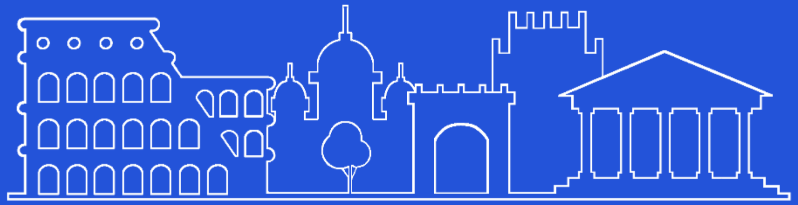
EUROVIS 2022 ME

Streaming Approach to In Situ Selection of Key Time Steps for Time-Varying Volume Data

Mengxi Wu, **Yi-Jen Chiang**, and Christopher Musco

New York University, NY, USA



• **24th EG Conference on Visualization • Rome • 13-17 June 2022 •**

Key Time Steps Selection

Data

- Scientific data, usually generated by simulations
- Regular-grid scalar field --- a scalar value at each vertex (e.g., temperature, pressure, etc.) of the regular-grid volume mesh
- Time-varying: (*T* time steps) x (*N* scalar values)

Motivation

- Expensive to visualize all time steps
- Typically only small changes are between consecutive time steps
- Select a few time steps with the most salient features to visualize
- **Pressing Need:** Perform in situ selection of key time steps, with good performance in theory and in practice.

In Situ Selection of Key Time Steps

Motivation

- Simulations often generate output that exceeds both the storage capacity and bandwidth to transfer the simulation output to disk --- disk just cannot keep up!
- It becomes necessary to select key time steps on the fly while simulation is running --- in situ selection of key time steps (also called triggers in the in-situ community)
- Select highly representative subset of time steps to facilitate post processing and reconstruction with high fidelity.

In Situ Setting

- Process the time-varying volume data in one pass in an online streaming fashion.
- Use only small main memory space and fast computing time.

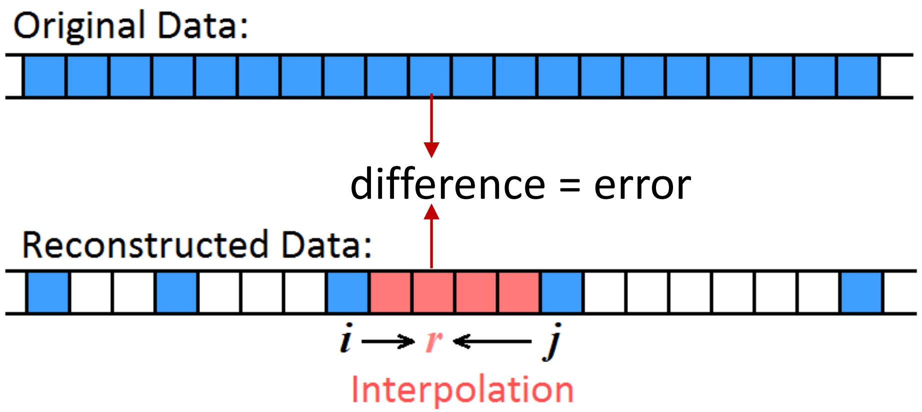
Goal

- Perform key time steps selection in the in situ setting, with theoretical guarantees and works well in practice. **(Extremely Challenging!!)**

Problem Formulation

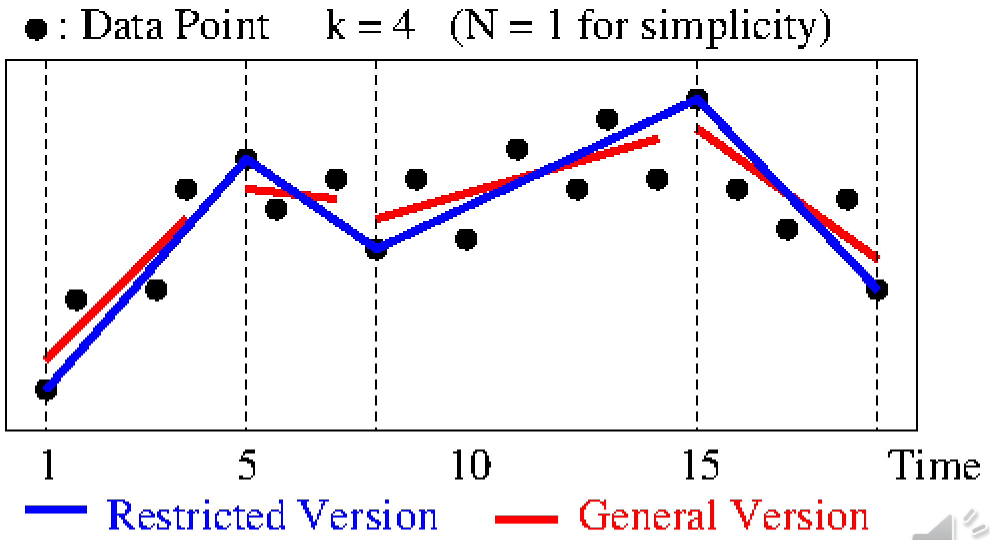
Solution Quality: Reconstruction Error

- A set of selected time steps is good if they can be used to accurately reconstruct the whole time series of volume data.
- For any consecutive selected time steps *i* and *j*, the skipped time steps in between are reconstructed by linear interpolation.



Problem Formulation

- Given an integer k > 0, ``fit'' the time series by a k-piece-wise linear function to minimize the total ``fitting error''.
- Restricted version: selected time steps are from the original data
- General version: no such restriction



Previous Work

In video processing

- *key frame selection* is well studied (large number of frames & small data size in each frame). Dynamic programming [Liu *et al*. 02]; many others are greedy methods --- excellent survey [Hu *et al.* 11]

In volume visualization --- restricted version & **post-simulation (not for in situ)**

- Many results are based on local/greedy considerations: [Akiba *et al.* 06 & 07], [Lu *et al.* 08]; *importance curves* [Wang *et al.* 08]; Time Activity Curve (TAC) [e.g., Woodring *et al.* 09, Lee *et al.* 09, Lee *et al.* 09]; *TransGraph* [Gu *et al.* 11].
- *Flow-based* approach [Frey *et al.* 17]: random sampling
- Globally Optimal: *Dynamic time warping* (DTW) [Tong *et al.* 12]: in-core dynamic programming (DP); [Zhou & Chiang, EuroVis 18]: accurate in-core DP and approximate out-of-core DP

Previous Work (cont.)

In volume visualization (cont.)

- In situ methods --- local considerations, **no theoretical guarantees**
  Restricted version --- triggers: domain-specific [Salloum *et al.* 15]; domain-agnostic [Yamaoka *et al.* 19, Larsen *et al.* 18, Kawakami *et al.* 20]
  General version --- based on piece-wise linear regression [Myers *et al.* 16]
- *Deep learning* method [Porter et al. 19]: for *multivariate data*

In machine learning

- Coreset method for segmenting streaming data [Feldman et al. 14]: starting point of our work, but too complicated and much worse bounds
- Approximation methods for piece-wise linear ([Acharya et al. 16]) and polynomial ([Lokshtanov et al. 21]) regression: non-streaming (i.e., not for in situ)

Our New Approaches

We solve the general version of key time steps selection problem

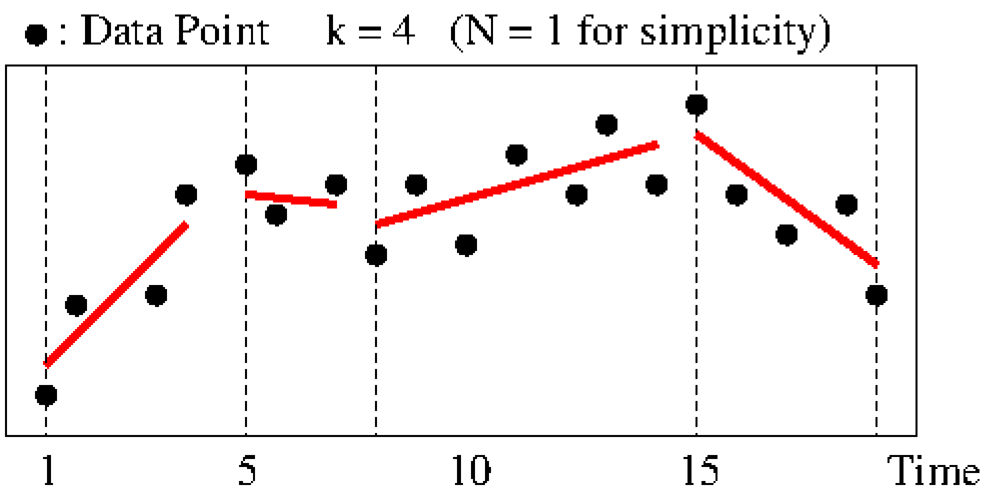
- Formulate the problem as optimal piece-wise linear least squares interpolation (same setting & error metric as previous in situ work [Myers *et al.* 16])
- Building block: online streaming method for computing linear interpolation solutions & their errors, by tools from numerical linear algebra
- Global optimal solution, by the building block and standard DP (improves over the previous state-of-the-art DP in [Zhou & Chiang 18])

- **Novel greedy, online streaming algorithm**
 + optimal I/O
 + very efficient main memory and computing time in theory & in practice; significant speed-up for large data (19.5 hrs -> 2.12 hrs)
 + first algorithm suitable for in situ setting with strong theoretical guarantees on the approximation quality and # of segments stored. Works well in practice

Problem Formulation

General version of key time steps selection

- Given an integer k > 0, ``fit'' the time series by a k-piece-wise linear function to minimize the total ``fitting error''.
- I.e., given k, partition the time steps into k ranges where for each range we perform linear least squares interpolation, to minimize the total interpolation errors from all ranges.



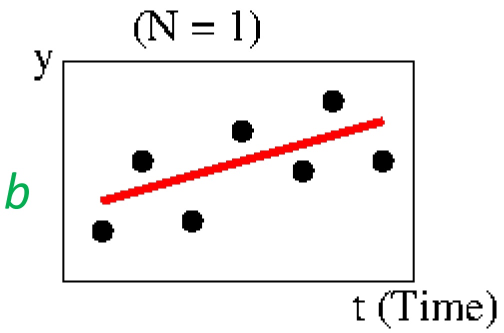
Basic Tools: numerical linear algebra

- Use matrices. They are huge, but we only maintain them implicitly.
- Building Block: Linear least squares interpolation, in a single range (one segment)

Building Block

Linear Least Squares Interpolation: one segment

- Time-varying data: (T time steps) x (N scalar values)
- When $N = 1$ --- Data $Y = [y_1\, y_2\, \dots\, y_T]$ is a length-T vector. A line segment is $y = m\,t + b$
  - ➔ Find **real numbers** m, b s.t. the error $\sum_{i=1}^{T}(m\,i + b - y_i)^2$ is minimized.



- For general N --- *Data Y: a T x N matrix, where row i (denoted by $Y_{i:}$) is the volume at time step i (each row has N scalar values).*
  - ➔ Find **length-N vectors** m, b s.t. the error $\sum_{i=1}^{T} ||m\,i + b - Y_{i:}||_2^2 = ||AZ - Y||_F^2$ is minimized, where $||x||_2$ is L_2 norm,

$$A = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ \vdots & \vdots \\ T & 1 \end{bmatrix} \quad Z = \overset{\xleftrightarrow{\quad N \quad}}{\begin{bmatrix} m \\ b \end{bmatrix}}$$

A: a T x 2 matrix
Z: a 2 x N matrix
AZ: a T x N matrix

For matrix X, $||X||_F$ is the Frobenius norm:

$$||X||_F = \sqrt{\sum_{i,j} X_{ij}^2}$$

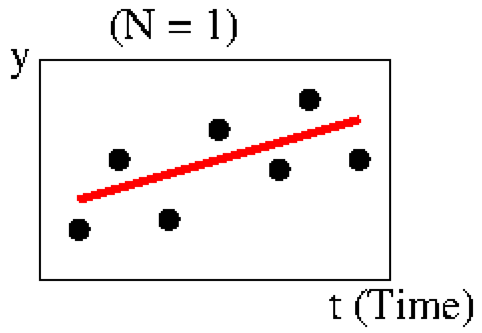
- Optimal Solution: $Z^* = (A'A)^{-1} A'Y$

A': transpose of A (usually A^T but T is already used for # time steps)

Building Block (cont.)

Linear Least Squares Interpolation: one segment

- Time-varying data: (*T* time steps) x (*N* scalar values)
- *Data Y: a T x N matrix, where row i is the volume at time step i.*
- Optimal Solution: $Z^* = (A'A)^{-1} A'Y$
- **In situ:** *Y* is given one time step (i.e., one row) at a time
- Online streaming computation for matrix operations in row-arrival order.
  E.g compute *A' A when A grows from 2 rows to 3 rows (A' from 2 columns to 3 columns)*



$$\begin{bmatrix} 1 & 2 & \boxed{\begin{matrix} 3 \\ 1 \end{matrix}} \\ 1 & 1 & \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ \boxed{3 \quad 1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}}_{\text{Current Result}} + \underbrace{\boxed{\begin{bmatrix} 3 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 1 \end{bmatrix}}}_{\text{New Update}}.$$

Optimal Solution *Z** & Optimal Error *Err**
can each be computed online streaming,
using O(N) time per time step, with O(N)
main memory space; both are optimal

(See paper)

Dynamic Programming

EUROVIS 2022 ME

Initial Computation

- *e(i, j)*: linear least squares interpolation error for time range *[i, j] --- e(i, j) is the optimal error Err**
- *Compute e(i, j) for each possible range [i, j], for* $1 \le i < j \le T$.
- *Use the building block: online streaming method*

  Pass 1: compute *e(1, 2), e(1, 3), …, e(1, T),* in that order

  Pass 2: compute *e(2, 3), e(2, 4), …, e(2, T),* in that order

  Pass *i* (*i = 1, 2, … T-1*): compute *e(i, i+1), e(i, i+2), …, e(i, T)*

Each pass: incremental update, in *O(NT)* time.
Total: $O(NT^2)$ *time*

Dynamic Programming (DP)

- $L(i,k)$: *minimum total error of partitioning the range [1, i] into k segments.*

$$L(i,1) = e(1,i)$$

$$L(i,k) = min_{k\le p\le i-1}\{L(p-1,k-1)+e(p,i)\},$$ for *k = 2 -> T & i = 2 -> T.* $O(T^3)$ *time*

Overall: $O(NT^2 + T^3)$ *time* (NT^2 dominates). Out-of-Core: $O(NT^2/B)$ I/O cost
(B: # items fitting in one disk block) **Main issue: Does NOT work for in situ setting**

Approximate Greedy Algorithm

Basic Greedy Algorithm (online streaming, suitable for in situ setting)

Input: threshold parameter $E > 0$, data matrix Y (a T x N matrix), where row i ($Y_{i:}$) is the volume at time step i (each row has N scalar values).

1. *Let s be the starting time step for the current segment. Initially s ← 1.*
2. *For j = 2, …, T do*

    Compute the linear least squares interpolation in time range [s, j], i.e., try to include current time step j into the current segment: [m, b*, Err*] ← Best-Linear-Fit([s,…,j] [$Y_{s:}, \dots, Y_{j:}$])*

    If Err $\geq E$ (resulting error Err* is too large) then*

    End the current segment at time step j-1, and start a new segment at time step j: s ← j

    Else

    Update the current segment to [m, b*, Err*] (i.e., to include time step j)*

    end for

Analysis: Use the building block to compute *Best-Linear-Fit()* in online streaming fashion

Overall: running time O(NT), I/O cost O(NT / B) --- both linear in dataset size

main memory: keep O(1) time steps, i.e, O(N) space. **All bounds are optimal**

Approximate Greedy Algorithm

Theorem *For any integer k > 0, let optimal piece-wise linear interpolation solution with k pieces have error Cost*, then Basic Greedy Algorithm produces a piece-wise linear solution with error Cost ≤ Cost* + Ek and q segments where q ≤ k + (2 Cost*) / E.*

See paper for high-level intuition, and Appendix A for a formal proof.

Corollary (Bi-criteria Approximation) *For any accuracy parameter $0 < \varepsilon \leq 1$, if we set E = (ε Cost*) / k in Basic Greedy Algorithm, then its solution has error Cost ≤ (1+ε) Cost*, with q ≤ (3/ε) k segments.*

Directly plug in the value of E into Theorem.

Note: *The greedy solution is near optimal, no more than (1+ε) and (3/ε) times the optimal error and # of segments, respectively.* ***Issue: Cost* is unknown!***

Approximate Greedy Algorithm

Final Greedy Algorithm (online streaming; does not require the knowledge of Cost*)

- *From Corollary, Basic Greedy needs to set E = (ε Cost* / k)*
- *Observation: Let σ > 1 be some constant (e.g., σ = 5). If we set E = σ (ε Cost* / k), we get the same guarantee for q (# segments) as in Corollary, with error factor (1+ σε) instead of (1+ε). If we set E = $\frac{1}{\sigma}$ (ε Cost* / k), we get less error but q might be $\sigma(3/\varepsilon)k$ instead of $(3/\varepsilon)k$.*

- *Gridding strategy for choose the right value of E, with 3 key ideas*

(1) Identify lower and upper bounds E_{min} and E_{max} on E
(2) In parallel, compute solutions for a geometric grid of thresholds between these bounds
(3) Combine tasks (1) and (2) so that everything is done in one pass, in an online streaming fashion

Approximate Greedy Algorithm

Final Greedy Algorithm

- *Task (2): In parallel, compute solutions for a geometric grid of thresholds between E_{min} and E_{max}*

- *Run Basic Greedy Algorithm with thresholds (one value per parallel thread) in list:*

$$\widetilde{\boldsymbol{E}}_{\boldsymbol{min}} = \boldsymbol{\sigma}^{\lfloor \log_\sigma E_{min} \rfloor}, \boldsymbol{\sigma}^{\lfloor \log_\sigma E_{min} \rfloor + 1}, \ldots, \boldsymbol{\sigma}^{\lceil \log_\sigma E_{max} \rceil} = \widetilde{\boldsymbol{E}}_{\boldsymbol{max}}$$

- *# parallel threads: $O(\log_\sigma(E_{max}/E_{min}))$; contains a value by a factor σ from ideal E*

- *Tasks (1) and (3):*

- *Compute E_{max}: set E_{max} = cost of best linear fit with 1 piece.*
  Compute on the fly; the value monotonically goes up, which may create new threads: OK, no need to re-run earlier time steps, as the result is the same (1 piece)

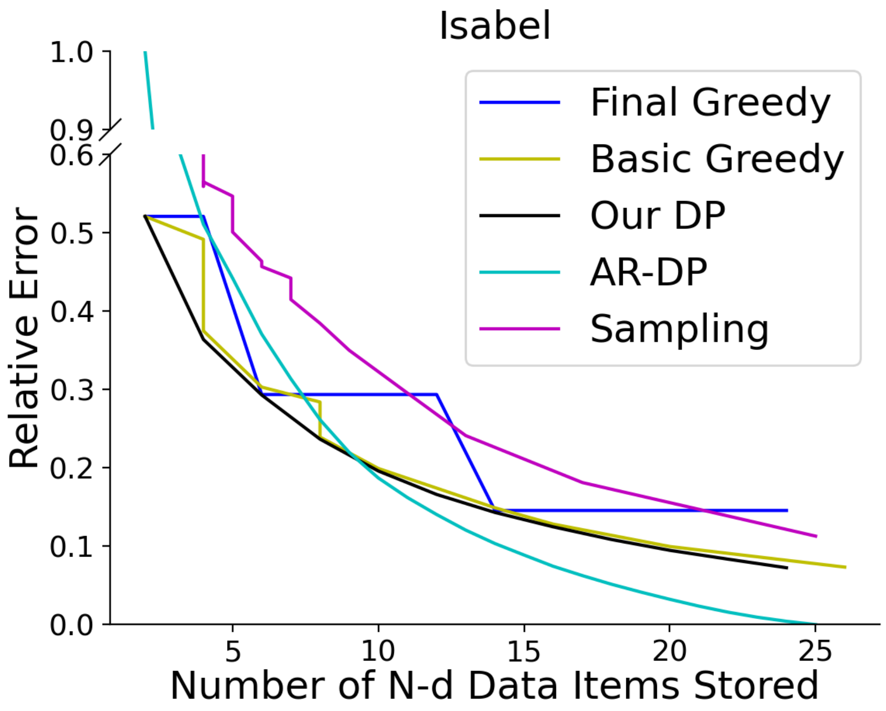
- *Compute E_{min}: similar (roughly T/2 pieces). See paper.*

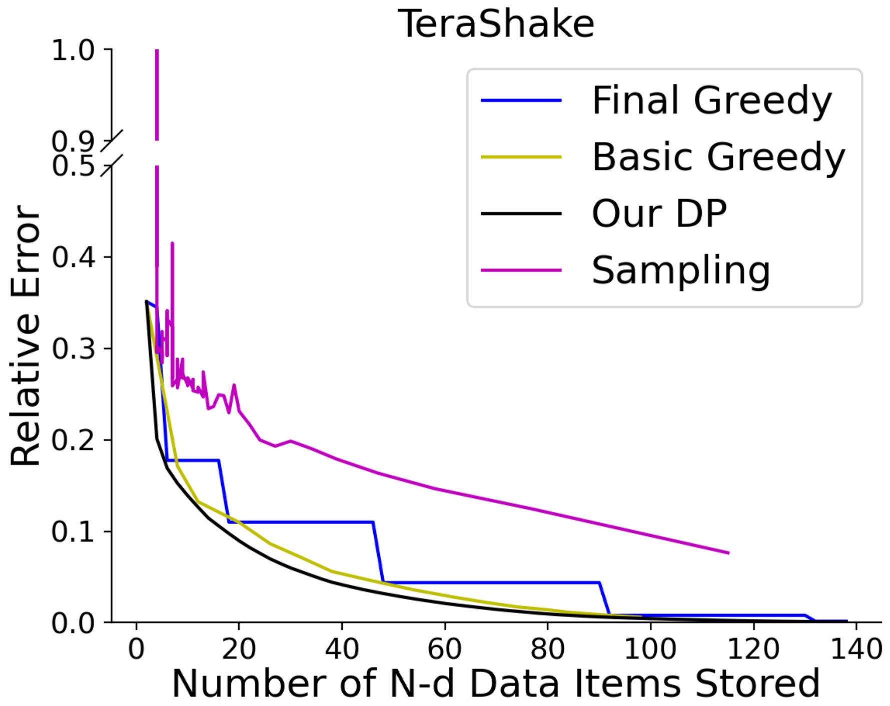
- **Overall:** *one pass, online streaming.*

Results

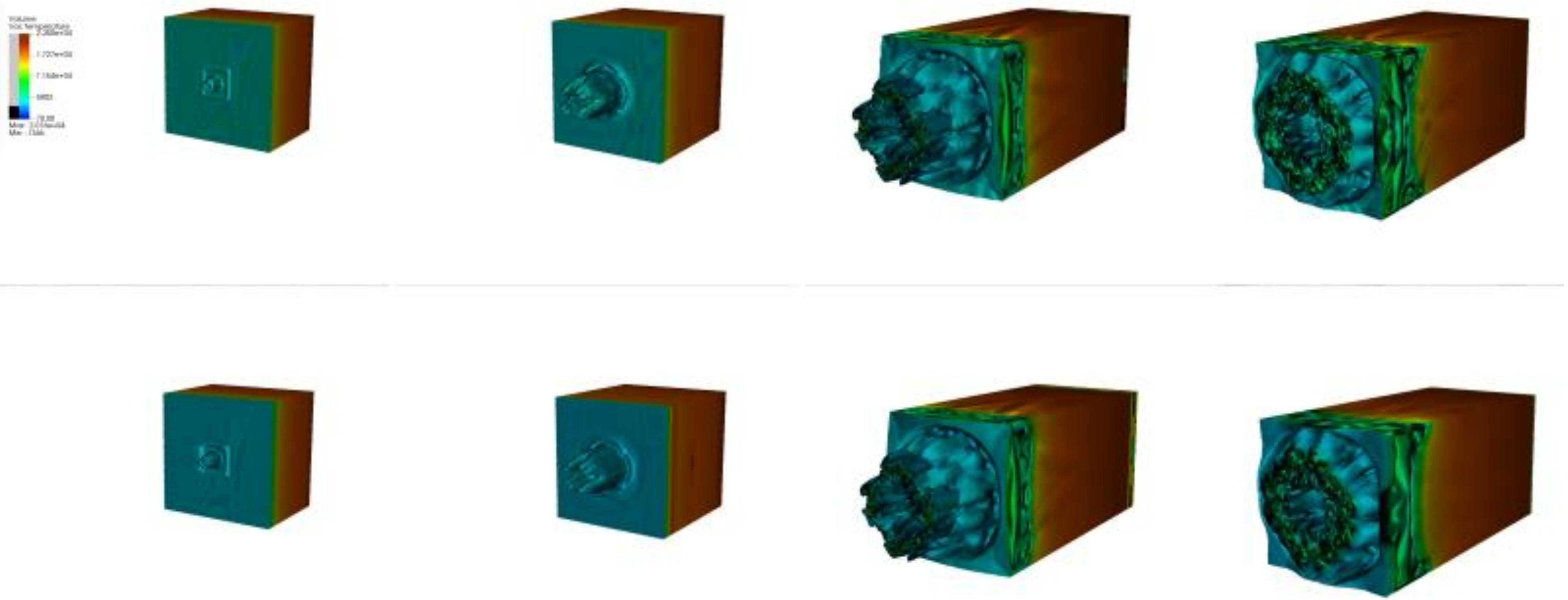
Compare:

- Final Greedy ($\sigma = 5$)
- Basic Greedy
- Our DP
- AR-DP (Accurate Restricted DP) [Zhou & Chiang 18]
- Uniform Sampling





Results



Ground Truth

Final Greedy

NRMSE: 3.85% 4.51% 2.61% 7.57%

Results

Compare:

- AR-DP (Accurate Restricted DP) [Zhou & Chiang 18]
- Our DP
- Basic Greedy
- Final Greedy (σ = 5)

In-Core Data: Efficiency Analysis

Dataset	Method	Runtime	I/O	DP	*e*-time	Mem
Vortex (T: 100)	AR-DP	3957	5.48	1.25	3951	831MB
(784 MB)	Our DP	164	5.72	1.38	158	881MB
$N = 2.1$M	Basic Greedy	14	4.26	N/A	N/A	50.3MB
(Th: 9)	Final Greedy	17.4	4.35	N/A	N/A	304MB
Isabel (T: 48)	AR-DP	5528	6.72	0.15	5821	4.8GB
(4.46GB)	Our DP	743	6.43	0.16	731	5.4GB
$N = 25$M	Basic Greedy	44	6.88	N/A	N/A	600MB
(Th: 6)	Final Greedy	86.52	6.26	N/A	N/A	2480MB

Th: # threads in Final Greedy

Runtime, I/O & *e*-time: in **seconds**

DP: in **milliseconds**

Results

EUROVIS 2022 ME

Compare:

- Our DP
- Basic Greedy
- Final Greedy

(σ = 5)

Larger Data: Efficiency Analysis

Dataset	Method	R-Time	I/O	DP	e-time	Mem
TeraShake (23.7GB, T: 227) $N = 28$M (Th: 14)	Our DP	6.00h*	39m*	12	6.00	676MB
	Basic Greedy	3.4m	0.50m	N/A	N/A	675MB
	Final Greedy	11.56m	0.49m	N/A	N/A	6223MB
Radiation (27.4GB, T: 200) $N = 32$M (Th: 10)	Our DP	6.09h*	58m*	11	6.09	886.5MB
	Basic Greedy	4.0m	0.53m	N/A	N/A	885.7MB
	Final Greedy	12.46m	0.54m	N/A	N/A	5912MB

Th: # threads in Final Greedy

DP: in **milliseconds**

e-time: in **hours**

Results

EUROVIS 2022 ME

Largest Data: Efficiency of Final Greedy

Dataset	Size	T	Total	I/O	Th	Memory
Radiation	27.4GB	200	12.46m	0.54m	10	5912MB
Radiation2	54.8GB	400	27.68m	1.04m	11	6475MB
Radiation4	109.6GB	800	59.69m	2.14m	11	6475MB
Radiation8	219.2GB	1600	127.37m	3.81m	12	7039MB

Th: # threads; *N = 32M*

- # threads ($\log_\sigma(E_{max}/E_{min})$): roughly the same, around 10
- I/O time & Total time: both roughly linear in the dataset size
- Total time for Radiation8: 2.12 hrs vs. 19.5 hrs in approximate out-of-core method in [Zhou & Chiang 18]
- Online streaming, suitable for in situ setting

Conclusions

Our New Approaches

- Building block: online streaming method for computing linear interpolation solutions & their errors, using tools from numerical linear algebra
- Global optimal solution, by the building block and standard DP (improves over the previous state-of-the-art DP in [Zhou & Chiang 18])

- **Novel greedy, online streaming algorithm**

  + optimal I/O

  + very efficient main memory and computing time in theory & in practice; significant speed-up for large data (19.5 hrs -> 2.12 hrs)

  + first algorithm suitable for in situ setting with strong theoretical guarantees on the approximation quality and # of segments stored. Works well in practice

Acknowledgement

NSF grants CCF-2008768 and CCF-2045590.