Contour Forests: Fast Multi-threaded Augmented Contour Trees

Journée Visu 2017

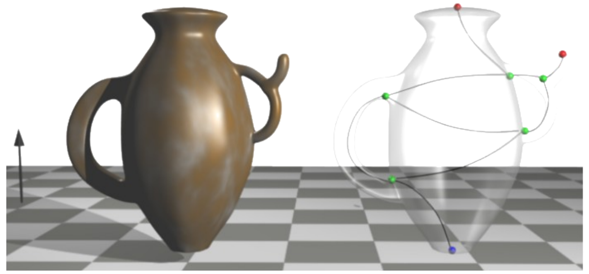
Charles Gueunet, UPMC and Kitware
Pierre Fortin, UPMC
Julien Jomier, Kitware
Julien Tierny, UPMC

VIS 2016

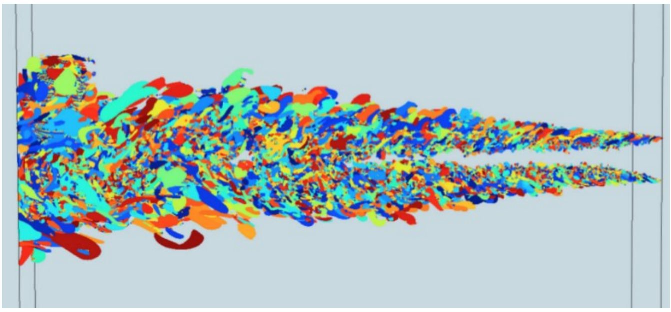
Introduction

- Context
- Related Works
- Challenges
- Contributions

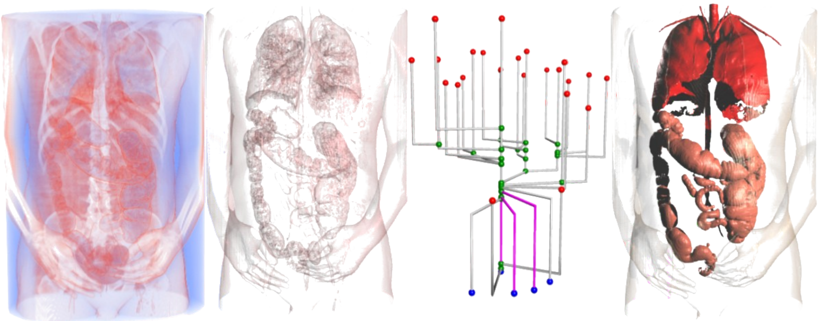
- Topological analysis: classical approach in Sci Viz
  - **Contour Trees**, Reeb Graphs, Morse-Smale Complexes...



[DoraiswamyTVCG13]



[LandgeSC14]



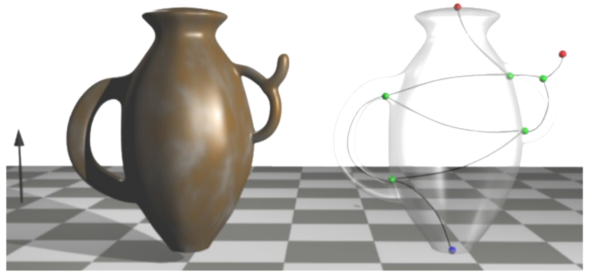
[MaadasamyHiPC12]

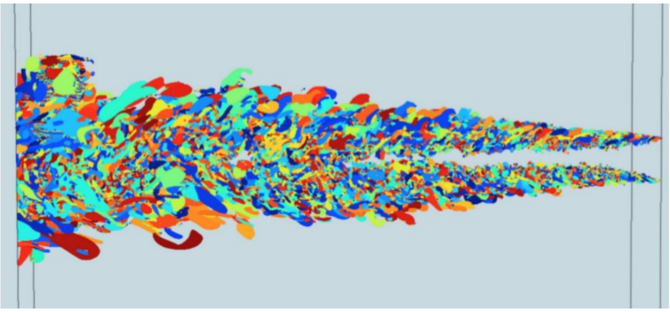
Introduction

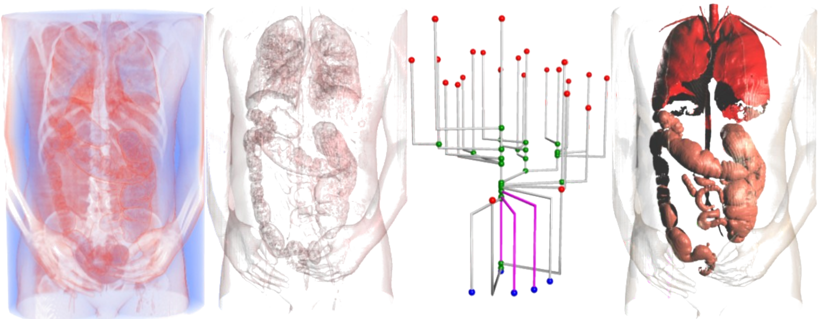
- Context
- Related Works
- Challenges
- Contributions

- Topological analysis: classical approach in Sci Viz
  - **Contour Trees**, Reeb Graphs, Morse-Smale Complexes...
- Increasing data size and complexity
  - Challenge for interactive exploration
  - Multi-core architectures are common

$\}$ Motivation for multi-threaded parallelism


[DoraiswamyTVCG13]


[LandgeSC14]


[MaadasamyHiPC12]

Introduction

- Context
- Related Works
- Challenges
- Contributions

Sequential:

- [CarrSODA00,ChiangCG05]

Introduction

- Context
- Related Works
- Challenges
- Contributions

Sequential:

- [CarrSODA00,ChiangCG05]

Ref	CT	Tet. mesh	Augmented	Combination
Pascucci	✓	~	✓	✗

Parallel:

- [PascucciAlgorithmica04]

Introduction

- Context
- Related Works
- Challenges
- Contributions

<u>Sequential</u>:

- [CarrSODA00,ChiangCG05]

Ref	CT	Tet. mesh	Augmented	Combination
Pascucci	✓	∼	✓	✗
Maadasamy	✓	✓	✗	✗

<u>Parallel</u>:

- [PascucciAlgorithmica04]
- [MaadasamyHiPC12]

Introduction

- Context
- Related Works
- Challenges
- Contributions

Sequential:

- [CarrSODA00,ChiangCG05]

Ref	CT	Tet. mesh	Augmented	Combination
Pascucci	✓	~	✓	✗
Maadasamy	✓	✓	✗	✗
Natarajan	✓	✗	✗	✗

Parallel:

- [PascucciAlgorithmica04]
- [MaadasamyHiPC12]
- [NatarajanPV15]

Introduction

- Context
- Related Works
- Challenges
- Contributions

Sequential:

- [CarrSODA00,ChiangCG05]

Ref	CT	Tet. mesh	Augmented	Combination
Pascucci	✓	∼	✓	✗
Maadasamy	✓	✓	✗	✗
Natarajan	✓	✗	✗	✗
Morozov MT	✗	∼	✓	-

Parallel:

- [PascucciAlgorithmica04]
- [MaadasamyHiPC12]
- [NatarajanPV15]

Distributed:

- [MorozovPPoPP13]

Introduction

- Context
- Related Works
- Challenges
- Contributions

Sequential:

- [CarrSODA00,ChiangCG05]

Ref	CT	Tet. mesh	Augmented	Combination
Pascucci	✓	∼	✓	✗
Maadasamy	✓	✓	✗	✗
Natarajan	✓	✗	✗	✗
Morozov MT	✗	∼	✓	-
Morozov CT	✓	∼	✓	✗

Parallel:

- [PascucciAlgorithmica04]
- [MaadasamyHiPC12]
- [NatarajanPV15]

Distributed:

- [MorozovPPoPP13]
- [MorozovTopoInVis13]

Introduction

- Context
- Related Works
- Challenges
- Contributions

Sequential:

- [CarrSODA00,ChiangCG05]

Ref	CT	Tet. mesh	Augmented	Combination
Pascucci	✓	∼	✓	✗
Maadasamy	✓	✓	✗	✗
Natarajan	✓	✗	✗	✗
Morozov MT	✗	∼	✓	-
Morozov CT	✓	∼	✓	✗
Landge	✗	✗	✓	-

Parallel:

- [PascucciAlgorithmica04]
- [MaadasamyHiPC12]
- [NatarajanPV15]

Distributed:

- [MorozovPPoPP13]
- [MorozovTopoInVis13]
- [LandgeTopoInVis15]

Introduction

- Context
- Related Works
- Challenges
- Contributions

- Topological data analysis algorithms
  - Intrinsically sequential approaches
  - Challenging parallelization

Introduction

- Context
- Related Works
- Challenges
- Contributions

- Topological data analysis algorithms
  - Intrinsically sequential approaches
  - Challenging parallelization

- Contour Tree:
  - No complete parallelization (only subroutines)
  - No efficient parallel algorithm for augmented trees

Introduction

- Context
- Related Works
- Challenges
- Contributions

- Efficient multi-threaded algorithm for contour tree computation
  - Simple approach
  - Good parallel efficiency on workstations

Introduction

- Context
- Related Works
- Challenges
- Contributions

- Efficient multi-threaded algorithm for contour tree computation
  - Simple approach
  - Good parallel efficiency on workstations

- Ready-to-use VTK-based C++ implementation: in TTK
  - Generic input (VTU/VTI, 2D/3D)
  - Generic output (augmented trees)

Summary

1) Introduction

2) Preliminaries
- Background
- Overview

3) Algorithm
- Domain partitioning
- Local computation
- Contour forest stitching

4) Experimental results
- Scalability
- Efficiency
- Limitations

5) Application

6) Conclusion
- Recall
- Perspective

Preliminaries

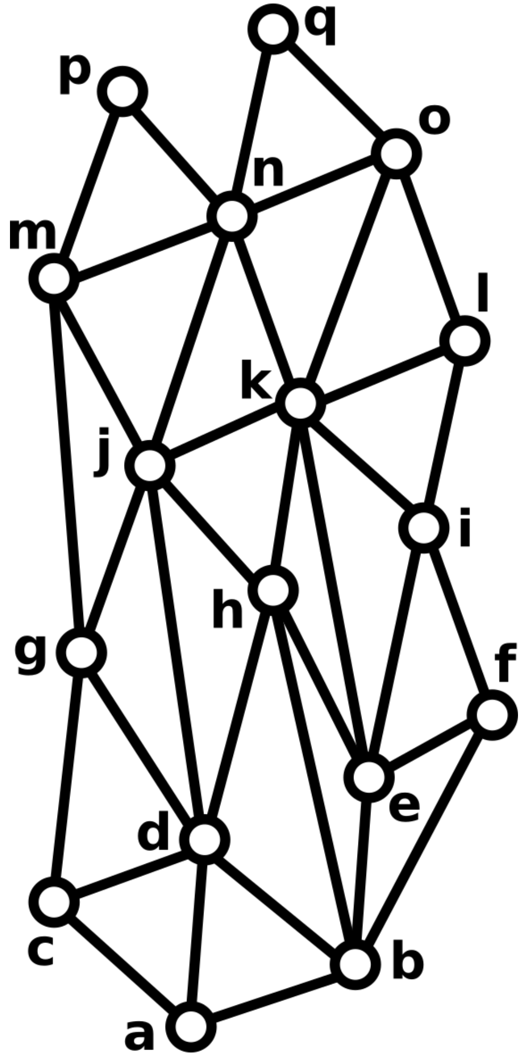
Preliminaries

- Background
- Overview

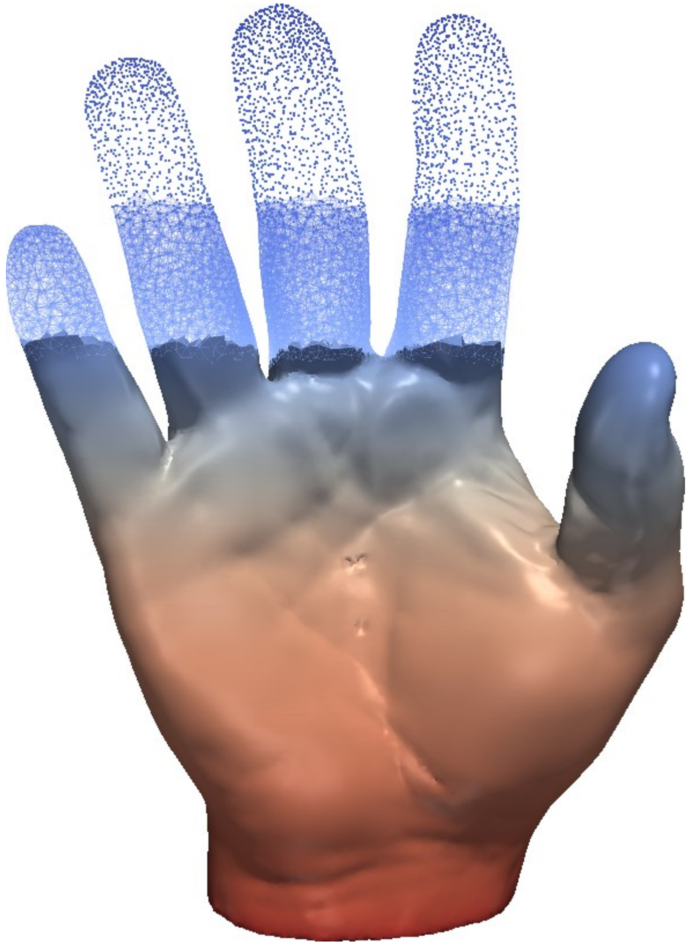
Input data:

- Piecewise linear scalar field

$$f : \mathcal{M} \to \mathbb{R}$$



2D Example



3D Example

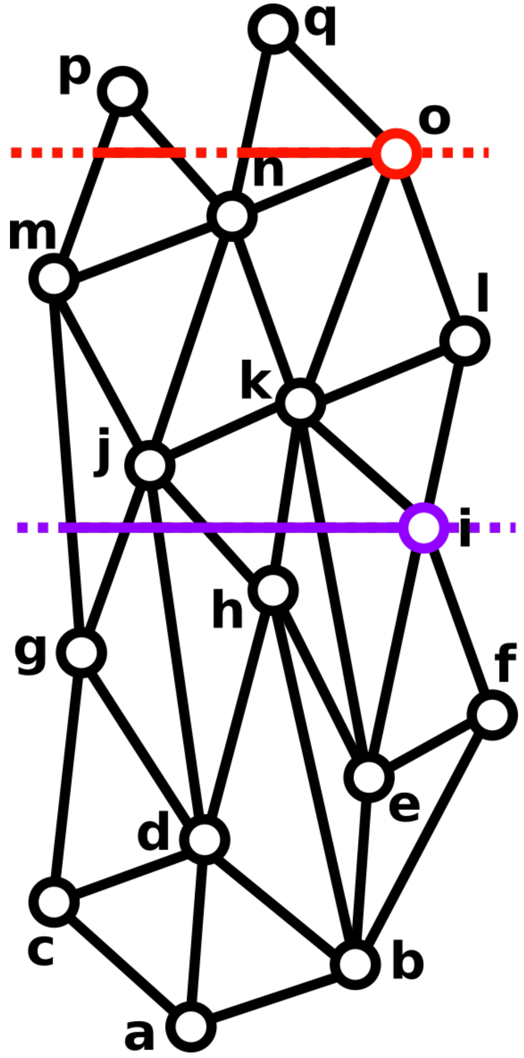
Preliminaries

- Background
- Overview

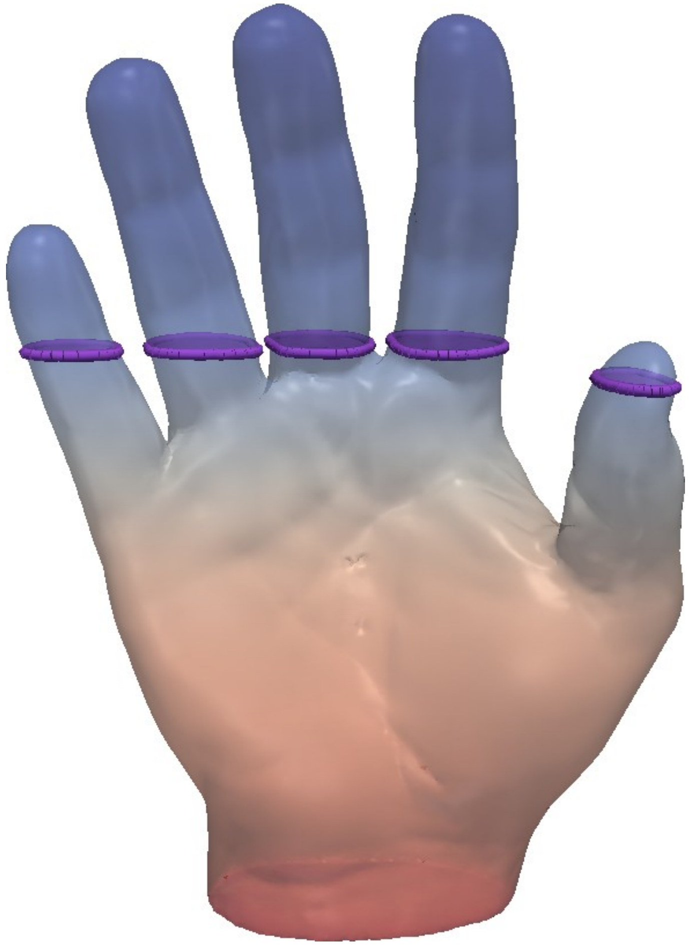
Level set:

- Preimage of a scalar value
- Isovalue: $i \in \mathbb{R}$ onto $\mathcal{M}$

$$f^{-1}(i) = \{p \in \mathcal{M} | f(p) = i\}$$



2D Example



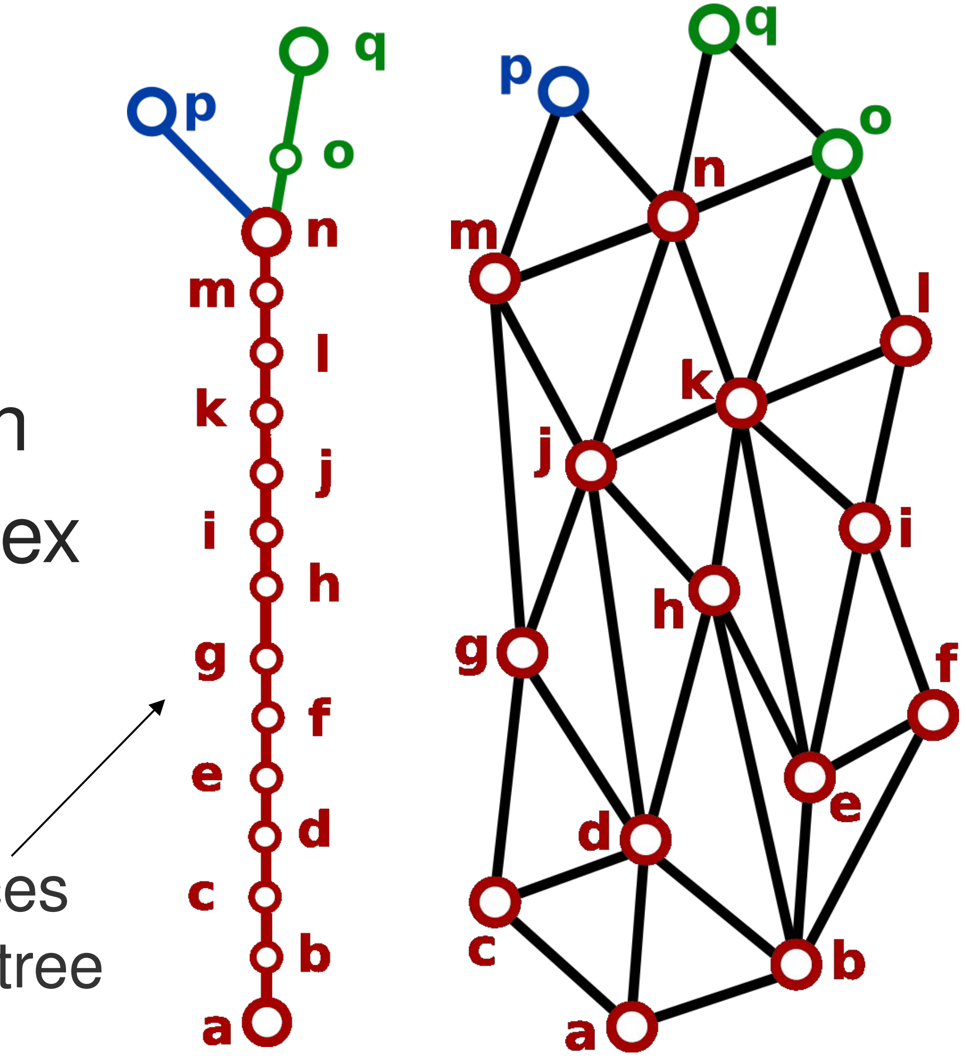
3D Example

Preliminaries

- Background
- Overview

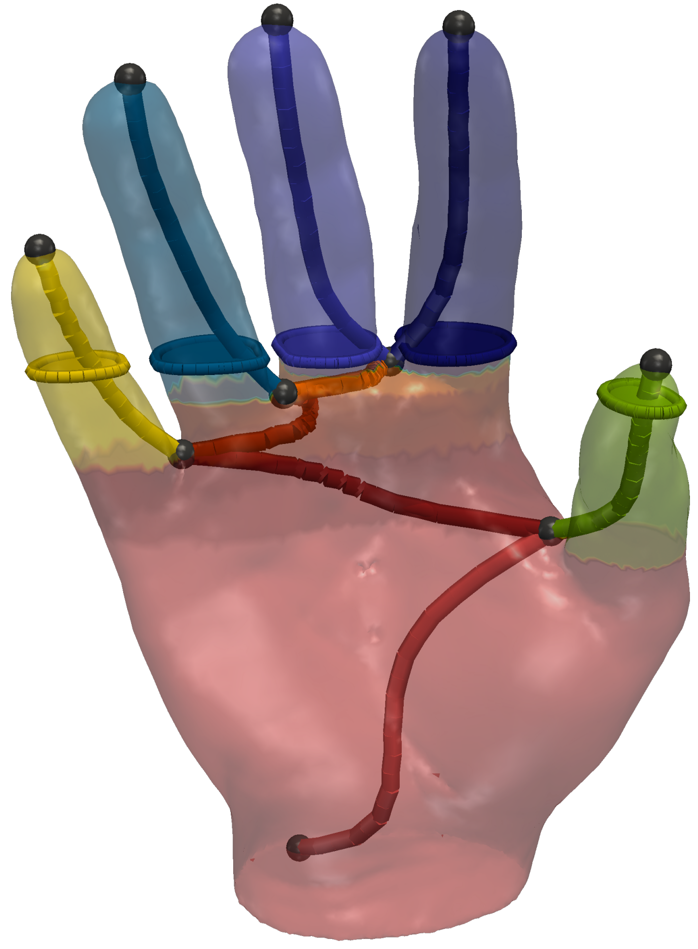
Contour tree:

- Simply connected input domain
- 1-dimensional simplicial complex



Regular vertices in augmented tree

2D Example



3D Example

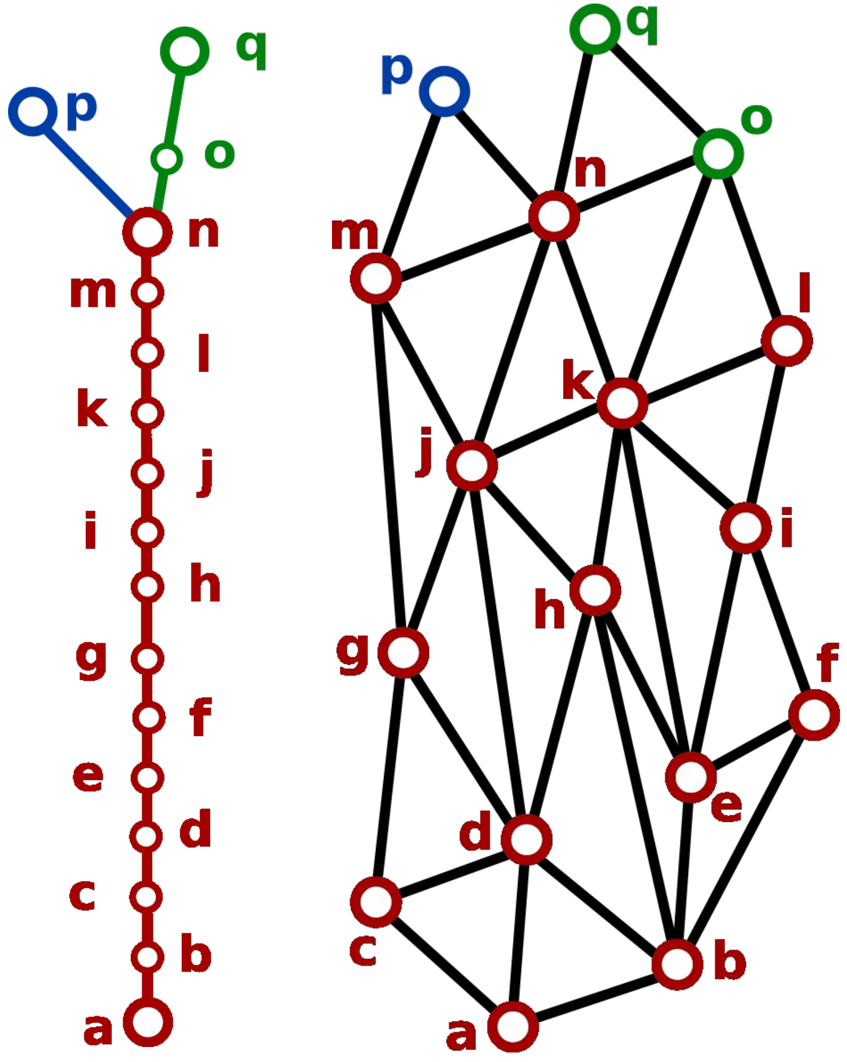
Preliminaries

- Background
- Overview

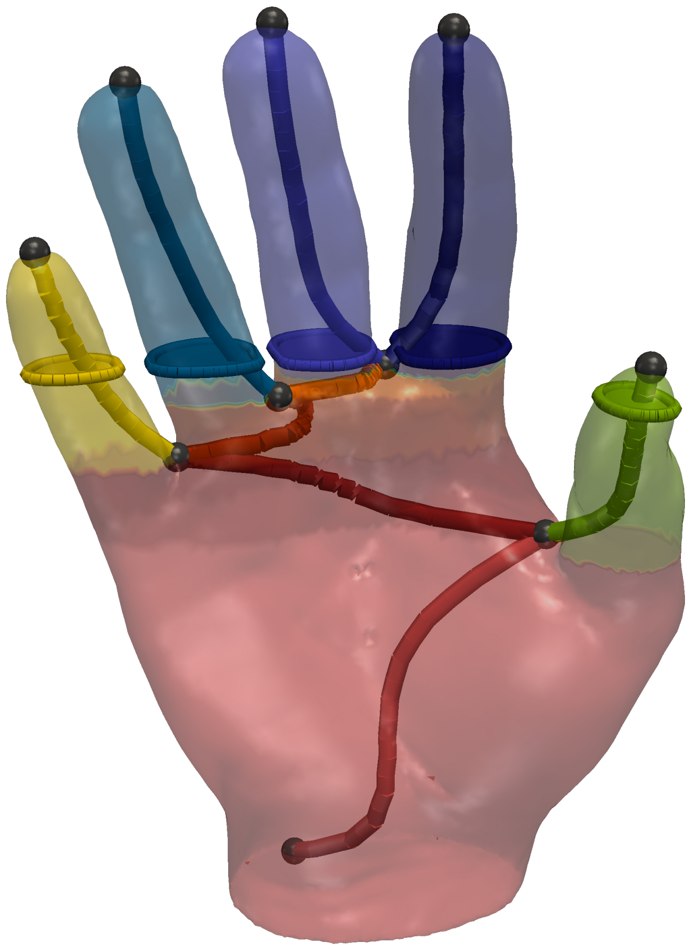
Contour tree:

- Quotient space: $\mathcal{C}(f) = \mathcal{M}/\sim$
- Equivalence relation: $p_1 \sim p_2$

$$\begin{cases} f(p_1) = f(p_2) \\ p_2 \in f^{-1}(f(p_1))_{p_1} \end{cases}$$



2D Example



3D Example

Preliminaries

- Background
- Overview

<u>Main steps of our approach</u>:

1) Sort vertices
2) Create partitions using level sets
3) Compute local trees
4) Stitch local trees

Preliminaries

Main steps of our approach:

1) Sort vertices
2) Create partitions using level sets
3) Compute local trees
4) Stitch local trees

Pros:

- Simple approach
- Local computation of CT
- Augmented trees
- Simple stitching of the forests

Preliminaries

- Background
- Overview

Main steps of our approach:

1) Sort vertices
2) Create partitions using level sets
3) Compute local trees
4) Stitch local trees

Pros:

- Simple approach
- Local computation of CT
- Augmented trees
- Simple stitching of the forests

Cons:

- Depends on interface level sets

Algorithm

Algorithm

- Domain partitioning
- Local computation
- Contour forest stitching

- Range driven partitions
- Sorted scalars ⇒ balanced partitions
- Use $n_t/2$ partitions

Algorithm

- Domain partitioning
- Local computation
- Contour forest stitching

- Range driven partitions
- Sorted scalars ⇒ balanced partitions
- Use $n_t/2$ partitions

$$f(\mathcal{M}) = \mathcal{I}_0 \cup \mathcal{I}_1 \cup \cdots \cup \mathcal{I}_{(n_t/2)-1}$$

$$|\sigma_0|_i \approx |\sigma_0|_j \quad \forall i \neq j$$

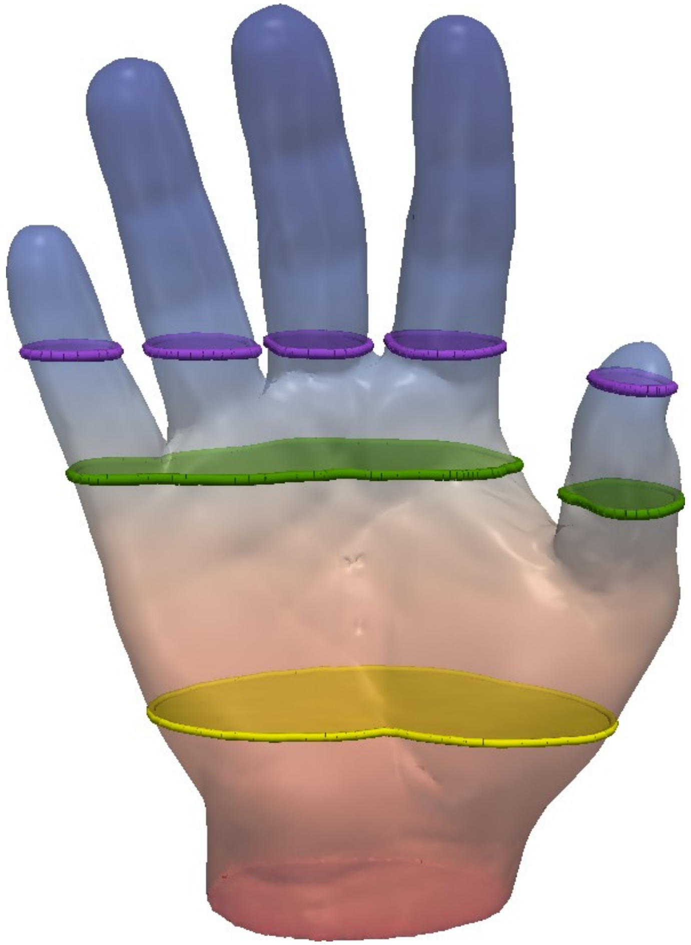
Algorithm

- Domain partitioning
- Local computation
- Contour forest stitching

- Range driven partitions
- Sorted scalars ⇒ balanced partitions
- Use $n_t/2$ partitions

$$f(\mathcal{M}) = \mathcal{I}_0 \cup \mathcal{I}_1 \cup \cdots \cup \mathcal{I}_{(n_t/2)-1}$$

$$|\sigma_0|_i \approx |\sigma_0|_j \quad \forall i \neq j$$



3D Example

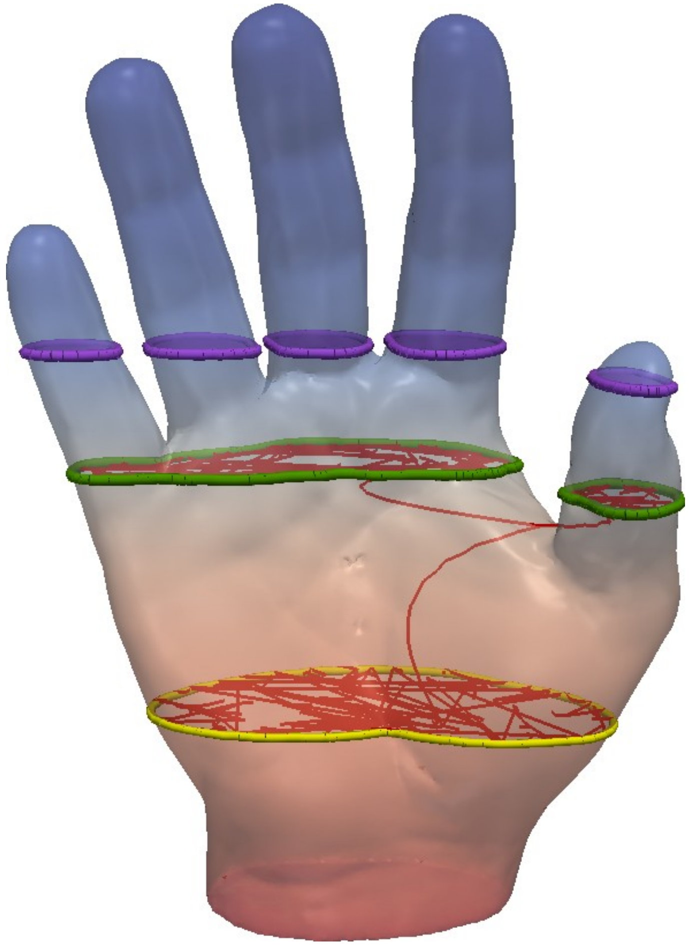
Algorithm

- Domain partitioning
- Local computation
- Contour forest stitching

- Range driven partitions
- Sorted scalars ⇒ balanced partitions
- Use $n_t/2$ partitions

$$f(\mathcal{M}) = \mathcal{I}_0 \cup \mathcal{I}_1 \cup \cdots \cup \mathcal{I}_{(n_t/2)-1}$$

$$|\sigma_0|_i \approx |\sigma_0|_j \quad \forall i \neq j$$



3D Example

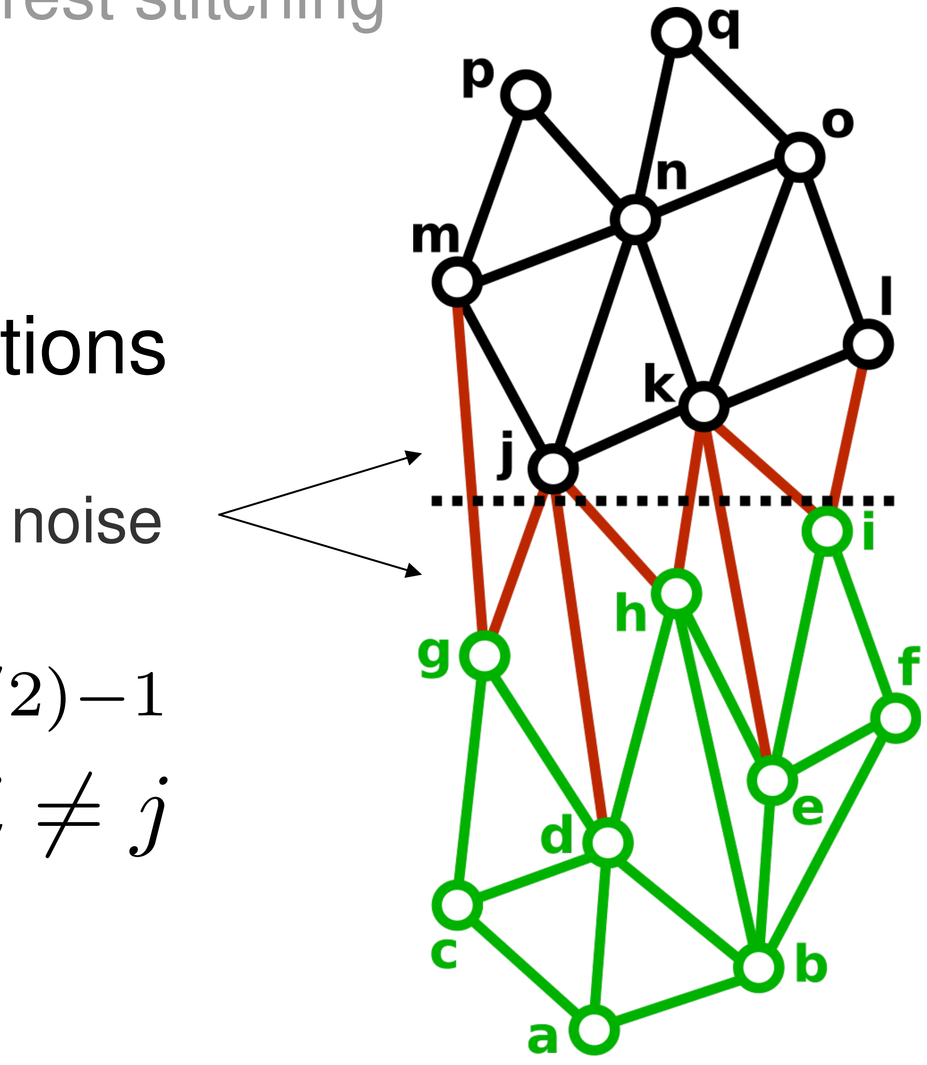
Algorithm

- Domain partitioning
- Local computation
- Contour forest stitching

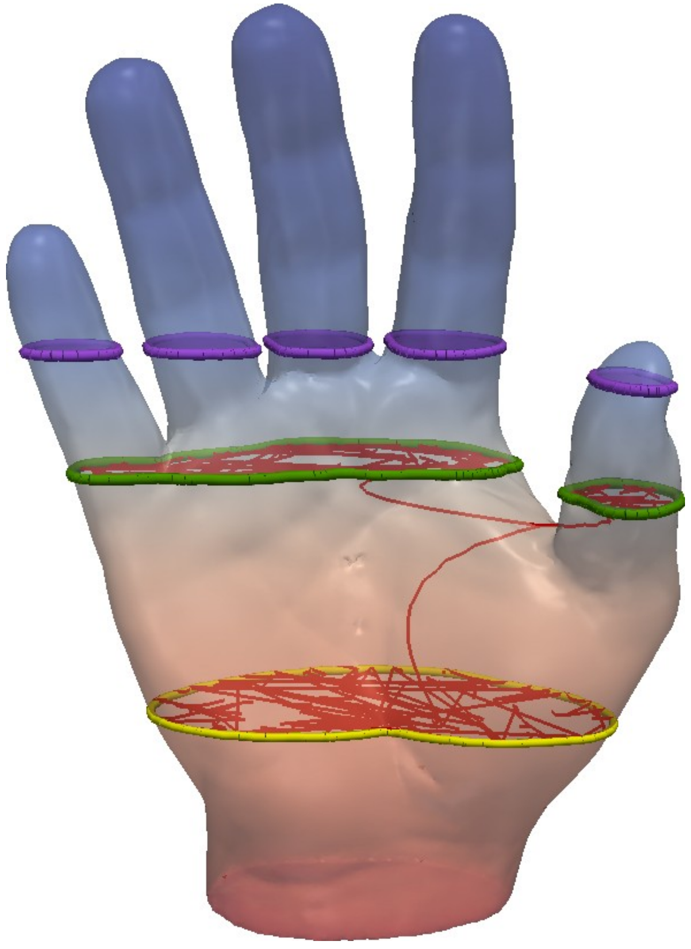
- Range driven partitions
- Sorted scalars ⇒ balanced partitions
- Use $n_t/2$ partitions

$$f(\mathcal{M}) = \mathcal{I}_0 \cup \mathcal{I}_1 \cup \cdots \cup \mathcal{I}_{(n_t/2)-1}$$

$$|\sigma_0|_i \approx |\sigma_0|_j \quad \forall i \neq j$$



2D Example



3D Example

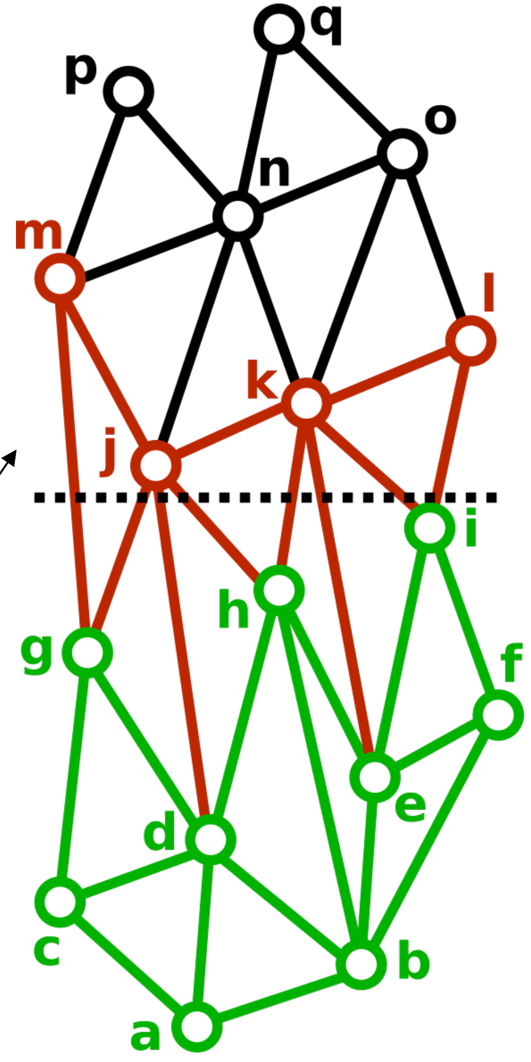
Algorithm

- Domain partitioning
- Local computation
- Contour forest stitching

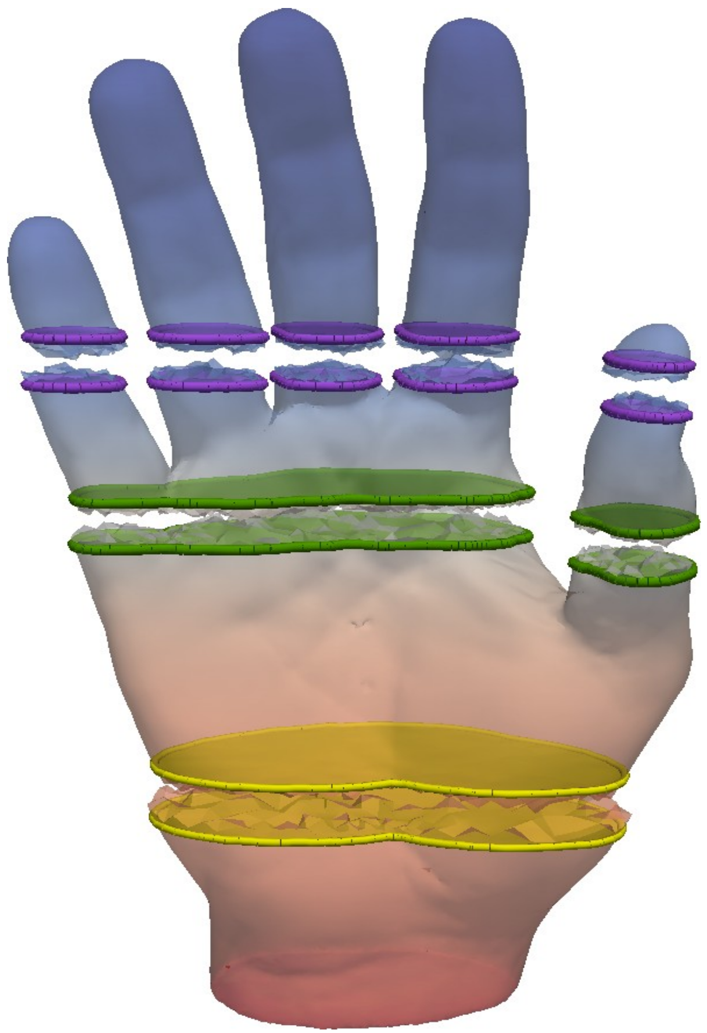
Redundant computations:

- Extended partition:
  initial partition + boundary vertices
- Correct tree in initial partition
- Visit all edges in parallel

extended
green + red



2D Example



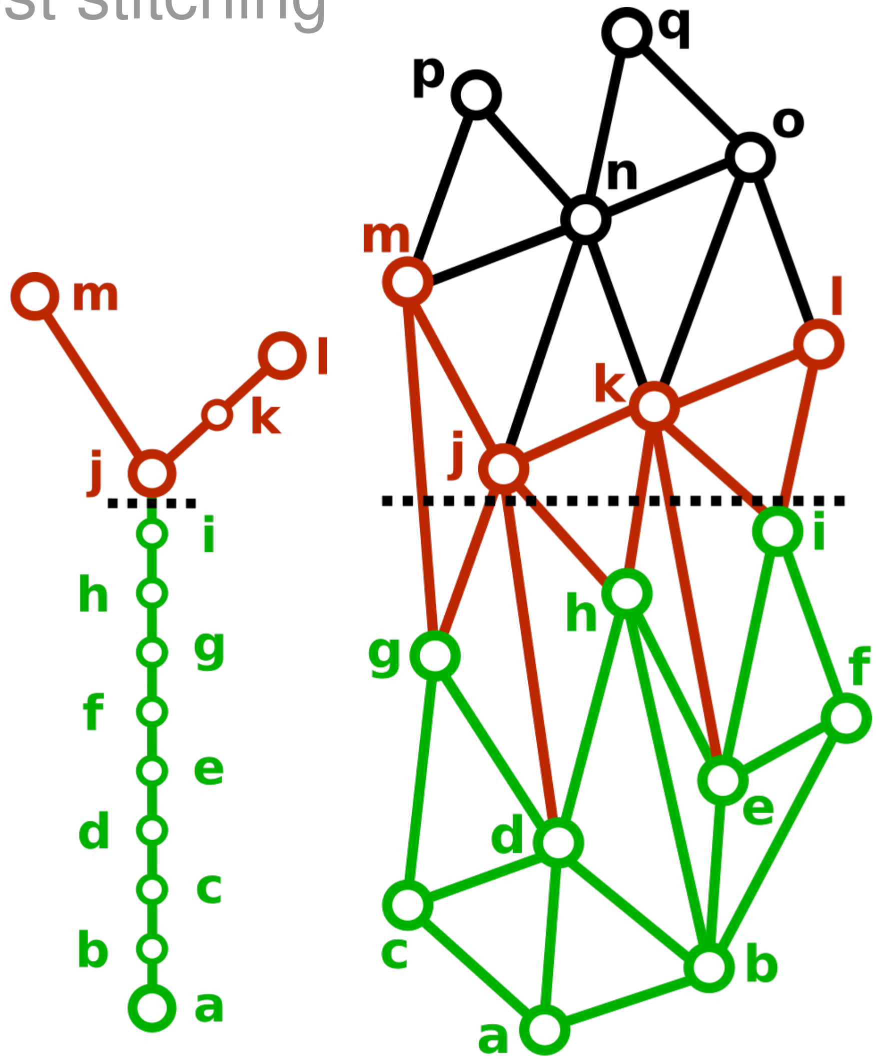
3D Example

Algorithm

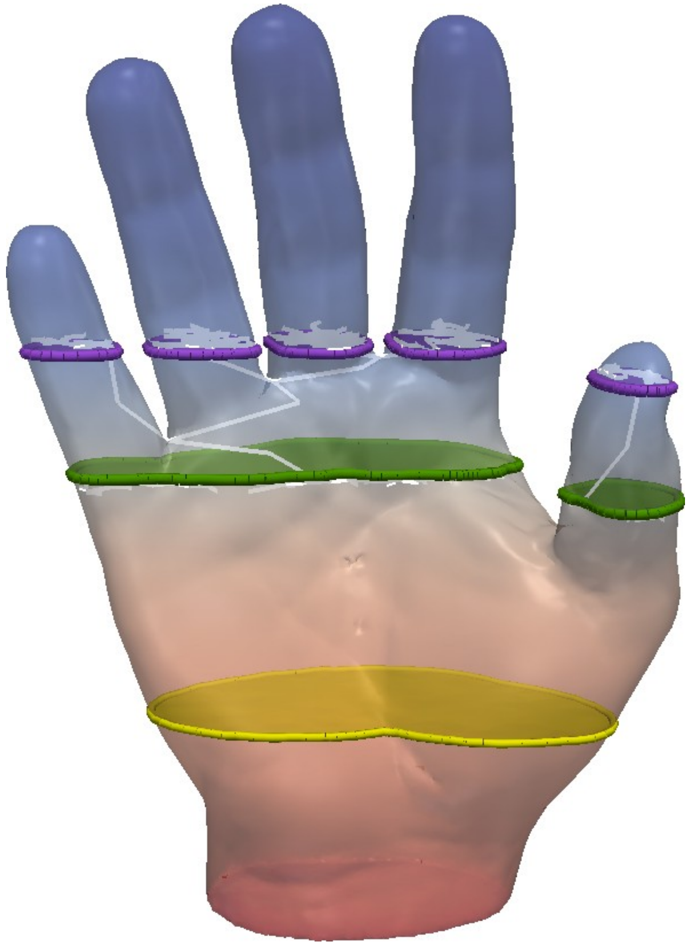
- Domain partitioning
- Local computation
- Contour forest stitching

- In extended partitions
- Using Carr *et al.* Algorithm:
  - Join Tree
  - Split Tree
  - Combination
- Union-Find [CormenItA01]

Independent



2D Example



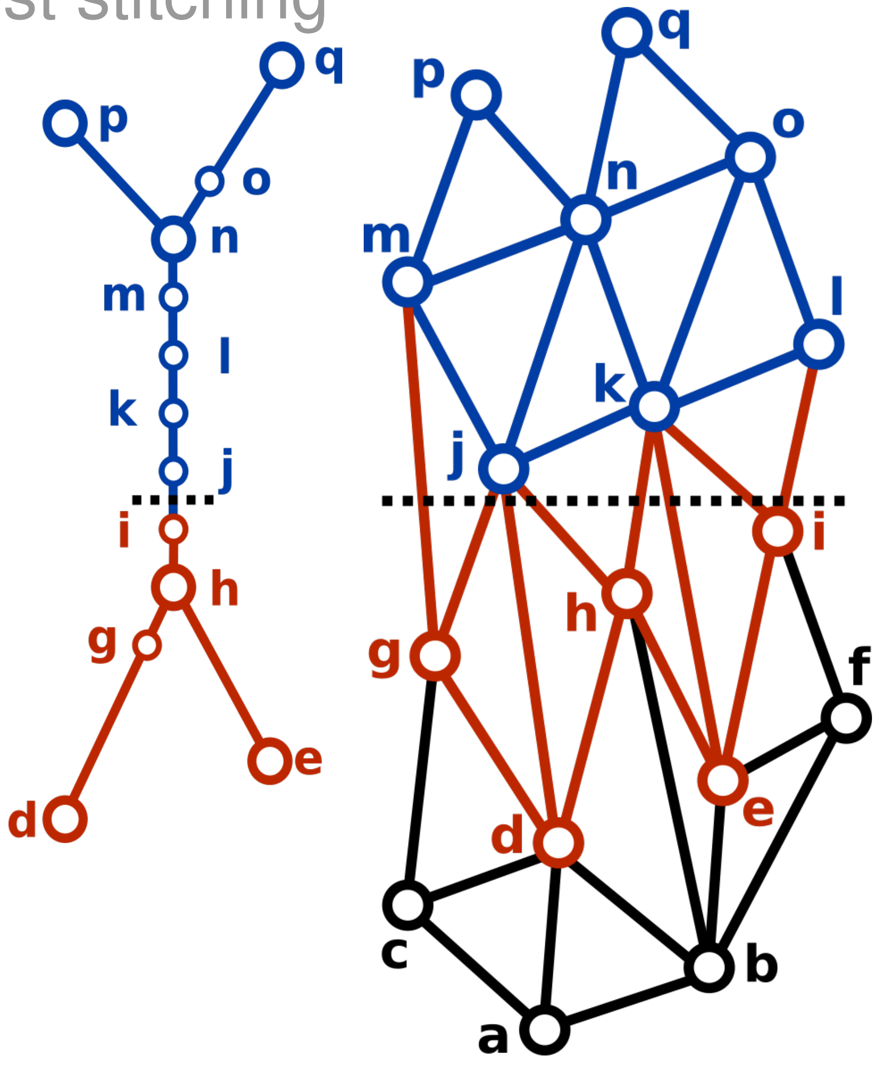
3D Example

Algorithm

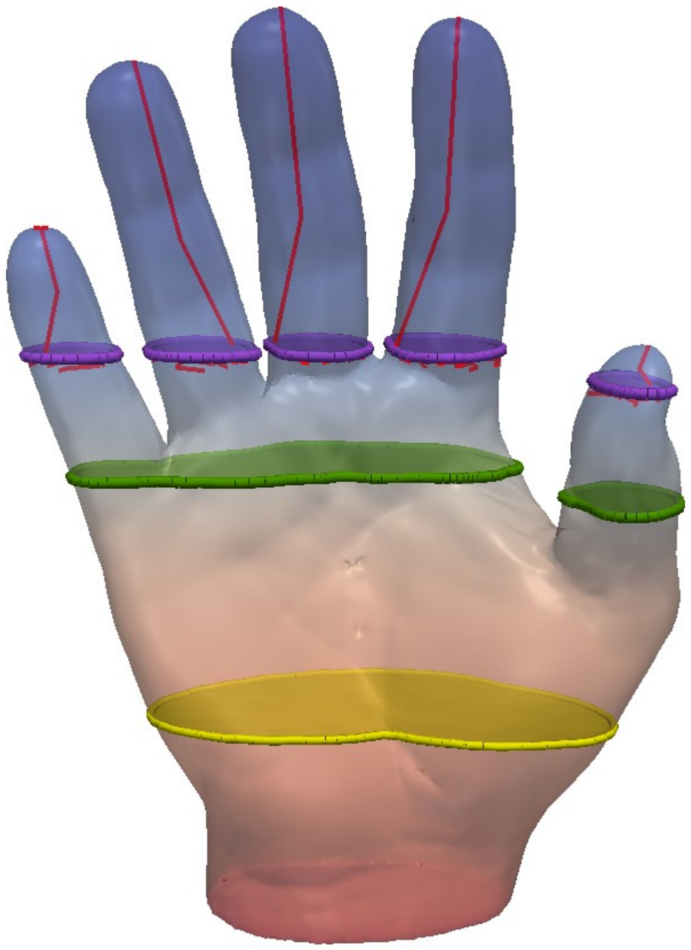
- Domain partitioning
- Local computation
- Contour forest stitching

Each partition:

- Join and Split trees: 2 threads
- Combine: 1 thread
- $O(|\sigma^i| \times \alpha(|\sigma^i|)) + O(|\mathcal{C}(f)|)$
- Keep arcs on the boundary (n-h)



2D Example

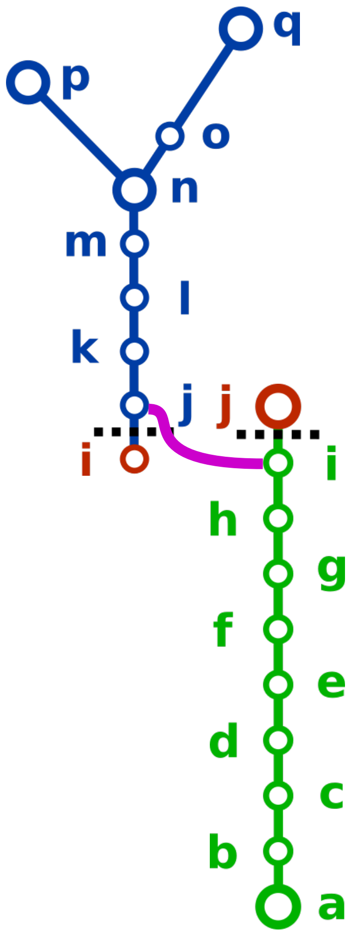


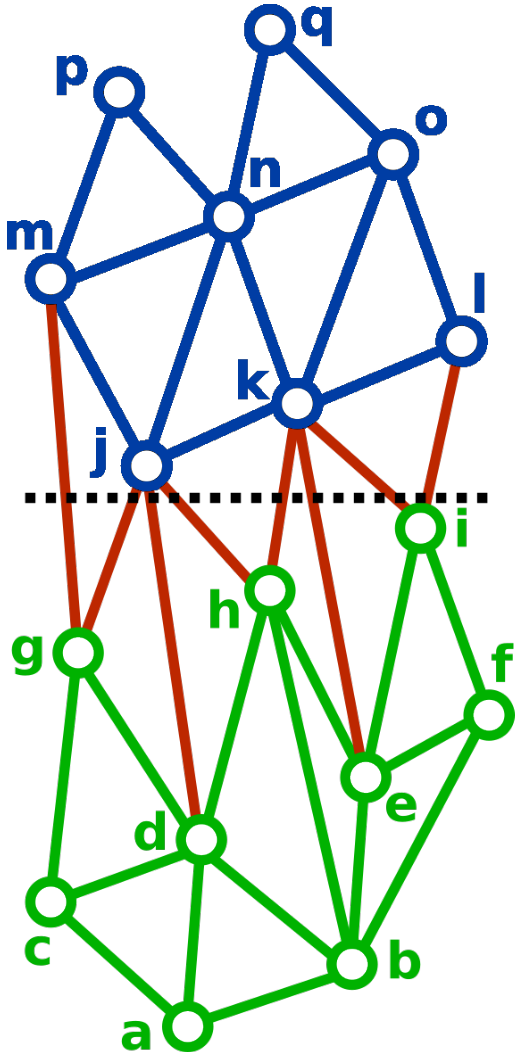
3D Example

Algorithm

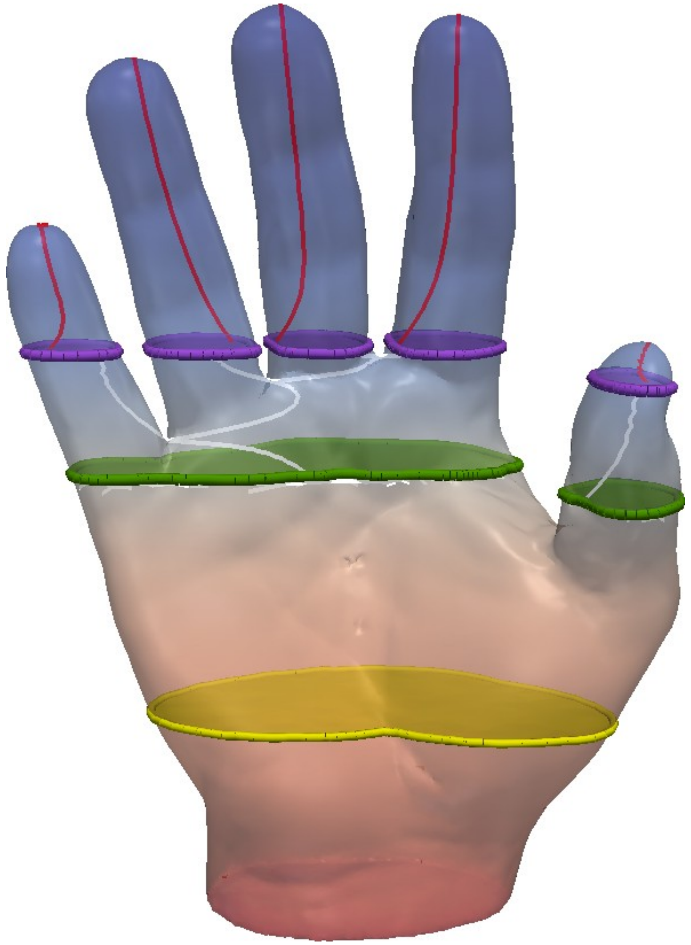
- Domain partitioning
- Local computation
- Contour forest stitching

- Simple step
- Visit crossing arcs
- Cut *interface* arcs
  - Segmentation: fast lookup





2D Example

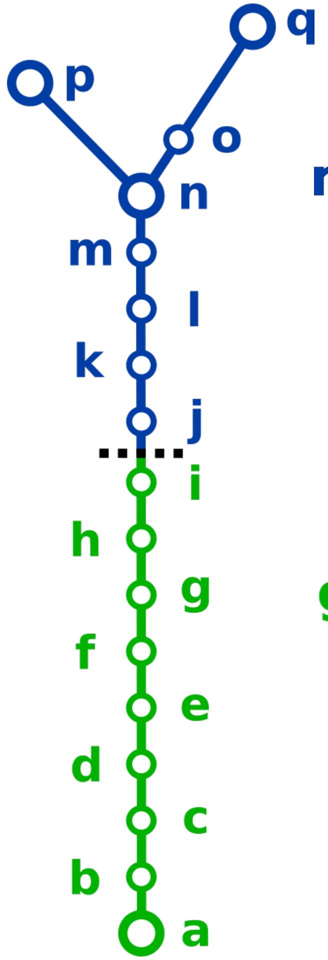


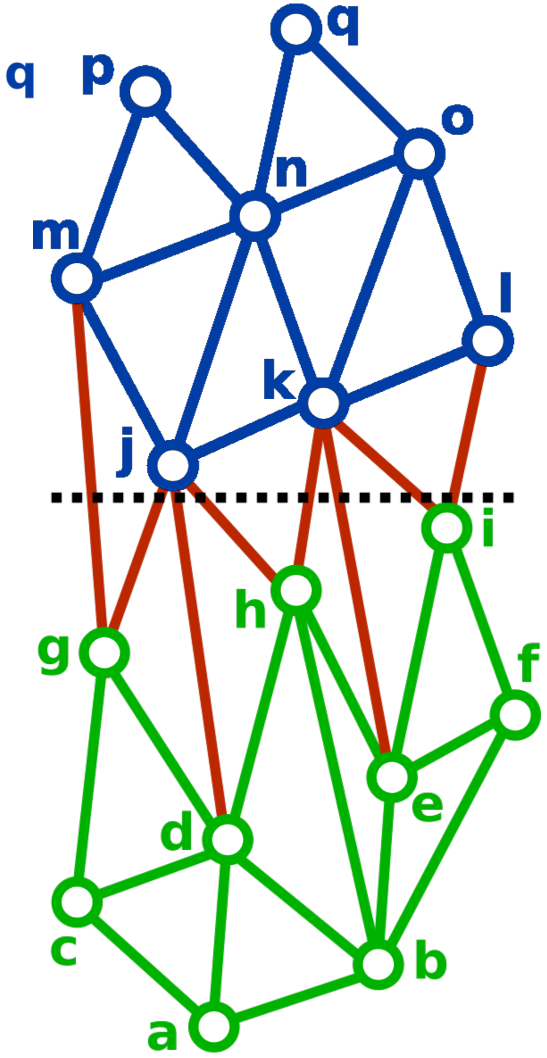
3D Example

Algorithm

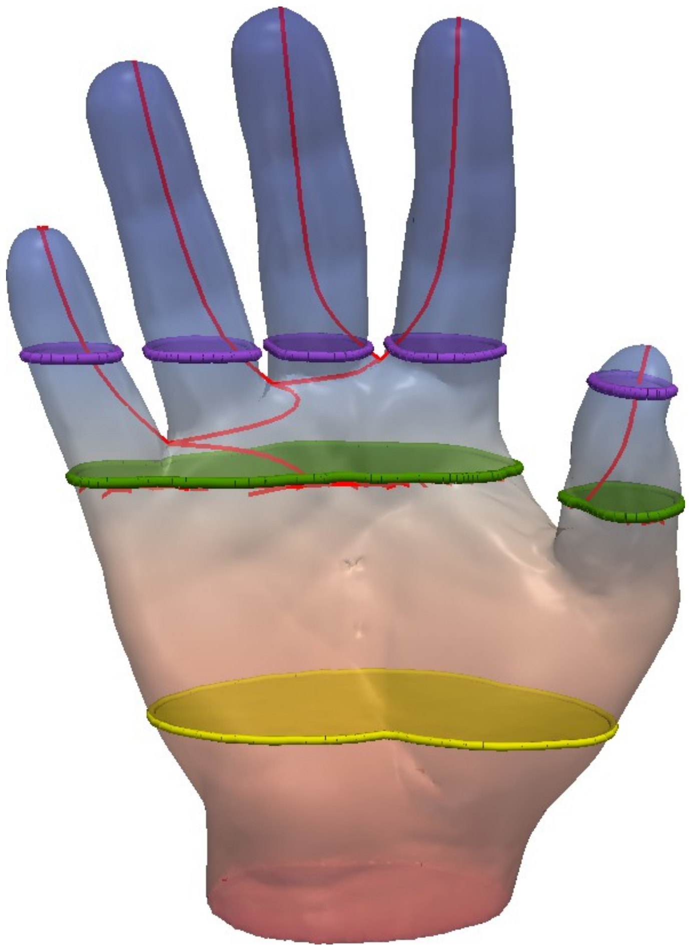
- Domain partitioning
- Local computation
- Contour forest stitching

- New super-arc:
  - Union of the two facing halves
- Once per connected component of level set (~ crossing arc)





2D Example

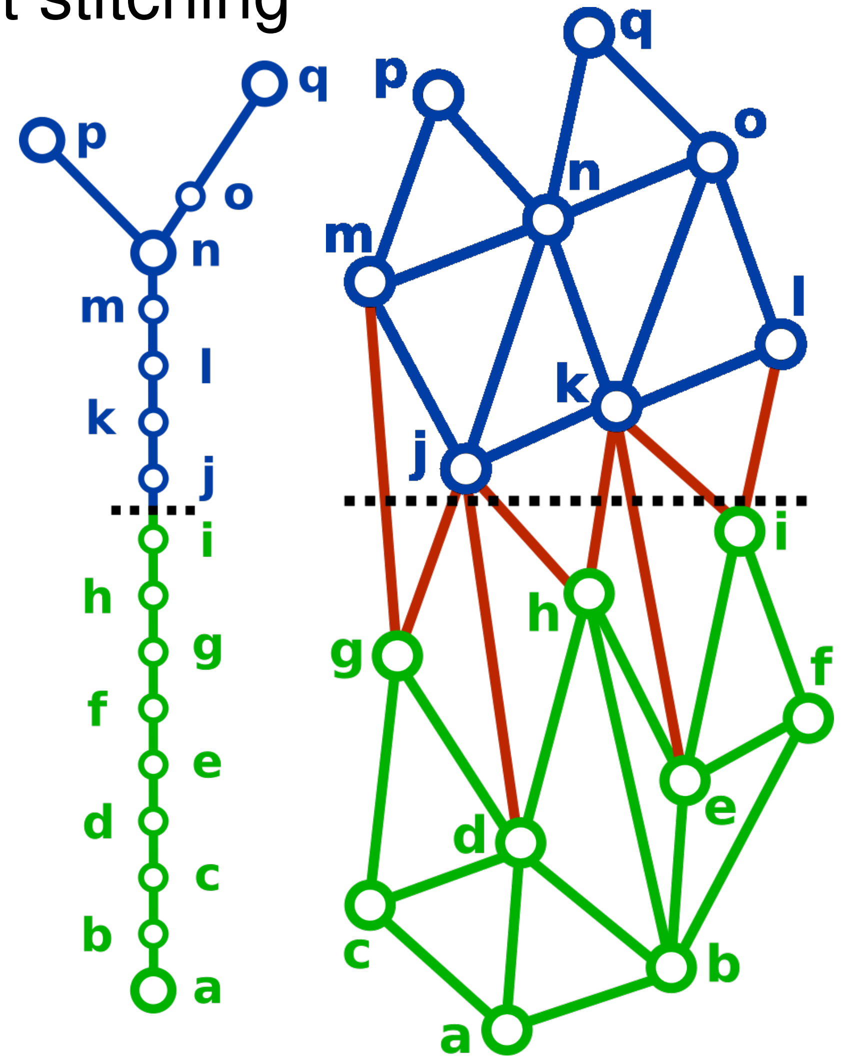


3D Example

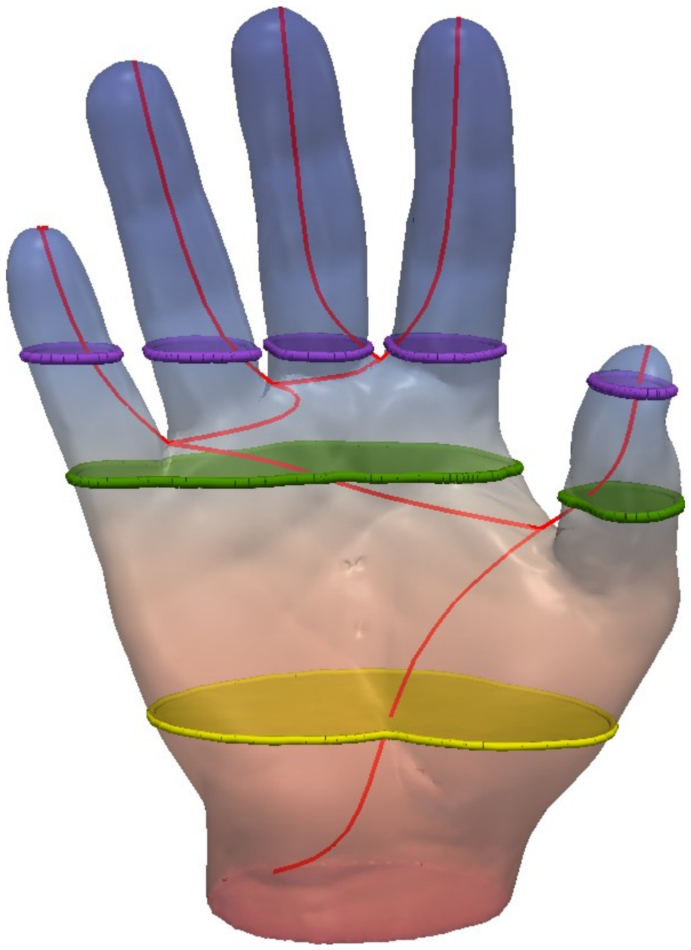
Algorithm

- Domain partitioning
- Local computation
- Contour forest stitching

- Repeat for each interface
- Fast in practice



2D Example



3D Example

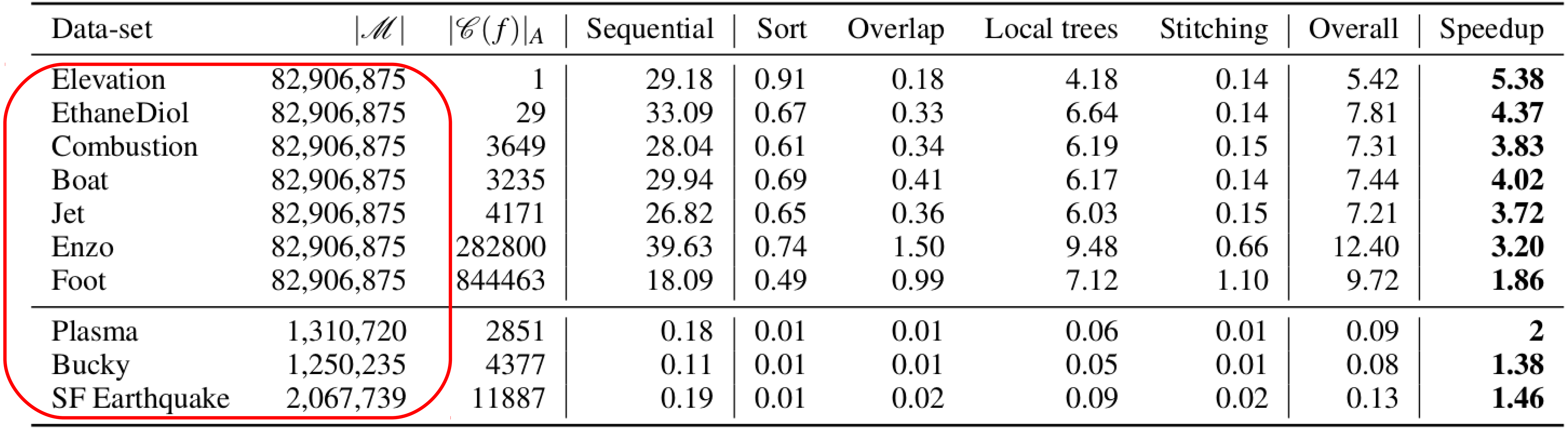
Experimental results

Intel Xeon CPU E5-2630 v3 (2.4 Ghz, 8 cores)
64GB of RAM
C++ (GCC-4.9), VTK, OpenMP (additional material)

Exp. Results

- Efficiency
- Comparison
- Speed
- Pros & cons

8 threads, 4 partitions



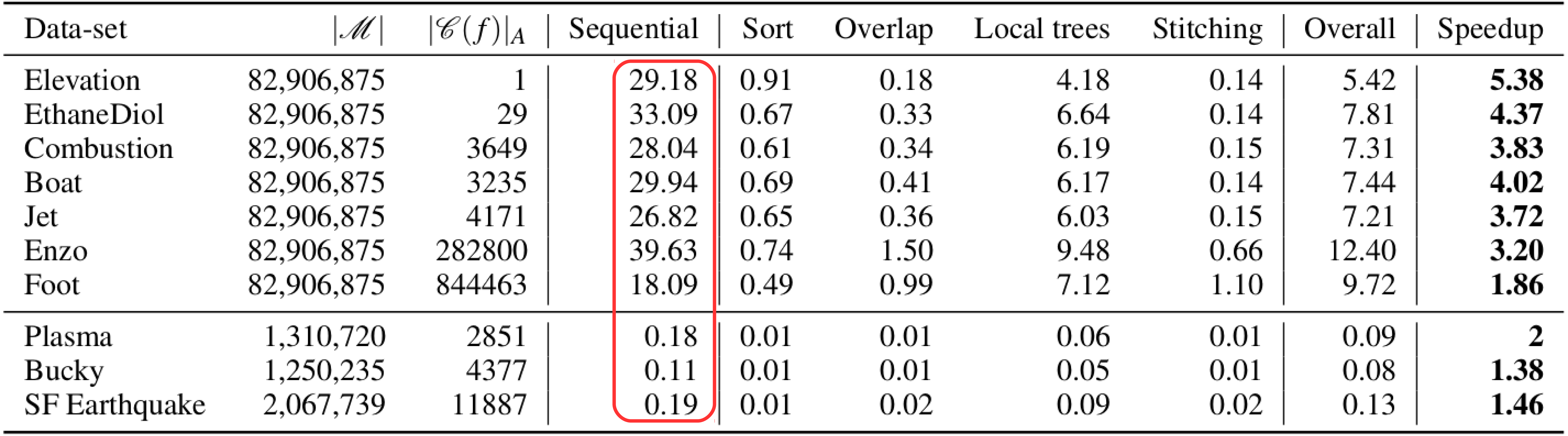
Data-set	$\lvert\mathscr{M}\rvert$	$\lvert\mathscr{C}(f)\rvert_A$	Sequential	Sort	Overlap	Local trees	Stitching	Overall	Speedup
Elevation	82,906,875	1	29.18	0.91	0.18	4.18	0.14	5.42	**5.38**
EthaneDiol	82,906,875	29	33.09	0.67	0.33	6.64	0.14	7.81	**4.37**
Combustion	82,906,875	3649	28.04	0.61	0.34	6.19	0.15	7.31	**3.83**
Boat	82,906,875	3235	29.94	0.69	0.41	6.17	0.14	7.44	**4.02**
Jet	82,906,875	4171	26.82	0.65	0.36	6.03	0.15	7.21	**3.72**
Enzo	82,906,875	282800	39.63	0.74	1.50	9.48	0.66	12.40	**3.20**
Foot	82,906,875	844463	18.09	0.49	0.99	7.12	1.10	9.72	**1.86**
Plasma	1,310,720	2851	0.18	0.01	0.01	0.06	0.01	0.09	**2**
Bucky	1,250,235	4377	0.11	0.01	0.01	0.05	0.01	0.08	**1.38**
SF Earthquake	2,067,739	11887	0.19	0.01	0.02	0.09	0.02	0.13	**1.46**

- Uniform sampling

Exp. Results

- Efficiency
- Comparison
- Speed
- Pros & cons

8 threads, 4 partitions



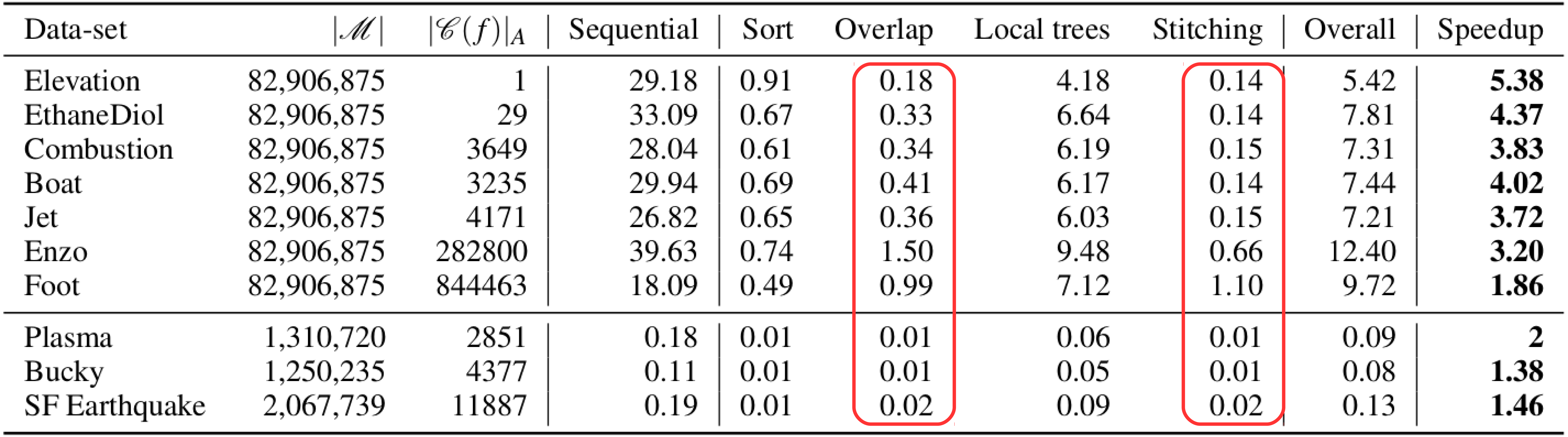
Data-set	$\lvert\mathscr{M}\rvert$	$\lvert\mathscr{C}(f)\rvert_A$	Sequential	Sort	Overlap	Local trees	Stitching	Overall	Speedup
Elevation	82,906,875	1	29.18	0.91	0.18	4.18	0.14	5.42	**5.38**
EthaneDiol	82,906,875	29	33.09	0.67	0.33	6.64	0.14	7.81	**4.37**
Combustion	82,906,875	3649	28.04	0.61	0.34	6.19	0.15	7.31	**3.83**
Boat	82,906,875	3235	29.94	0.69	0.41	6.17	0.14	7.44	**4.02**
Jet	82,906,875	4171	26.82	0.65	0.36	6.03	0.15	7.21	**3.72**
Enzo	82,906,875	282800	39.63	0.74	1.50	9.48	0.66	12.40	**3.20**
Foot	82,906,875	844463	18.09	0.49	0.99	7.12	1.10	9.72	**1.86**
Plasma	1,310,720	2851	0.18	0.01	0.01	0.06	0.01	0.09	**2**
Bucky	1,250,235	4377	0.11	0.01	0.01	0.05	0.01	0.08	**1.38**
SF Earthquake	2,067,739	11887	0.19	0.01	0.02	0.09	0.02	0.13	**1.46**

- Foot: fast computation

Exp. Results

- Efficiency
- Comparison
- Speed
- Pros & cons

8 threads, 4 partitions



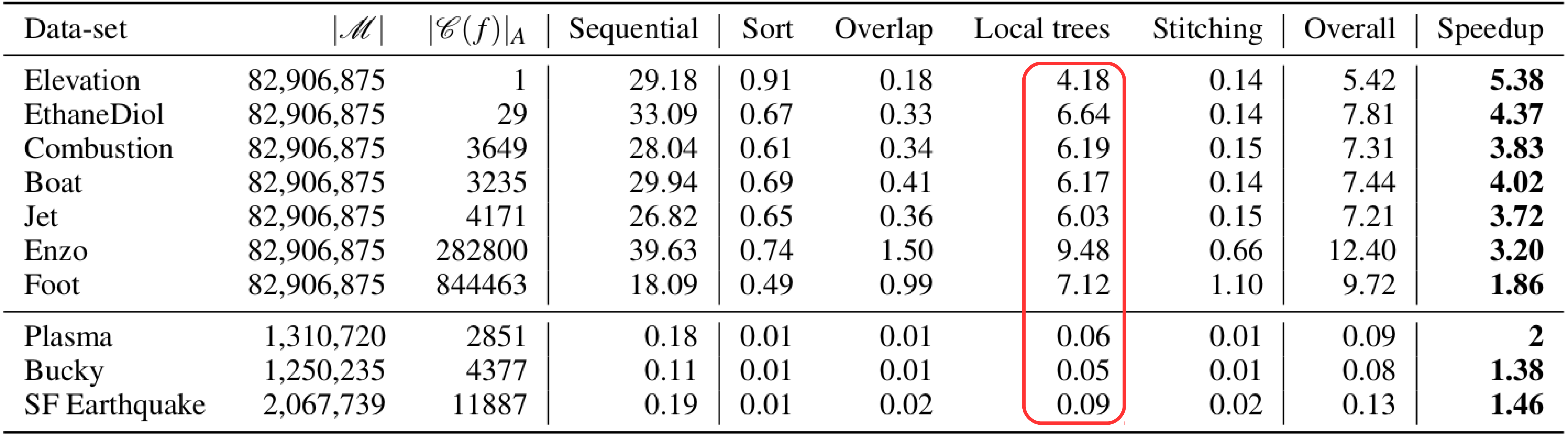
Data-set	$\lvert\mathscr{M}\rvert$	$\lvert\mathscr{C}(f)\rvert_A$	Sequential	Sort	Overlap	Local trees	Stitching	Overall	Speedup
Elevation	82,906,875	1	29.18	0.91	0.18	4.18	0.14	5.42	**5.38**
EthaneDiol	82,906,875	29	33.09	0.67	0.33	6.64	0.14	7.81	**4.37**
Combustion	82,906,875	3649	28.04	0.61	0.34	6.19	0.15	7.31	**3.83**
Boat	82,906,875	3235	29.94	0.69	0.41	6.17	0.14	7.44	**4.02**
Jet	82,906,875	4171	26.82	0.65	0.36	6.03	0.15	7.21	**3.72**
Enzo	82,906,875	282800	39.63	0.74	1.50	9.48	0.66	12.40	**3.20**
Foot	82,906,875	844463	18.09	0.49	0.99	7.12	1.10	9.72	**1.86**
Plasma	1,310,720	2851	0.18	0.01	0.01	0.06	0.01	0.09	**2**
Bucky	1,250,235	4377	0.11	0.01	0.01	0.05	0.01	0.08	**1.38**
SF Earthquake	2,067,739	11887	0.19	0.01	0.02	0.09	0.02	0.13	**1.46**

- Overlap and Stitching: efficient

Exp. Results

- Efficiency
- Comparison
- Speed
- Pros & cons

8 threads, 4 partitions



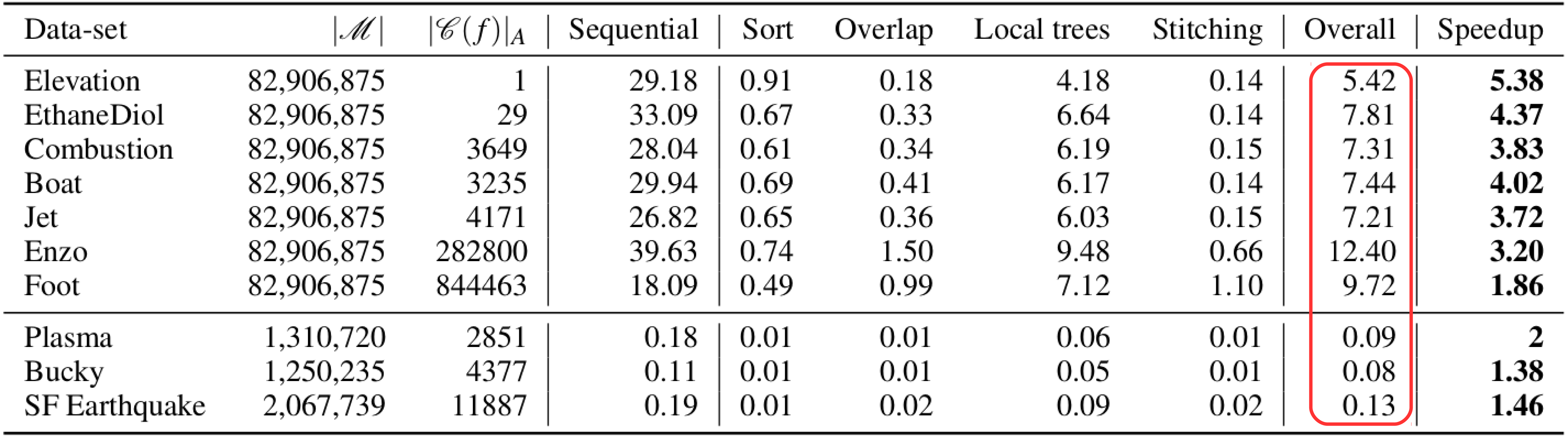
Data-set	$\lvert\mathscr{M}\rvert$	$\lvert\mathscr{C}(f)\rvert_A$	Sequential	Sort	Overlap	Local trees	Stitching	Overall	Speedup
Elevation	82,906,875	1	29.18	0.91	0.18	4.18	0.14	5.42	**5.38**
EthaneDiol	82,906,875	29	33.09	0.67	0.33	6.64	0.14	7.81	**4.37**
Combustion	82,906,875	3649	28.04	0.61	0.34	6.19	0.15	7.31	**3.83**
Boat	82,906,875	3235	29.94	0.69	0.41	6.17	0.14	7.44	**4.02**
Jet	82,906,875	4171	26.82	0.65	0.36	6.03	0.15	7.21	**3.72**
Enzo	82,906,875	282800	39.63	0.74	1.50	9.48	0.66	12.40	**3.20**
Foot	82,906,875	844463	18.09	0.49	0.99	7.12	1.10	9.72	**1.86**
Plasma	1,310,720	2851	0.18	0.01	0.01	0.06	0.01	0.09	**2**
Bucky	1,250,235	4377	0.11	0.01	0.01	0.05	0.01	0.08	**1.38**
SF Earthquake	2,067,739	11887	0.19	0.01	0.02	0.09	0.02	0.13	**1.46**

- High range of value

Exp. Results

- Efficiency
- Comparison
- Speed
- Pros & cons

8 threads, 4 partitions



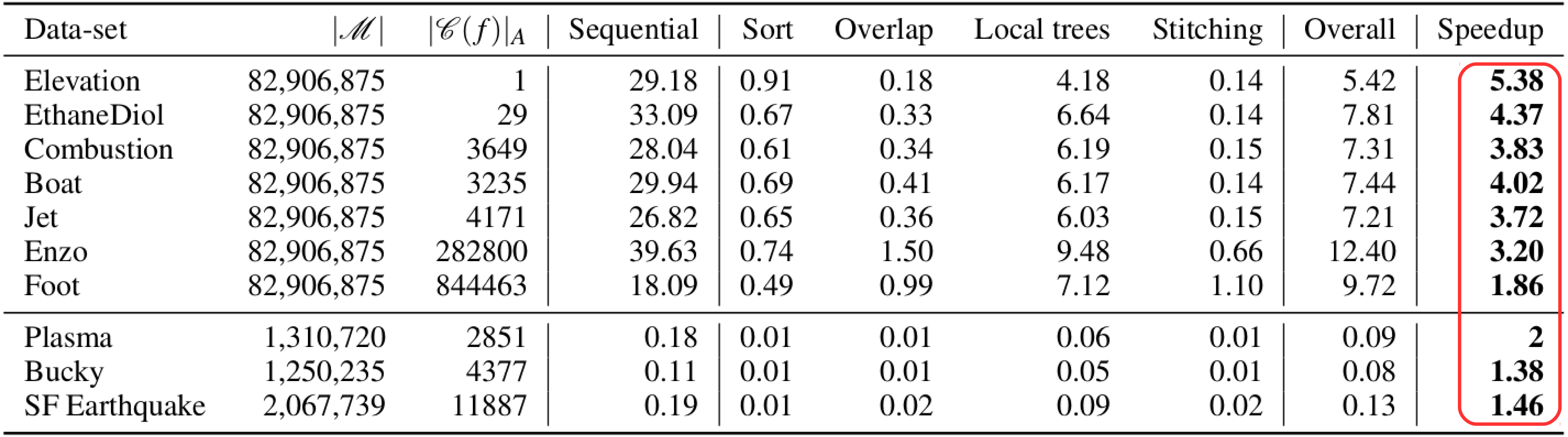
Data-set	$\|\mathcal{M}\|$	$\|\mathcal{C}(f)\|_A$	Sequential	Sort	Overlap	Local trees	Stitching	Overall	Speedup
Elevation	82,906,875	1	29.18	0.91	0.18	4.18	0.14	5.42	**5.38**
EthaneDiol	82,906,875	29	33.09	0.67	0.33	6.64	0.14	7.81	**4.37**
Combustion	82,906,875	3649	28.04	0.61	0.34	6.19	0.15	7.31	**3.83**
Boat	82,906,875	3235	29.94	0.69	0.41	6.17	0.14	7.44	**4.02**
Jet	82,906,875	4171	26.82	0.65	0.36	6.03	0.15	7.21	**3.72**
Enzo	82,906,875	282800	39.63	0.74	1.50	9.48	0.66	12.40	**3.20**
Foot	82,906,875	844463	18.09	0.49	0.99	7.12	1.10	9.72	**1.86**
Plasma	1,310,720	2851	0.18	0.01	0.01	0.06	0.01	0.09	**2**
Bucky	1,250,235	4377	0.11	0.01	0.01	0.05	0.01	0.08	**1.38**
SF Earthquake	2,067,739	11887	0.19	0.01	0.02	0.09	0.02	0.13	**1.46**

- About ten seconds

Exp. Results

- Efficiency
- Comparison
- Speed
- Pros & cons

8 threads, 4 partitions

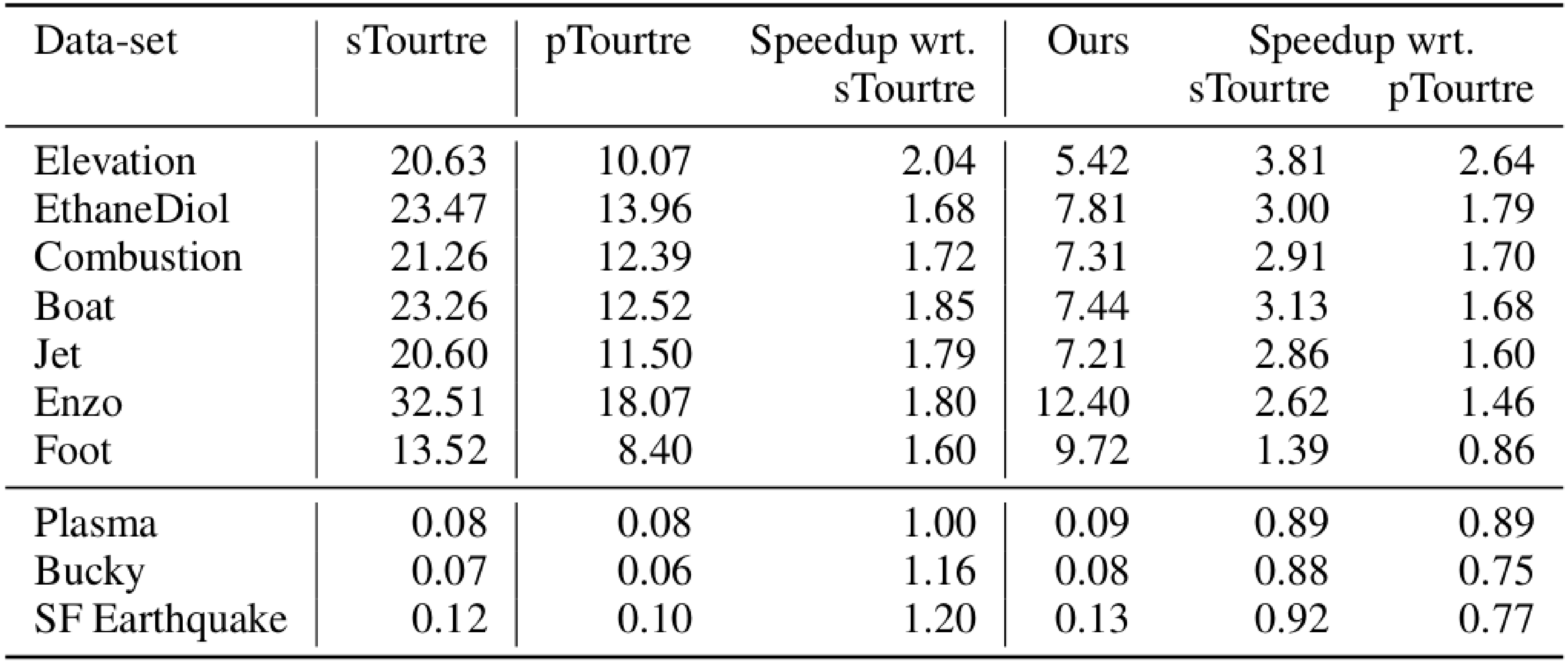


Data-set	$\|\mathscr{M}\|$	$\|\mathscr{C}(f)\|_A$	Sequential	Sort	Overlap	Local trees	Stitching	Overall	Speedup
Elevation	82,906,875	1	29.18	0.91	0.18	4.18	0.14	5.42	**5.38**
EthaneDiol	82,906,875	29	33.09	0.67	0.33	6.64	0.14	7.81	**4.37**
Combustion	82,906,875	3649	28.04	0.61	0.34	6.19	0.15	7.31	**3.83**
Boat	82,906,875	3235	29.94	0.69	0.41	6.17	0.14	7.44	**4.02**
Jet	82,906,875	4171	26.82	0.65	0.36	6.03	0.15	7.21	**3.72**
Enzo	82,906,875	282800	39.63	0.74	1.50	9.48	0.66	12.40	**3.20**
Foot	82,906,875	844463	18.09	0.49	0.99	7.12	1.10	9.72	**1.86**
Plasma	1,310,720	2851	0.18	0.01	0.01	0.06	0.01	0.09	**2**
Bucky	1,250,235	4377	0.11	0.01	0.01	0.05	0.01	0.08	**1.38**
SF Earthquake	2,067,739	11887	0.19	0.01	0.02	0.09	0.02	0.13	**1.46**

- Parallel efficiency max: 67% (speedup / n_t)
- Parallel efficiency: 40 ~ 55 % (except extrema)

Exp. Results

- Efficiency
- Comparison
- Speed
- Pros & cons



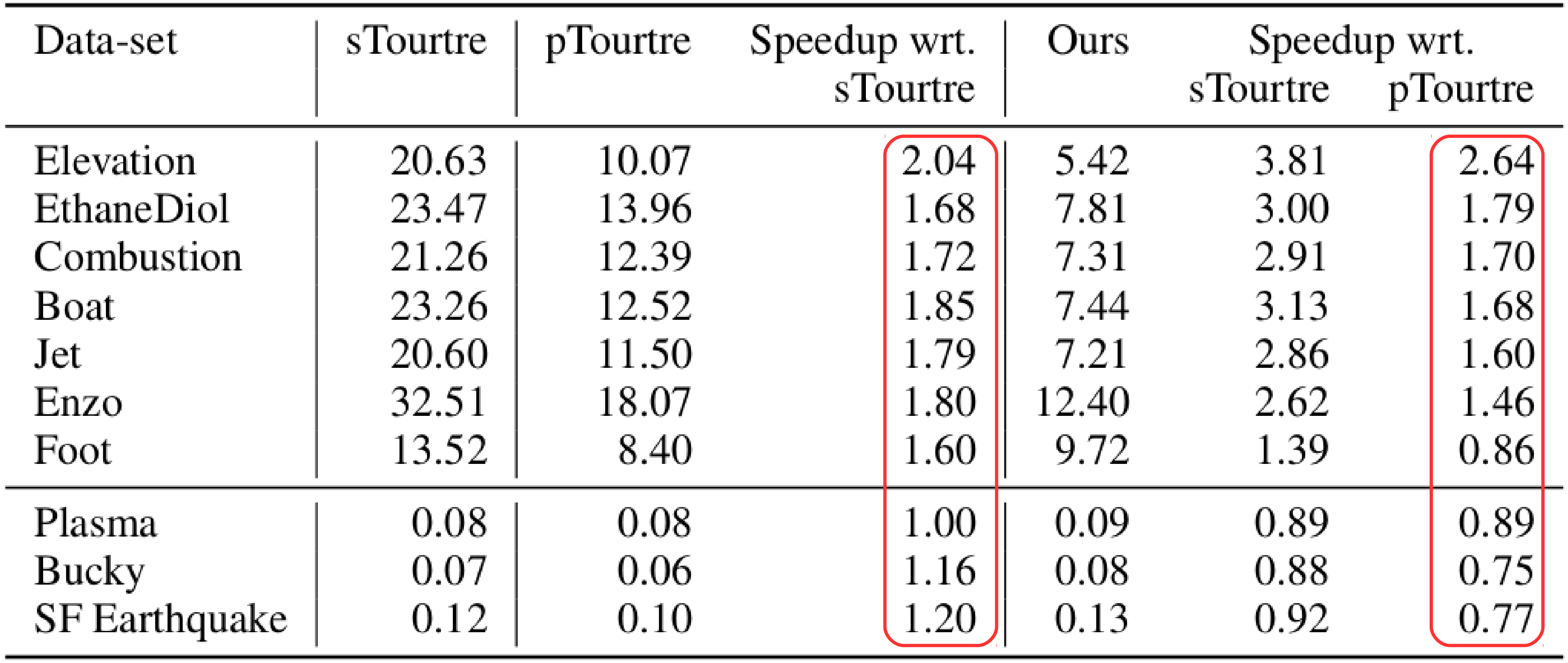
Data-set	sTourtre	pTourtre	Speedup wrt.	Ours	Speedup wrt.	
			sTourtre		sTourtre	pTourtre
Elevation	20.63	10.07	2.04	5.42	3.81	2.64
EthaneDiol	23.47	13.96	1.68	7.81	3.00	1.79
Combustion	21.26	12.39	1.72	7.31	2.91	1.70
Boat	23.26	12.52	1.85	7.44	3.13	1.68
Jet	20.60	11.50	1.79	7.21	2.86	1.60
Enzo	32.51	18.07	1.80	12.40	2.62	1.46
Foot	13.52	8.40	1.60	9.72	1.39	0.86
Plasma	0.08	0.08	1.00	0.09	0.89	0.89
Bucky	0.07	0.06	1.16	0.08	0.88	0.75
SF Earthquake	0.12	0.10	1.20	0.13	0.92	0.77

Libtourtre:

- Open-source reference implementation
- pTourtre: naive parallel version

Exp. Results

- Efficiency
- Comparison
- Speed
- Pros & cons



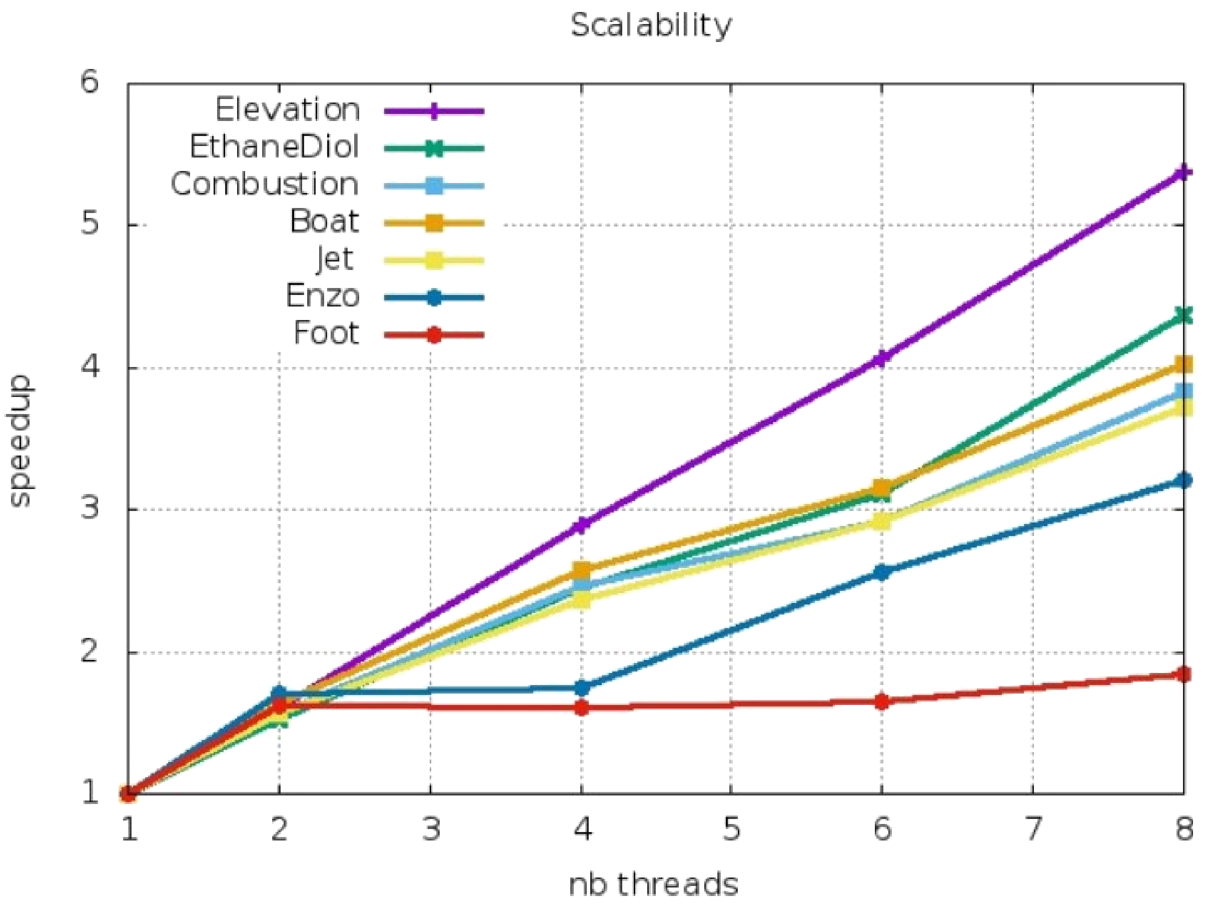
Data-set	sTourtre	pTourtre	Speedup wrt.	Ours	Speedup wrt.	
			sTourtre		sTourtre	pTourtre
Elevation	20.63	10.07	2.04	5.42	3.81	2.64
EthaneDiol	23.47	13.96	1.68	7.81	3.00	1.79
Combustion	21.26	12.39	1.72	7.31	2.91	1.70
Boat	23.26	12.52	1.85	7.44	3.13	1.68
Jet	20.60	11.50	1.79	7.21	2.86	1.60
Enzo	32.51	18.07	1.80	12.40	2.62	1.46
Foot	13.52	8.40	1.60	9.72	1.39	0.86
Plasma	0.08	0.08	1.00	0.09	0.89	0.89
Bucky	0.07	0.06	1.16	0.08	0.88	0.75
SF Earthquake	0.12	0.10	1.20	0.13	0.92	0.77

Libtourtre:

- Open-source reference implementation
- pTourtre: naive parallel version
- Always better than sequential libtourtre for big data sets
- Better than parallel except for the foot data set

Exp. Results

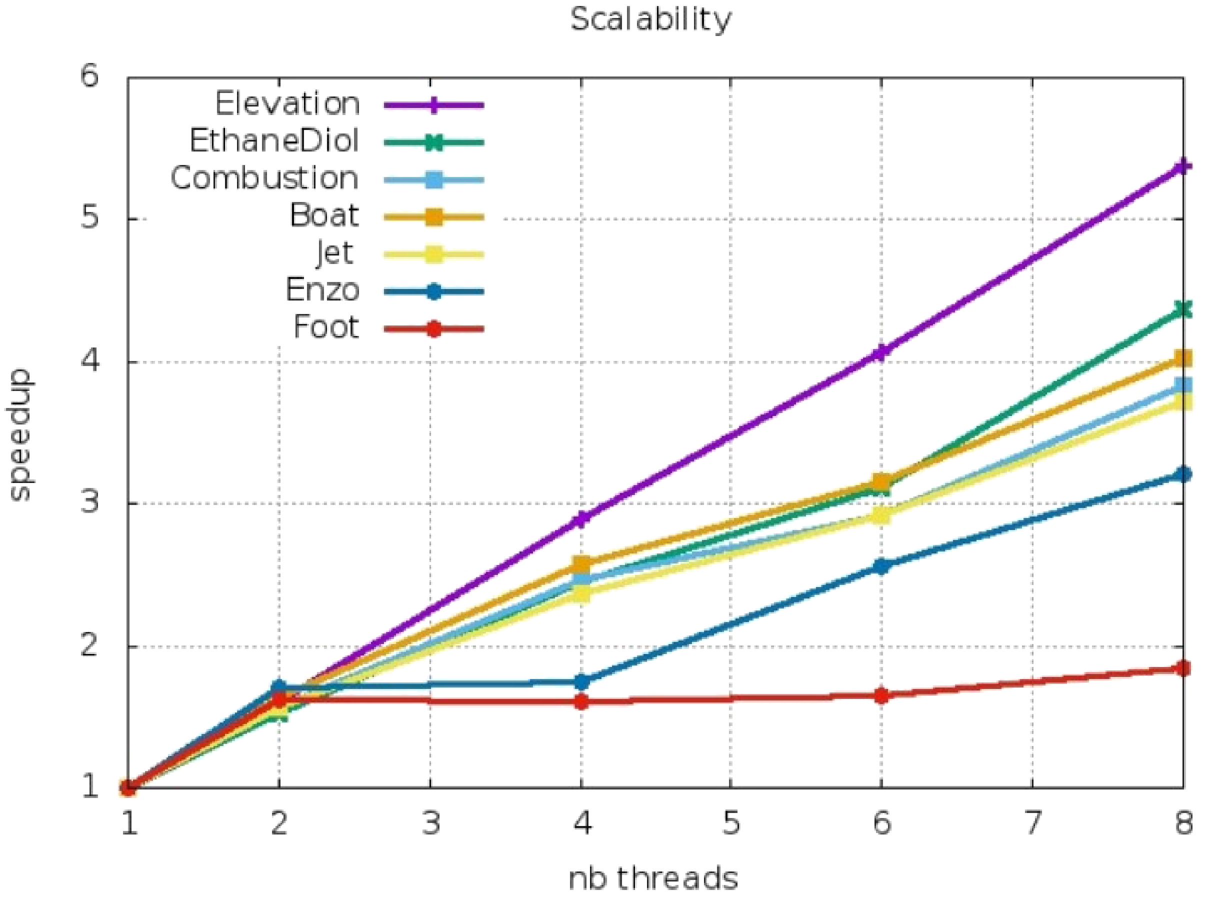
- Efficiency
- Comparison
- Speed
- Pros & cons



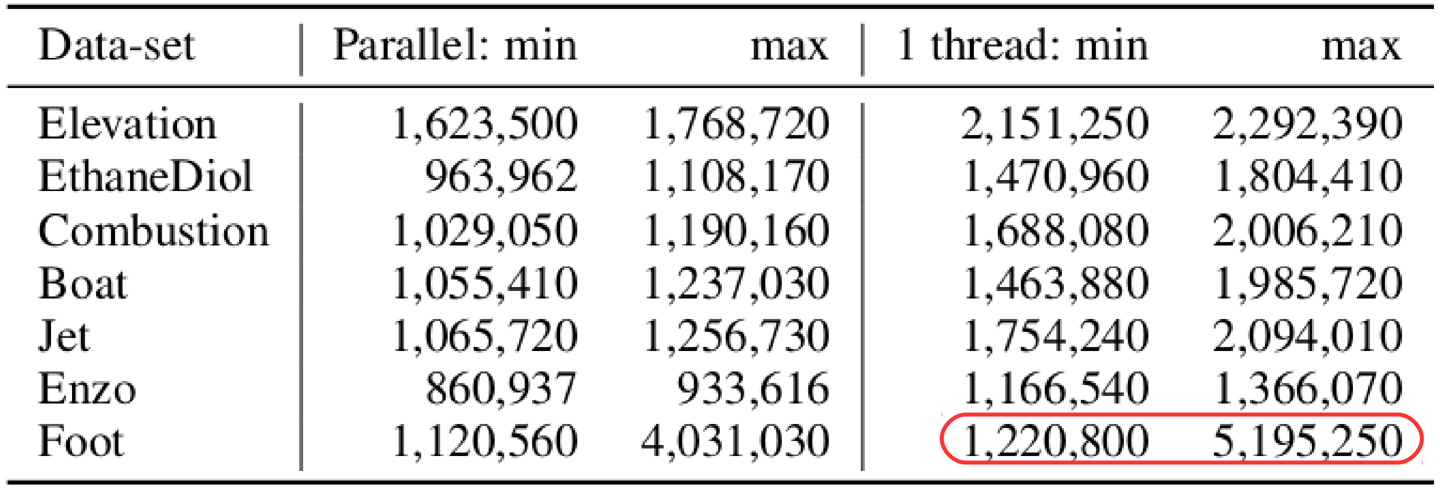
- Different slopes
- Few arithmetic operations per memory access

Exp. Results

- Efficiency
- Comparison
- Speed
- Pros & cons



- Different slopes
- Few arithmetic operations per memory access
- Load imbalance

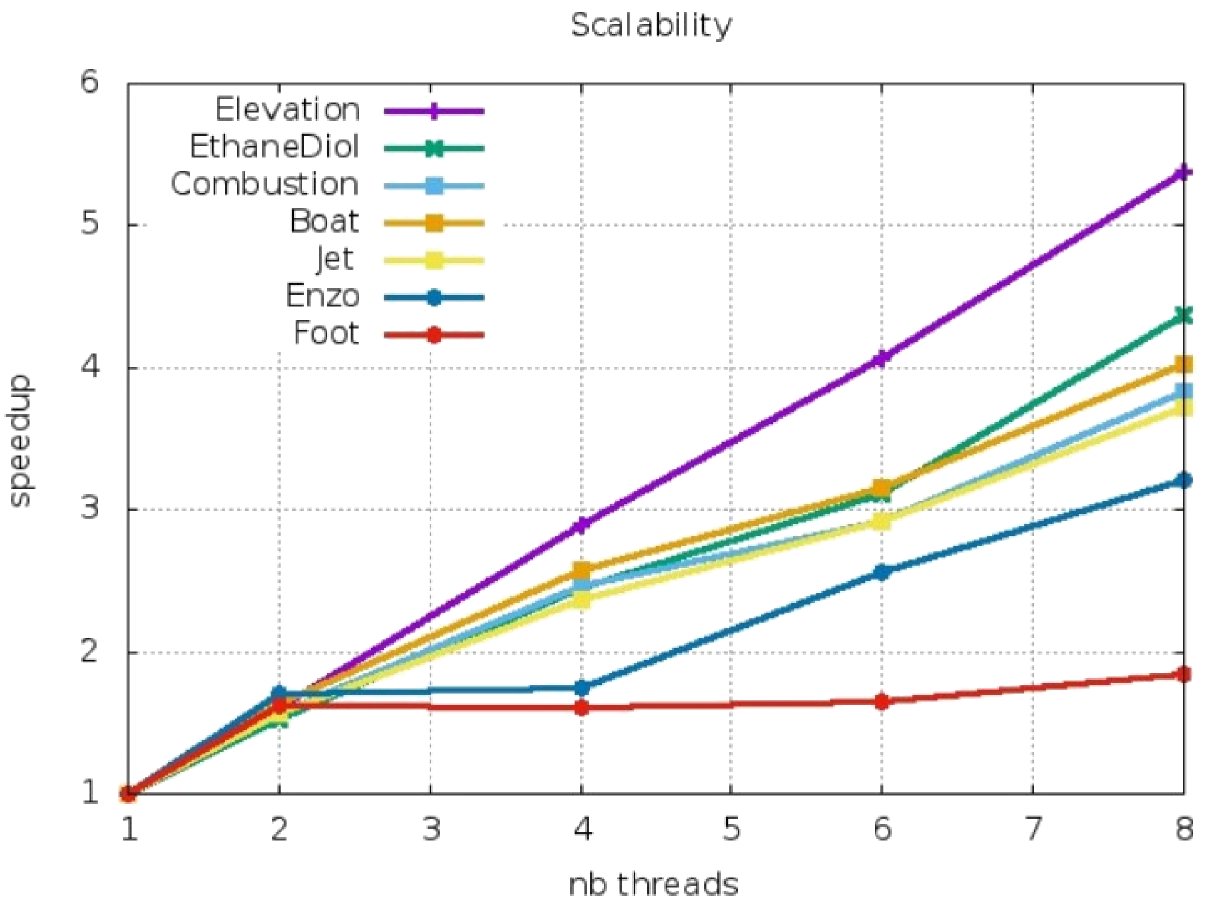


Data-set	Parallel: min	max	1 thread: min	max
Elevation	1,623,500	1,768,720	2,151,250	2,292,390
EthaneDiol	963,962	1,108,170	1,470,960	1,804,410
Combustion	1,029,050	1,190,160	1,688,080	2,006,210
Boat	1,055,410	1,237,030	1,463,880	1,985,720
Jet	1,065,720	1,256,730	1,754,240	2,094,010
Enzo	860,937	933,616	1,166,540	1,366,070
Foot	1,120,560	4,031,030	1,220,800	5,195,250

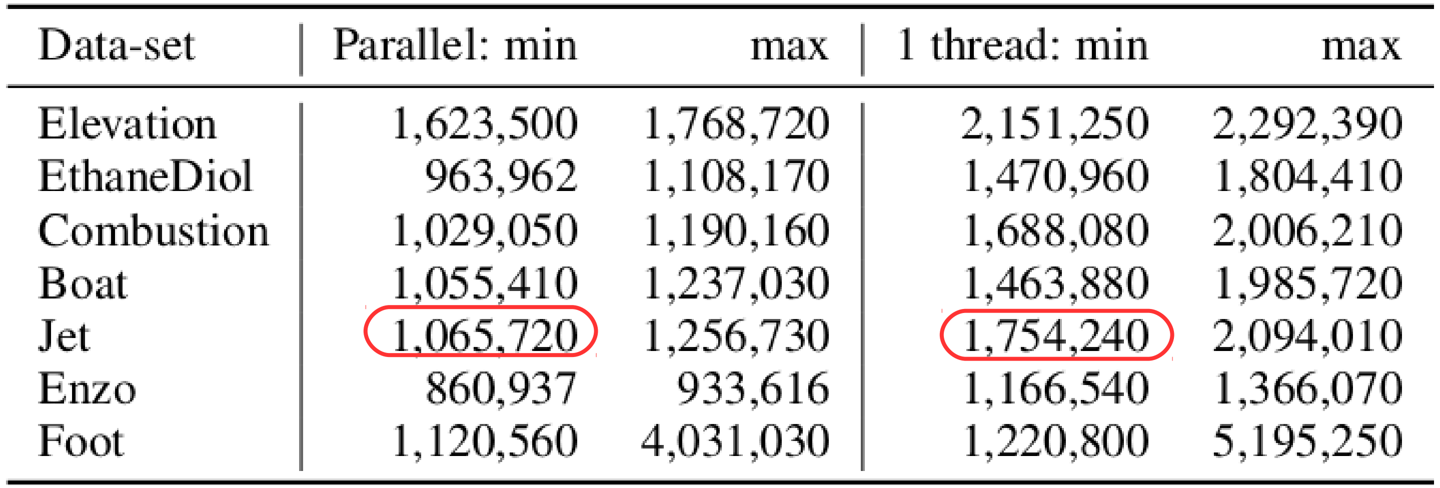
Computational speed: vertices/second

Exp. Results

- Efficiency
- Comparison
- Speed
- Pros & cons



- Different slopes
- Few arithmetic operations per memory access
- Load imbalance
- Memory congestion

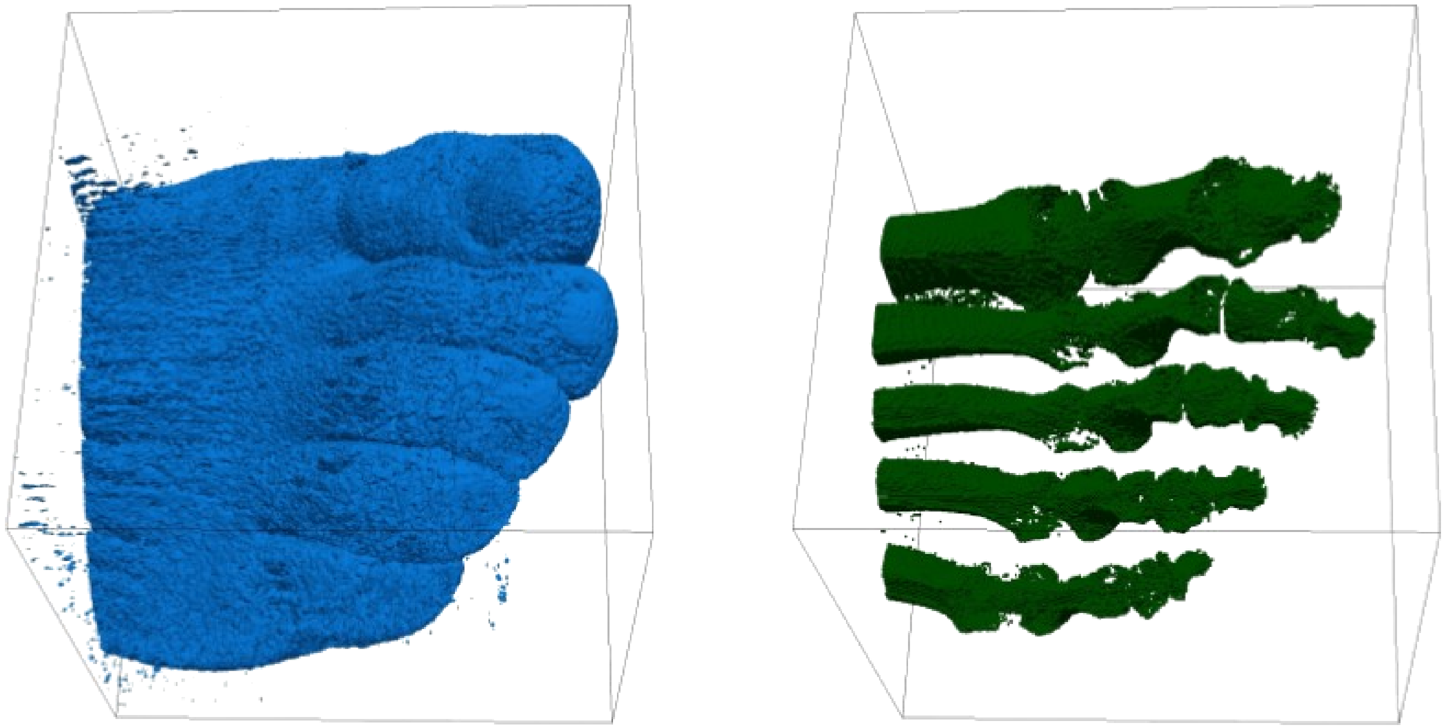


Data-set	Parallel: min	max	1 thread: min	max
Elevation	1,623,500	1,768,720	2,151,250	2,292,390
EthaneDiol	963,962	1,108,170	1,470,960	1,804,410
Combustion	1,029,050	1,190,160	1,688,080	2,006,210
Boat	1,055,410	1,237,030	1,463,880	1,985,720
Jet	1,065,720	1,256,730	1,754,240	2,094,010
Enzo	860,937	933,616	1,166,540	1,366,070
Foot	1,120,560	4,031,030	1,220,800	5,195,250

Computational speed: vertices/second

Exp. Results

- Efficiency
- Comparison
- Speed
- Pros & cons


(a)

(b)

- Different slopes
- Few arithmetic operations per memory access
- Load imbalance
- Memory congestion
- Redundant computations

Data-set	ideal	min	max
Elevation	4,194,304	4,259,840	4,325,376
EthaneDiol	4,194,304	4,362,086	4,616,938
Combustion	4,194,304	4,353,986	4,635,078
Boat	4,194,304	4,418,409	4,791,092
Jet	4,194,304	4,358,176	4,701,586
Enzo	4,194,304	5,234,144	6,474,322
Foot	4,194,304	4,499,572	6,044,708

Partitions sizes in vertices

Exp. Results

- Efficiency
- Comparison
- Speed
- Pros & cons

Pros:

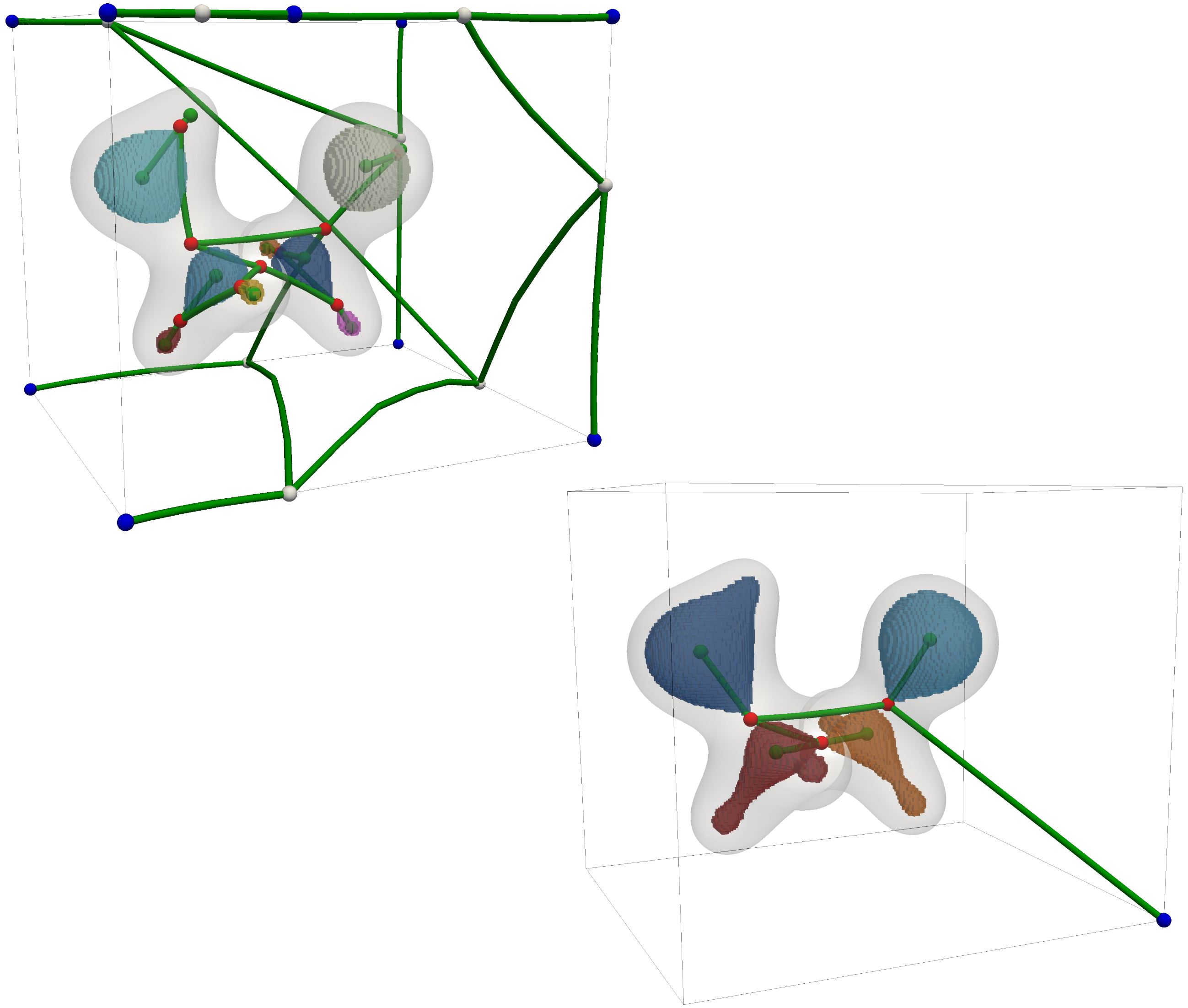
- Good parallel efficiency
- Faster than reference implementation
- Large spectrum of data

Cons:

- Redundant computation
- Load imbalance
- Memory congestion (augmented tree)

Application

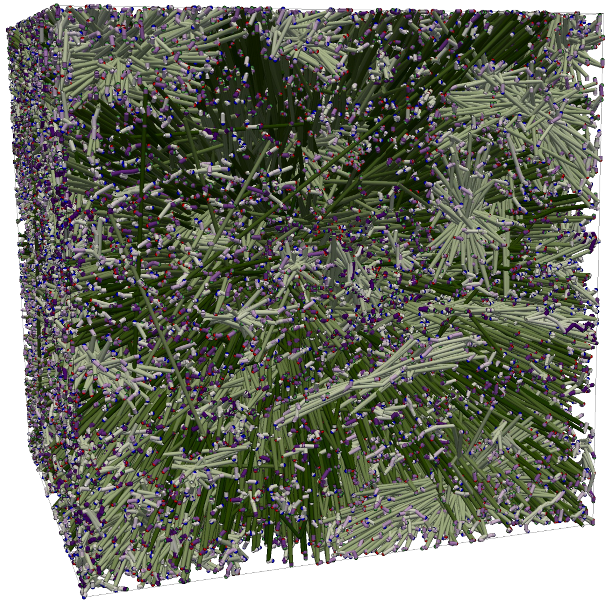
Application

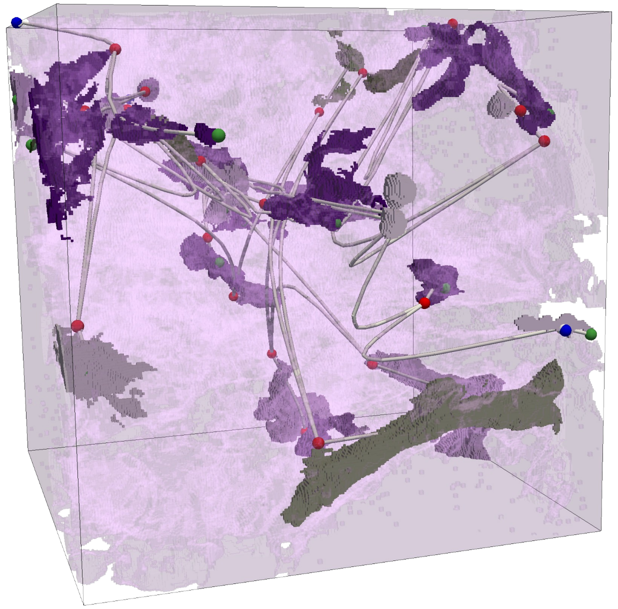


Exploration:

- Highlight features
- Allows features grouping

Application





Occlusion reduction:

- Remove noise
- Keep the most important features

Conclusion

Conclusion

- Recall
- Perspective

Take home message:

- Efficient algorithm:
    - Multi-threaded
    - Augmented
    - Simple approach, subtle details

Conclusion

- Recall
- Perspective

Take home message:

- Efficient algorithm:
  - Multi-threaded
  - Augmented
  - Simple approach, subtle details

- VTK-based implementation:
  - Generic input
    - VTU,VTI
    - 2D/3D
  - Generic output (augmented trees)
  - Ready-to-use
  - Integrated in TTK

Conclusion

- Recall
- Perspective

Take home message:

- Efficient algorithm:
  - Multi-threaded
  - Augmented
  - Simple approach, subtle details

- VTK-based implementation:
  - Generic input
    - VTU,VTI
    - 2D/3D
  - Generic output (augmented trees)
  - Ready-to-use
  - Integrated in TTK
- Lesson learned
  - Memory bound
  - Memory congestion: price to pay for augmented trees
  - Efficient implementation: hard

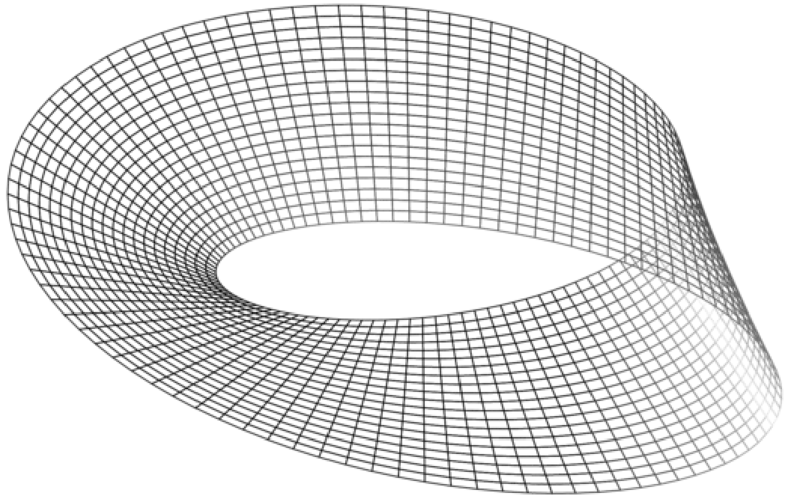
Conclusion

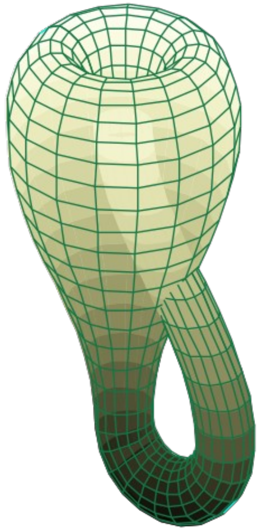
- Recall
- Perspective

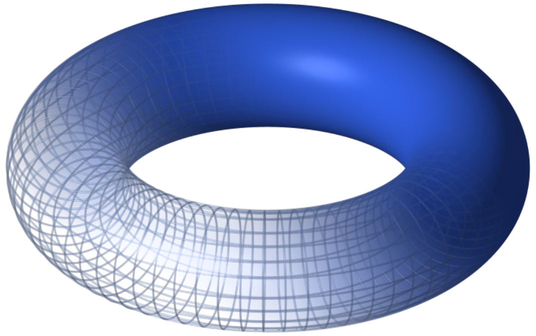
- Improve partitioning:
  - Better cutting isovalue selection
  - Contour Spectrum

- Distributed systems

The end







Thank you for your attention,

Do you have any question ?

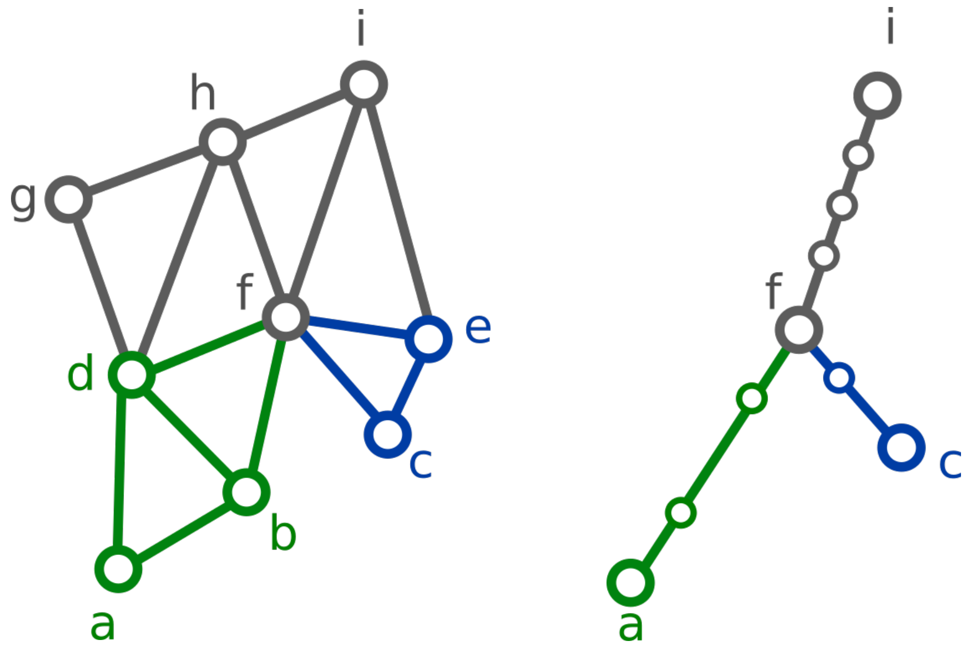
Paper at: http://www-pequan.lip6.fr/~tierny/
Code at: https://github.com/topology-tool-kit/ttk

This work is partially supported by the Bpifrance grant “AVIDO” (Programme d’Investissements d’Avenir, reference P112017-2661376/DOS0021427) and by the French National Association for Research and Technology (ANRT), in the framework of the LIP6 - Kitware SAS CIFRE partnership reference 2015/1039.

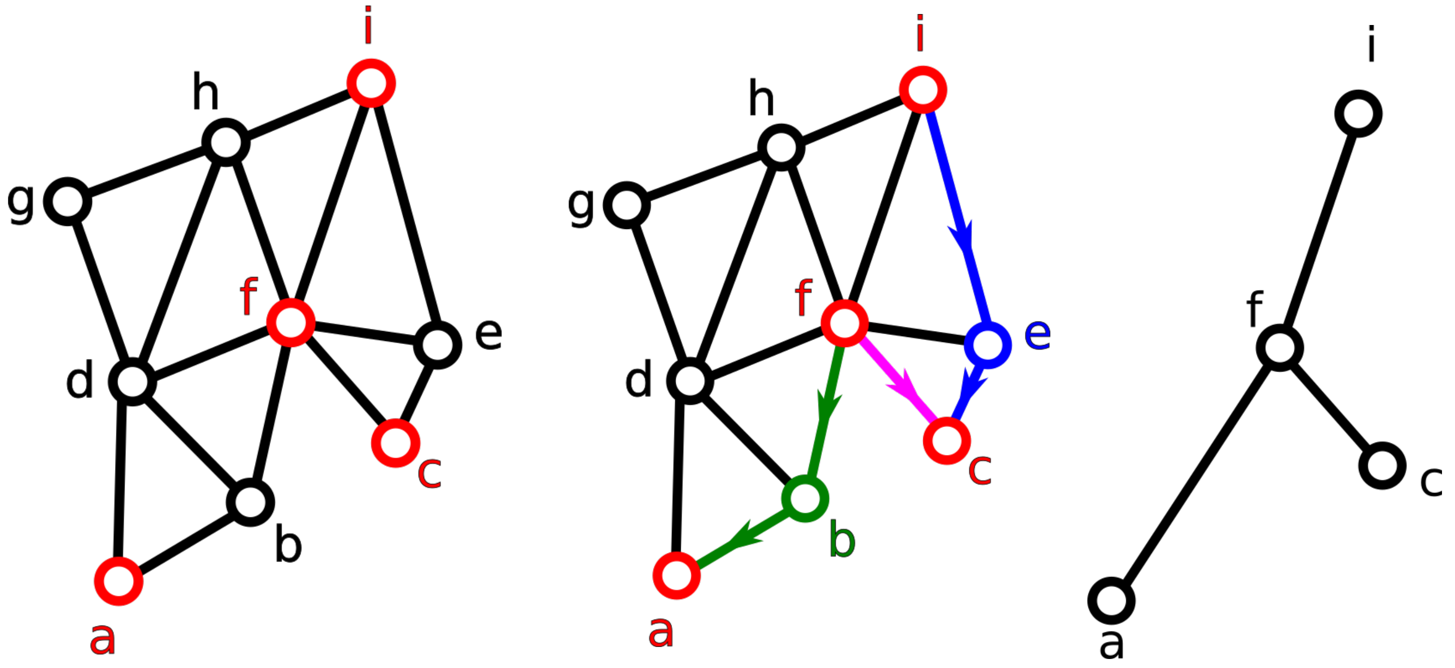
Appendice

Seq. Approaches

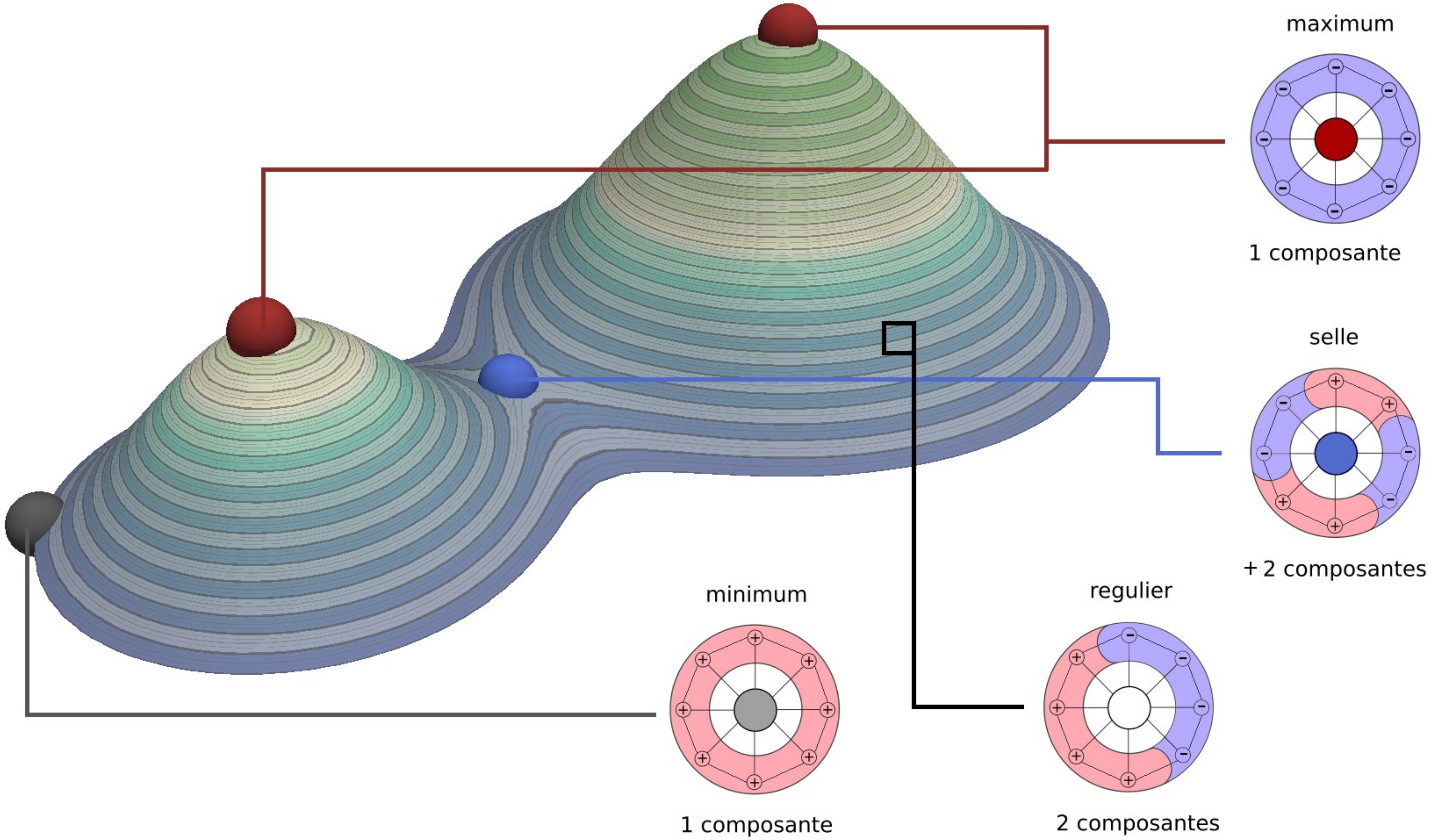
Union find:



Monotone paths:



Critical points


maximum
1 composante
selle
+ 2 composantes
minimum
1 composante
regulier
2 composantes