Distributed Computation of Persistent Homology

*Ulrich Bauer**   Michael Kerber†   Jan Reininghaus*

* IST Austria

† MPI Saarbrücken

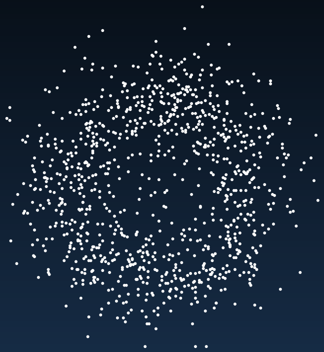
ALENEX14
Jan 5, 2014

What is persistent homology?

Turn data into growing sequence of topological spaces

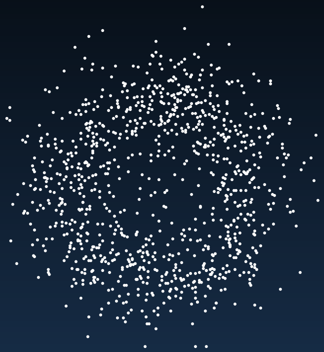
What is persistent homology?

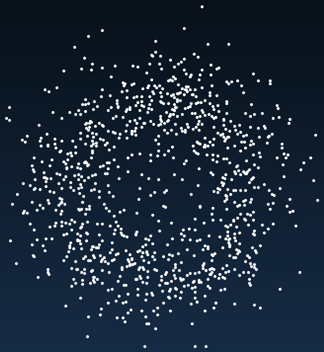
Turn data into growing sequence of topological spaces



What is persistent homology?

Turn data into growing sequence of topological spaces

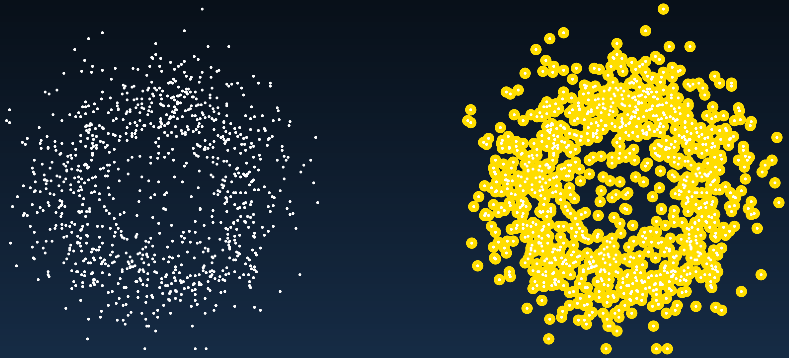




Union of balls: all points below a certain distance

What is persistent homology?

Turn data into growing sequence of topological spaces

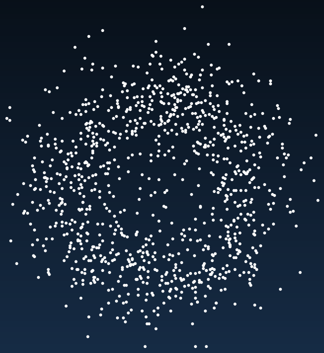


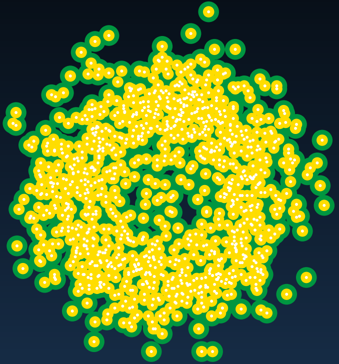
Union of balls: all points below a certain distance

Track changes in connectivity

What is persistent homology?

Turn data into growing sequence of topological spaces



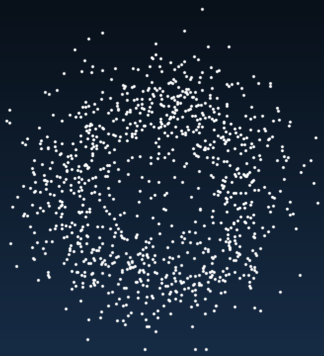


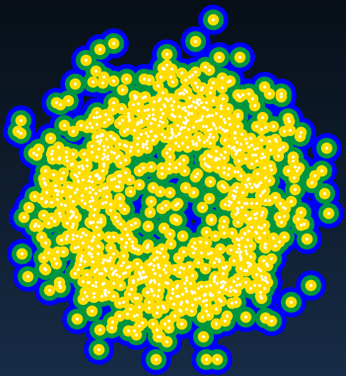
Union of balls: all points below a certain distance

Track changes in connectivity

What is persistent homology?

Turn data into growing sequence of topological spaces





Union of balls: all points below a certain distance

Track changes in connectivity

What is persistent homology?

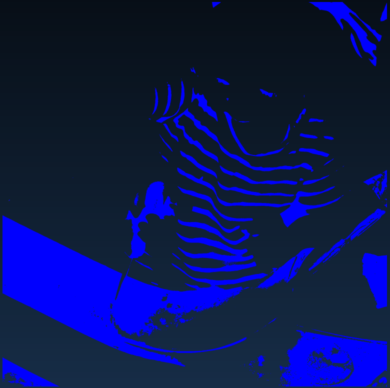
Turn data into growing sequence of topological spaces

What is persistent homology?

Turn data into growing sequence of topological spaces

What is persistent homology?

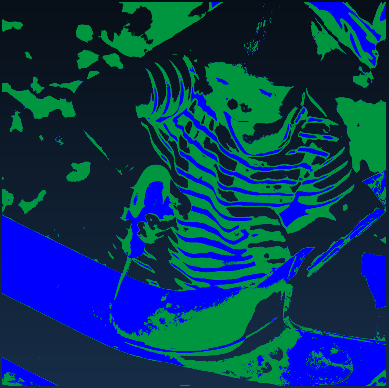
Turn data into growing sequence of topological spaces



Sublevel sets: all points below a certain level

What is persistent homology?

Turn data into growing sequence of topological spaces

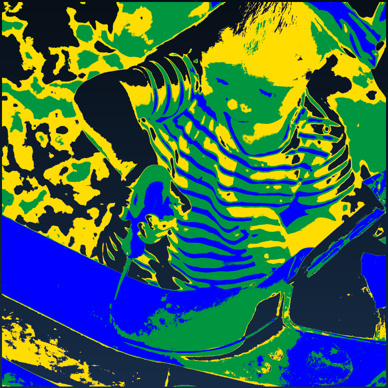


Sublevel sets: all points below a certain level

Track changes in connectivity

What is persistent homology?

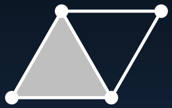
Turn data into growing sequence of topological spaces



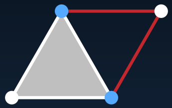
Sublevel sets: all points below a certain level

Track changes in connectivity

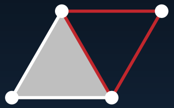
What is homology?



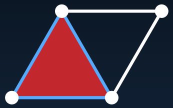
What is homology?



What is homology?

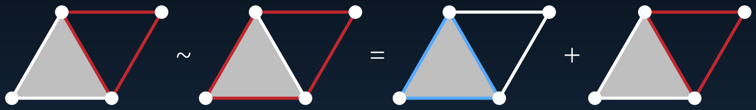


What is homology?



What is homology?

What is homology?



Example: filtration and boundary matrix

$D =$

	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							

Matrix reduction

	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1				
4				1			
5					1		
6						1	
7							1

Matrix reduction

	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1				
4				1			
5					1		
6						1	
7							1

Pivot of column j:

- largest index with nonzero entry

Matrix reduction

	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1				
4				1			
5					1		
6						1	
7							1

Eliminating pivot of col j:

- adding col i to col j with $i < j$ and $\operatorname{pivot}(i) = \operatorname{pivot}(j)$

Matrix reduction

	1	2	3	4	5	6	7
1			1		1	1	
2			1			1	
3							1
4					1	0	
5							1
6							1
7							

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1				
4				1			
5					1	1	
6						1	
7							1

Eliminating pivot of col j:

- adding col i to col j with $i < j$ and $\mathrm{pivot}(i) = \mathrm{pivot}(j)$

Matrix reduction

	1	2	3	4	5	6	7
1			1		1	1	
2			1			1	
3							1
4					1		
5							1
6							1
7							

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1				
4				1			
5					1	1	
6						1	
7							1

Eliminating pivot of col j:

- adding col i to col j with $i < j$ and $\text{pivot}(i) = \text{pivot}(j)$

Matrix reduction

	1	2	3	4	5	6	7
1			1		1	0	
2			1			0	
3							1
4					1		
5							1
6							1
7							

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1			1	
4				1			
5					1	1	
6						1	
7							1

Eliminating pivot of col j:

- adding col i to col j with $i < j$ and $\mathrm{pivot}(i) = \mathrm{pivot}(j)$

Matrix reduction

	1	2	3	4	5	6	7
1			1		1		
2			1				
3							1
4					1		
5							1
6							1
7							

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1			1	
4				1			
5					1	1	
6						1	
7							1

Column j is *reduced:*

- pivot of col j minimal under left-to-right column additions

Matrix reduction

	1	2	3	4	5	6	7
1			1		1		
2			1				
3							1
4					1		
5							1
6							1
7							

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1			1	
4				1			
5					1	1	
6						1	
7							1

Matrix M is *reduced:*

- all columns are reduced (equivalently: pivots are unique)

Matrix reduction

	1	2	3	4	5	6	7
1			1		1		
2			1				
3							1
4					1		
5							1
6							1
7							

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1			1	
4				1			
5					1	1	
6						1	
7							1

Matrix M is *reduced at index* (i, j) :

- submatrix with rows $\geq i$ and cols $\leq j$ (lower left) is reduced

Matrix reduction

	1	2	3	4	5	6	7
1			1		1		
2			1				
3							1
4					1		
5							1
6							1
7							

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1			1	
4				1			
5					1	1	
6						1	
7							1

$i = \operatorname{pivot}(j)$ and M is reduced at index $(i, j) \Rightarrow$

- column j is reduced
- (i, j) is a *persistence pair:*
  homology is created at step i and killed at step j

Matrix reduction

	1	2	3	4	5	6	7
1			1		1		
2			1				
3							1
4					1		
5							1
6							1
7							

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1			1	
4				1			
5					1	1	
6						1	
7							1

$i = \operatorname{pivot}(j)$ and M is reduced at index $(i, j) \Rightarrow$

- column j is reduced
- (i, j) is a *persistence pair:*

  homology is created at step i and killed at step j

Matrix reduction

	1	2	3	4	5	6	7
1			1		1		
2			1				
3							1
4					1		
5							1
6							1
7							

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1			1	
4				1			
5					1	1	
6						1	
7							1

$i = \mathrm{pivot}(j)$ and M is reduced at index $(i, j) \Rightarrow$

- column j is reduced
- (i, j) is a *persistence pair:*
  homology is created at step i and killed at step j

Matrix reduction

	1	2	3	4	5	6	7
1			1		1		
2			1				
3							1
4					1		
5							1
6							1
7							

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1			1	
4				1			
5					1	1	
6						1	
7							1

$i = \mathrm{pivot}(j)$ and M is reduced at index $(i, j) \Rightarrow$

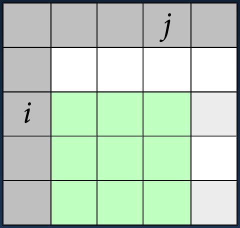
- column j is reduced
- (i, j) is a *persistence pair:*
  homology is created at step i and killed at step j

Matrix reduction in blocks

Partition $1 \ldots n$ into p *ranges:* $0 = r_0 < \cdots < r_i < \cdots < r_p = n$

Range i: $\{k : r_{i-1} < k \leq r_i\}$

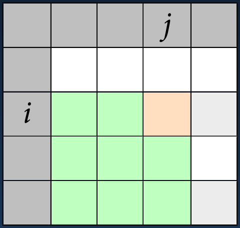
- Matrix M is *reduced at block* (i, j)*:*
  submatrix of row range $\geq i$ and col range $\leq j$ is reduced



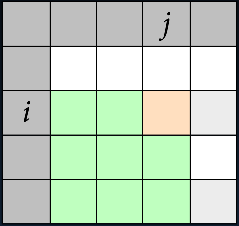
Matrix reduction in blocks

Partition $1 \dots n$ into p *ranges:* $0 = r_0 < \cdots < r_i < \cdots < r_p = n$
Range i: $\{k : r_{i-1} < k \leq r_i\}$

- Matrix M is *reduced at block* (i, j)*:*
  submatrix of row range $\geq i$ and col range $\leq j$ is reduced
- Matrix M is *reducible in block* (i, j)*:*
  reduced at blocks $(i, j-1)$ and $(i+1, j)$



Matrix reduction in blocks



To reduce a reducible block (i, j):

- Only pivots in block (i, j) need to be eliminated
- Only cols in range j are modified
- Only reduced cols with pivot in range i are used

Distributed matrix reduction

Assign row ranges to nodes:

- Nodes collect reduced cols with pivot in their range
- Nodes eliminate pivots in their range

Distributed matrix reduction

Assign row ranges to nodes:

- Nodes collect reduced cols with pivot in their range
- Nodes eliminate pivots in their range

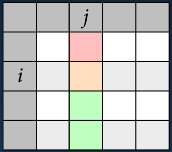
Distributed reduction in blocks

- Node i initially holds columns in range i
- Node i repeats for $j = i \dots n$:
  - reduce matrix at block (i, j) and collect reduced columns
  - send columns in range j with pivot range $< i$ to node $(i - 1)$
  - receive columns in range $(j + 1)$ from node $(i + 1)$

Distributed matrix reduction

Distributed reduction in blocks

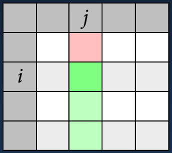
- Node i initially holds columns in range i
- Node i repeats for $j = i \dots n$:
  - reduce matrix at block (i, j) and collect reduced columns
  - send columns in range j with pivot range $< i$ to node $(i - 1)$
  - receive columns in range $(j + 1)$ from node $(i + 1)$



Distributed matrix reduction

Distributed reduction in blocks

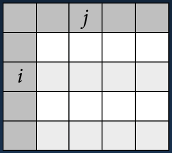
- Node i initially holds columns in range i
- Node i repeats for $j = i \dots n$:
  - reduce matrix at block (i, j) and collect reduced columns
  - send columns in range j with pivot range $< i$ to node $(i-1)$
  - receive columns in range $(j+1)$ from node $(i+1)$



Distributed matrix reduction

Distributed reduction in blocks

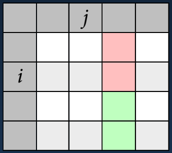
- Node i initially holds columns in range i
- Node i repeats for $j = i \dots n$:
  - reduce matrix at block (i, j) and collect reduced columns
  - send columns in range j with pivot range $< i$ to node $(i - 1)$
  - receive columns in range $(j + 1)$ from node $(i + 1)$



Distributed matrix reduction

Distributed reduction in blocks

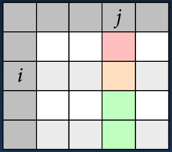
- Node i initially holds columns in range i
- Node i repeats for $j = i \dots n$:
  - reduce matrix at block (i, j) and collect reduced columns
  - send columns in range j with pivot range $< i$ to node $(i - 1)$
  - receive columns in range $(j + 1)$ from node $(i + 1)$



Distributed matrix reduction

Distributed reduction in blocks

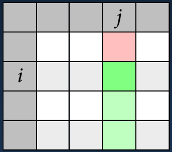
- Node i initially holds columns in range i
- Node i repeats for $j = i \dots n$:
  - reduce matrix at block (i, j) and collect reduced columns
  - send columns in range j with pivot range $< i$ to node $(i-1)$
  - receive columns in range $(j+1)$ from node $(i+1)$



Distributed matrix reduction

Distributed reduction in blocks

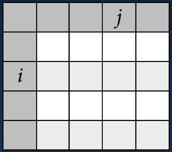
- Node i initially holds columns in range i
- Node i repeats for $j = i \ldots n$:
  - reduce matrix at block (i, j) and collect reduced columns
  - send columns in range j with pivot range $< i$ to node $(i - 1)$
  - receive columns in range $(j + 1)$ from node $(i + 1)$



Distributed matrix reduction

Distributed reduction in blocks

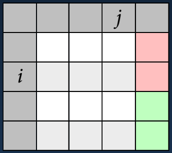
- Node i initially holds columns in range i
- Node i repeats for $j = i \dots n$:
  - reduce matrix at block (i, j) and collect reduced columns
  - send columns in range j with pivot range $< i$ to node $(i - 1)$
  - receive columns in range $(j + 1)$ from node $(i + 1)$



Distributed matrix reduction

Distributed reduction in blocks

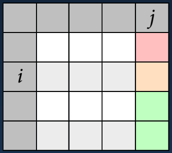
- Node i initially holds columns in range i
- Node i repeats for $j = i \ldots n$:
  - reduce matrix at block (i, j) and collect reduced columns
  - send columns in range j with pivot range $< i$ to node $(i - 1)$
  - receive columns in range $(j + 1)$ from node $(i + 1)$



Distributed matrix reduction

Distributed reduction in blocks

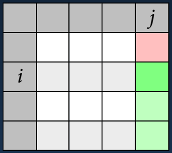
- Node i initially holds columns in range i
- Node i repeats for $j = i \dots n$:
  - reduce matrix at block (i, j) and collect reduced columns
  - send columns in range j with pivot range $< i$ to node $(i - 1)$
  - receive columns in range $(j + 1)$ from node $(i + 1)$



Distributed matrix reduction

Distributed reduction in blocks

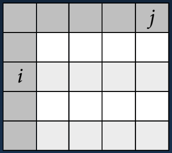
- Node i initially holds columns in range i
- Node i repeats for $j = i \ldots n$:
  - reduce matrix at block (i, j) and collect reduced columns
  - send columns in range j with pivot range $< i$ to node $(i-1)$
  - receive columns in range $(j+1)$ from node $(i+1)$



Distributed matrix reduction

Distributed reduction in blocks

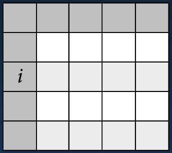
- Node i initially holds columns in range i
- Node i repeats for $j = i \dots n$:
  - reduce matrix at block (i, j) and collect reduced columns
  - send columns in range j with pivot range $< i$ to node $(i - 1)$
  - receive columns in range $(j + 1)$ from node $(i + 1)$



Distributed matrix reduction

Distributed reduction in blocks

- Node i initially holds columns in range i
- Node i repeats for $j = i \ldots n$:
  - reduce matrix at block (i, j) and collect reduced columns
  - send columns in range j with pivot range $< i$ to node $(i-1)$
  - receive columns in range $(j+1)$ from node $(i+1)$



Result: node i has all reduced columns with pivot in range i

Experimental results

Running times:

	n	PHAT		DIPHA				
cores/nodes (p)		1	16	2	4	8	16	32
`GRF2-256`	1.3×10^8	15 s	5.2 s	8.9 s	5.9 s	3.0 s	2.3 s	1.5 s
`GRF1-256`	1.3×10^8	29 s	13 s	30 s	22 s	16 s	13 s	12 s
`GRF2-512`	1.1×10^9					32 s	22 s	16 s
`GRF1-512`	1.1×10^9					147 s	116 s	100 s
`vertebra16`	1.1×10^9					45 s	42 s	34 s

Peak memory consumption per node:

	PHAT		DIPHA				
cores/nodes (p)	1	16	2	4	8	16	32
`GRF2-256`	10 GB	10 GB	5.8 GB	3.0 GB	1.5 GB	0.8 GB	0.4 GB
`GRF1-256`	10 GB	11 GB	6.1 GB	3.0 GB	1.5 GB	0.8 GB	0.4 GB
`GRF2-512`					11 GB	5.5 GB	3.0 GB
`GRF1-512`					12 GB	6.1 GB	3.1 GB
`vertebra16`					14 GB	9.8 GB	5.6 GB

Experimental results

Running times:

	n	PHAT		DIPHA				
cores/nodes (p)		1	16	2	4	8	16	32
`GRF2-256`	1.3×10^8	15 s	5.2 s	8.9 s	5.9 s	3.0 s	2.3 s	1.5 s
`GRF1-256`	1.3×10^8	29 s	13 s	30 s	22 s	16 s	13 s	12 s
`GRF2-512`	1.1×10^9					32 s	22 s	16 s
`GRF1-512`	1.1×10^9					147 s	116 s	100 s
`vertebra16`	1.1×10^9					45 s	42 s	34 s

Peak memory consumption per node:

	PHAT		DIPHA				
cores/nodes (p)	1	16	2	4	8	16	32
`GRF2-256`	10 GB	10 GB	5.8 GB	3.0 GB	1.5 GB	0.8 GB	0.4 GB
`GRF1-256`	10 GB	11 GB	6.1 GB	3.0 GB	1.5 GB	0.8 GB	0.4 GB
`GRF2-512`					11 GB	5.5 GB	3.0 GB
`GRF1-512`					12 GB	6.1 GB	3.1 GB
`vertebra16`					14 GB	9.8 GB	5.6 GB

Experimental results

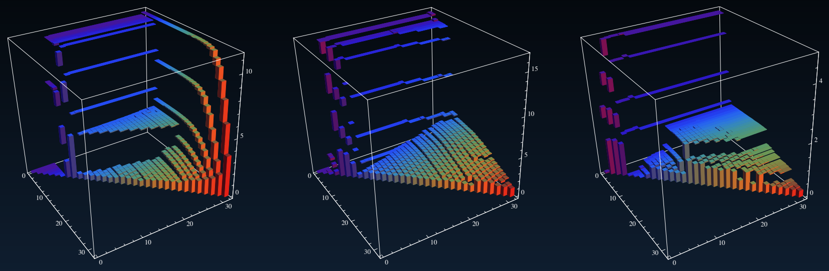
Total network transfer:

nodes (p)	2	4	8	16	32
`GRF2-256`	5.6 MB	15.1 MB	32.5 MB	67.7 MB	136 MB
`GRF1-256`	69.2 MB	218 MB	497 MB	1.0 GB	2.0 GB
`GRF2-512`			117 MB	237 MB	475 MB
`GRF1-512`			3.6 GB	7.3 GB	14.8 GB
`vertebra16`			3.3 GB	7.6 GB	15.7 GB

Maximal pairwise network transfer:

nodes (p)	2	4	8	16	32
`GRF2-256`	5.6 MB	5.6 MB	5.6 MB	6.5 MB	8.7 MB
`GRF1-256`	69.2 MB	90.3 MB	109 MB	162 MB	238 MB
`GRF2-512`			21.2 MB	21.2 MB	21.2 MB
`GRF1-512`			658 MB	658 MB	663 MB
`vertebra16`			2.9 GB	2.9 GB	2.9 GB

Running times per block



Data set `vertebra16`

- left axis: nodes
- right axis: column ranges
- vertical axis: running time
- persistence computation in dimensions 2, 1, 0

Conclusion

A fast distributed algorithm for persistent homology

- Constant (small) number of messages exchanged
- Amount of transferred data comparable to input data size
- Memory consumption scales very well
- Computation time is not an issue
- Able to compute huge instances

Source code available

- `http://dipha.googlecode.com`