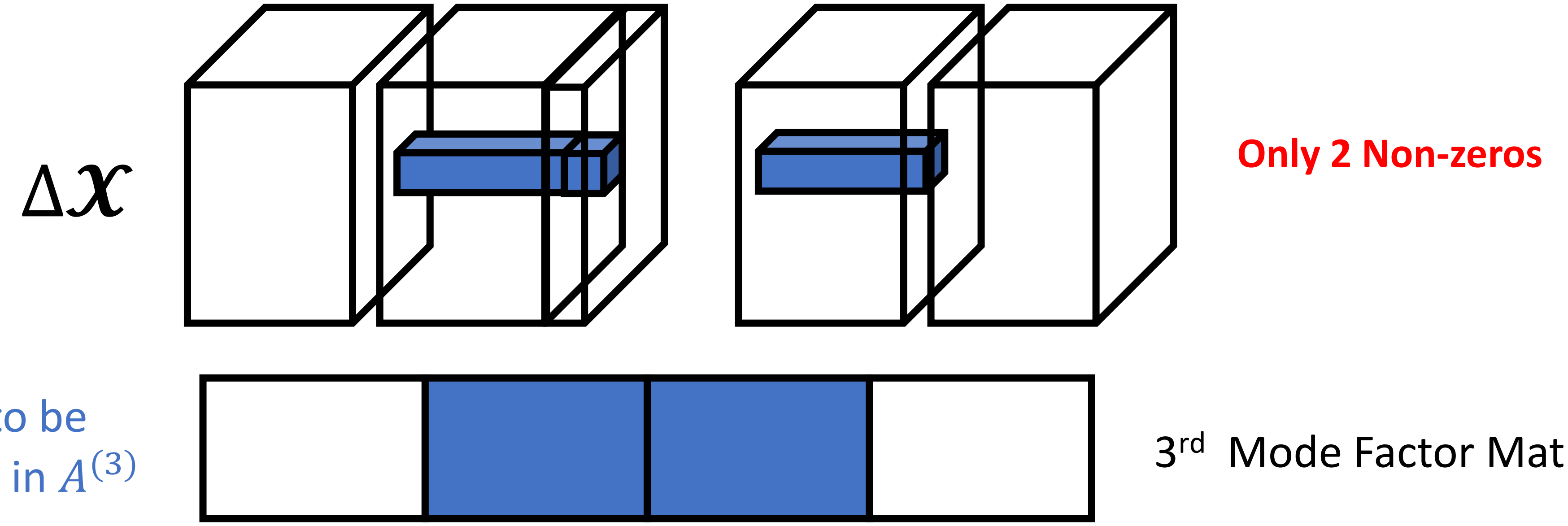


SliceNStitch-Vector: $\mathrm{SNS}_{\mathrm{VEC}}$

For the time mode,

- **Approximate** $\mathcal{X}$ as $\tilde{\mathcal{X}}$ (the **output** of **CP decomposition**) and solve the least square problem

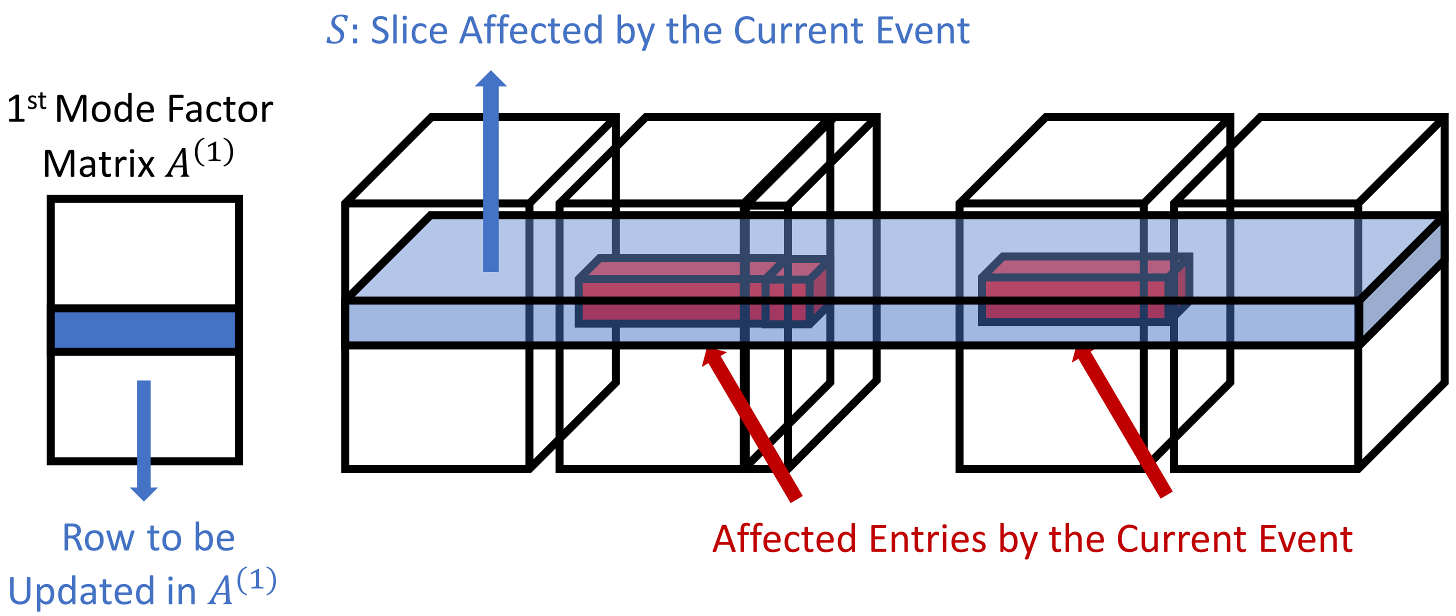
- Computation is proportional to the number of non-zeros in $\Delta\boldsymbol{\mathcal{X}}$



SliceNStitch-Vector: $\mathrm{SNS}_{\mathrm{VEC}}$

For non-time modes,

- Solve the original problem for only a single row
- Time complexity is proportional to the number of non-zeros in S



Pros

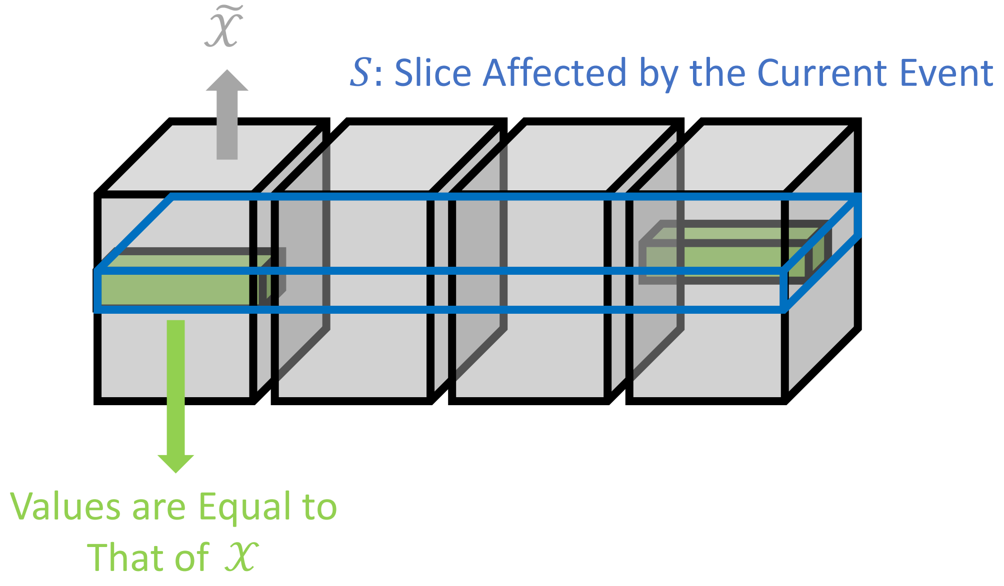
1. Significantly faster than $\mathrm{SNS}_{\mathrm{MAT}}$

Cons

1. Numerically unstable
2. Slow downs if many non-zeros are of the same index

SliceNStitch-Random: $\mathrm{SNS}_{\mathrm{RND}}$

- Given a threshold θ
  - **If** the number of non-zeros in $S \leq \theta$ **then** solve the original problem
  - **Else**
    - Approximate $\mathcal{X}$ as $\tilde{\mathcal{X}}$
    - correcting at most θ randomly chosen entries in $\tilde{\mathcal{X}}$ to $\mathcal{X}$
    - Solve the least square problem
- Time complexity is proportional to θ



Pros

1. Time complexity becomes constant

Cons

1. Numerically unstable
2. Reduction in the quality of the solution compared to $\mathrm{SNS}_{\mathrm{VEC}}$

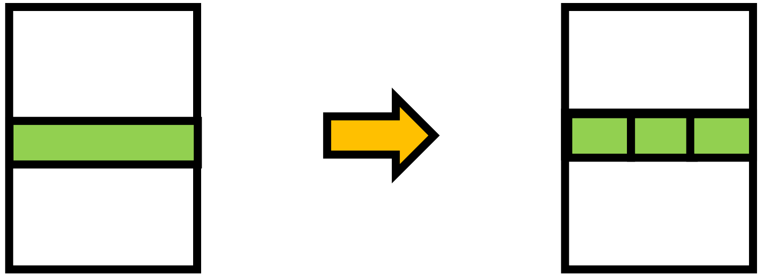
SliceNStitch-Stable: $\mathrm{SNS}^{+}_{\mathrm{VEC}}$ and $\mathrm{SNS}^{+}_{\mathrm{RND}}$

- Problem: $\mathrm{SNS}_{\mathrm{VEC}}$ and $\mathrm{SNS}_{\mathrm{RND}}$ are unstable
  - Many products (e.g. $\bigodot_{i\neq m} A^{(i)}$ and $*_{i\neq m} {A^{(m)}}^{T} A^{(m)}$) are required
  - These result in too large numbers and impair the accuracy of the calculation

- Solution: update entries **one by one**, and **clip** each update value if it is larger than a threshold η

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

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



Guarantees that the updated entry does not increase the objective function although it is clipped

Road Map

- Introduction
- Problem Definition
- Data Model: Continuous Tensor Model
- Optimization Algorithms: SliceNStitch
- **Experiment Results**
- Conclusions

Experiments Settings

- 4 tensors from traffic and crime data

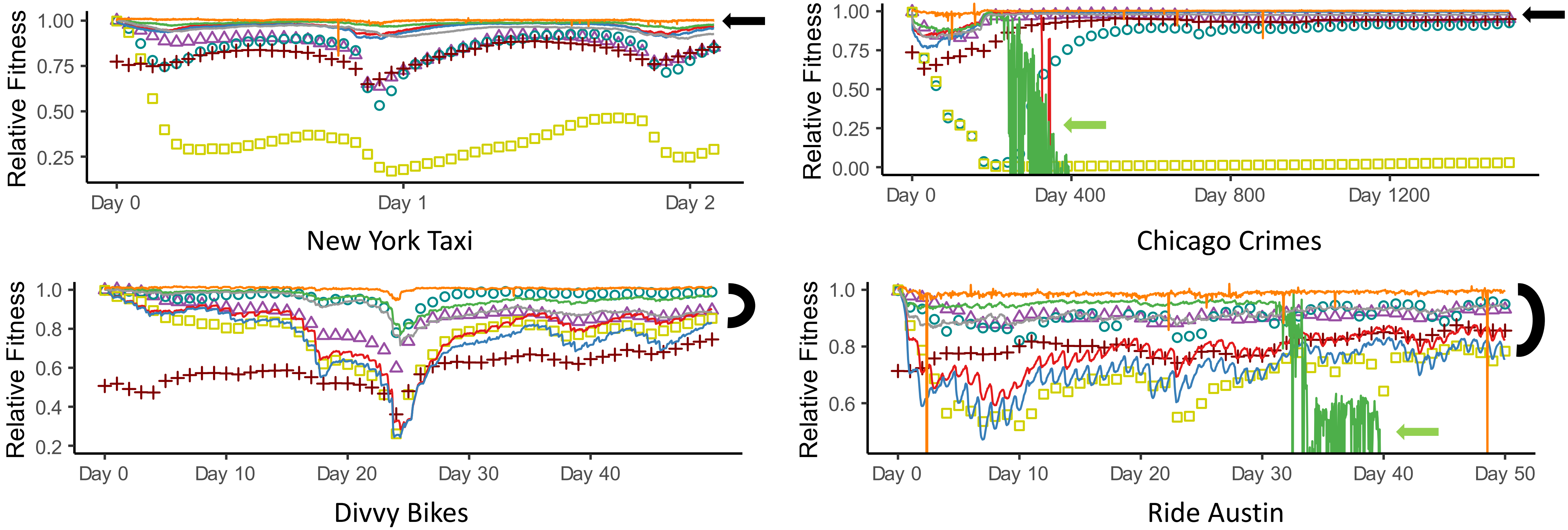
Name	Size	# Non-zeros
Divvy Bikes	673 x 673 x 525594 [minutes]	3.82M
Chicago Crime	77 x 32 x 148464 [hours]	5.33M
New York Taxi	265 x 265 x 5184000 [seconds]	84.39M
Ride Austin	219 x 219 x 24 x 285136 [minutes]	0.89M

- 4 baselines that update CPD periodically
  - ALS [CC70], onlineSCP [ZEB18], CP-stream [SHSK18], NeCPD [ASZ20]

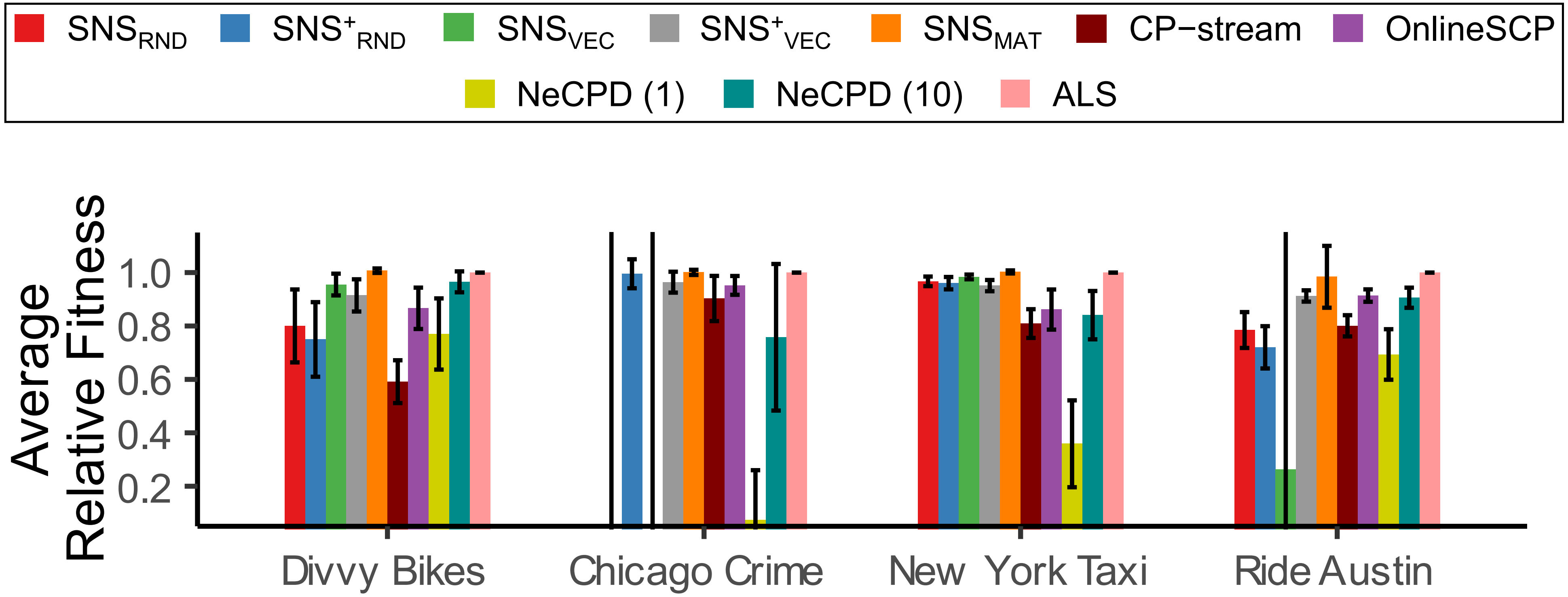
SliceNStitch is Accurate

SNS_{RND} SNS^{+}_{RND} SNS_{VEC} SNS^{+}_{VEC} SNS_{MAT} CP-stream OnlineSCP NeCPD (1) NeCPD (10)

$$Relative\ Fitenss = \frac{Fitness_{target}}{Fitness_{ALS}}$$

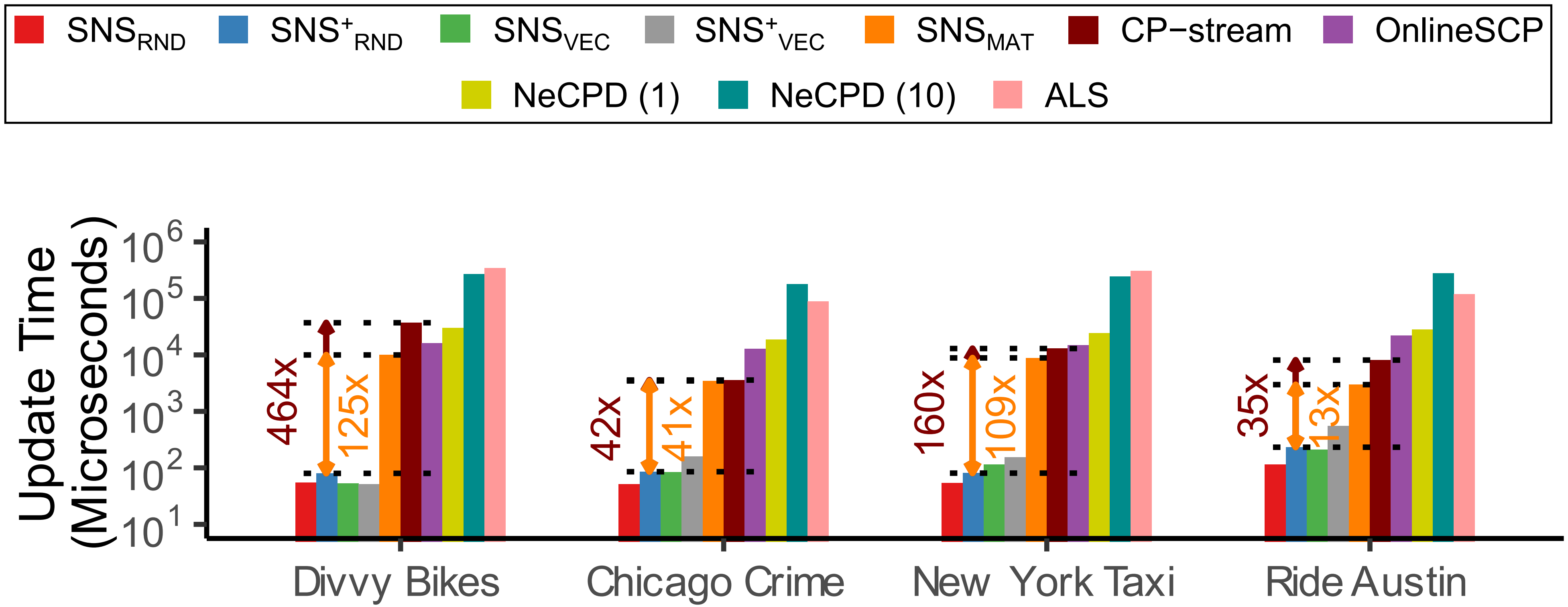


SliceNStitch is Accurate



72 ~ 100% relative fitness to the most accurate baseline

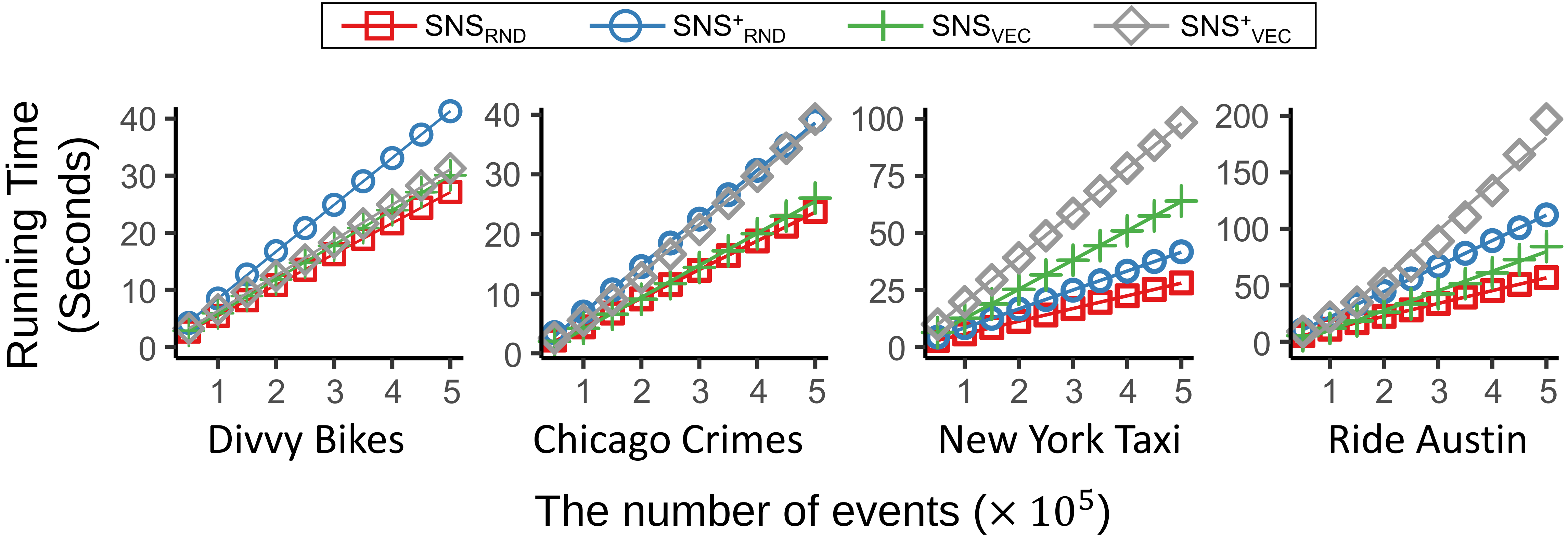
SliceNStitch is Fast



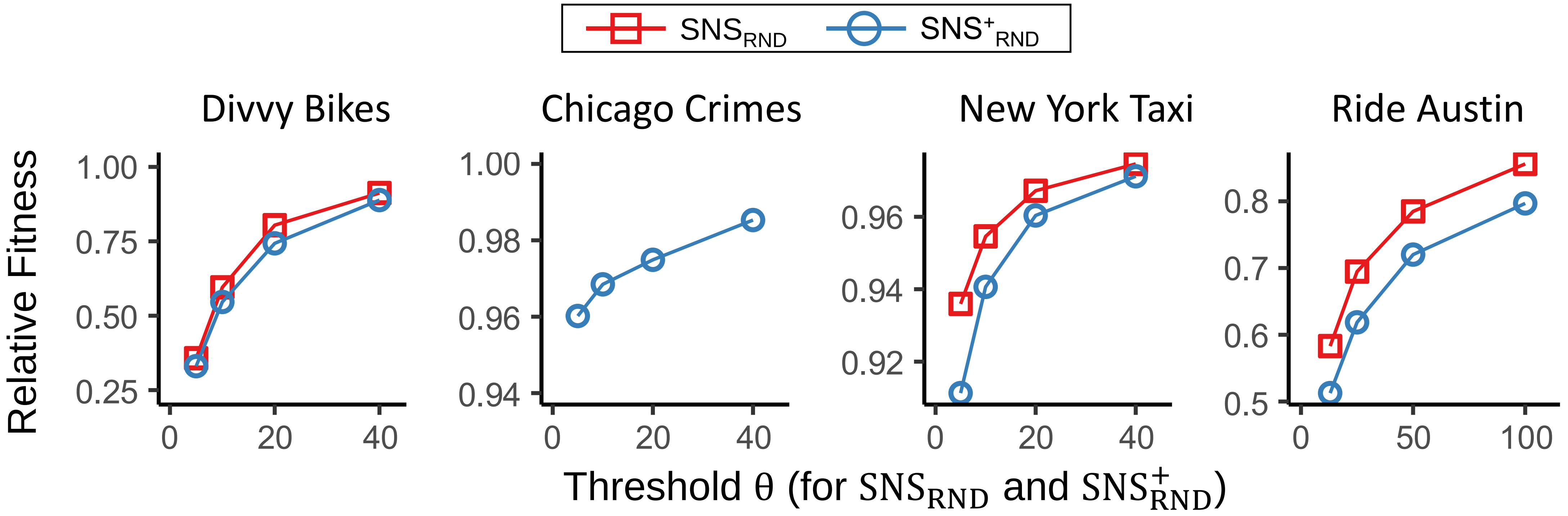
$\mathrm{SNS}^{+}_{\mathrm{RND}}$ is up to 464 times faster than CP-stream

SliceNStitch is Scalable

The total runtime of all SliceNStitch versions was linear in the number of events.

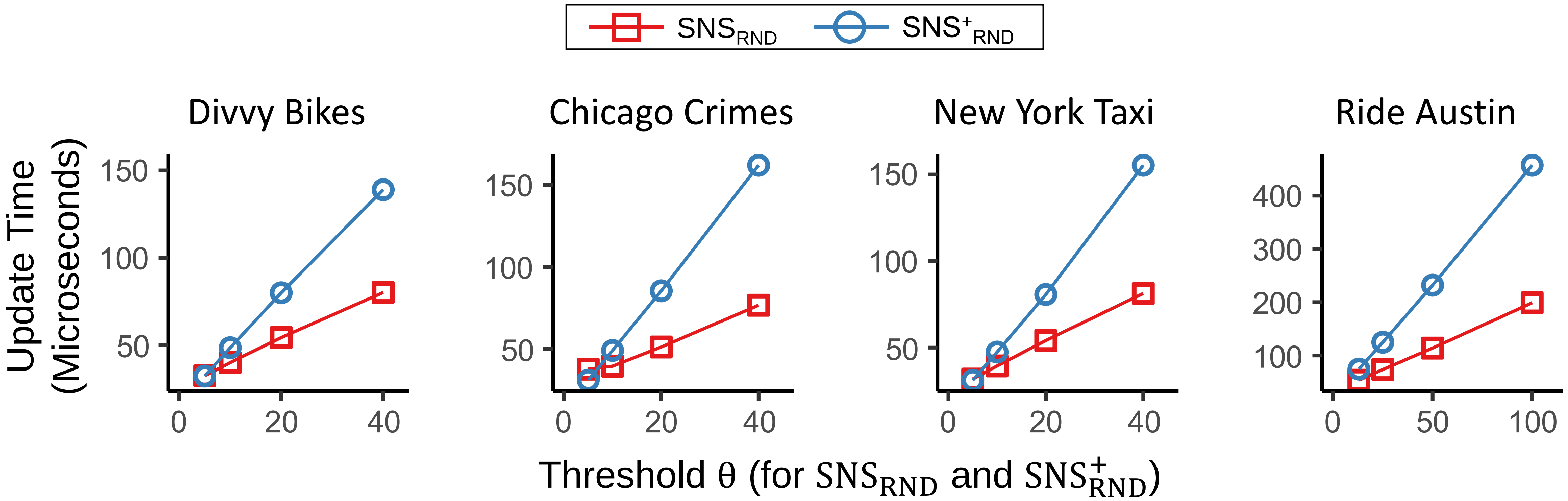


Effect of Sampling Parameter θ (for $\mathrm{SNS_{RND}}$ and $\mathrm{SNS^{+}_{RND}}$)



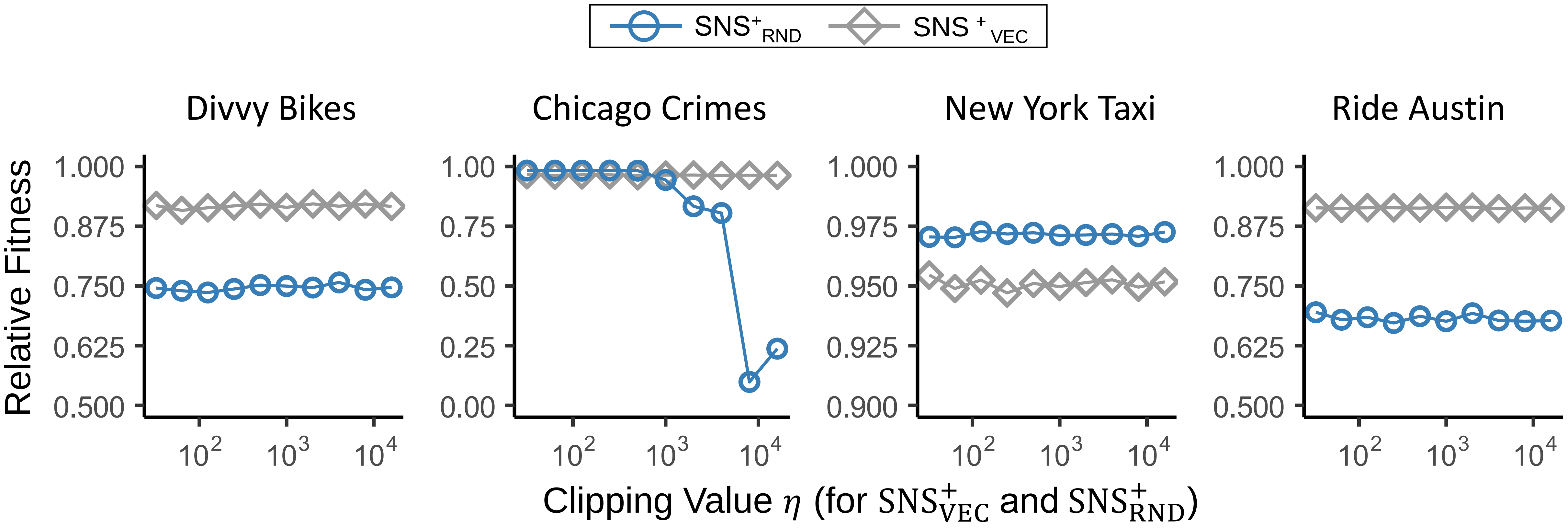
As θ increases, the fitness increases with diminishing returns

Effect of Sampling Parameter θ (for $\mathrm{SNS}_{\mathrm{RND}}$ and $\mathrm{SNS}^{+}_{\mathrm{RND}}$)



As θ increases, runtime grows linearly

Effect of Clipping Value η (for $\mathrm{SNS}^{+}_{\mathrm{VEC}}$ and $\mathrm{SNS}^{+}_{\mathrm{RND}}$)

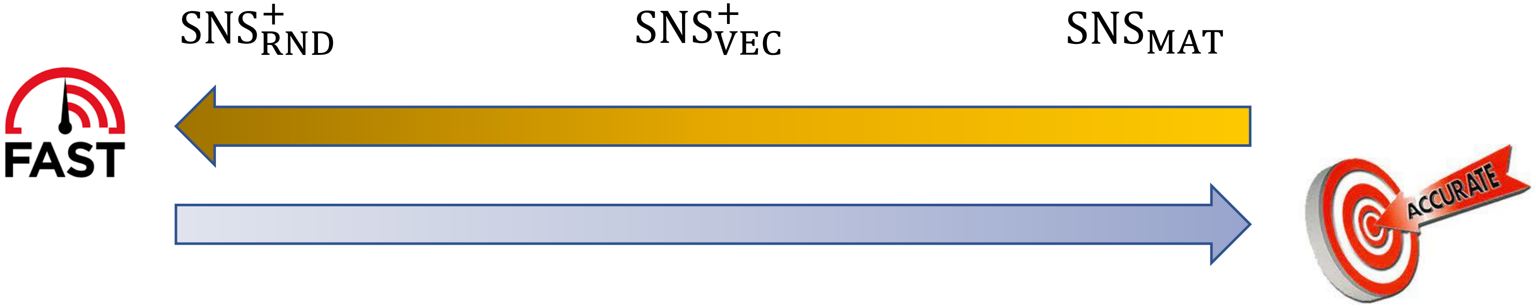


The fitness is insensitive to η as long as η is small enough

Practitioner's Guide

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

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



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

Road Map

- Introduction
- Problem Definition
- Data Model: Continuous Tensor Model
- Optimization Algorithms: SliceNStitch
- Experiment Results
- **Conclusions**

Conclusion

Continuous CPD with SliceNStitch achieved **near-instant updates**, **high fitness**, and a **small number of parameters**.