20th EG/VGTC
Conference on Visualization

Key Time Steps Selection for Large-Scale Time-Varying Volume Datasets Using an Information-Theoretic Storyboard

Bo Zhou and **Yi-Jen Chiang**

New York University, NY, USA

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* vertices)

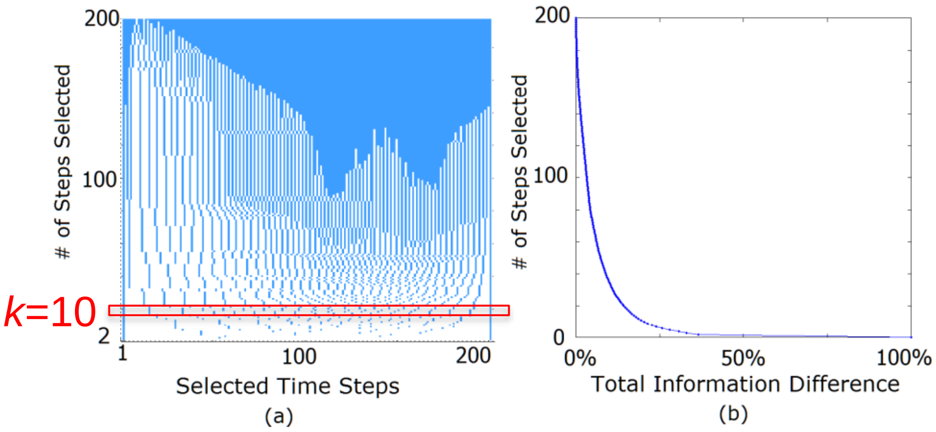
Motivation

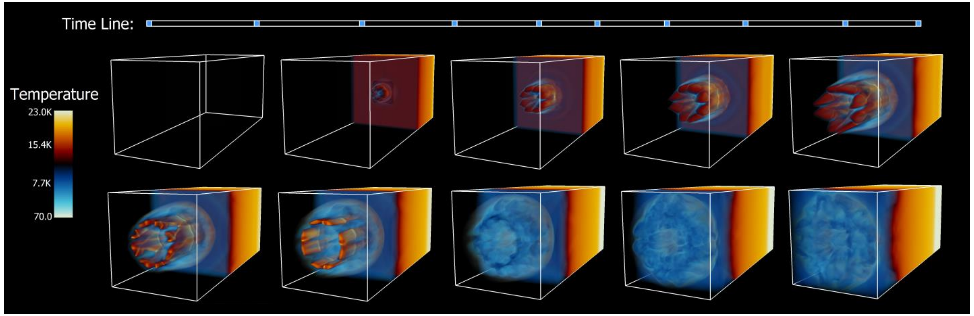
- Expensive to visualize all time steps
- Typically only small changes are between consecutive time steps
- Select a few time steps with the most salient features to visualize, with theoretical guarantees
- Provide a storyboard to guide the user during data exploration

Proposed Scheme

After fully-automatic preprocessing, storyboard to answer in run-time:

- User inputs *k*<=*T*; return a selection of *k* time steps that best represent the data



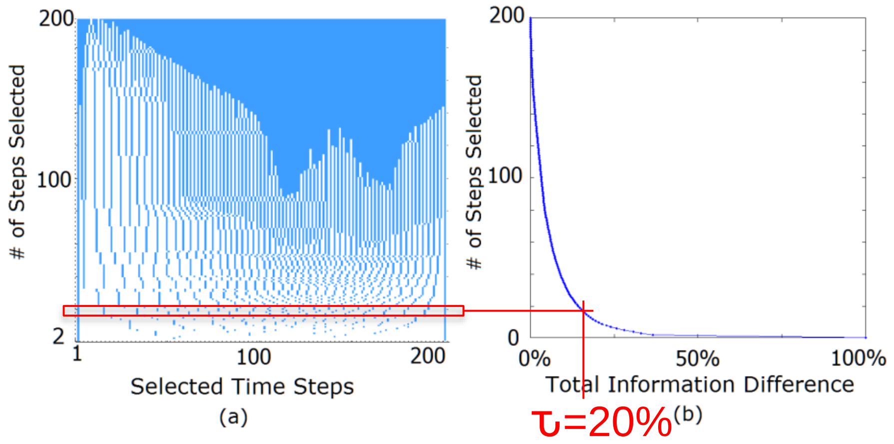


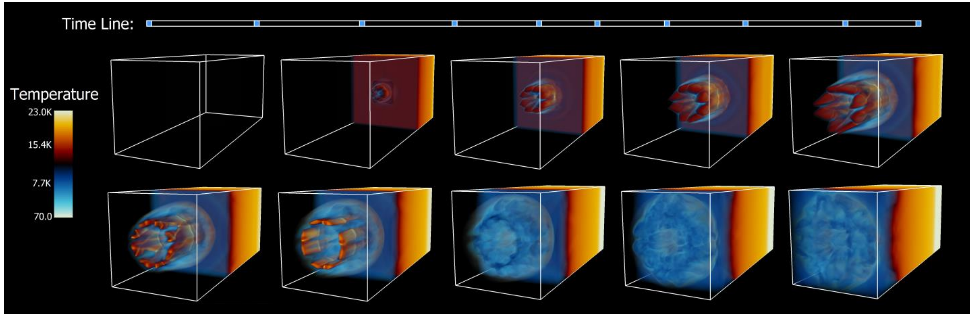
Proposed Scheme

After fully-automatic preprocessing, storyboard to answer in run-time:

- User inputs *k*<=*T*; return a selection of *k* time steps that best represent the data

- User inputs a tolerance τ (in %) of Total Information Difference; return the fewest time steps to satisfy Information Difference <= τ





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

- 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]; *in-situ* method [Myers *et al.* 16].

- *Flow-based* approach [Frey *et al.* 17]: random sampling
- Globally Optimal: *Dynamic time warping* (DTW) [Tong *et al.* 12]:

dynamic programming; I/O issue not considered (can't handle large data)

Our New Approaches

- Fully automatic preprocessing; globally optimal by dynamic programming
- Provide a storyboard of the data to guide data exploration in run time
- Out-of-Core approximate method (multi-pass dynamic programming)

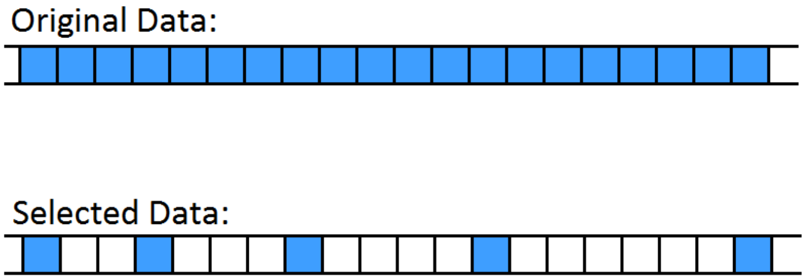
+ optimal I/O
+ significant speed-up for large data (1000+ hrs → < 20 hrs!)
+ close-to-optimal selection qualities

- Independent of the selection-quality metrics used

+ We give Information Difference (InfoD) based on information theory (could extract unknown salient data features [Wang et al. 11])
+ All experiments were under both InfoD and root-mean-square error (RMSE)

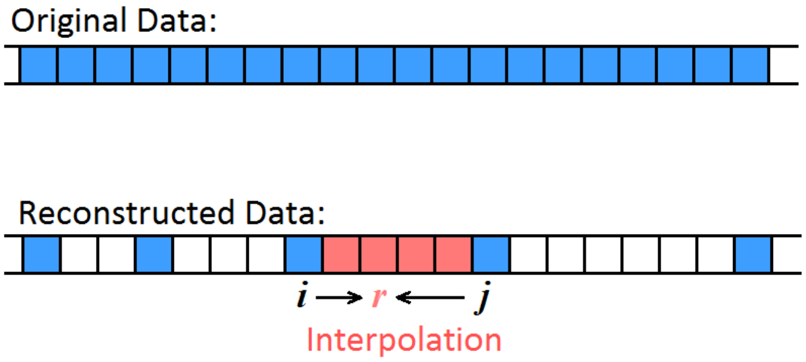
Basic Idea

- How to quantify the quality of selected time steps?



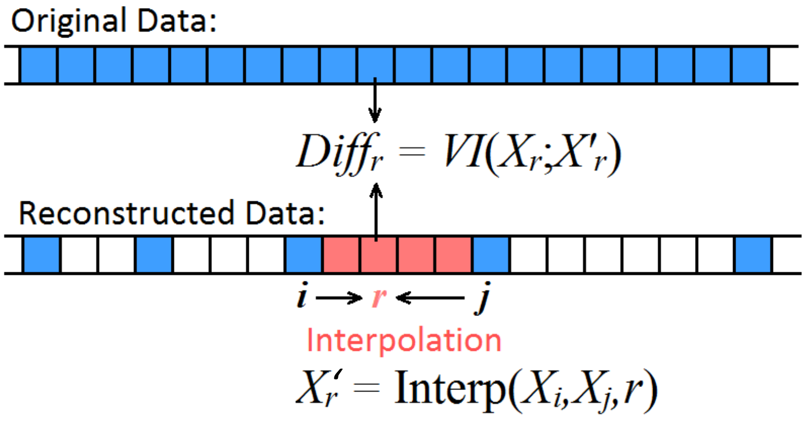
Basic Idea

- How to quantify the quality of selected time steps?
- When looking at the missing time steps, the user would probably *reconstruct* the missing data *in mind* to understand what is going on.



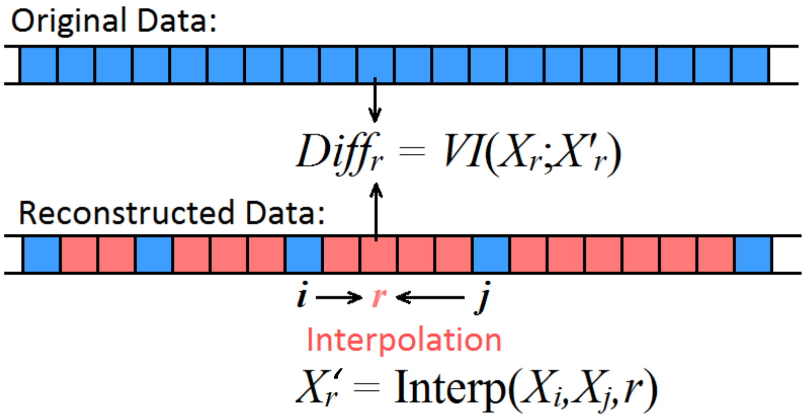
Basic Idea

- How to quantify the quality of selected time steps?
- When looking at the missing time steps, the user would probably *reconstruct* the missing data *in mind* to understand what is going on.
- We use linear interpolation to "simulate" that process. Quantify the difference of information between the reconstructed and the original data.



Basic Idea

- How to quantify the quality of selected time steps?
- When looking at the missing time steps, the user would probably *reconstruct* the missing data *in mind* to understand what is going on.
- We use linear interpolation to “simulate” that process. Quantify the difference of information between the reconstructed and the original data.

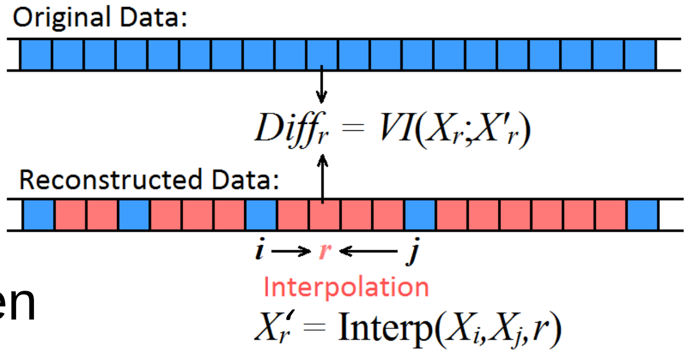


- Find k time steps that minimize the Total Information Difference.

Dynamic Programming

- $Diff_r = VI(X_r; X'_r)$: Difference between reconstructed data and original data

- $c(i,j) = \sum_{i<r<j} Diff_r$: Cost for selecting time steps *i* and *j* while skipping all others in between



- $D^{(k)}(T)$: Minimum cost for selecting *k* time steps from *{1… T}*

The first and last time steps must be selected (linear interp.)

$$D^{(k)}(T) = \min_{1<p<T}\{D^{(k-1)}(p) + c(p,T)\}$$

(Subproblem: $D^{(k)}(i) = \min_{1<p<i}\{D^{(k-1)}(p) + c(p,i)\}$ for i :1 → T and k: 2 → T)

EuroVis 2018

Complexity Analysis

- T time steps (50 ~ 500)
- N vertices for each time step (100M ~ 500M)
- Time to compute $c(i,j)$ in table C for all pairs $\{i,j\}$: $O(T^3N)$
  - Esp. if not fit in memory, #blocks disk read: $O(T^3N/B)$ ← Too Slow
- Time to compute $D^{(k)}(i)$ in the memorization table D for all tuples $\{i,k\}$: $O(T^3)$

- Total in-core time: $O(T^3N + T^3)$
- Total I/O cost: $O(T^3N/B)$

(B: # items fitting in one disk block)

Key Insight to Overcome the Bottleneck

- **Bottleneck:** Computing $c(i,j)$'s, especially when i,j are far apart.

- How are the $c(i,j)$'s used?

  Recall the DP recurrence: $D^{(k)}(i) = \min_{1<p<i}\{D^{(k-1)}(p) + c(p,i)\}$

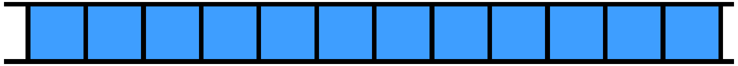
  Note: $c(i,j) = \Sigma_{i<r<j} Diff_r$: *c(i, j) is large when i, j are far apart.*

➔ Such expensive (to compute) $c(i,j)$'s are rarely used!!

Our Solution – Multi-pass Approximate Approach

- First pass, working set $\boldsymbol{S} = \{1, 2, \ldots, T\}$.
  - Use a sliding window (in-core memory) of size $\boldsymbol{t}$, only compute $C(i, j)$ in the sliding window
  - In the cost table C, $c(i, j) = \infty$, for i and j far away (not both in the window)
  - Run DP, compute memoization table D

Original Data: T=12

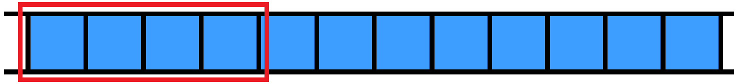


$$
C(i,j) \qquad
\begin{array}{c|cccccccccccc}
 & & & & & & j & & & & & & \\
\hline
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
i & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\end{array}
$$

|S|=12

t=4

Working Set $S=\{\boxed{1,2,3,4},5,6,7,8,9,10,11,12\}$



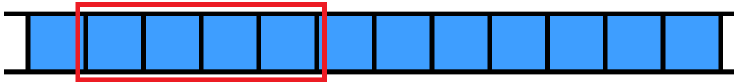
$C(i, j)$ — rows i, columns j

0	0	2.7	5.2	∞	∞	∞	∞	∞	∞	∞	∞
∞	0	0	2.3	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	0	0	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

$|S|=12$

$t=4$

Working Set $S=\{1,\boxed{2,3,4,5},6,7,8,9,10,11,12\}$

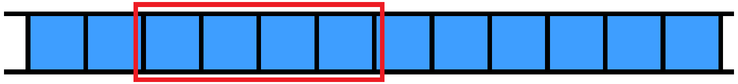


$$
\begin{array}{cc}
C(i,j) & j \\
i & \begin{array}{cccccccccccc}
0 & 0 & 2.7 & 5.2 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & 0 & 0 & 2.3 & 4.7 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & 0 & 0 & 2.0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & 0 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty
\end{array}
\end{array}
$$

|S|=12

t=4

Working Set $S=\{1,2,\boxed{3,4,5,6},7,8,9,10,11,12\}$

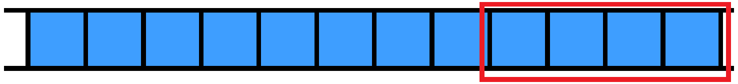


$C(i, j)$						j						
	0	0	2.7	5.2	∞	∞	∞	∞	∞	∞	∞	∞
	∞	0	0	2.3	4.7	∞	∞	∞	∞	∞	∞	∞
	∞	∞	0	0	2.0	4.8	∞	∞	∞	∞	∞	∞
	∞	∞	∞	0	0	1.8	∞	∞	∞	∞	∞	∞
	∞	∞	∞	∞	0	0	∞	∞	∞	∞	∞	∞
i	∞	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	∞
	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

$|S|=12$

$t=4$

Working Set $S=\{1,2,3,4,5,6,7,8,\boxed{9,10,11,12}\}$



$$
C(i,j) \qquad
\begin{array}{c|cccccccccccc}
 & & & & & & j & & & & & & \\
\hline
 & 0 & 0 & 2.7 & 5.2 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & 0 & 0 & 2.3 & 4.7 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & 0 & 0 & 2.0 & 4.8 & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & 0 & 0 & 1.8 & 4.5 & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & \infty & 0 & 0 & 1.1 & 3.5 & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 1.8 & 2.1 & \infty & \infty & \infty \\
i & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 3.8 & 6.1 & \infty & \infty \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 2.3 & 6.8 & \infty \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 2.8 & 6.5 \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 1.2 \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0
\end{array}
$$

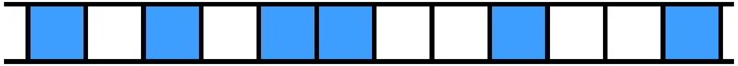
Our Solution – Multi-pass Approximate Approach

- First pass, working set $\boldsymbol{S} = \{1, 2, \ldots, T\}$.
  - Use a sliding window (in-core memory) of size $\boldsymbol{t}$, only compute $C(i, j)$ in the sliding window
  - In the cost table C, $c(i, j) = \infty$, for i and j far away (not both in the window)
  - Run DP, compute memoization table D
- Second pass, working set $\boldsymbol{S} = \{$best $k = \boldsymbol{T/2}$ time steps from previous DP result$\}$.
  - Use a sliding window of size t, update those $c(i, j) = \infty$ which are now close enough in the current sliding window, by estimating only using S
  - Run DP, update memoization table D

|S|=6

t=4

Working Set S={1, 3, 5,6, 9, 12}



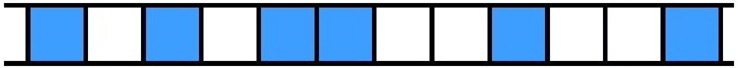
$C(i, j)$ j

i												
	0	0	2.7	5.2	∞	∞	∞	∞	∞	∞	∞	∞
	∞	0	0	2.3	4.7	∞	∞	∞	∞	∞	∞	∞
	∞	∞	0	0	2.0	4.8	∞	∞	∞	∞	∞	∞
	∞	∞	∞	0	0	1.8	4.5	∞	∞	∞	∞	∞
	∞	∞	∞	∞	0	0	1.1	3.5	∞	∞	∞	∞
	∞	∞	∞	∞	∞	0	0	1.8	2.1	∞	∞	∞
	∞	∞	∞	∞	∞	∞	0	0	3.8	6.1	∞	∞
	∞	∞	∞	∞	∞	∞	∞	0	0	2.3	6.8	∞
	∞	∞	∞	∞	∞	∞	∞	∞	0	0	2.8	6.5
	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	0	1.2
	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	0
	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

|S|=6

t=4

Working Set S={1,3,5,6,9,12}

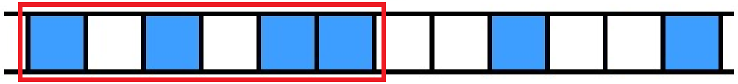


$$
C(i,j) \qquad
\begin{array}{c|cccccccccccc}
 & & & & & & j & & & & & & \\
\hline
 & 0 & 0 & 2.7 & 5.2 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & 0 & 0 & 2.3 & 4.7 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & 0 & 0 & 2.0 & 4.8 & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & 0 & 0 & 1.8 & 4.5 & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & \infty & 0 & 0 & 1.1 & 3.5 & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 1.8 & 2.1 & \infty & \infty & \infty \\
i & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 3.8 & 6.1 & \infty & \infty \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 2.3 & 6.8 & \infty \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 2.8 & 6.5 \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 1.2 \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0
\end{array}
$$

|S|=6

t=4

Working Set $S=\{\boxed{1,3,5,6}9,12\}$

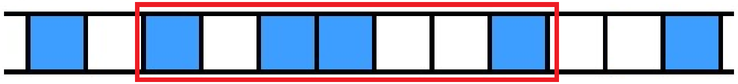


$$
\begin{array}{cc}
C(i,j) & j \\
i & \begin{matrix}
0 & 0 & 2.7 & 5.2 & 6.6 & 7.9 & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & 0 & 0 & 2.3 & 4.7 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & 0 & 0 & 2.0 & 4.8 & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & 0 & 0 & 1.8 & 4.5 & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & 0 & 0 & 1.1 & 3.5 & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & 0 & 0 & 1.8 & 2.1 & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 3.8 & 6.1 & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 2.3 & 6.8 & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 2.8 & 6.5 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 1.2 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0
\end{matrix}
\end{array}
$$

|S|=6

t=4

Working Set S={1,$\boxed{3,5,6,9}$,12}



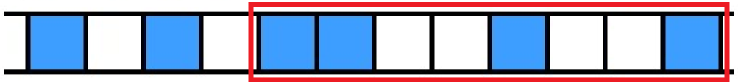
$C(i, j)$

	j											
i	0	0	2.7	5.2	6.6	7.9	∞	∞	∞	∞	∞	∞
	∞	0	0	2.3	4.7	∞	∞	∞	∞	∞	∞	∞
	∞	∞	0	0	2.0	4.8	∞	∞	∞	∞	∞	∞
	∞	∞	∞	0	0	1.8	4.5	∞	∞	∞	∞	∞
	∞	∞	∞	∞	0	0	1.1	3.5	5.6	∞	∞	∞
	∞	∞	∞	∞	∞	0	0	1.8	2.1	∞	∞	∞
	∞	∞	∞	∞	∞	∞	0	0	3.8	6.1	∞	∞
	∞	∞	∞	∞	∞	∞	∞	0	0	2.3	6.8	∞
	∞	∞	∞	∞	∞	∞	∞	∞	0	0	2.8	6.5
	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	0	1.2
	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	0
	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

$|S|=6$

$t=4$

Working Set $S=\{1,3,\boxed{5,6,9,12}\}$



$$
\begin{array}{c|cccccccccccc}
C(i,j) & & & & & & j & & & & & & \\
\hline
 & 0 & 0 & 2.7 & 5.2 & 6.6 & 7.9 & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & 0 & 0 & 2.3 & 4.7 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & 0 & 0 & 2.0 & 4.8 & \infty & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & 0 & 0 & 1.8 & 4.5 & \infty & \infty & \infty & \infty & \infty \\
 & \infty & \infty & \infty & \infty & 0 & 0 & 1.1 & 3.5 & 5.6 & \infty & \infty & 7.7 \\
i & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 1.8 & 2.1 & \infty & \infty & 6.4 \\
 & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 3.8 & 6.1 & \infty & \infty \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 2.3 & 6.8 & \infty \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 2.8 & 6.5 \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 & 1.2 \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0 \\
 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0
\end{array}
$$

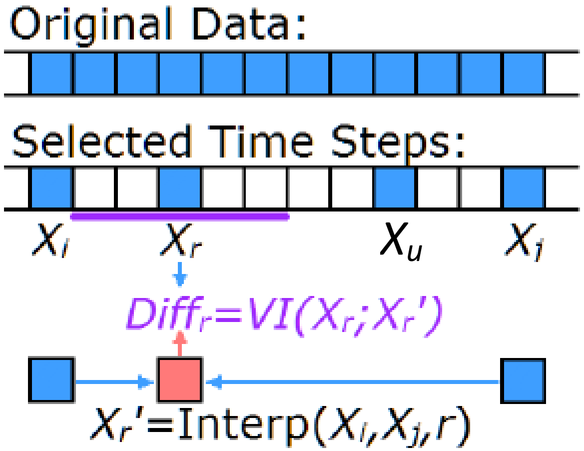
Key Technical Detail

In computing $c(i, j)$ = *sum of* $Diff_v$ $(i<v<j)$, use approximate information difference

- In 2nd row, blue squares (e.g., X_r) are in the sliding window (in-core). Easy to compute $Diff_r$

- Use $Diff_r$ to approximate the *Diff* of each of 5 time steps underlined in purple

➔ $c(i, j) = 5\ Diff_r + 5\ Diff_u$



Our Solution – Multi-pass Approximate Approach

- First pass, working set $\boldsymbol{S} = \{1, 2, \ldots, T\}$.
  - Use a sliding window (in-core memory) of size $\boldsymbol{t}$, only compute $C(i, j)$ in the sliding window
  - In the cost table C, $c(i, j) = \infty$, for i and j far away (not both in the window)
  - Run DP, compute memoization table D
- Second pass, working set $\boldsymbol{S} = \{\text{best } k = \boldsymbol{T}/\mathbf{2} \text{ time steps from previous DP result}\}$.
  - Use a sliding window of size t, update those $c(i, j) = \infty$ which are now close enough in the current sliding window, by estimating only using S
  - Run DP, update memoization table D
- $|\boldsymbol{S}| = \boldsymbol{T}/\mathbf{4}\,, \boldsymbol{T}/\mathbf{8}\,, \boldsymbol{T}/\mathbf{16}\ \ldots$
- Repeat until $|S| \leq t$ (last pass is $|S| <= t$)

$|S|=3$

$t=4$

Working Set $S=\{1,9,12\}$

$|S|<=4$

Last pass!

$C(i, j)$ $\qquad j$

	0	0	2.7	5.2	6.6	7.9	∞	∞	∞	∞	∞	∞
	∞	0	0	2.3	4.7	∞	∞	∞	∞	∞	∞	∞
	∞	∞	0	0	2.0	4.8	∞	∞	∞	∞	∞	∞
	∞	∞	∞	0	0	1.8	4.5	∞	∞	∞	∞	∞
	∞	∞	∞	∞	0	0	1.1	3.5	5.6	∞	∞	7.7
	∞	∞	∞	∞	∞	0	0	1.8	2.1	∞	∞	6.4
i	∞	∞	∞	∞	∞	∞	0	0	3.8	6.1	∞	∞
	∞	∞	∞	∞	∞	∞	∞	0	0	2.3	6.8	∞
	∞	∞	∞	∞	∞	∞	∞	∞	0	0	2.8	6.5
	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	0	1.2
	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	0
	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

Results:

- In each pass, we spend $O(t^2N \cdot |S|)$ to update the $c(i,j)$ values. Then we use $O(T^2 \cdot |S|)$ to run DP. Every pass the size of S is reduced in half, until $|S| \leq t$. So number of passes is $O(\log_2 \frac{T}{t})$.
- Total DP time: $O(T^2 \cdot (T + {}^{T}/_{2} + {}^{T}/_{4} + \ \ldots)) = O(T^3)$
- Total time for *c(i,j)*'s: $O(t^2N \cdot (T + {}^{T}/_{2} + {}^{T}/_{4} + \ \ldots)) = O(t^2TN)$ $\ll O(T^3N)$
- Overall – about linear to data size: $O(t^2TN + T^3) = O(t^2TN)$

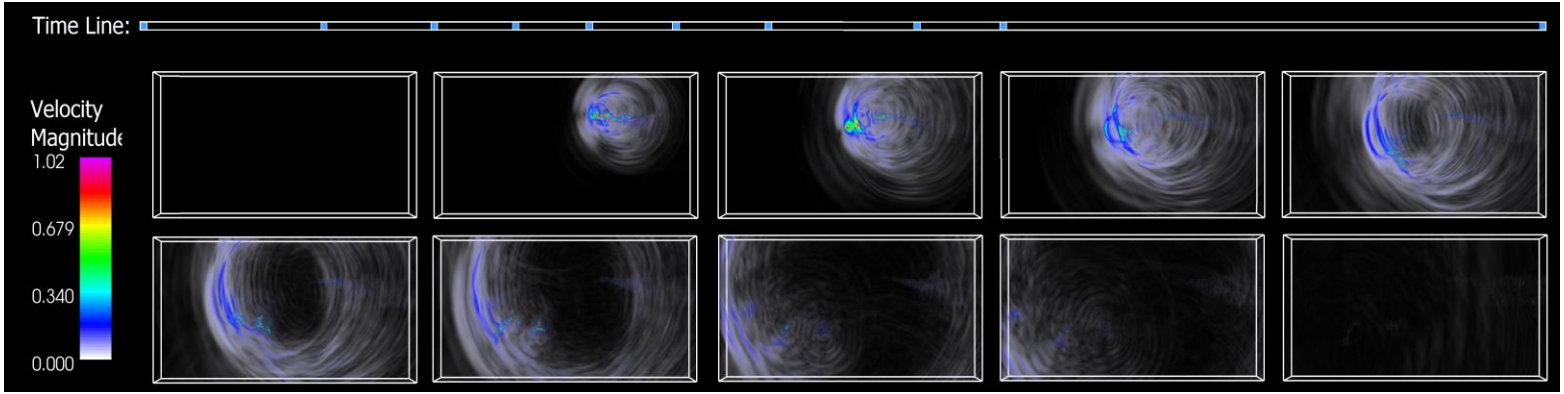
  $O(t^2TN)$ *(improved from $O(T^3N)$) * t is a chosen constant typically 10~15
- I/O – equivalent to 2 linear scans: $(N/B)(T + T/2 + T/4 + \ldots) \leq 2(N/B)T$

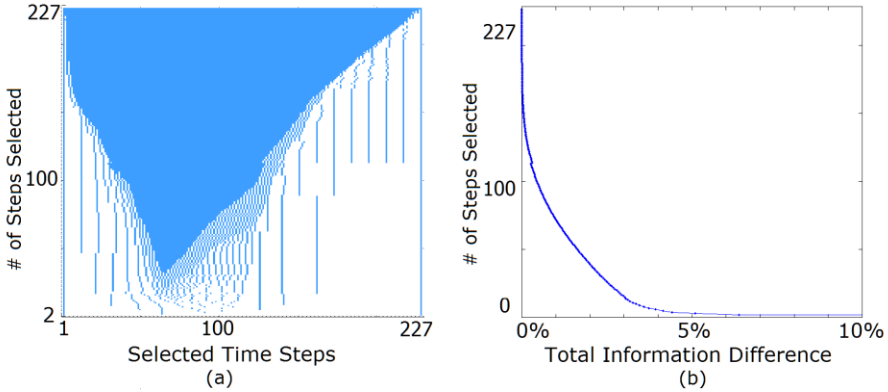
  $O(TN/B)$ (improved from $O(T^3N/B)$) $= O(TN / B)$ --- Optimal I/O!
- Error of approximation:

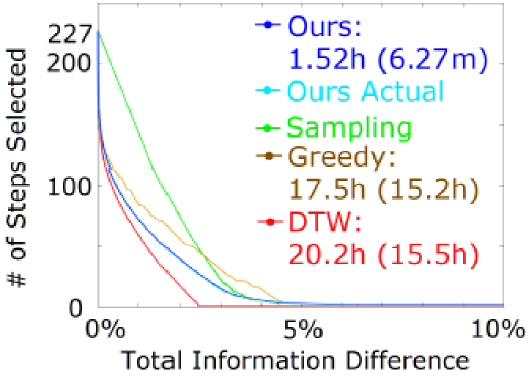
  Very low, similar results as accurate method

Results:

Compare:
- Ours (t = 12)
- Uniform Sampling
- Greedy
- DTW [Tong *et al.* 12]



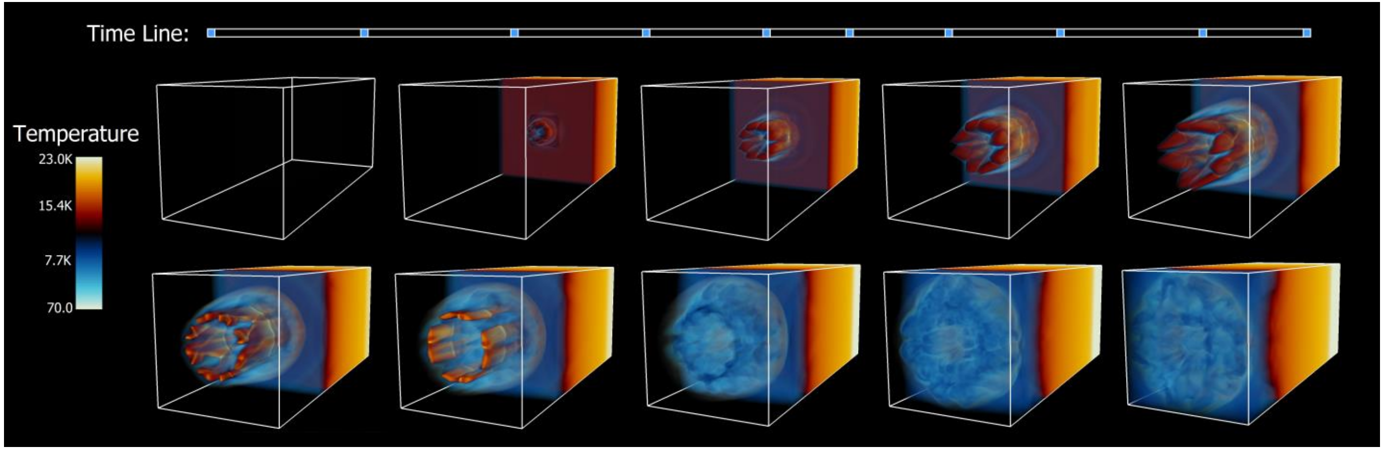


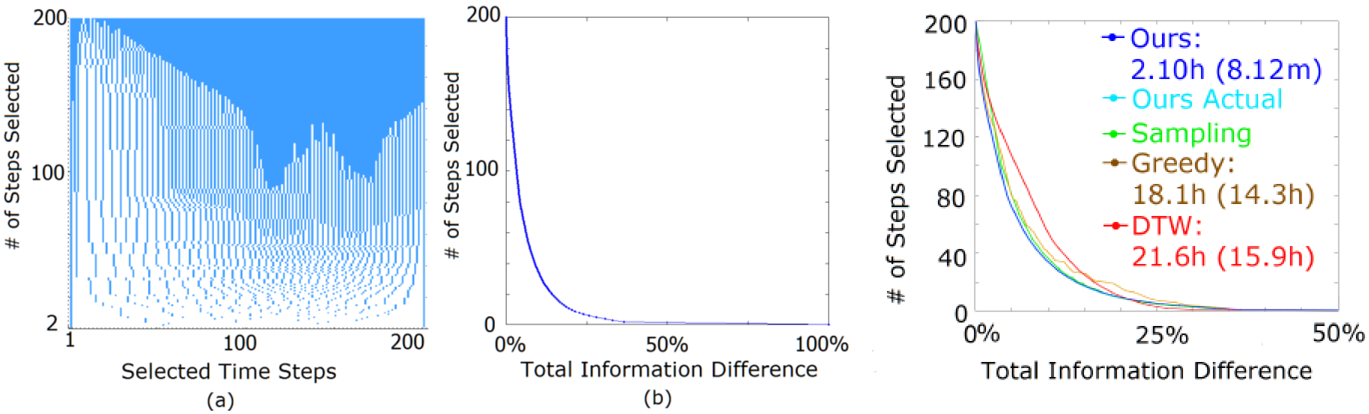


EVIS 2018 EURO

Results:

Compare:
• Ours
(t = 12)
• Uniform Sampling
• Greedy
• DTW [Tong *et al.* 12]





Results: Running Time Analysis

- In-Core Computation

+ Time for $c(i, j)$'s: $O(t^2TN) = O(TN)$ (t = 12) **Dominates!**
+ Time for DP: $O(T^3)$
+ Total in-core time: $O(t^2TN + T^3) = O(TN)$ **Linear in data size (>> I/O time!)**

- I/O: $O(TN/B)$ **Linear in data size**

Dataset	Size	T	Total (h)	I/O (m)	DP (s)
Radiation	27.4GB	200	2.1	8.1	0.026
Radiation2	54.8GB	400	4.5	17.8	0.19
Radiation4	109.6GB	800	9.4	36.6	1.35
Radiation8	219.2GB	1600	19.5	73.2	13.6

N: 37M, t = 12. Memory footprint: 1.91GB

Results:

Dataset Size	DTW	Our Method
27.4 GB	21.6 hours	2.1 hours
219.2 GB	> 1000 hours (estimated)	19.5 hours

Selection Quality of our method: very close to DTW (optimal)
Memory Footprints of our method: 1.91GB.
Running Time of our method: Significant Speed-up!

Conclusions

Our New Approches:

- Fully automatic; globally optimal qualities for the accurate method
- Provide a storyboard to guide data exploration
- Out-of-core approximate method:

+ optimal I/O
+ significant speed-up (1000+ hrs → < 20 hrs!)
+ close-to-optimal selection qualities

- Independent of the metrics used (InfoD proposed; RMSE also used)

Acknolwedgement:
DOE grant DE-SC0004874, program manager Lucy Nowell