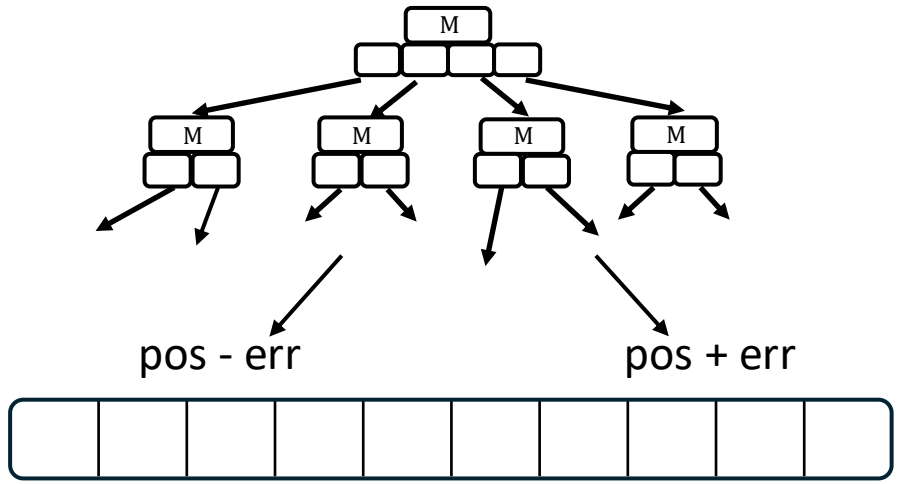
BOSTON UNIVERSITY

Improving the physical-to-sorted mapping for Learned Indexes

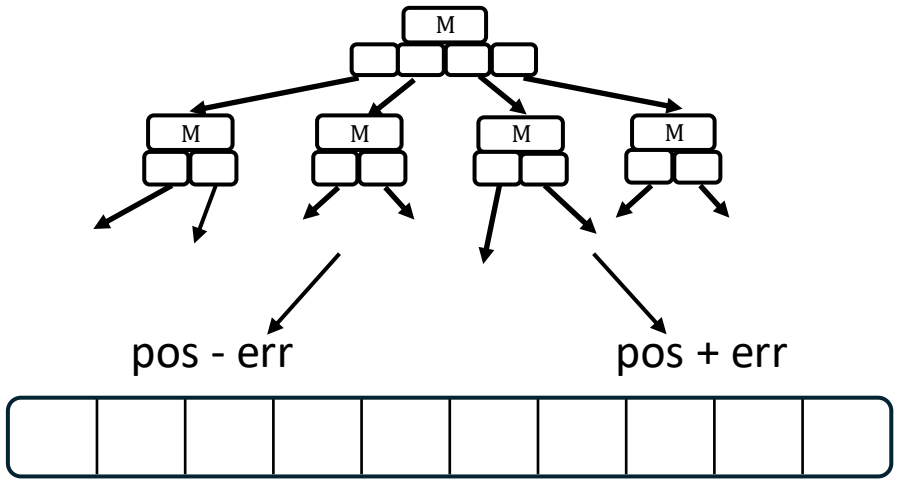
Anwesha Saha, Aneesh Raman, Ryan Marcus, Manos Athanassoulis

DiSC lab

Learned Indexes

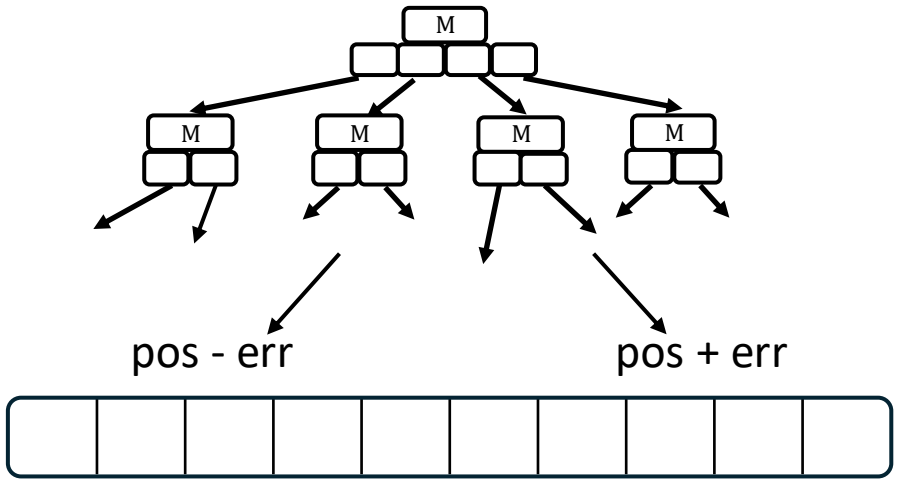


Learned Indexes



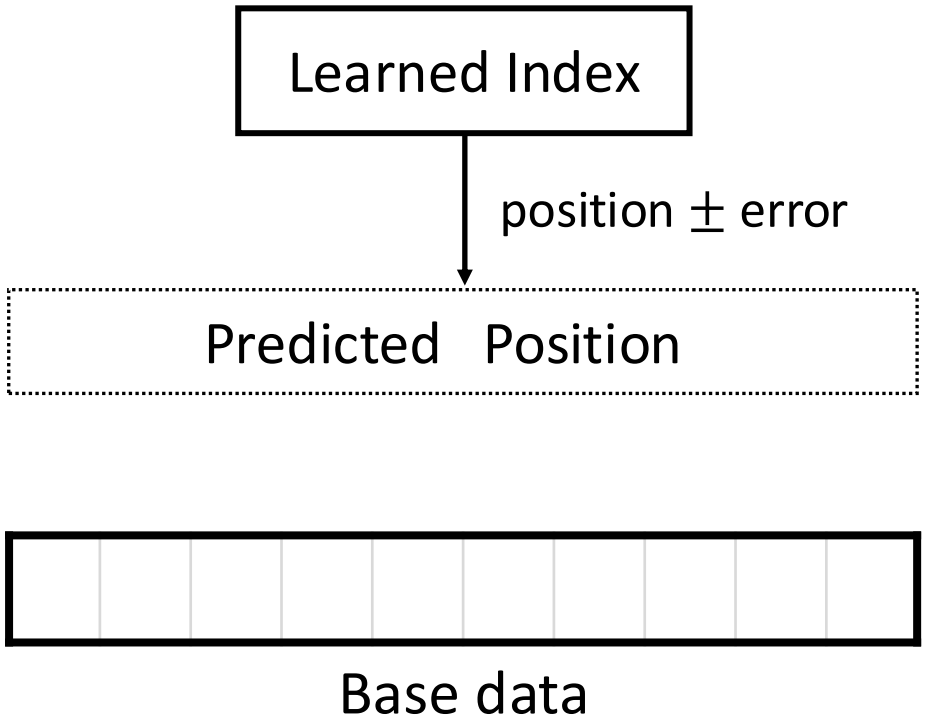
✓ Lower Memory

Learned Indexes

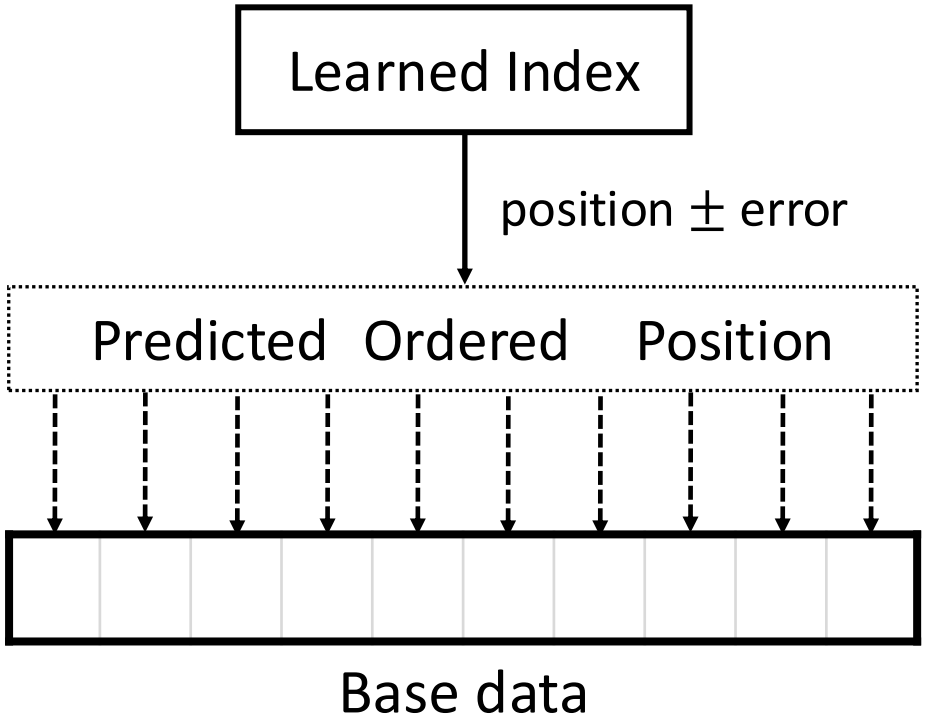


- ✓ Lower Memory
- ✓ Faster Access

Learned Indexes Assumptions

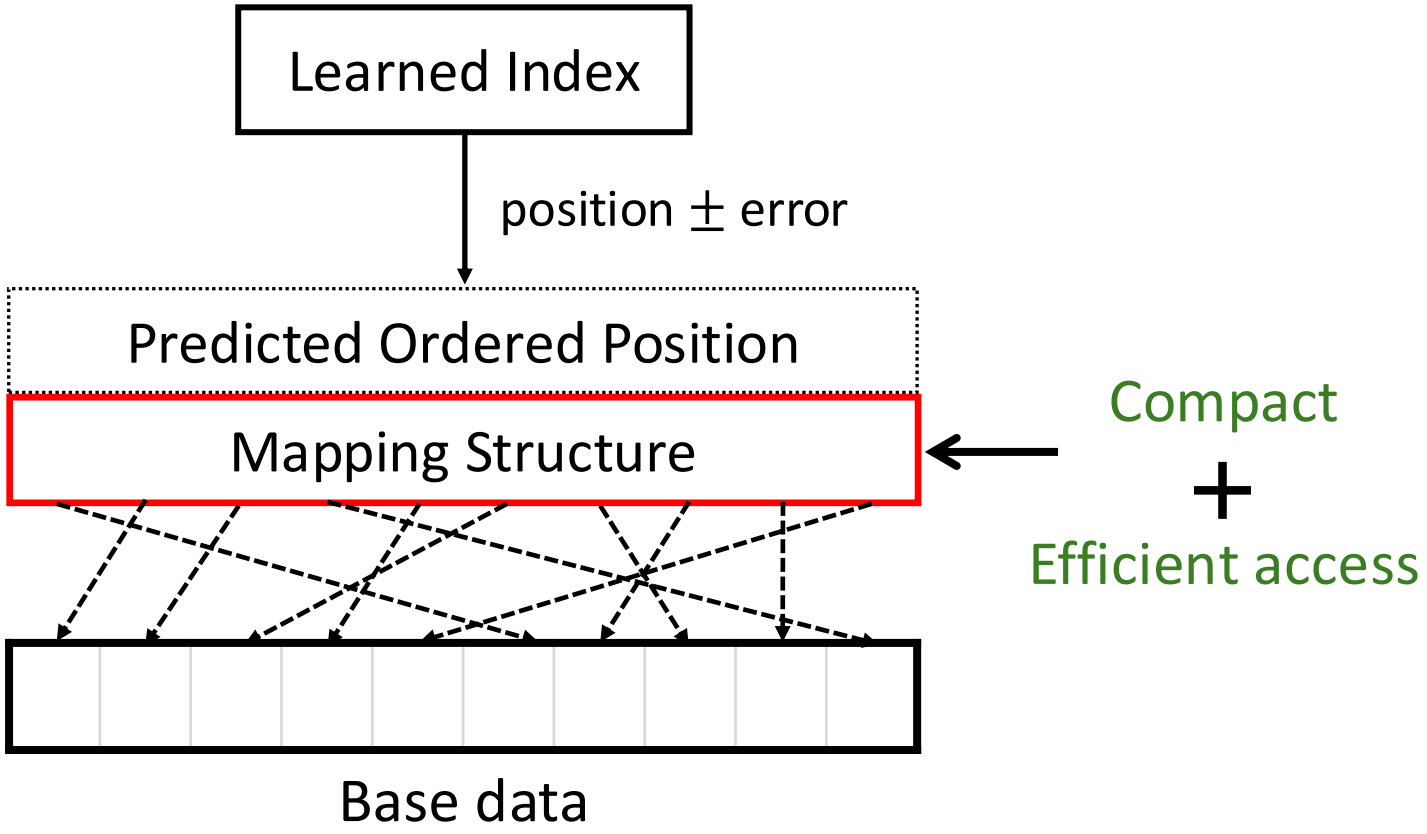


Learned Indexes Assumptions



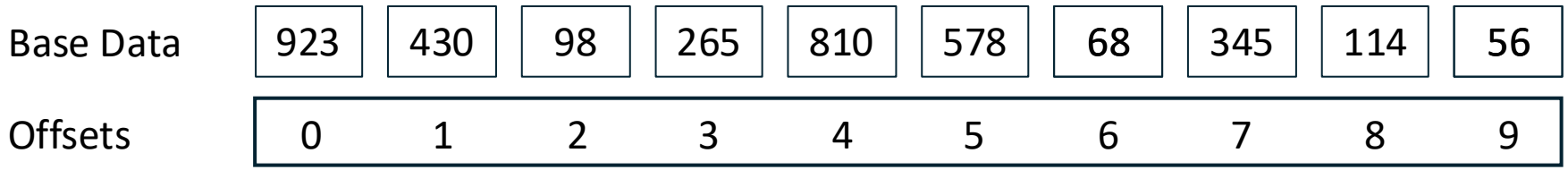
Assumes Sorted data underneath!

Proposed Design

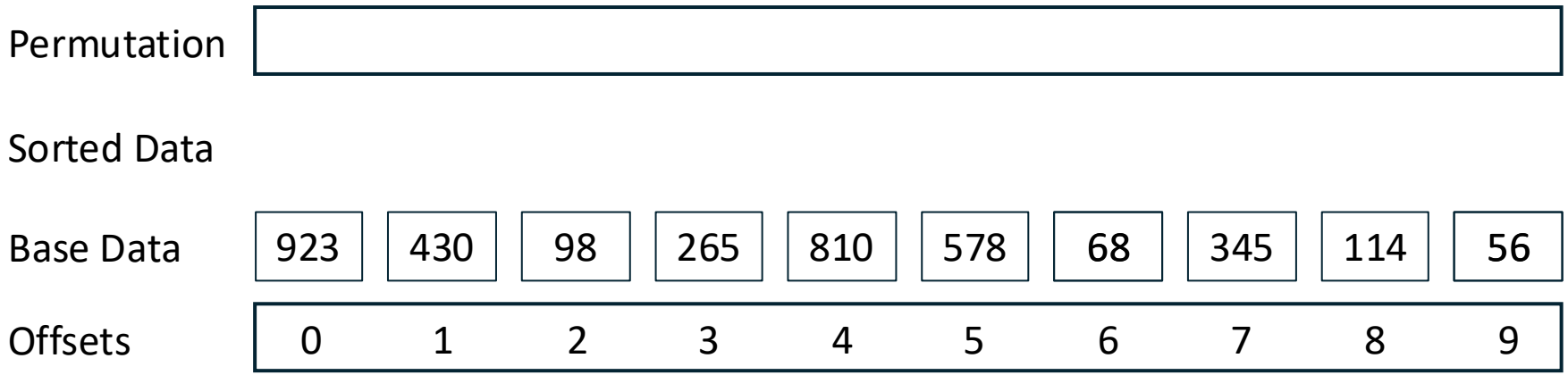


Access(Mapping structure) << Access(Traditional index)

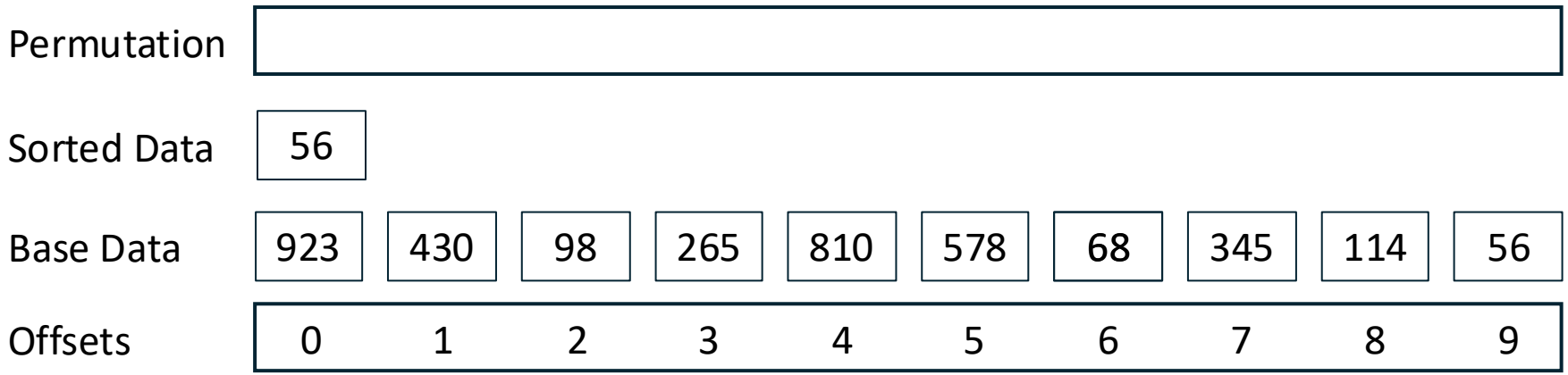
Prior Work


Base Data
923
430
98
265
810
578
68
345
114
56
Offsets
0
1
2
3
4
5
6
7
8
9

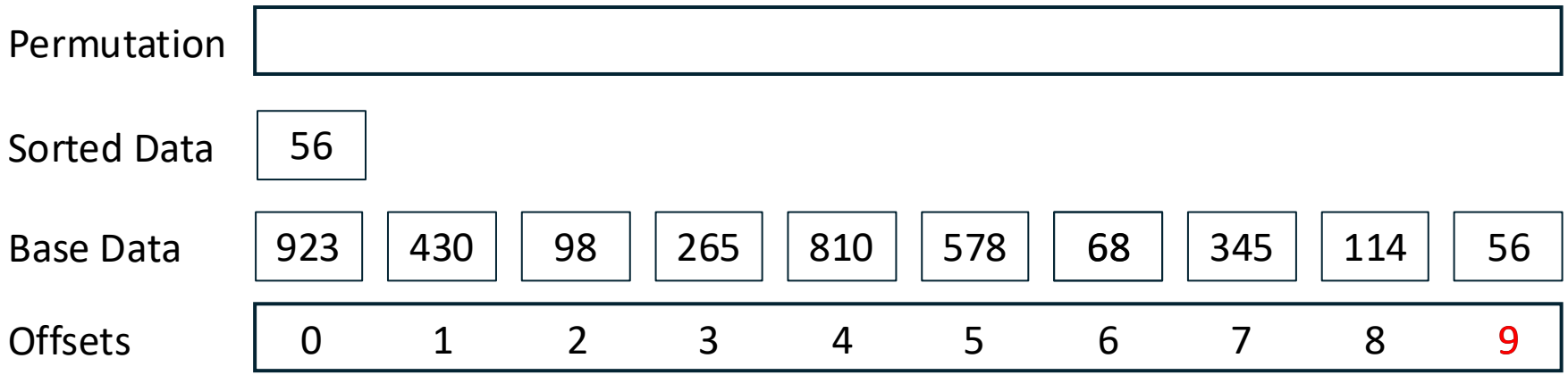
Prior Work


Permutation
Sorted Data
Base Data
923
430
98
265
810
578
68
345
114
56
Offsets
0
1
2
3
4
5
6
7
8
9

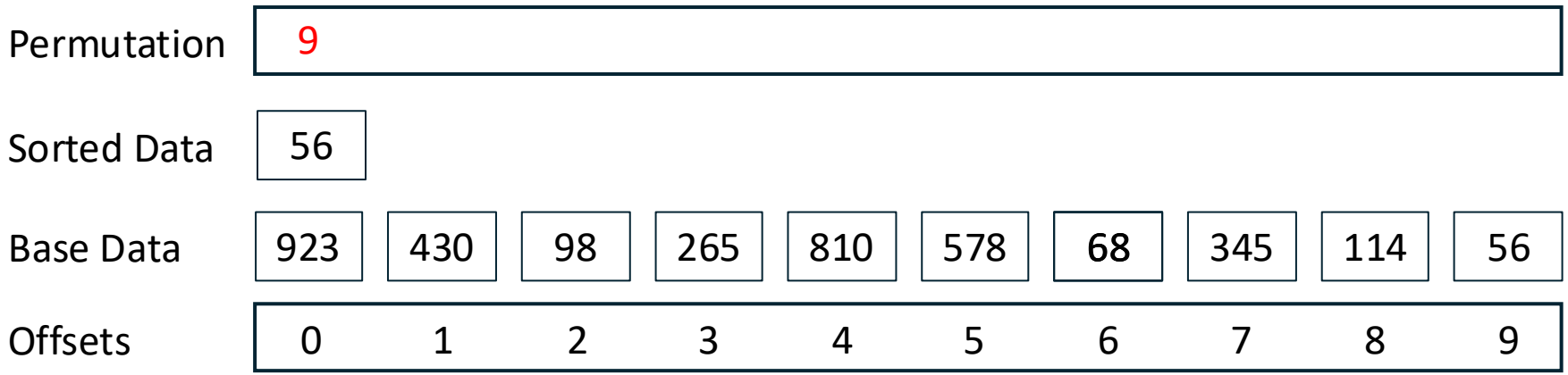
Prior Work


Permutation
Sorted Data
56
Base Data
923
430
98
265
810
578
68
345
114
56
Offsets
0
1
2
3
4
5
6
7
8
9

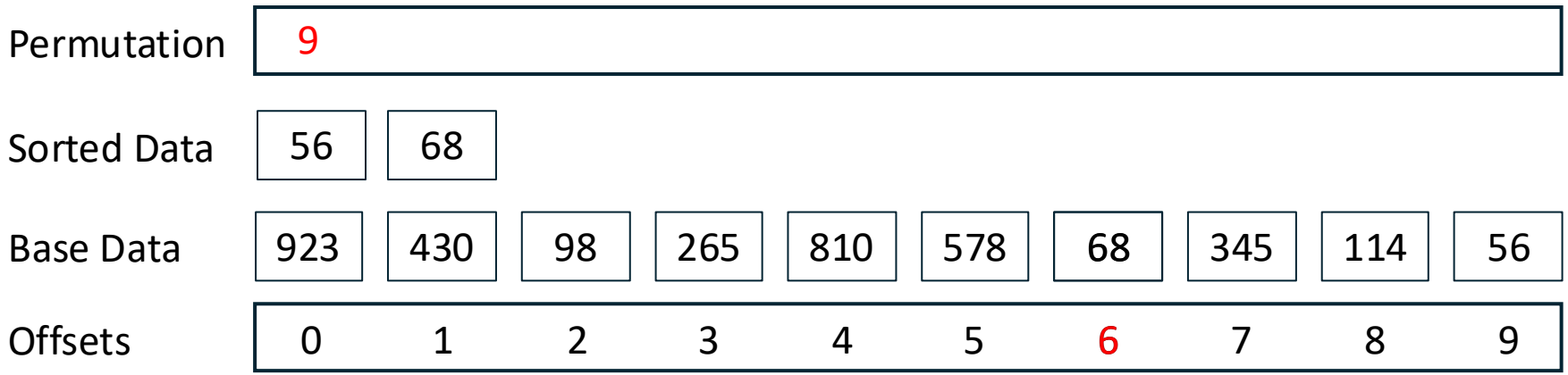
Prior Work


Permutation
Sorted Data
56
Base Data
923
430
98
265
810
578
68
345
114
56
Offsets
0
1
2
3
4
5
6
7
8
9

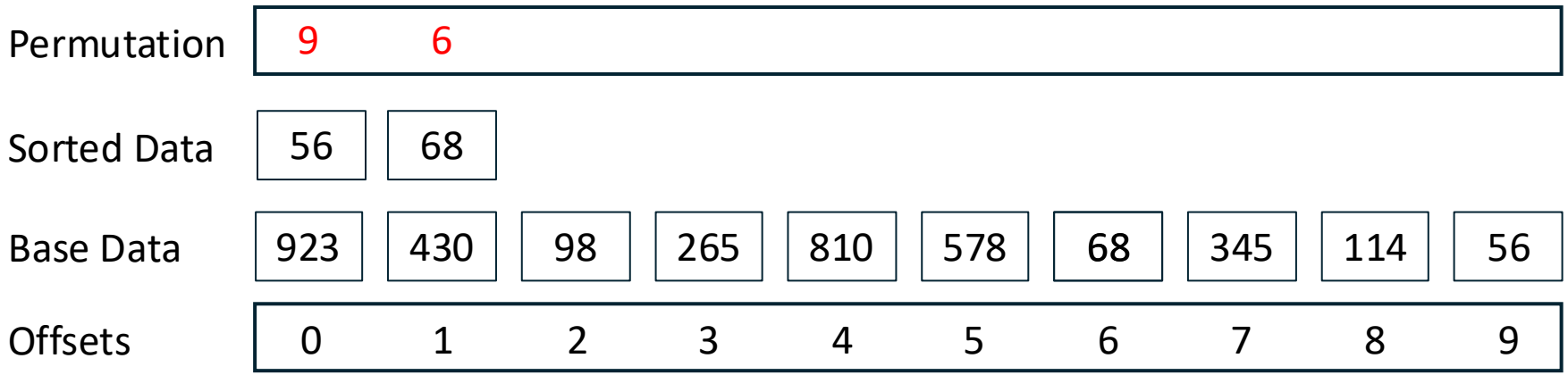
Prior Work


Permutation
9
Sorted Data
56
Base Data
923
430
98
265
810
578
68
345
114
56
Offsets
0
1
2
3
4
5
6
7
8
9

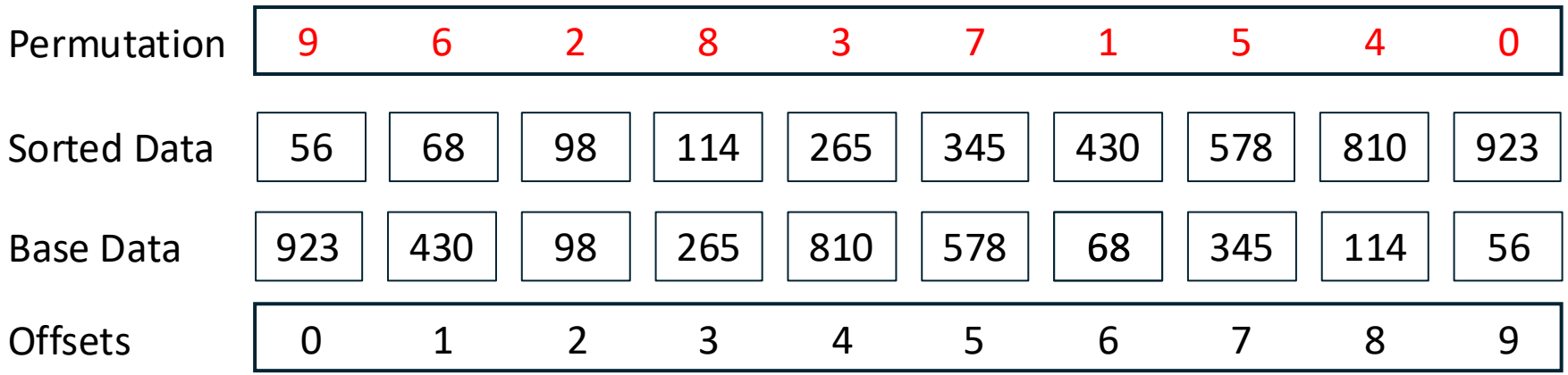
Prior Work


Permutation
9
Sorted Data
56
68
Base Data
923
430
98
265
810
578
68
345
114
56
Offsets
0
1
2
3
4
5
6
7
8
9

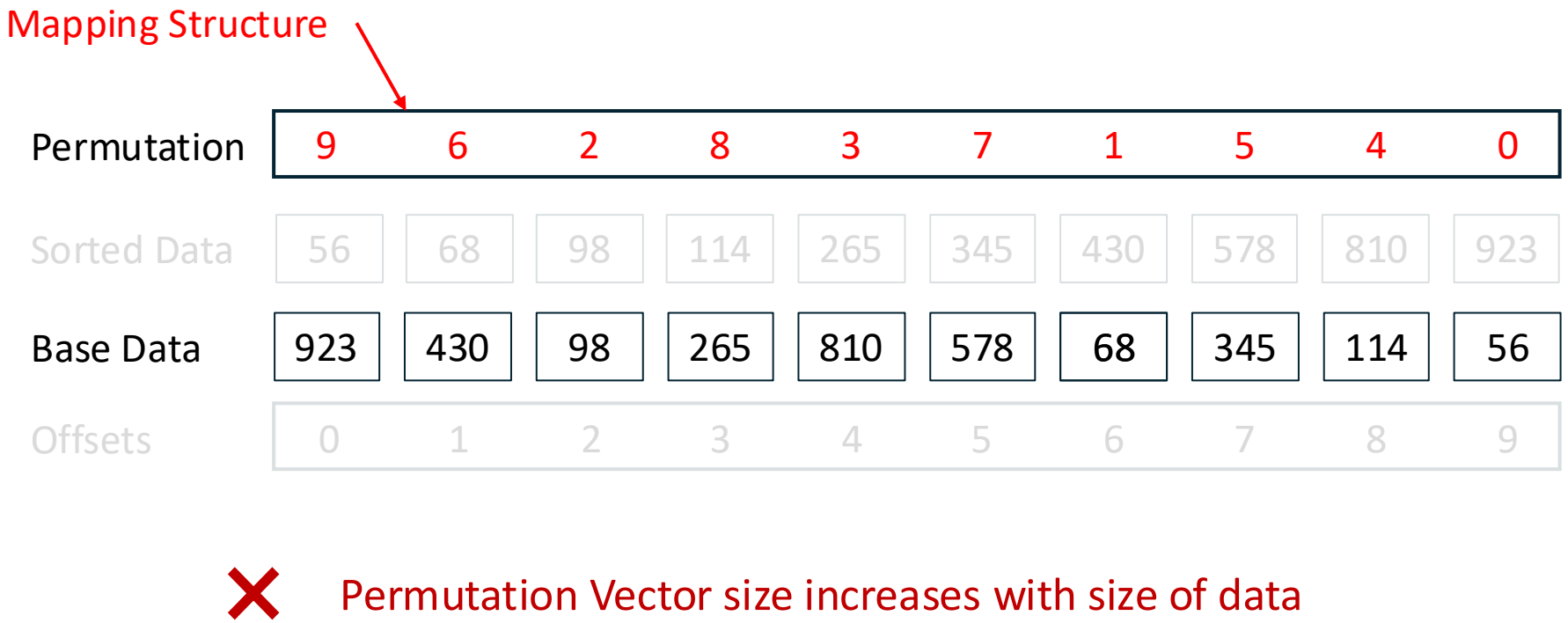
Prior Work


Permutation
9
6
Sorted Data
56
68
Base Data
923
430
98
265
810
578
68
345
114
56
Offsets
0
1
2
3
4
5
6
7
8
9

Prior Work

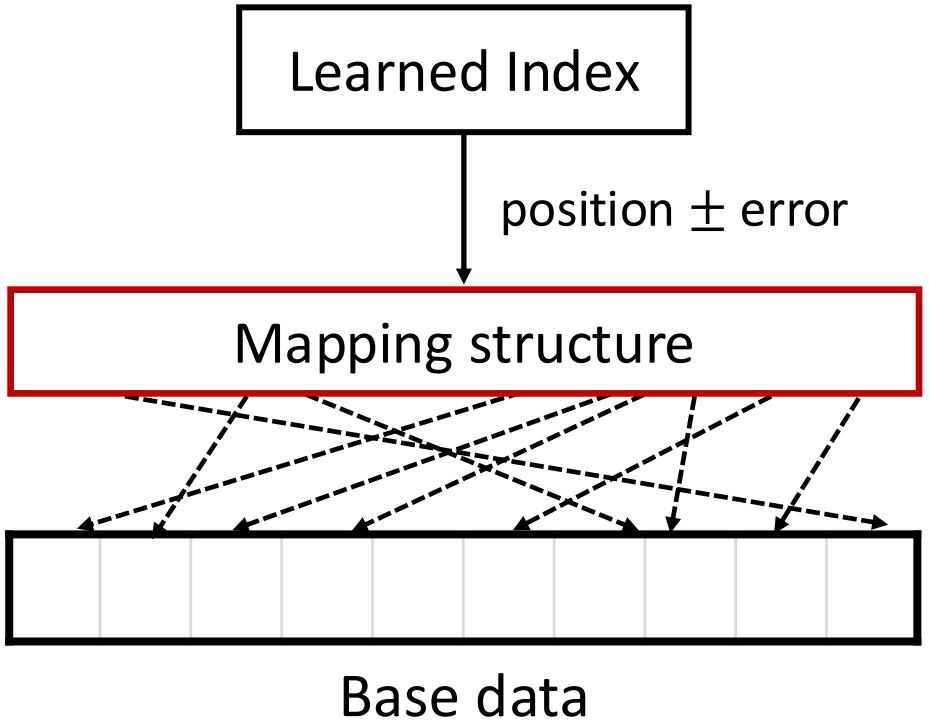

Permutation
9 6 2 8 3 7 1 5 4 0
Sorted Data
56 68 98 114 265 345 430 578 810 923
Base Data
923 430 98 265 810 578 68 345 114 56
Offsets
0 1 2 3 4 5 6 7 8 9

Prior Work

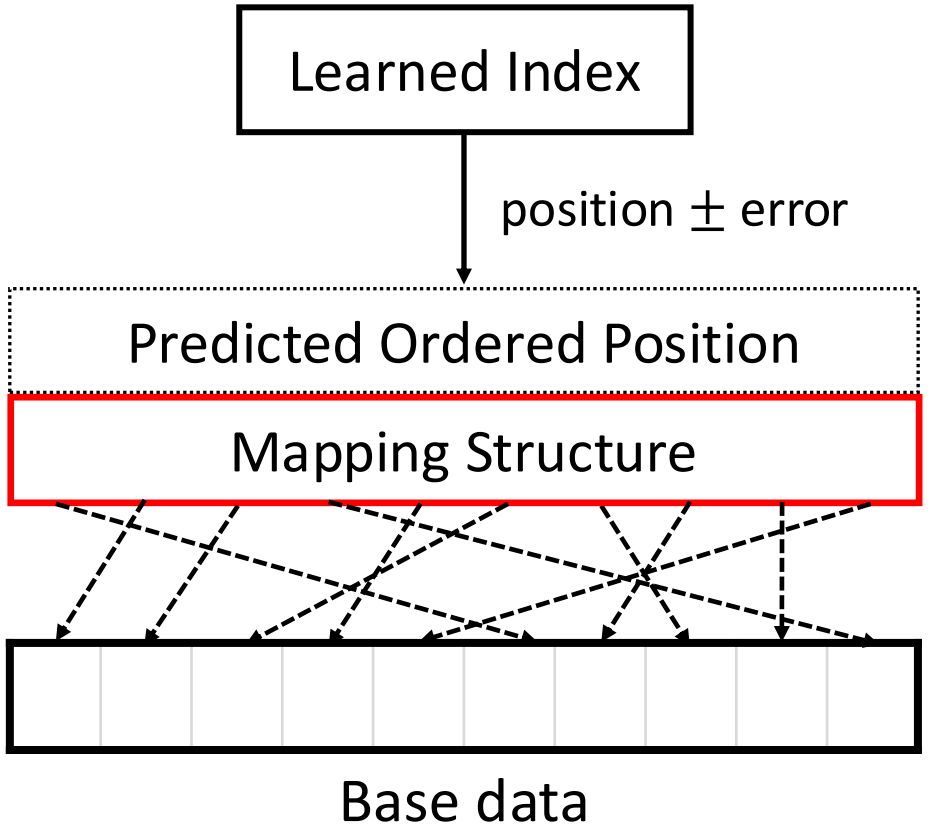


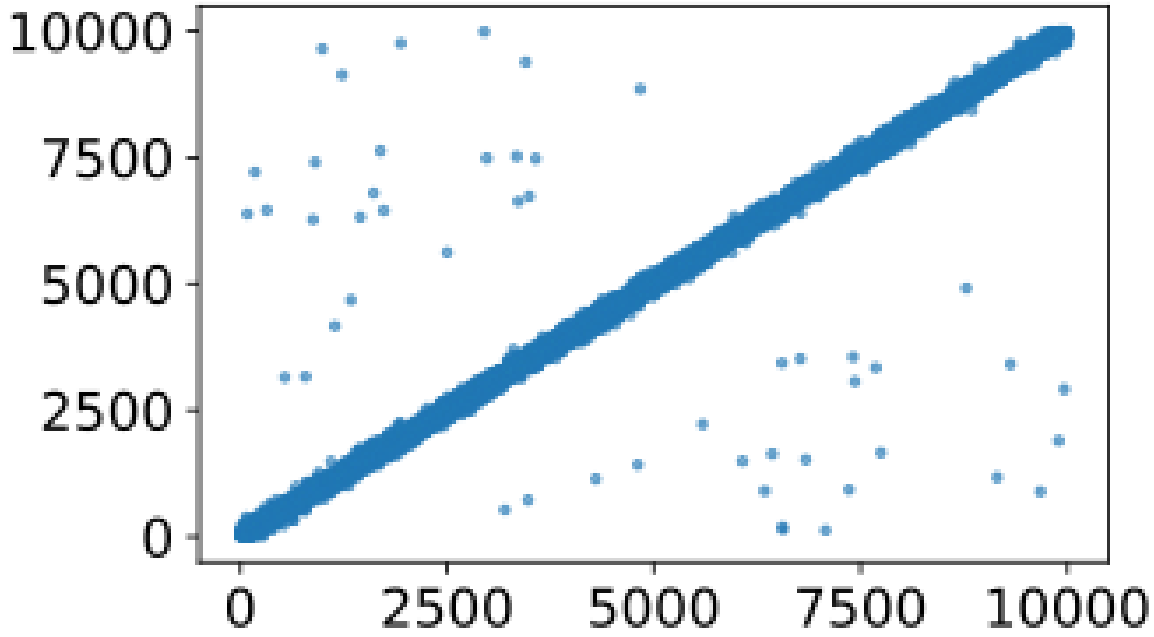
✗ Permutation Vector size increases with size of data

What do we Want?

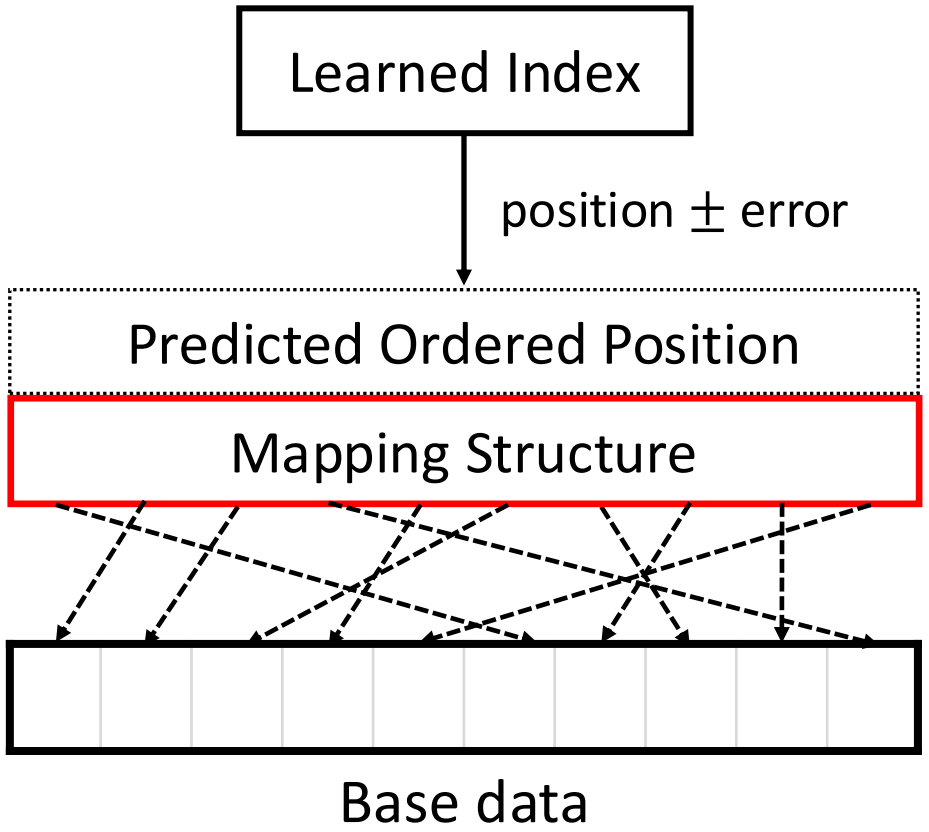


Constellation Maps as Mapping Structures

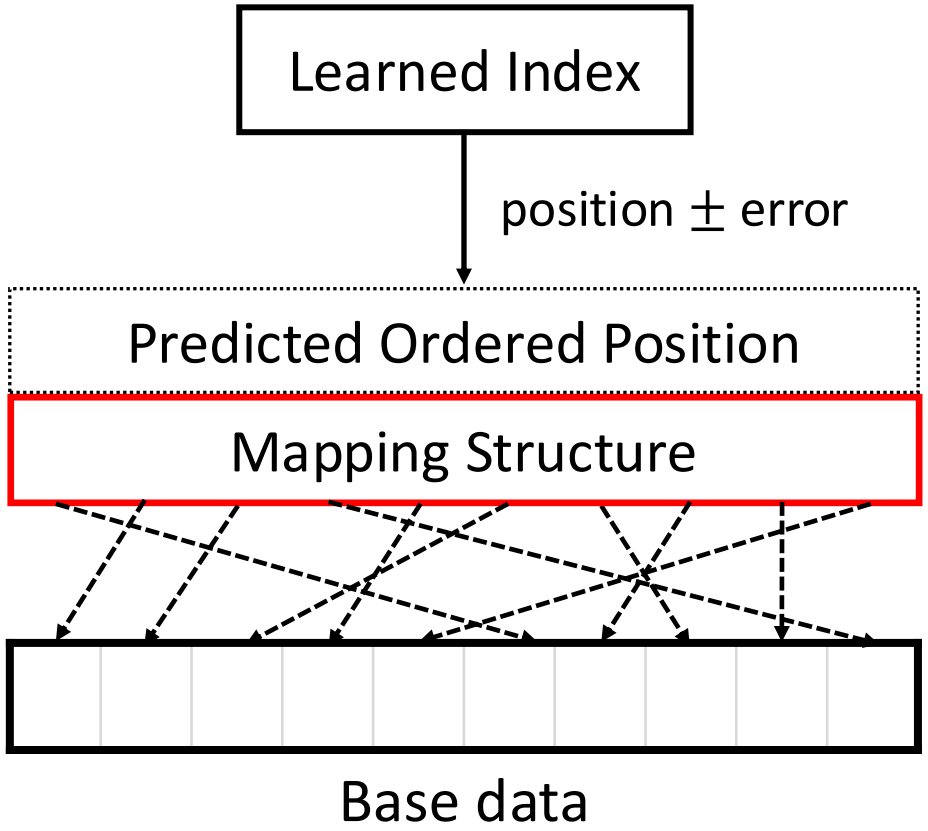


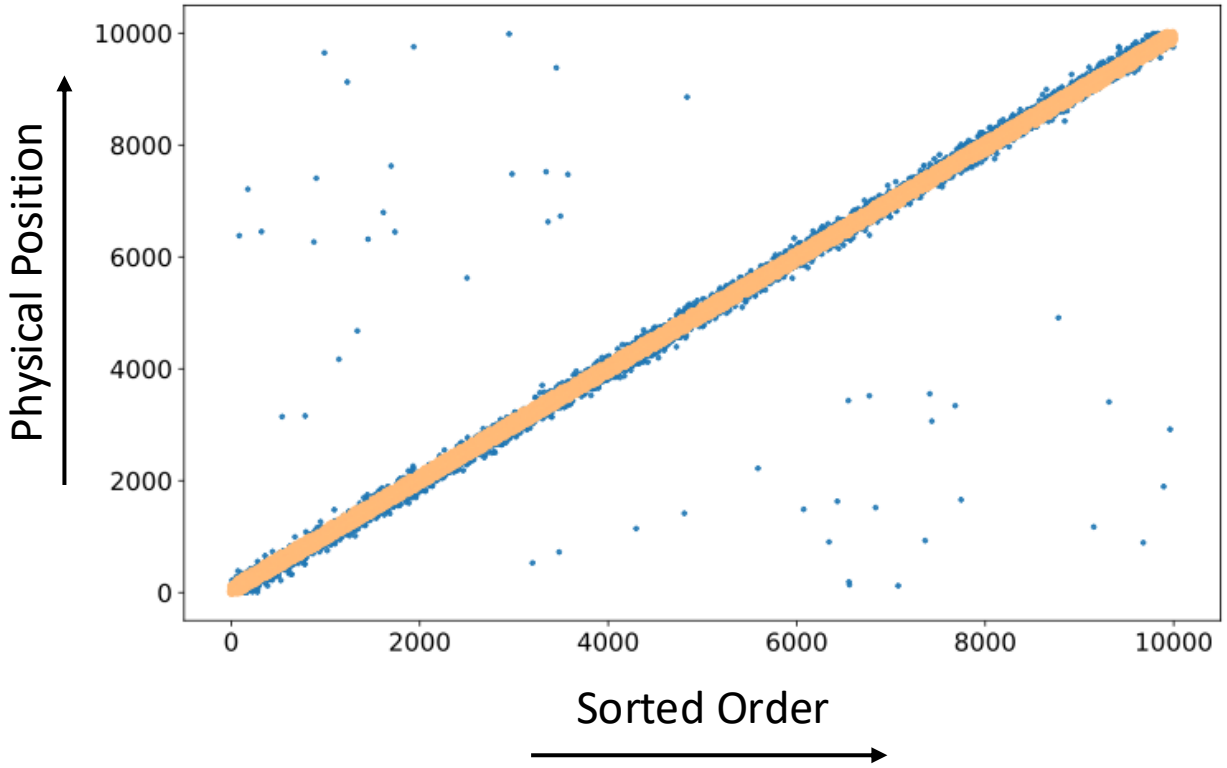


Constellation Maps as Mapping Structures

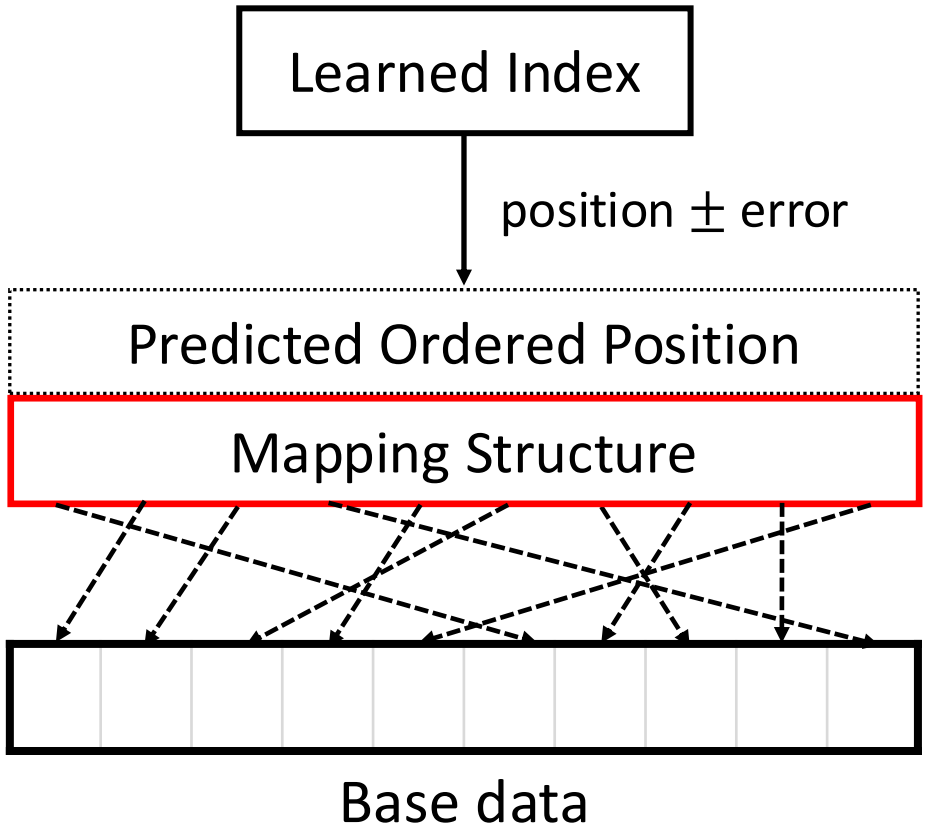


Constellation Maps as Mapping Structures

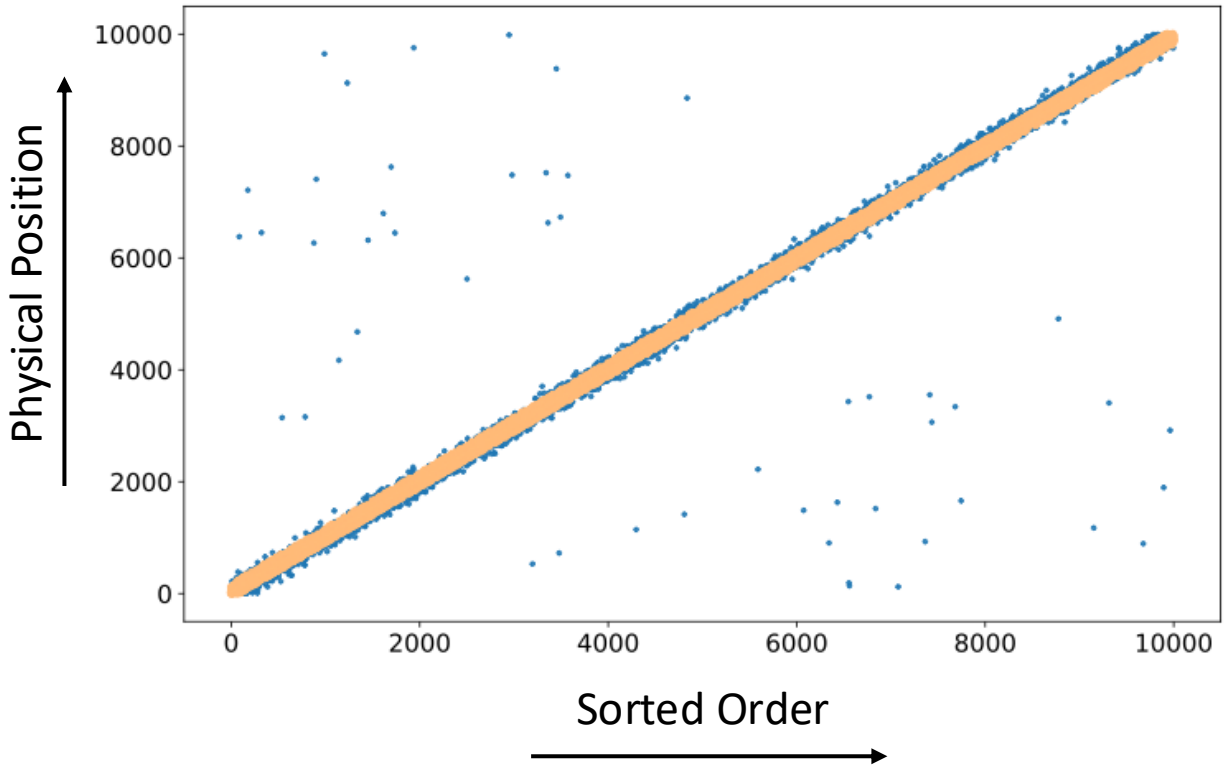




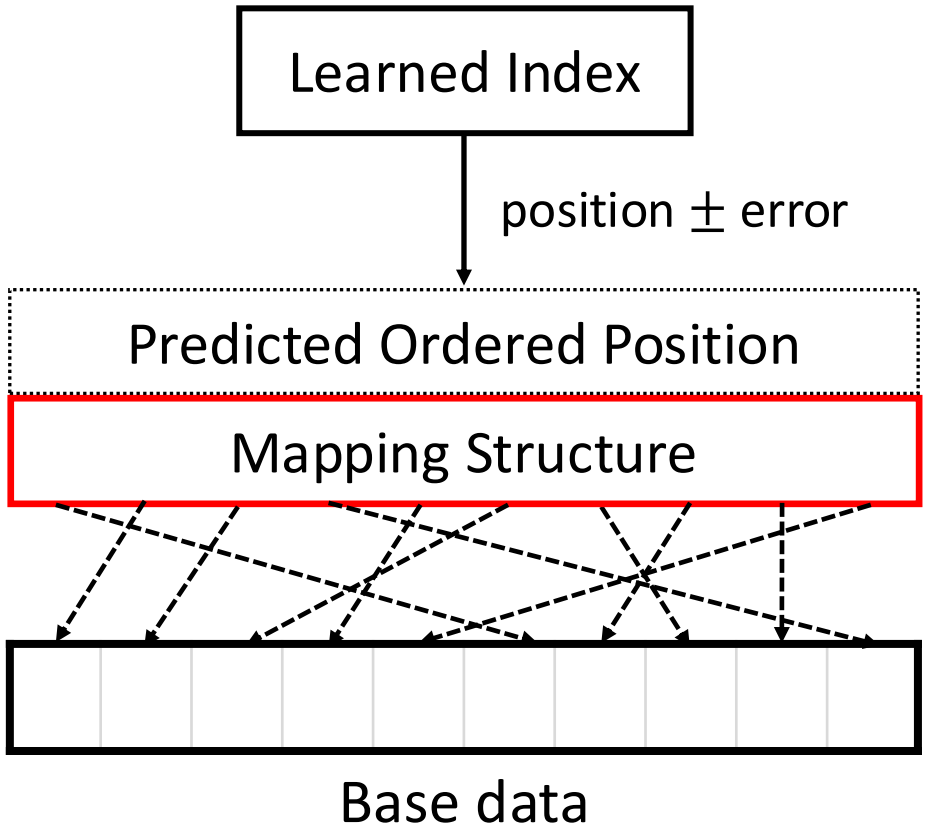
Constellation Maps as Mapping Structures

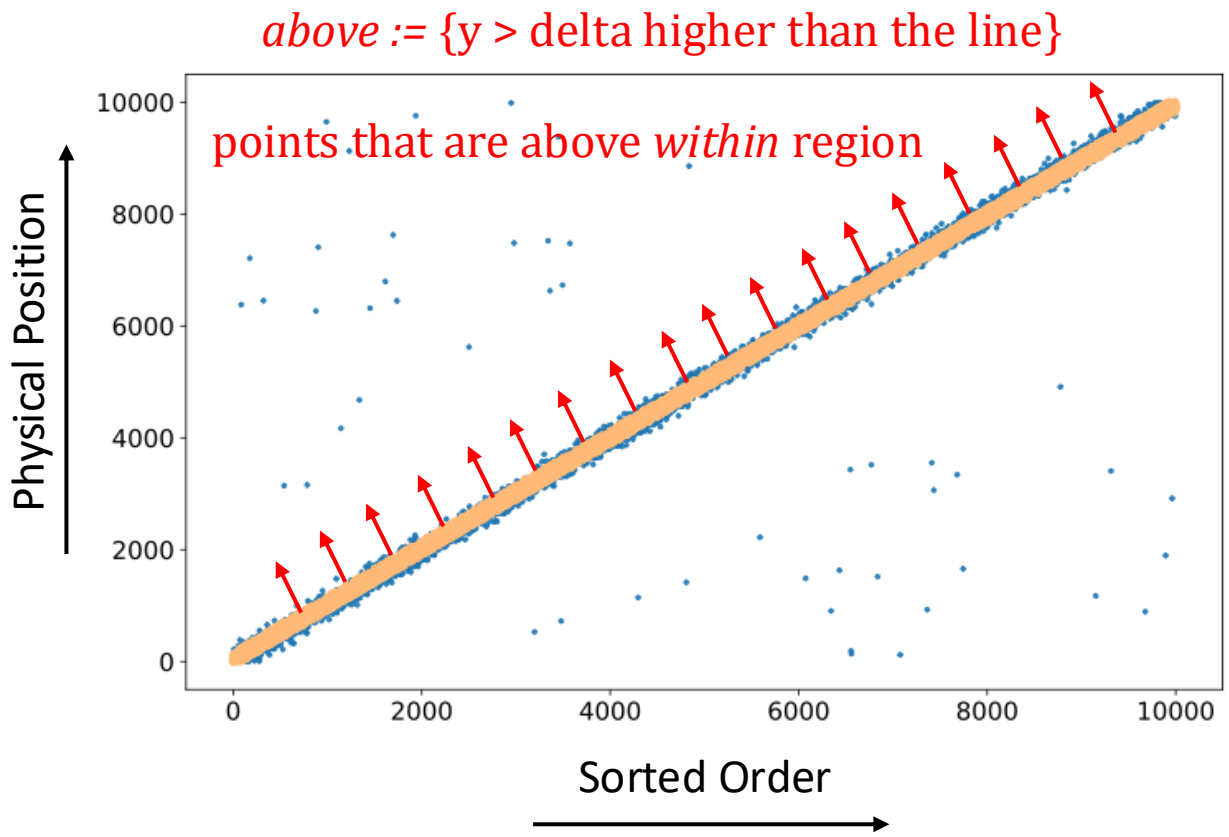


within := {y ≤ delta distance from line}

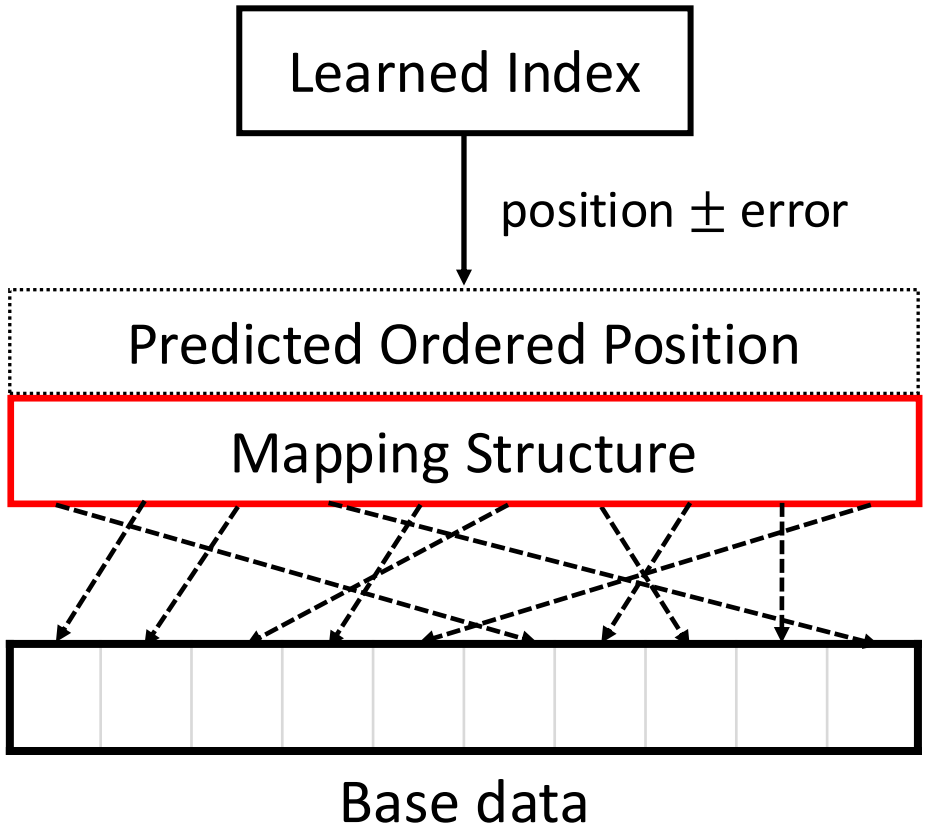


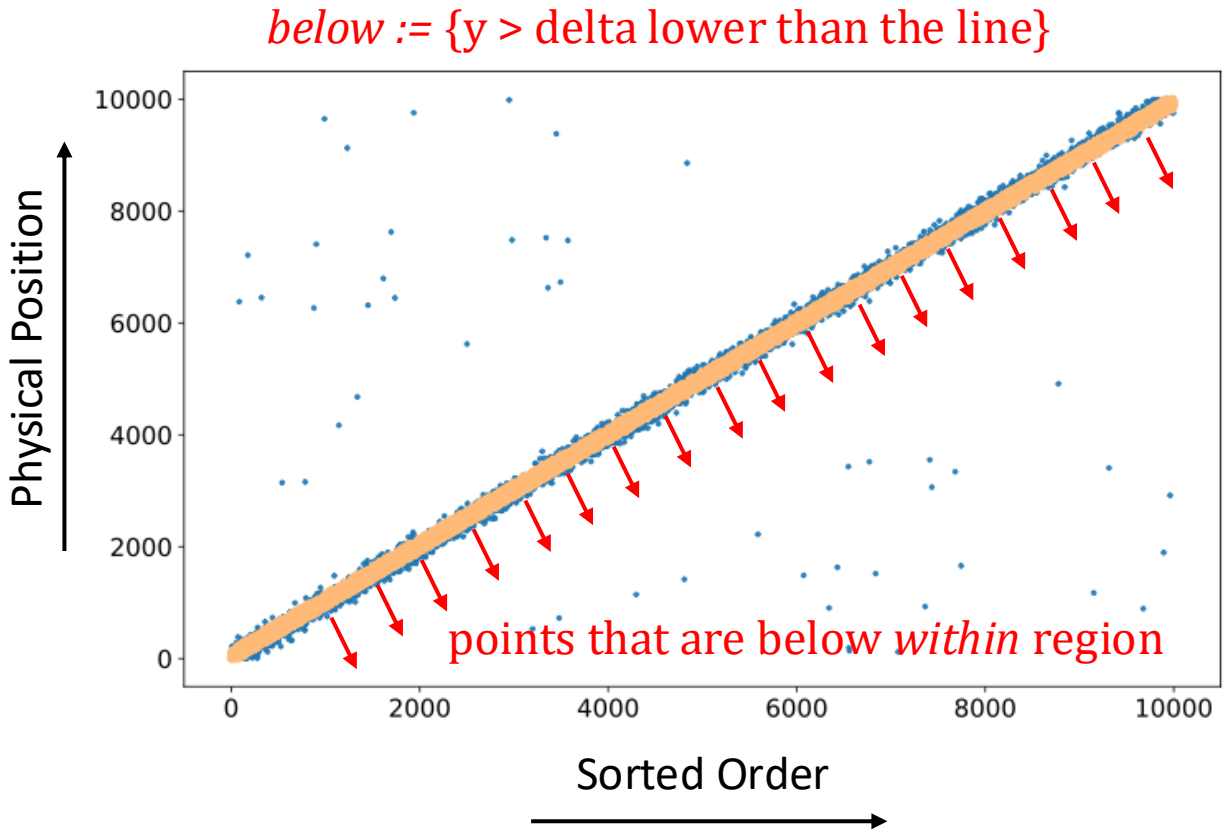
Constellation Maps as Mapping Structures



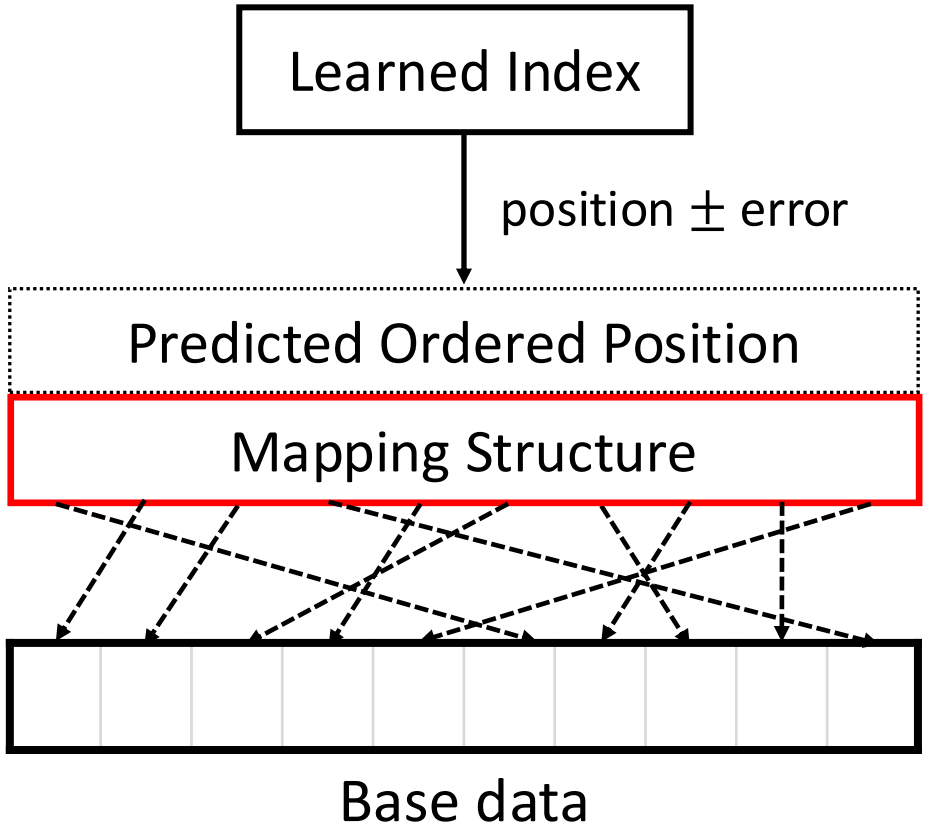


Constellation Maps as Mapping Structures

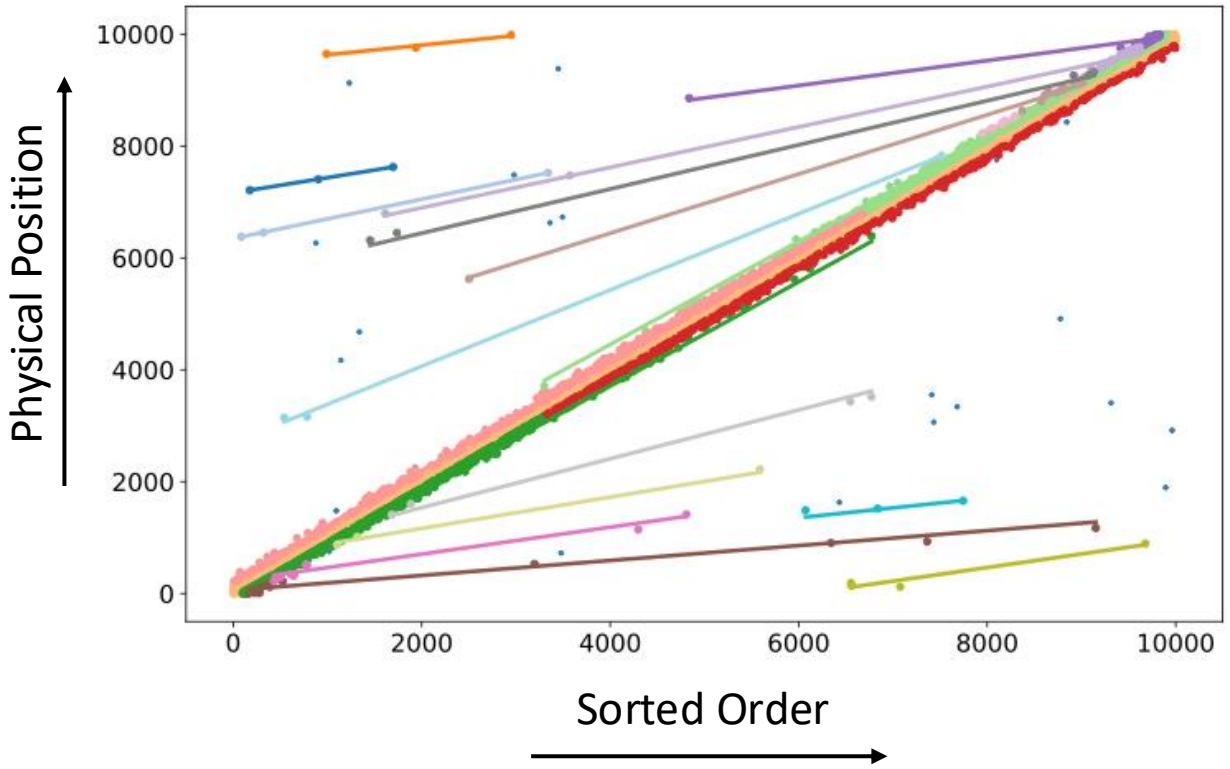




Constellation Maps as Mapping Structures

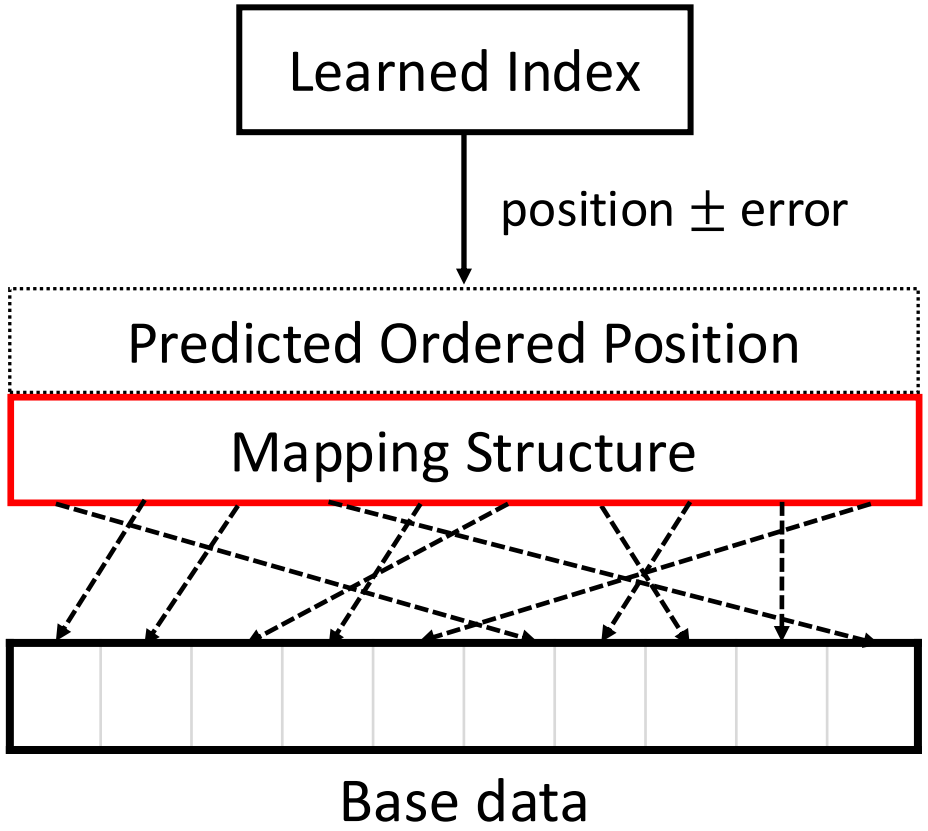


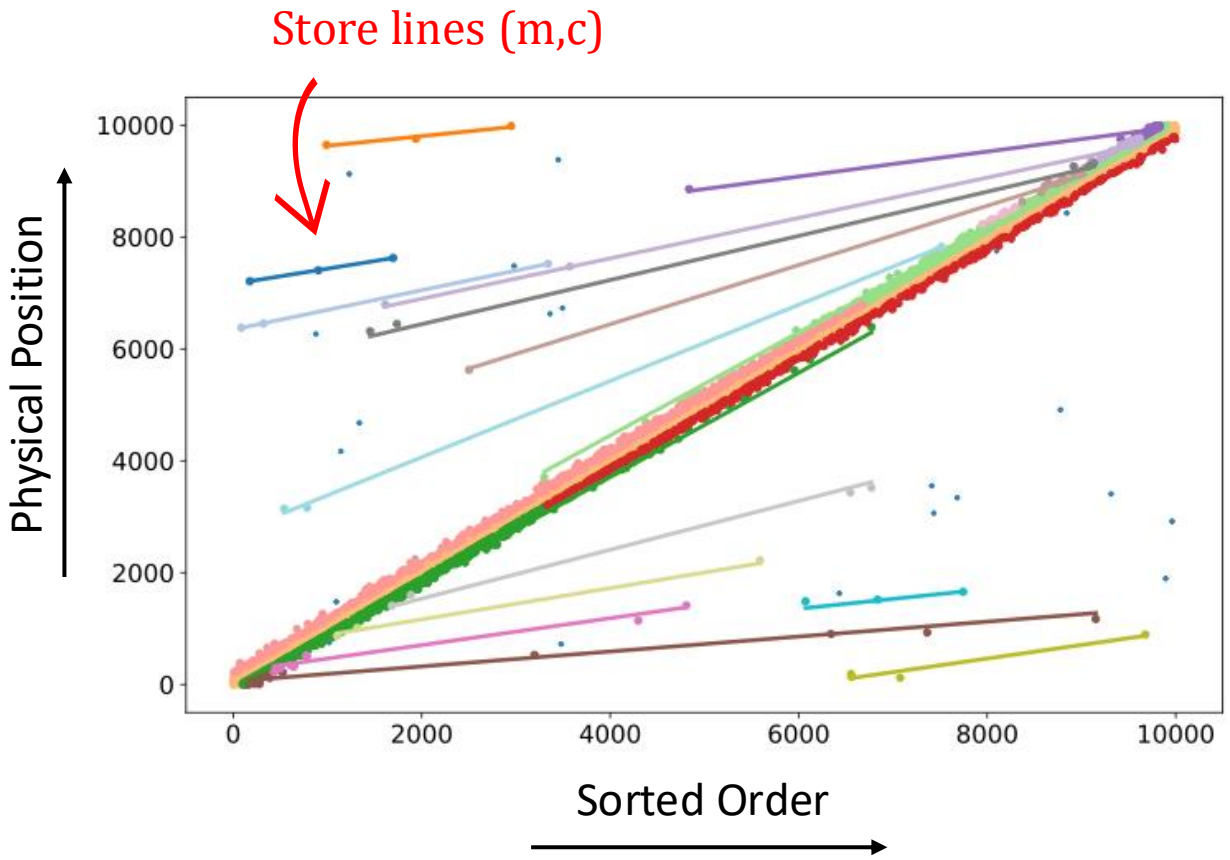
After 20 iterations



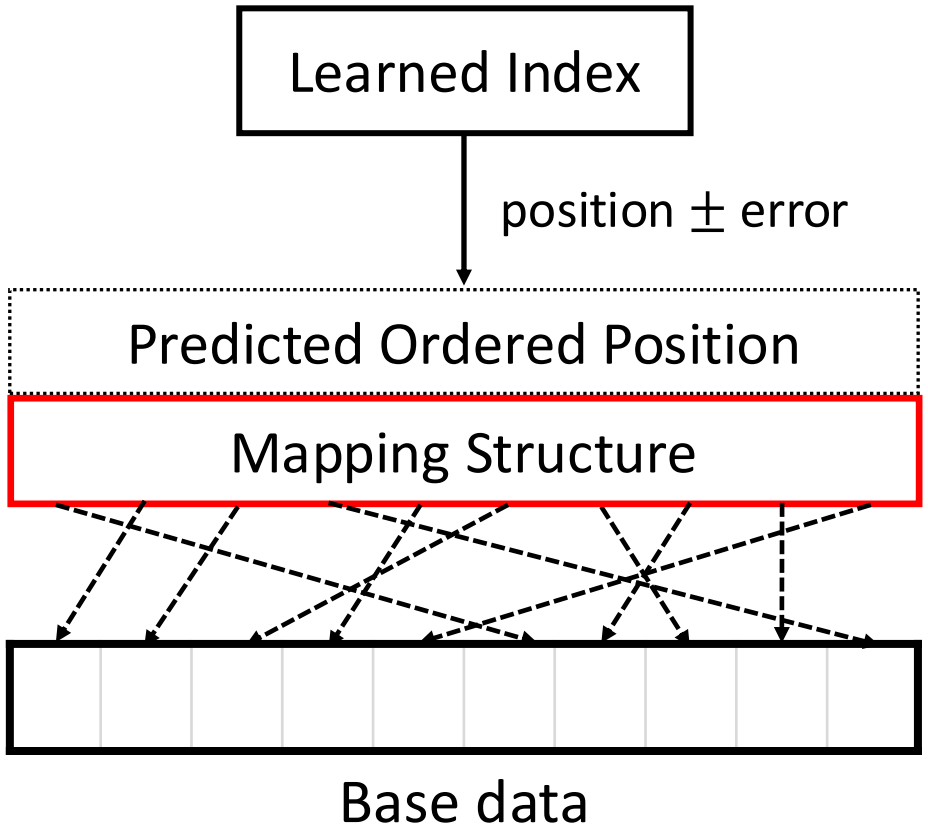
No line can cross any other line

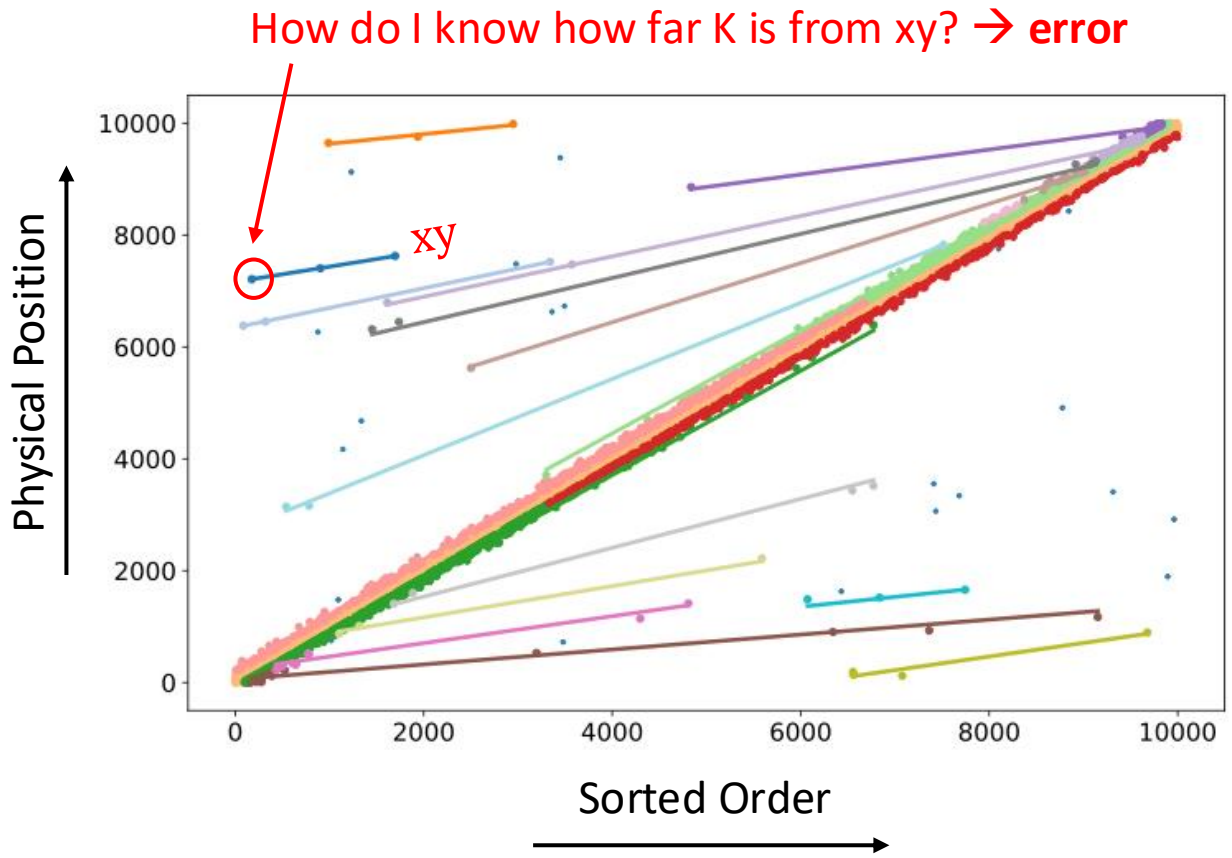
Constellation Maps as Mapping Structures





Constellation Maps as Mapping Structures

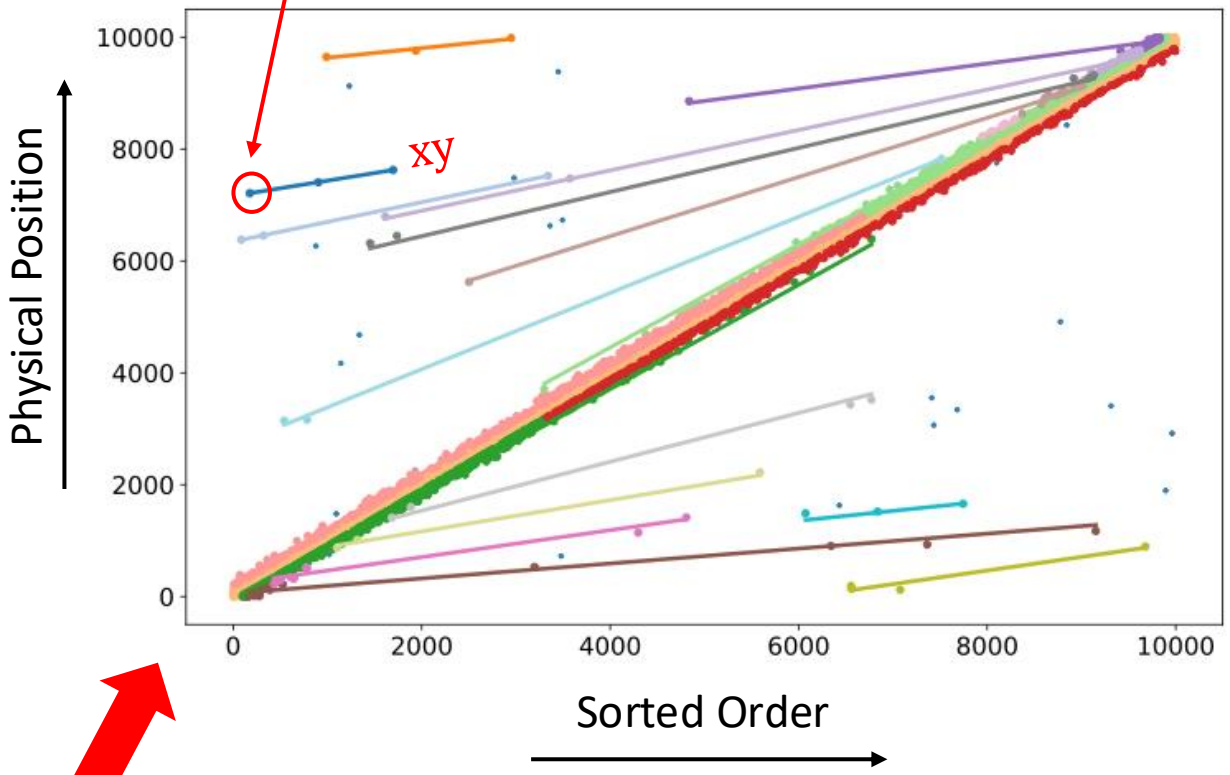




Constellation Maps as Mapping Structures

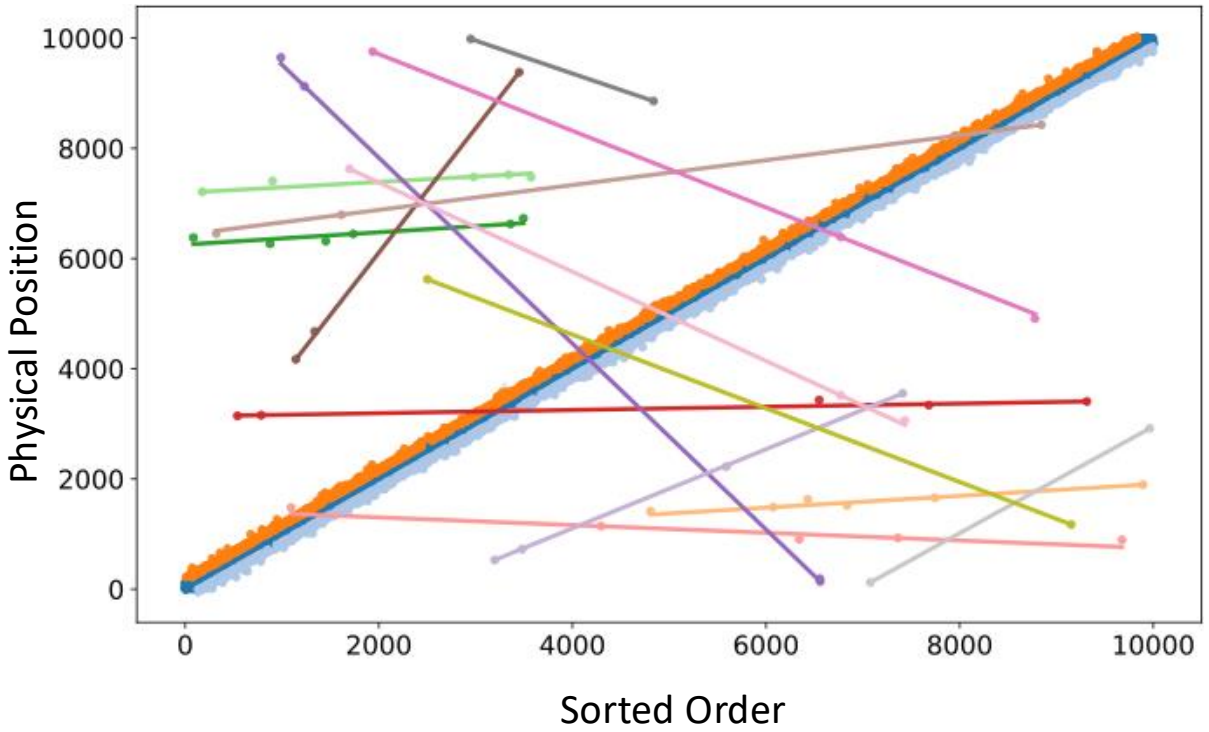
How do I know point is mapped by line xy? → mapping

How do I know how far K is from xy? → **error**

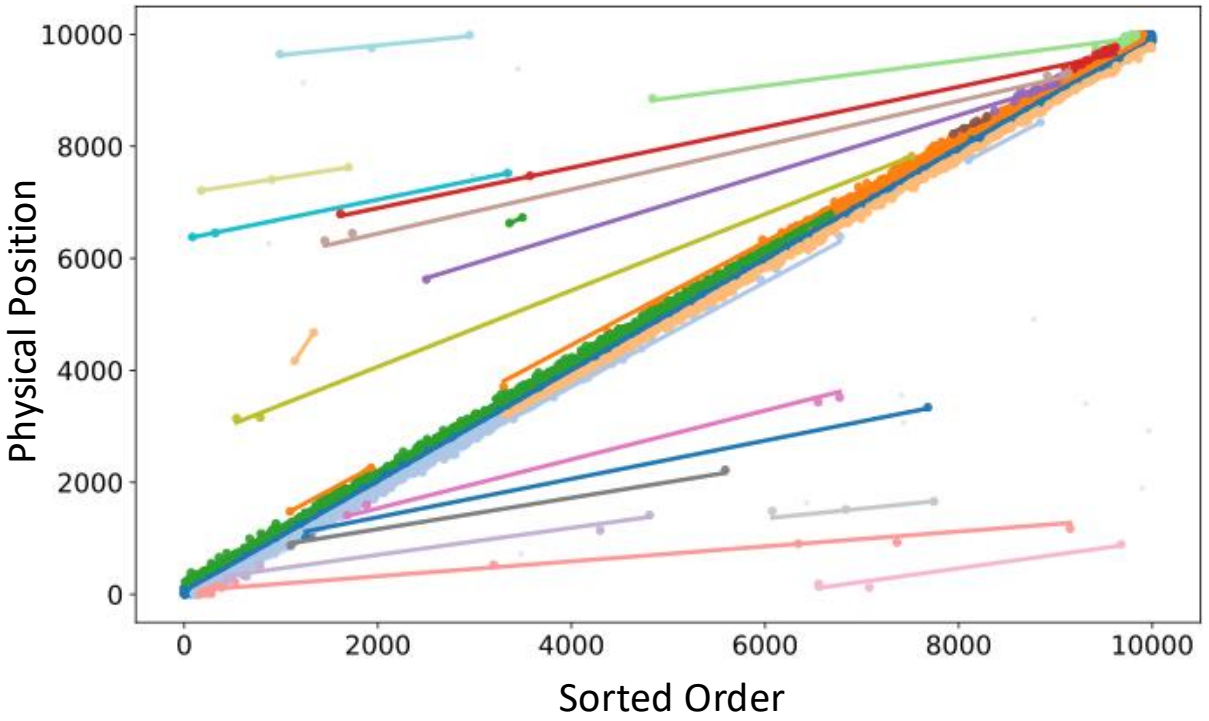


Constellation Maps!!!

Constellation Maps in Action



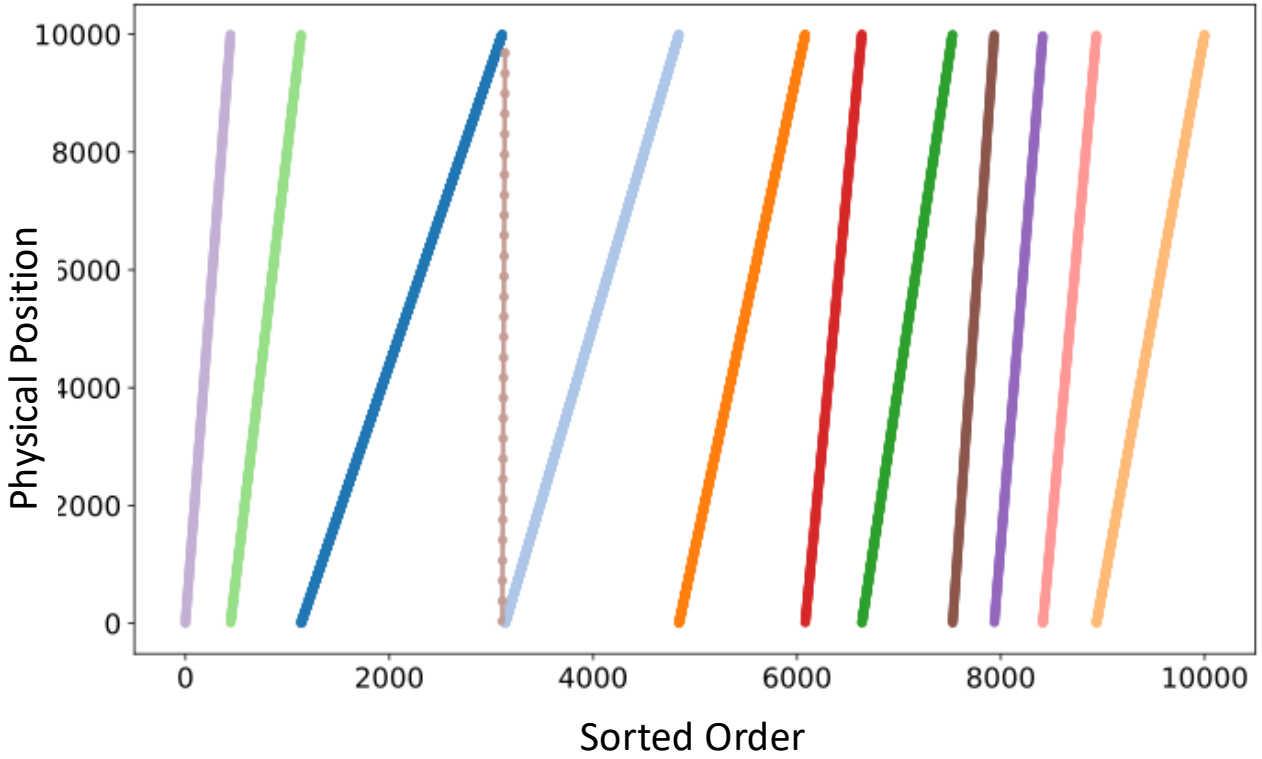
RANSAC = 17 lines



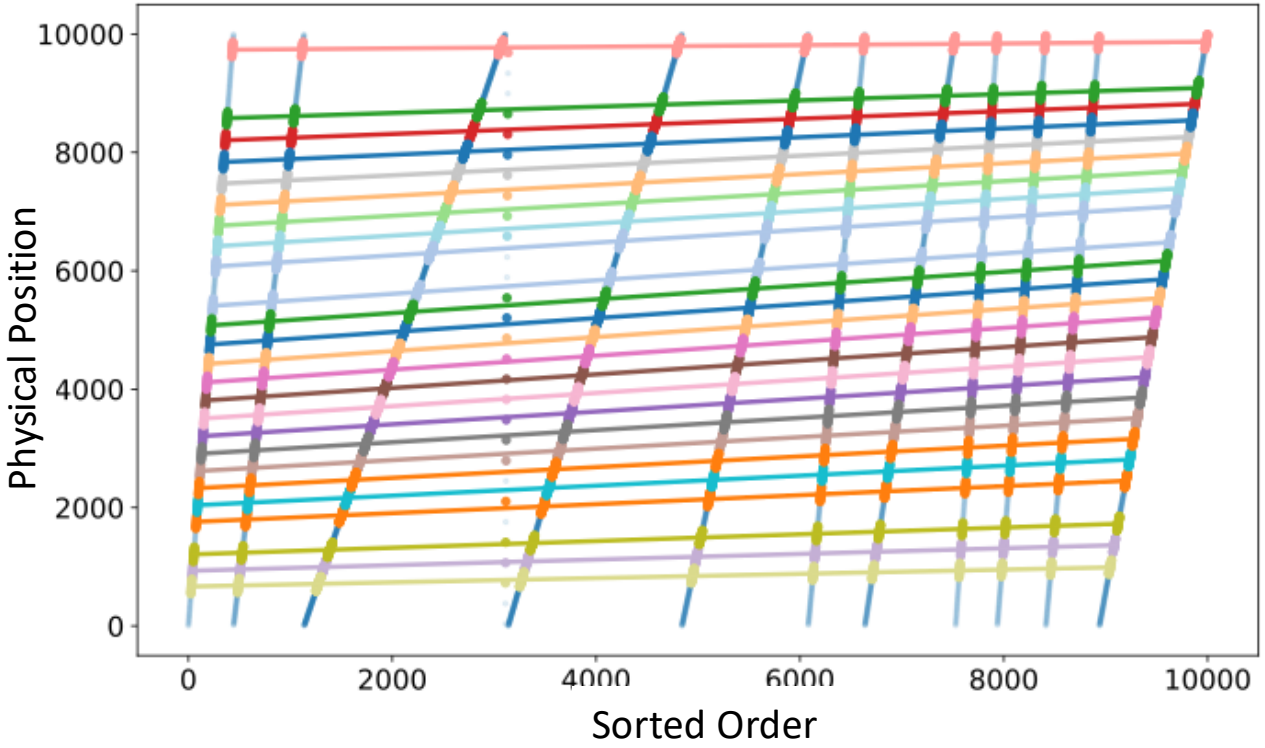
LR = 29 lines

Datasets were donated by an anonymous corporation

Constellation Maps in Action



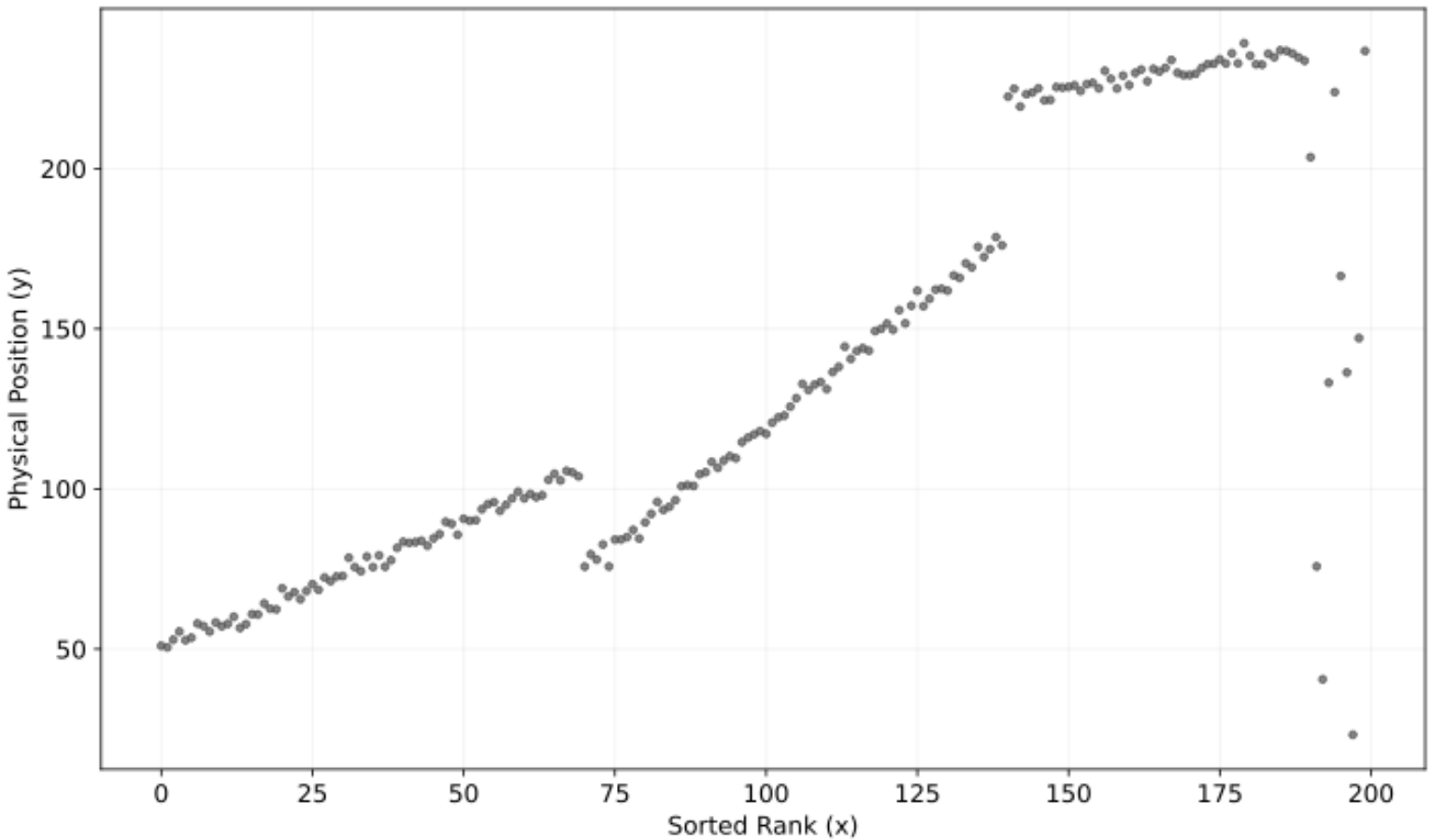
RANSAC = 12 lines



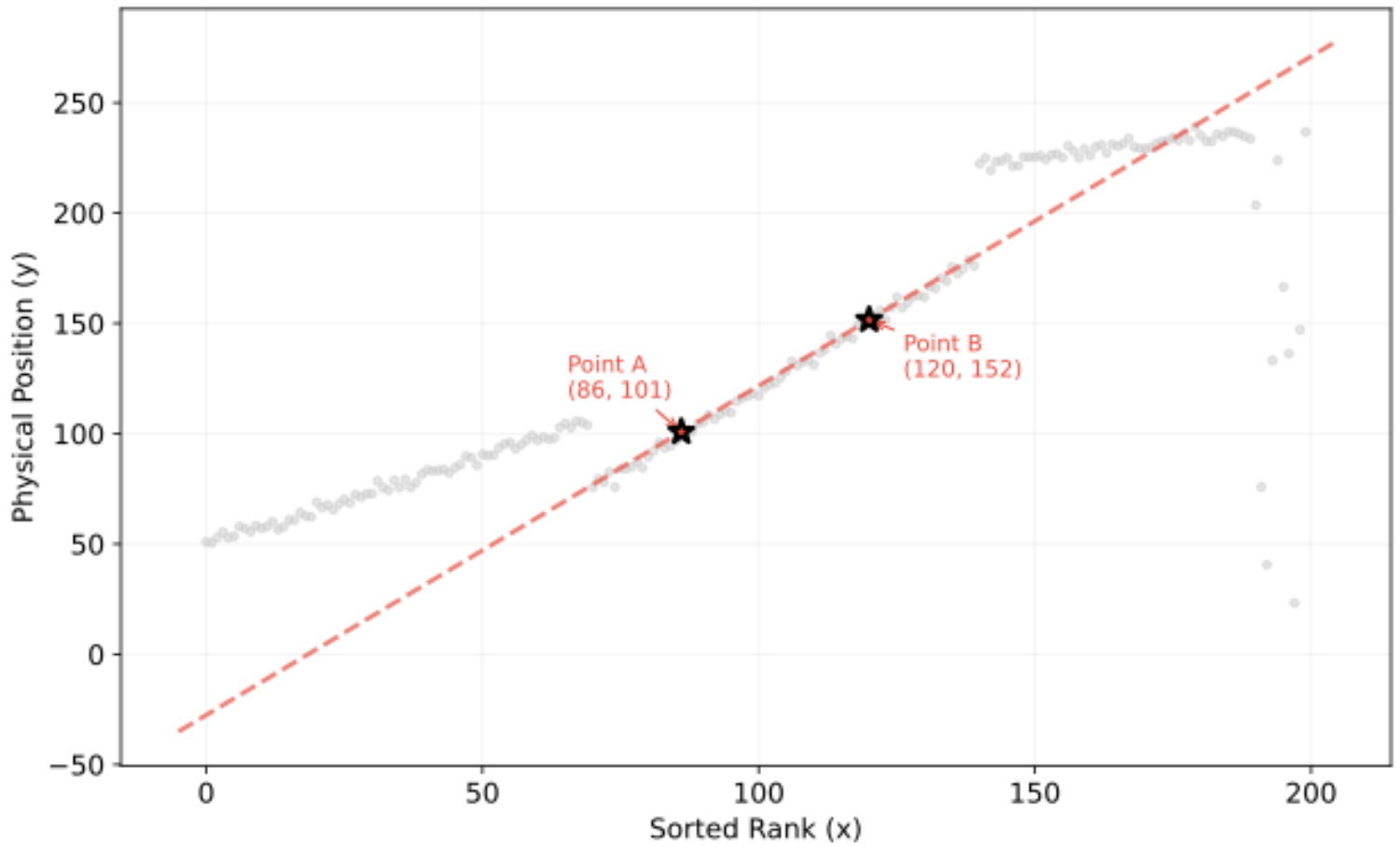
LR = 63 lines

Datasets were donated by an anonymous corporation

Problems with Default Algorithm

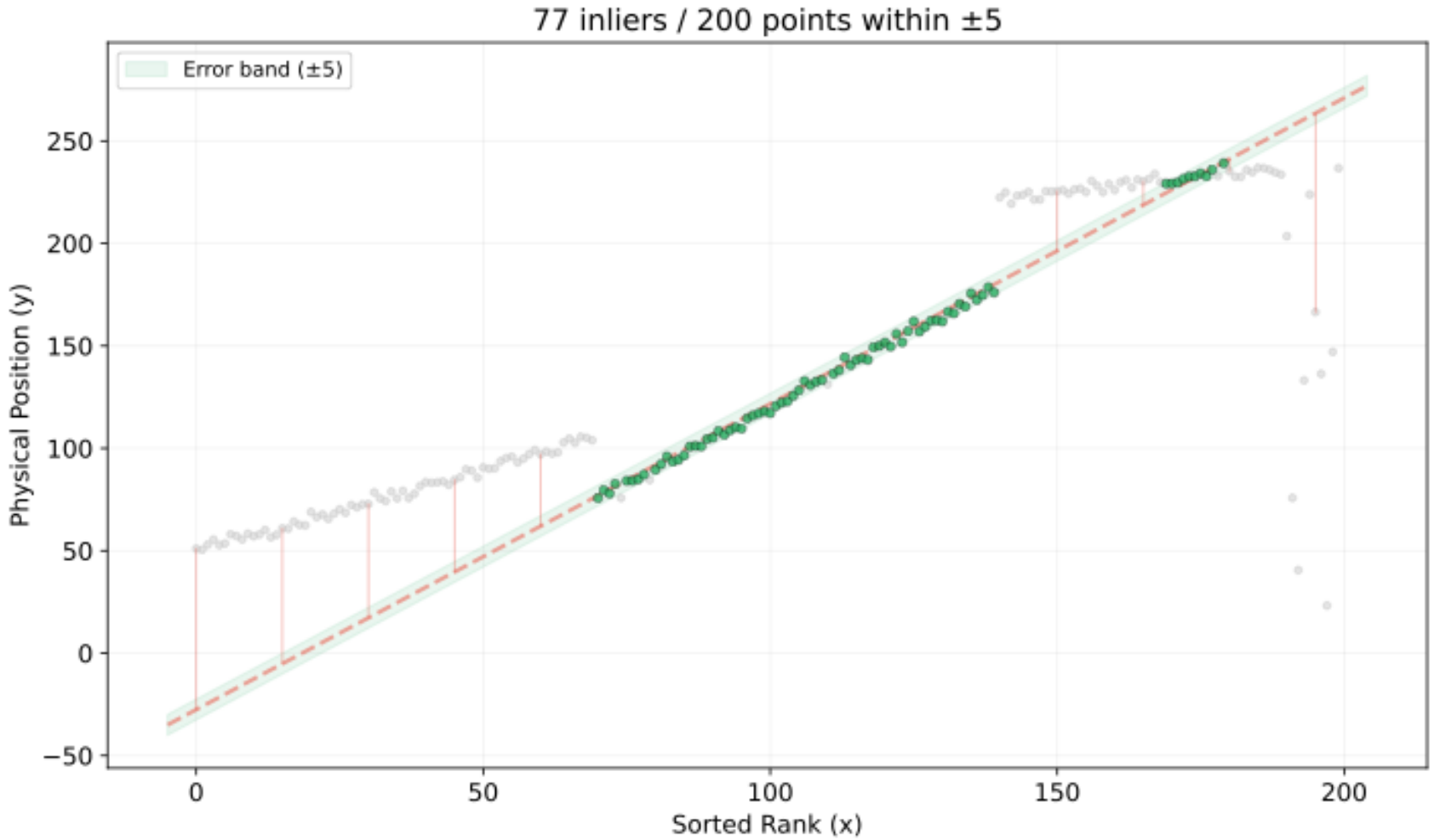


Problems with Default Algorithm



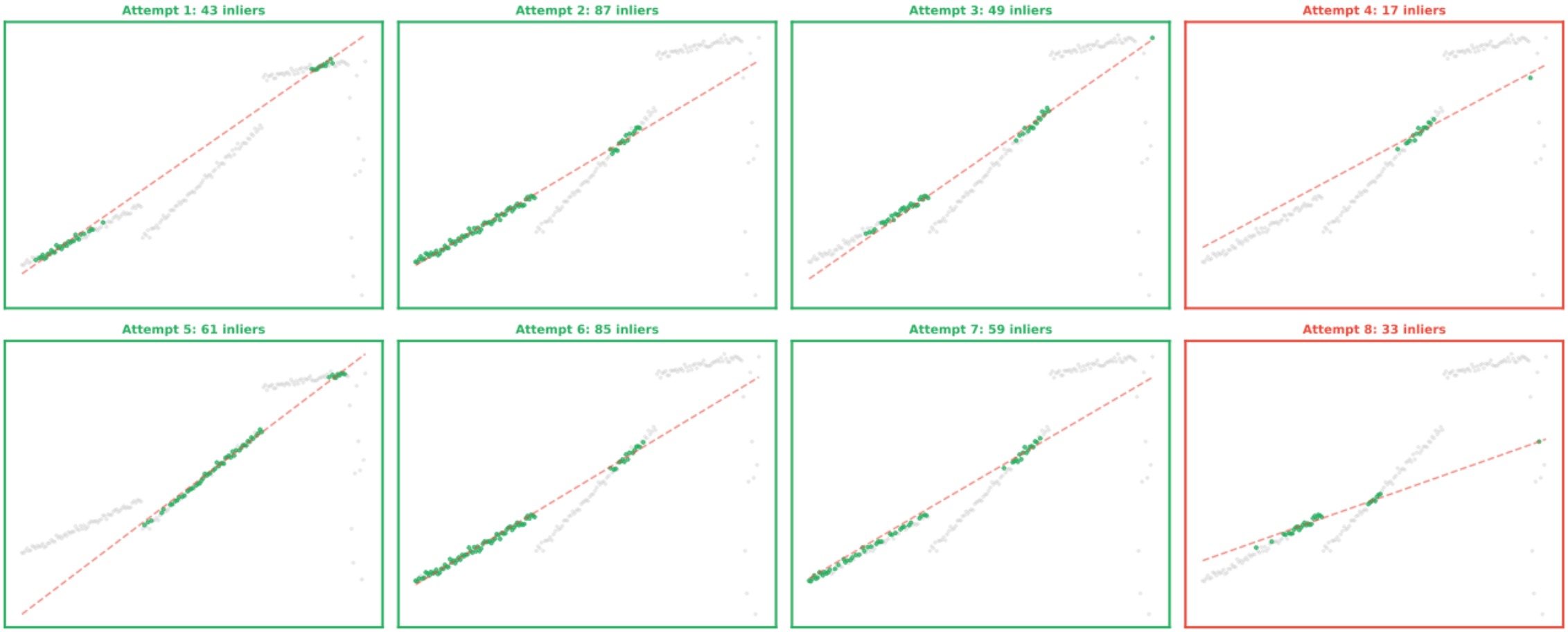
2 random points selected to form a line

Problems with Default Algorithm



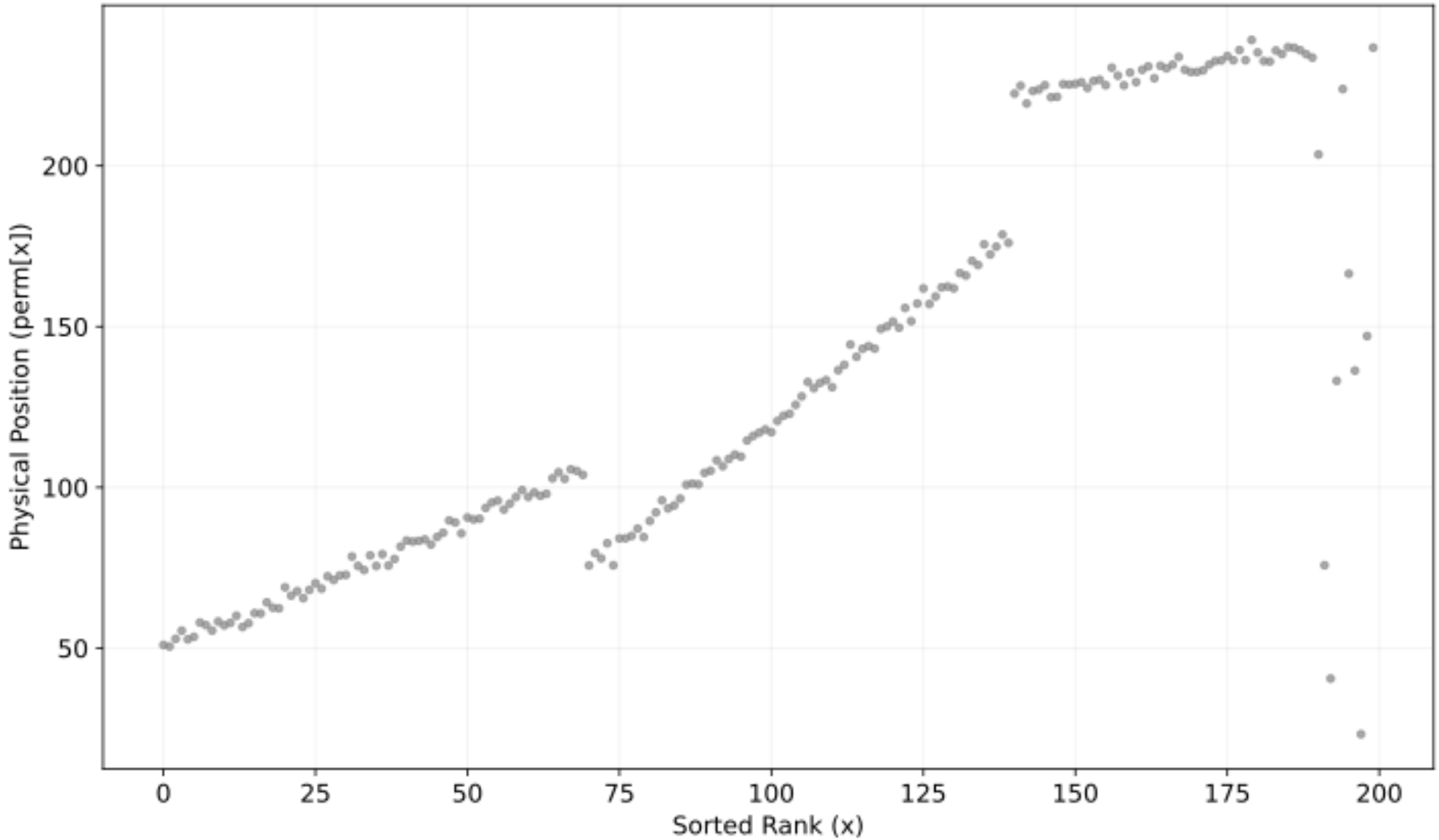
Inliers within error bounds

Problems with Default Algorithm

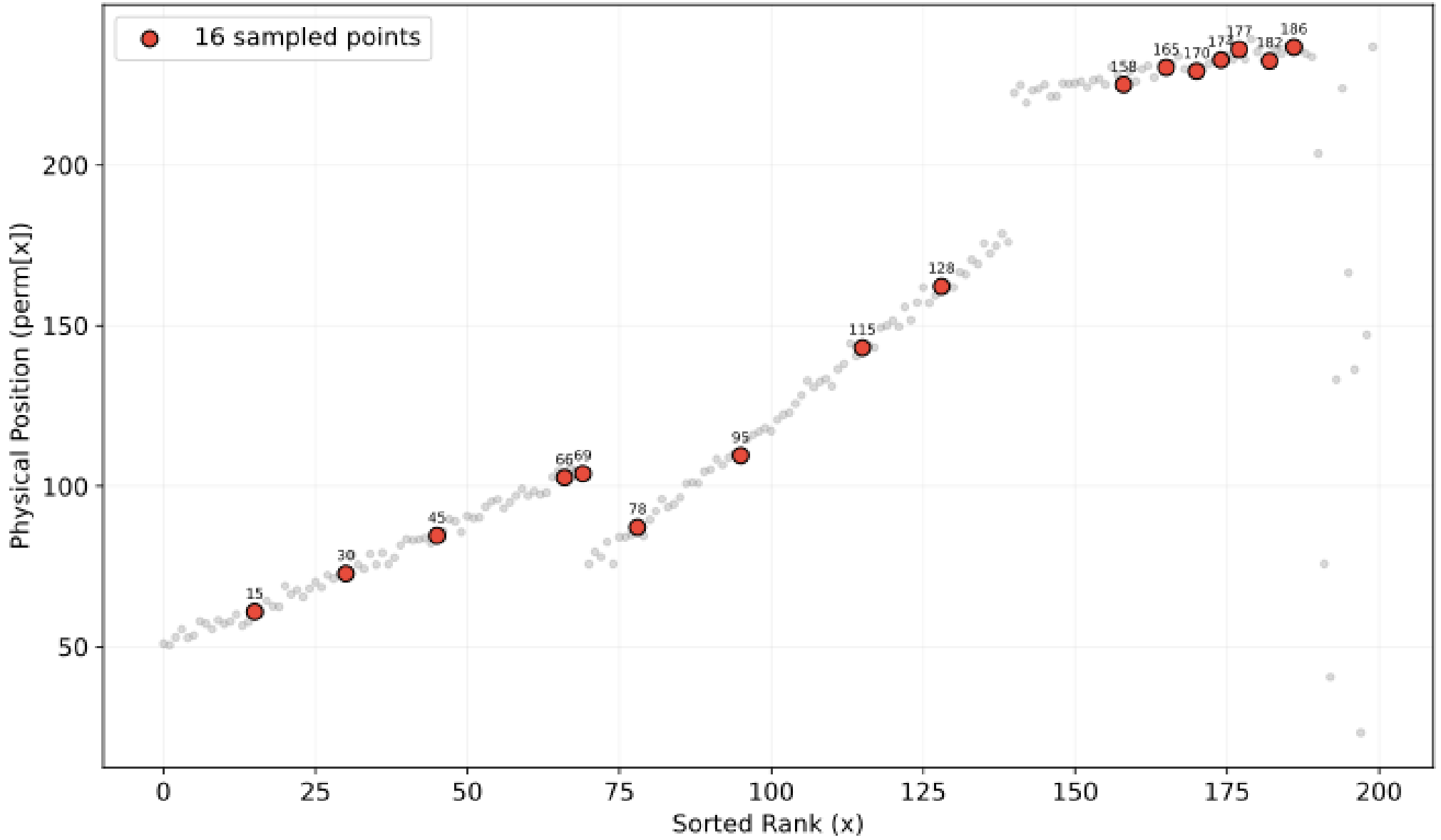


Multiple trials ≠ Better Lines

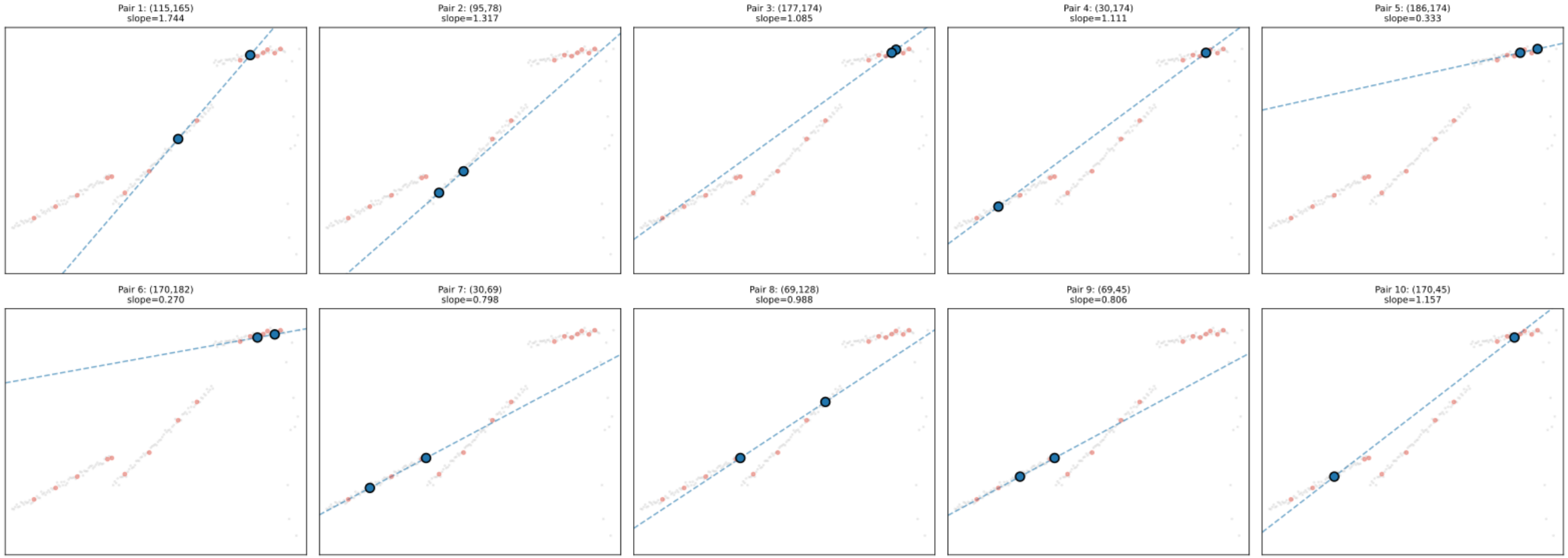
Building a Better Algorithm



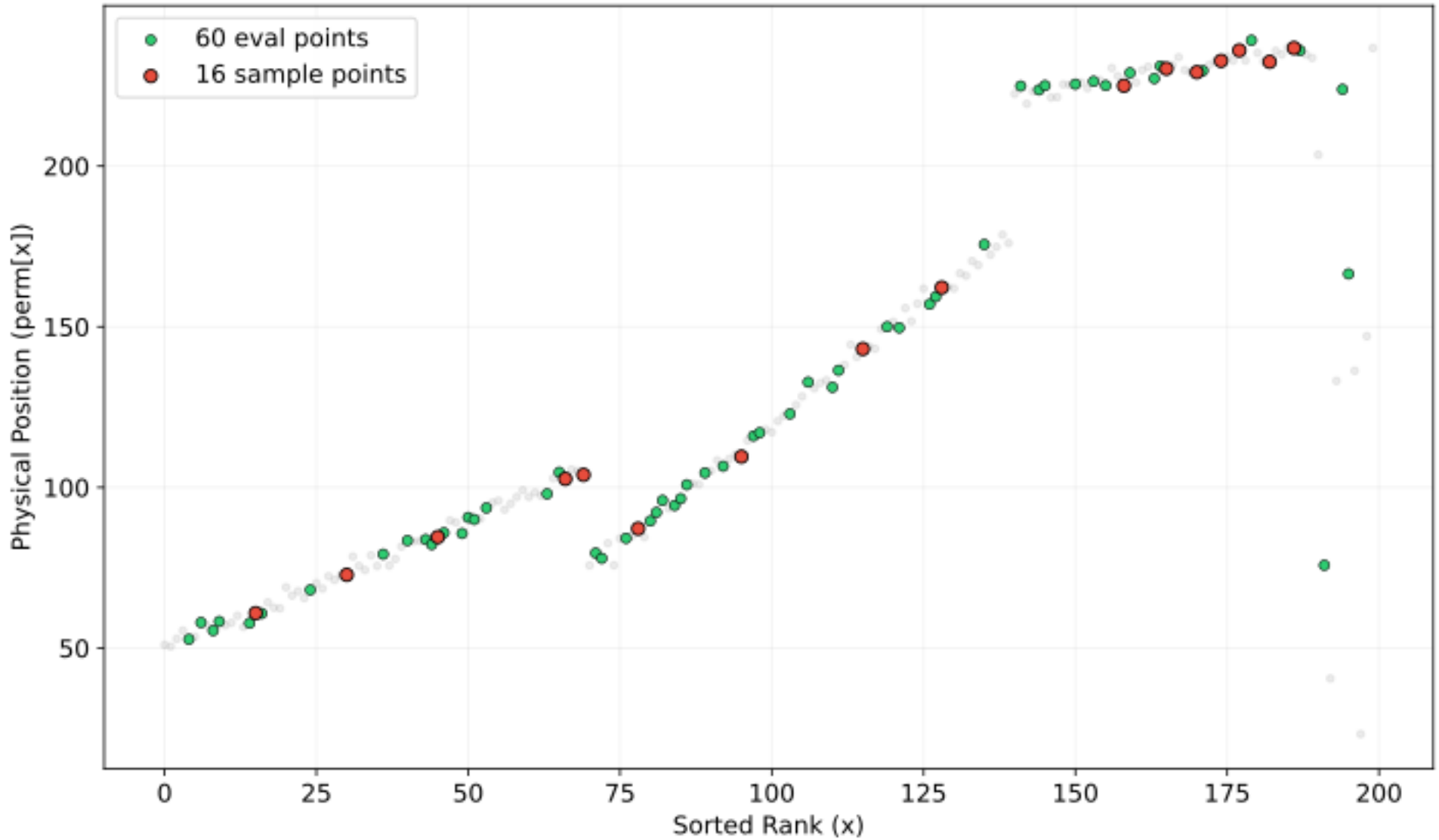
Sample x points



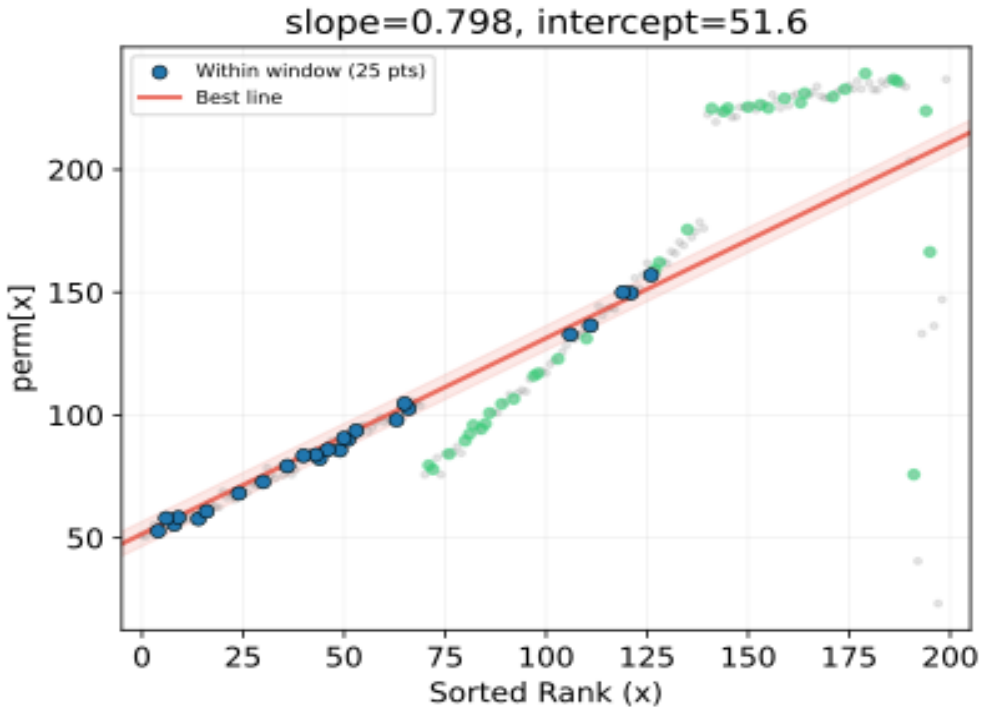
Compute slope for x points



Sample eval points



Find intercepts



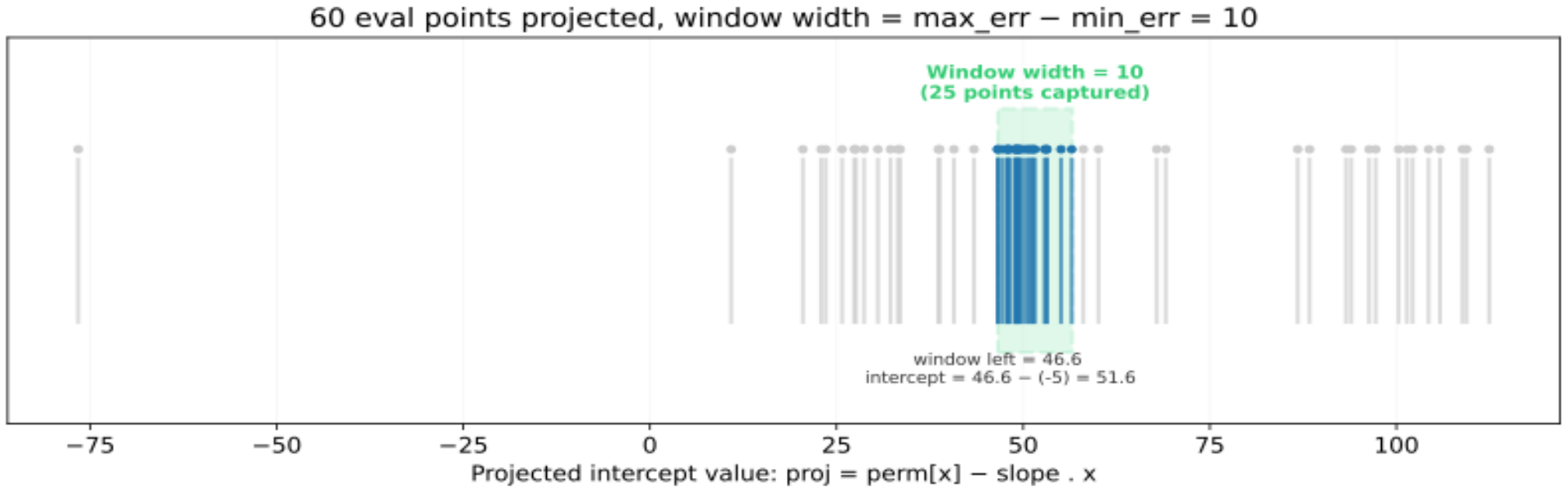
Find c for each eval point:

$c = y - m.x$

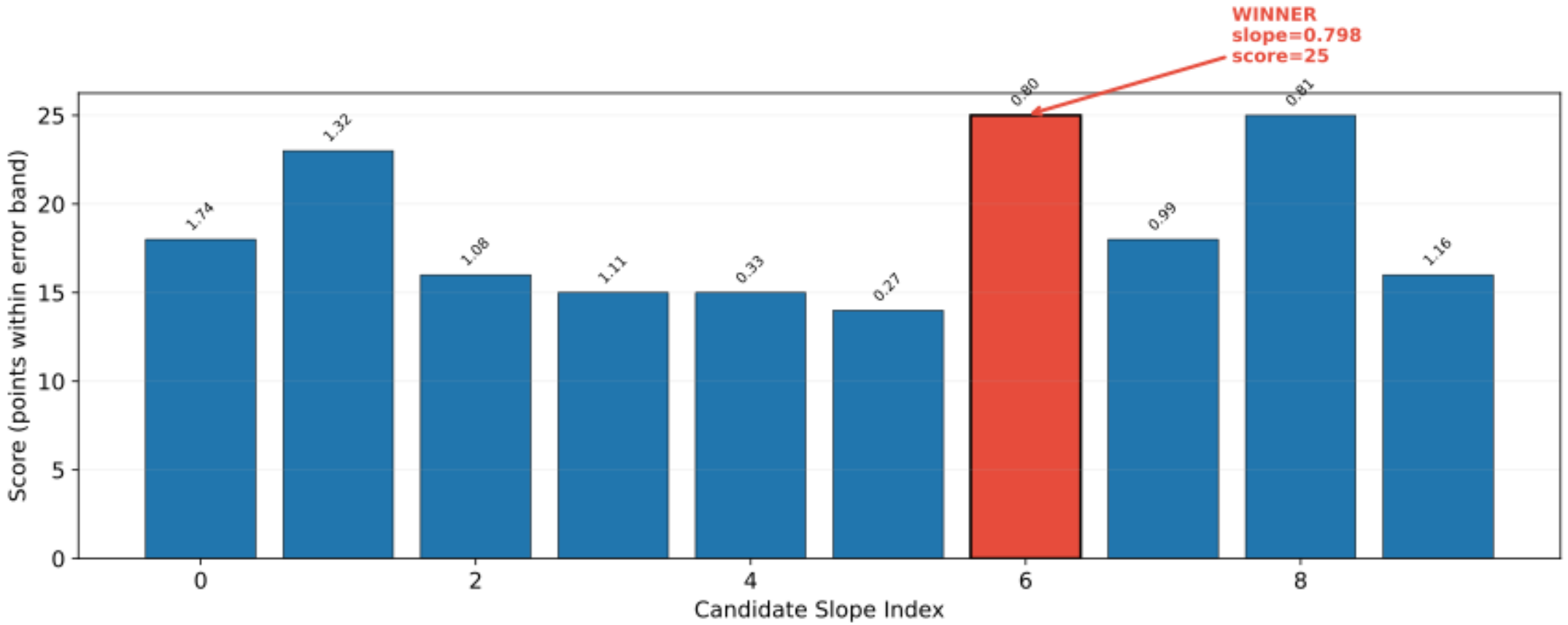
Aim: Find best slope

Find intercepts fitting in window

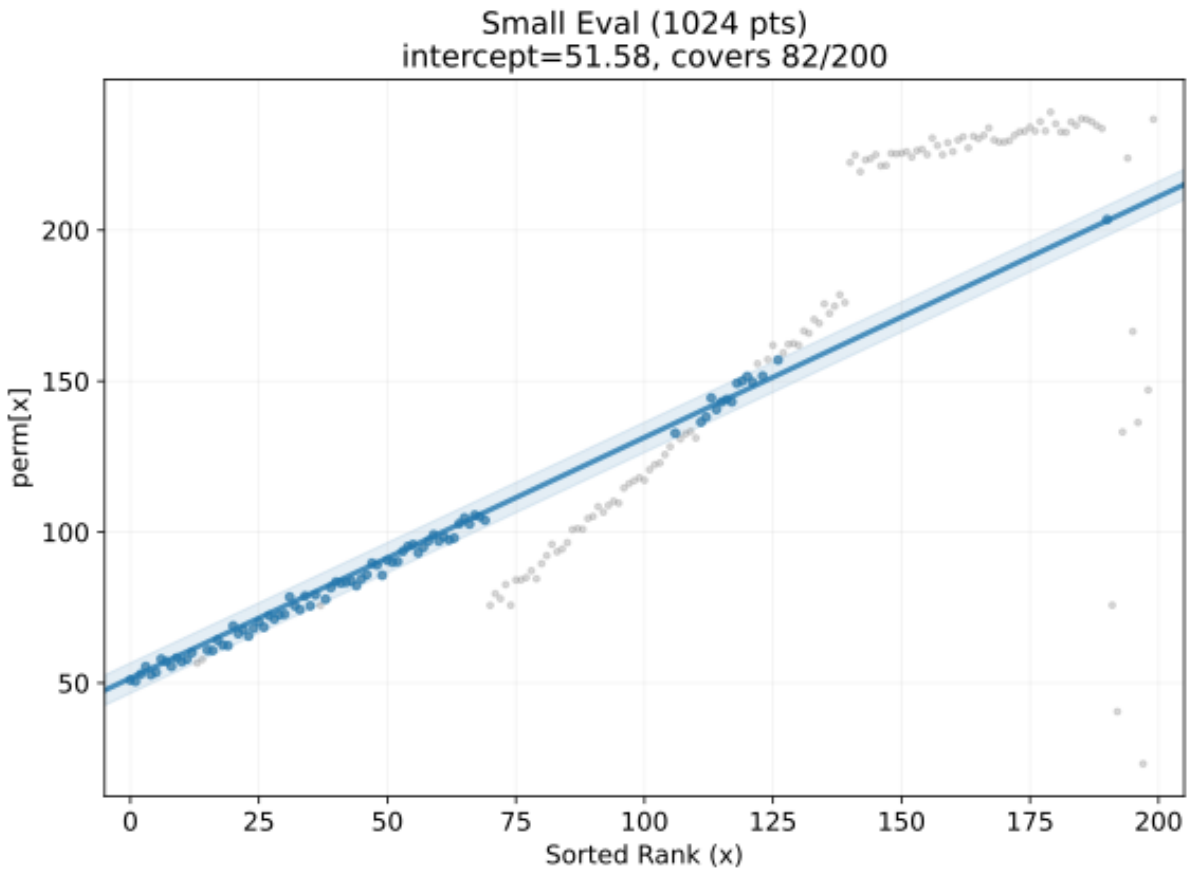
Sort the intercepts and slide a window to find the intercept fitting max points

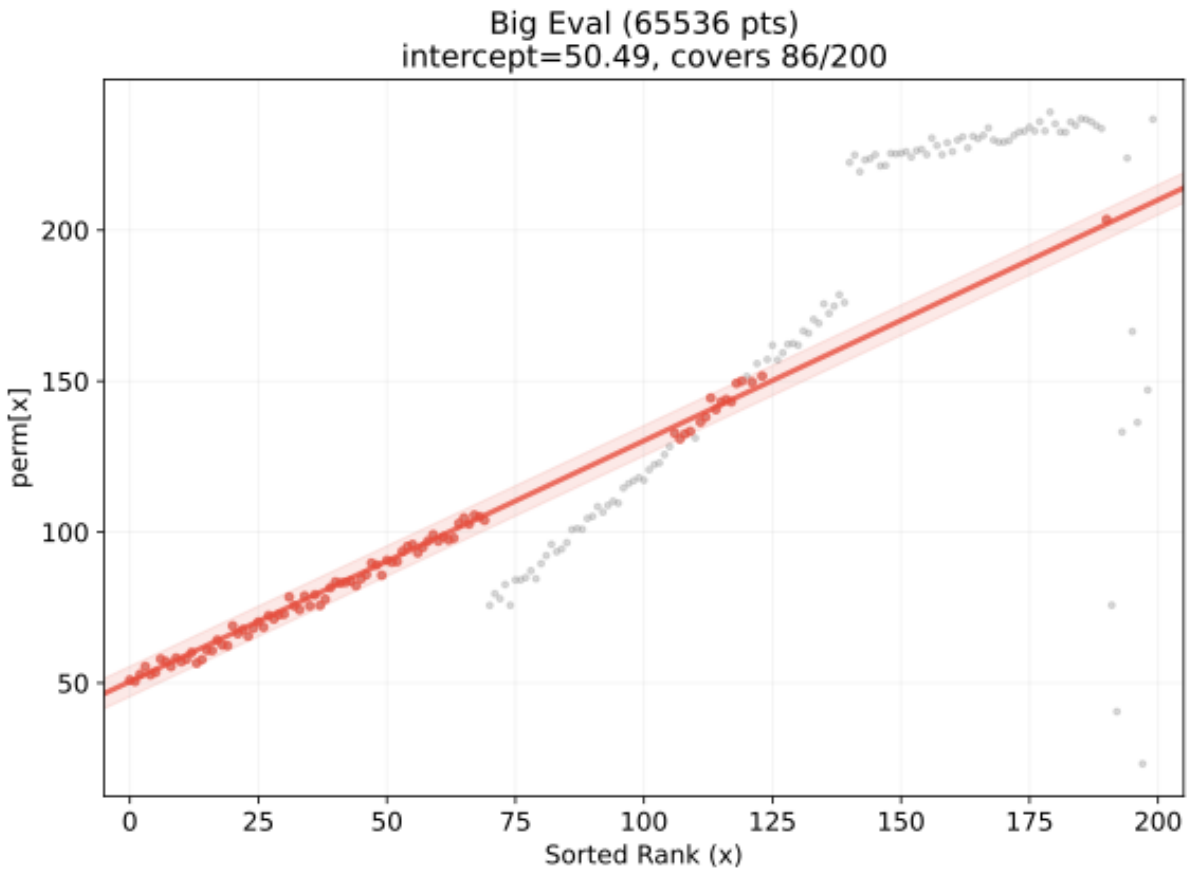


Find best slope

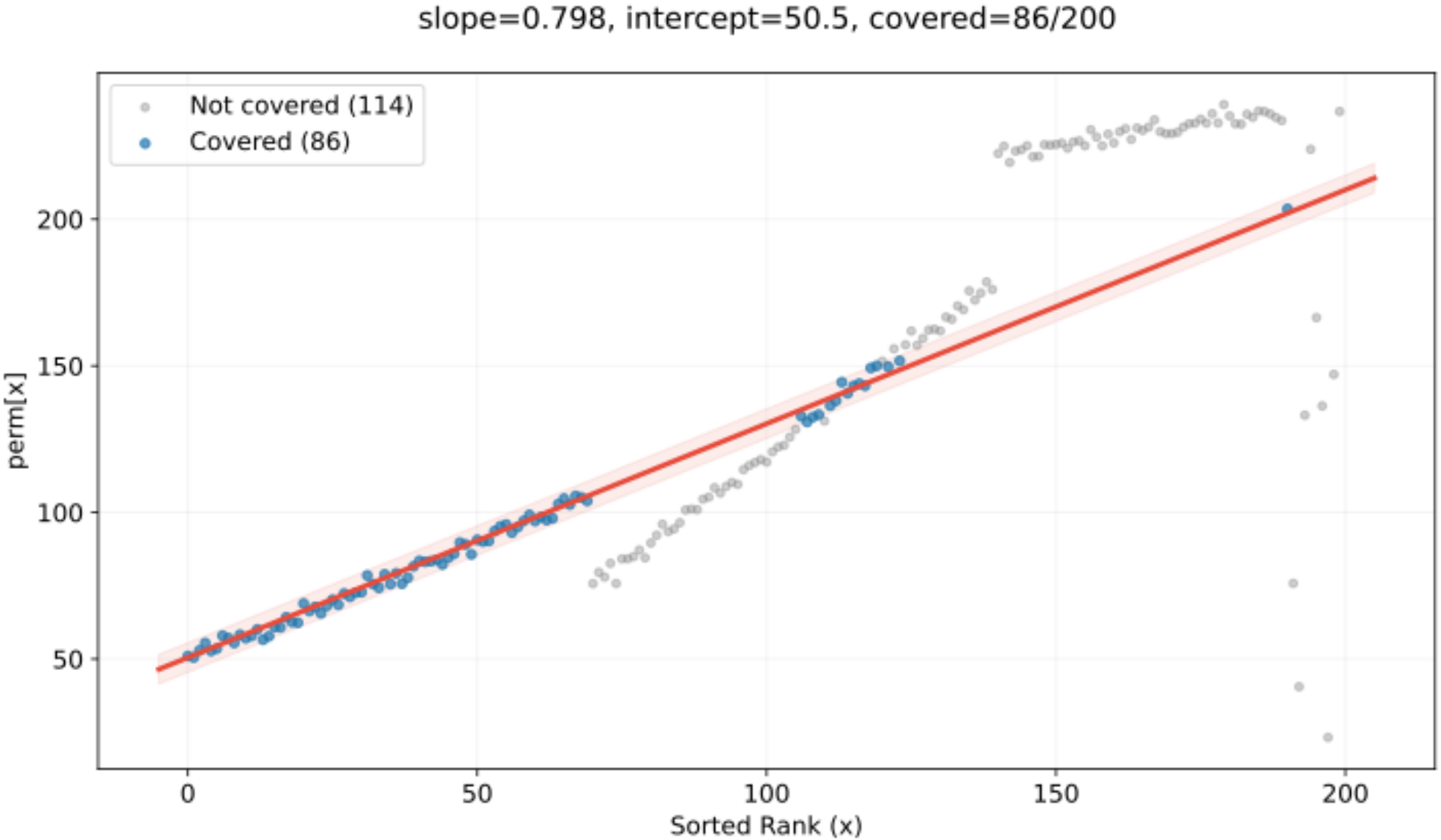


Refine intercept of best slope

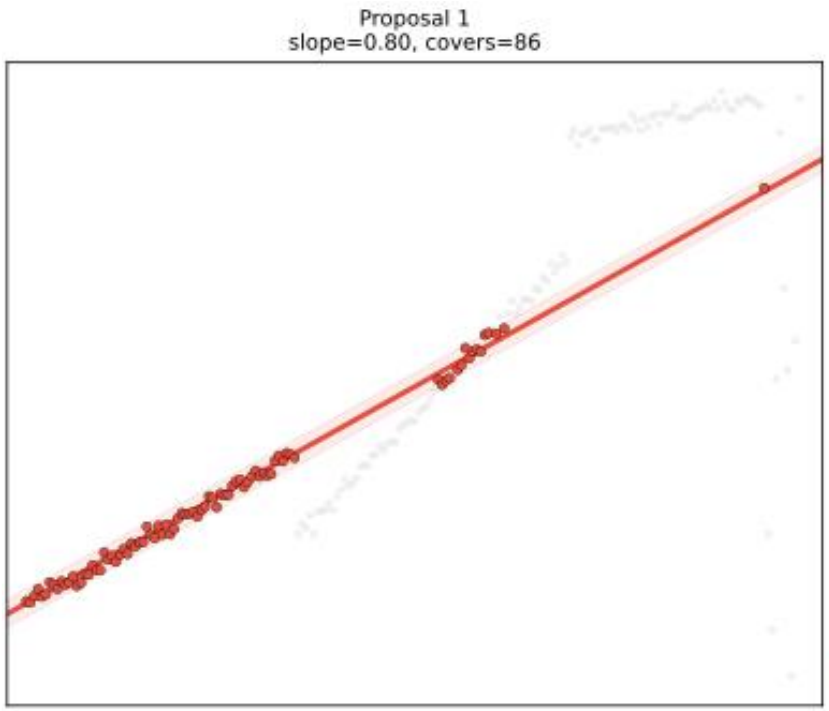


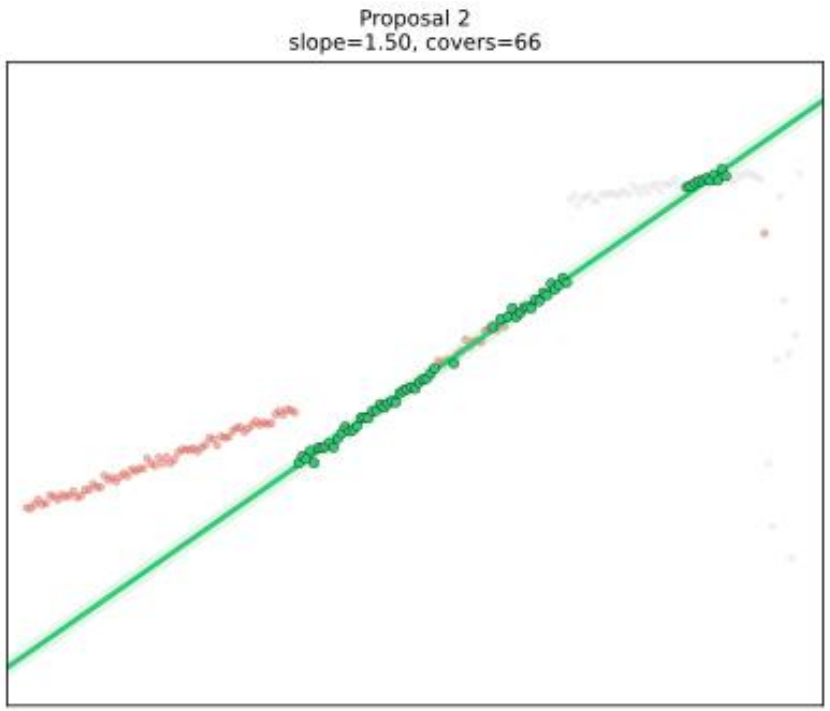


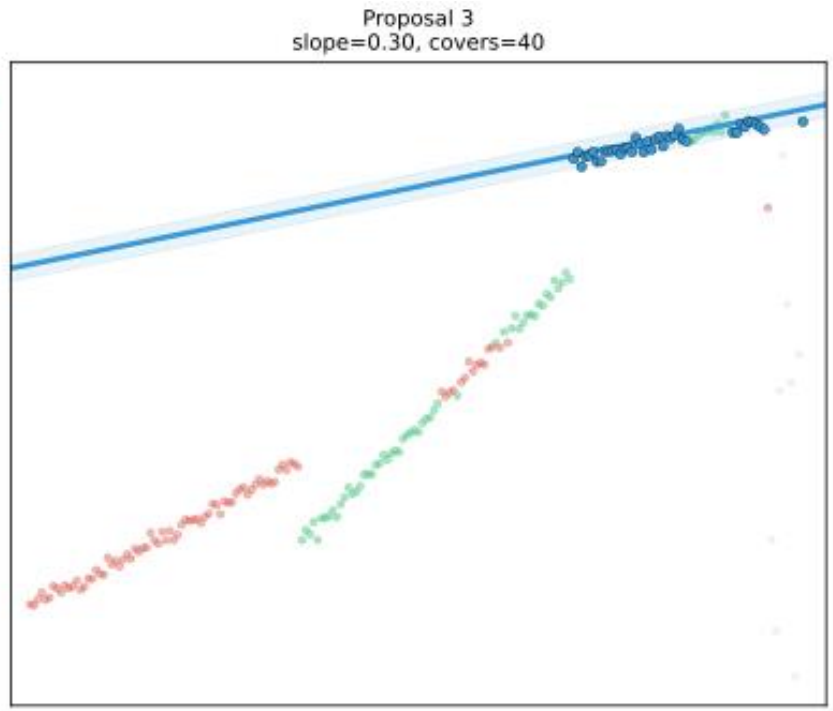
Found first line!



Incremental line cover

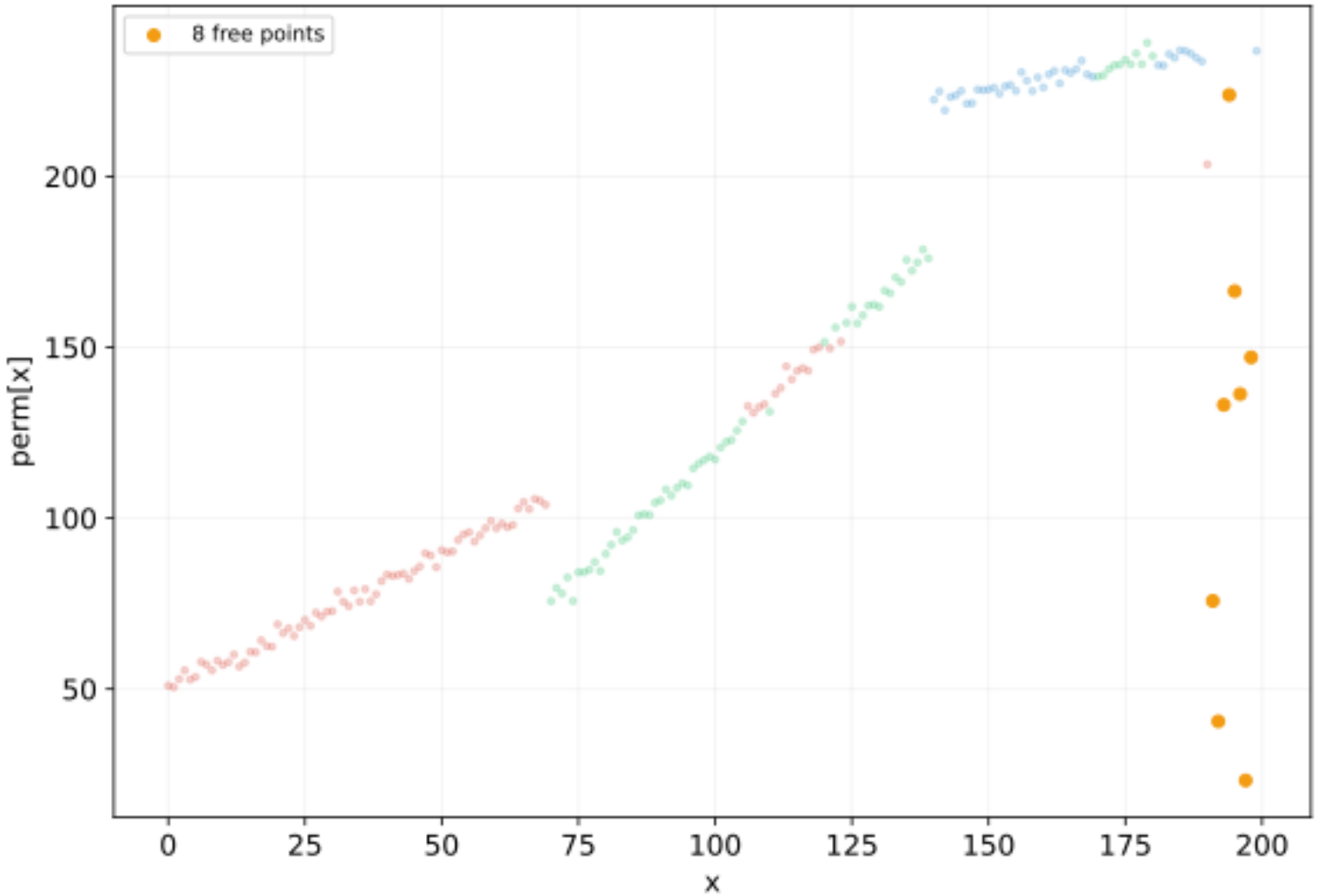




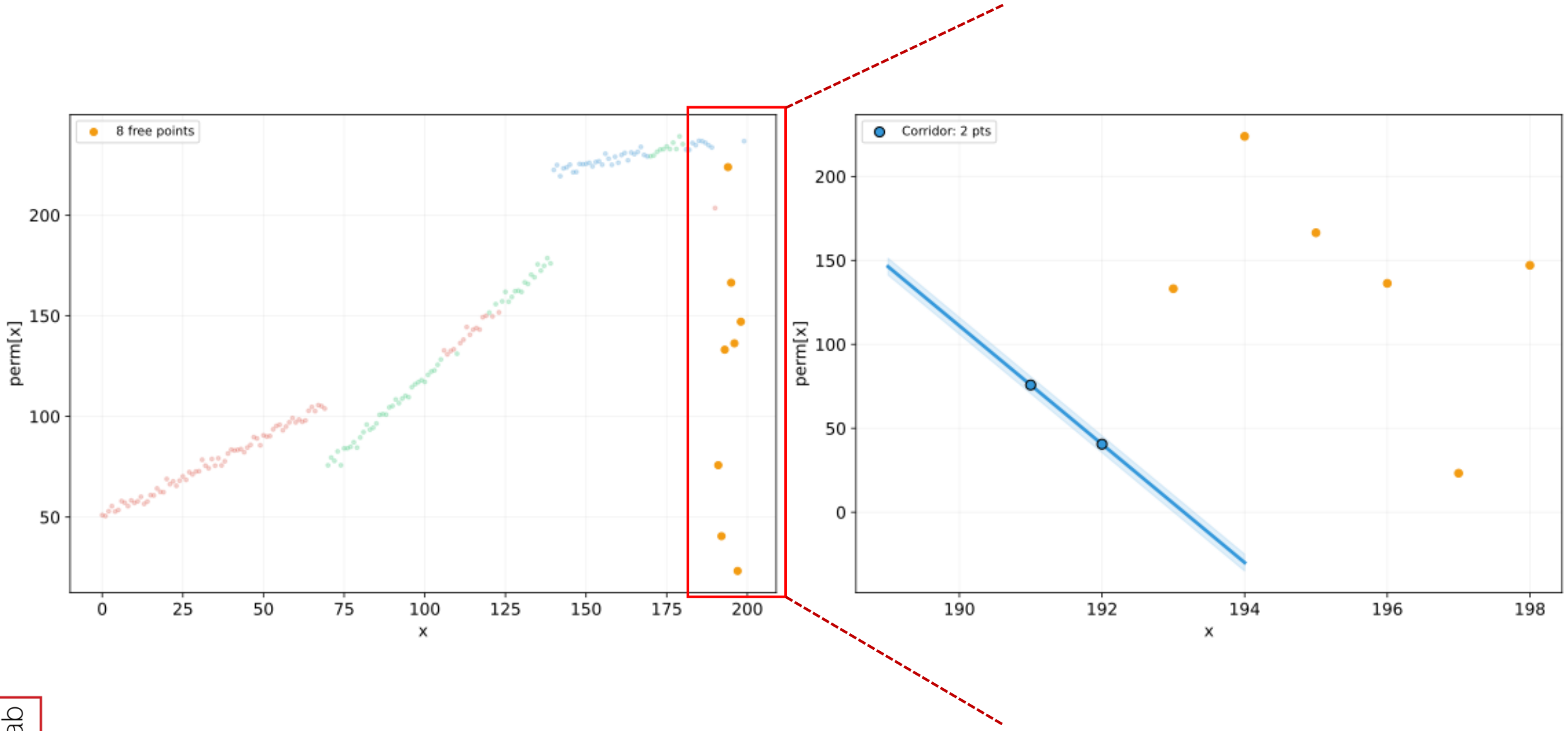


What about the uncovered points?

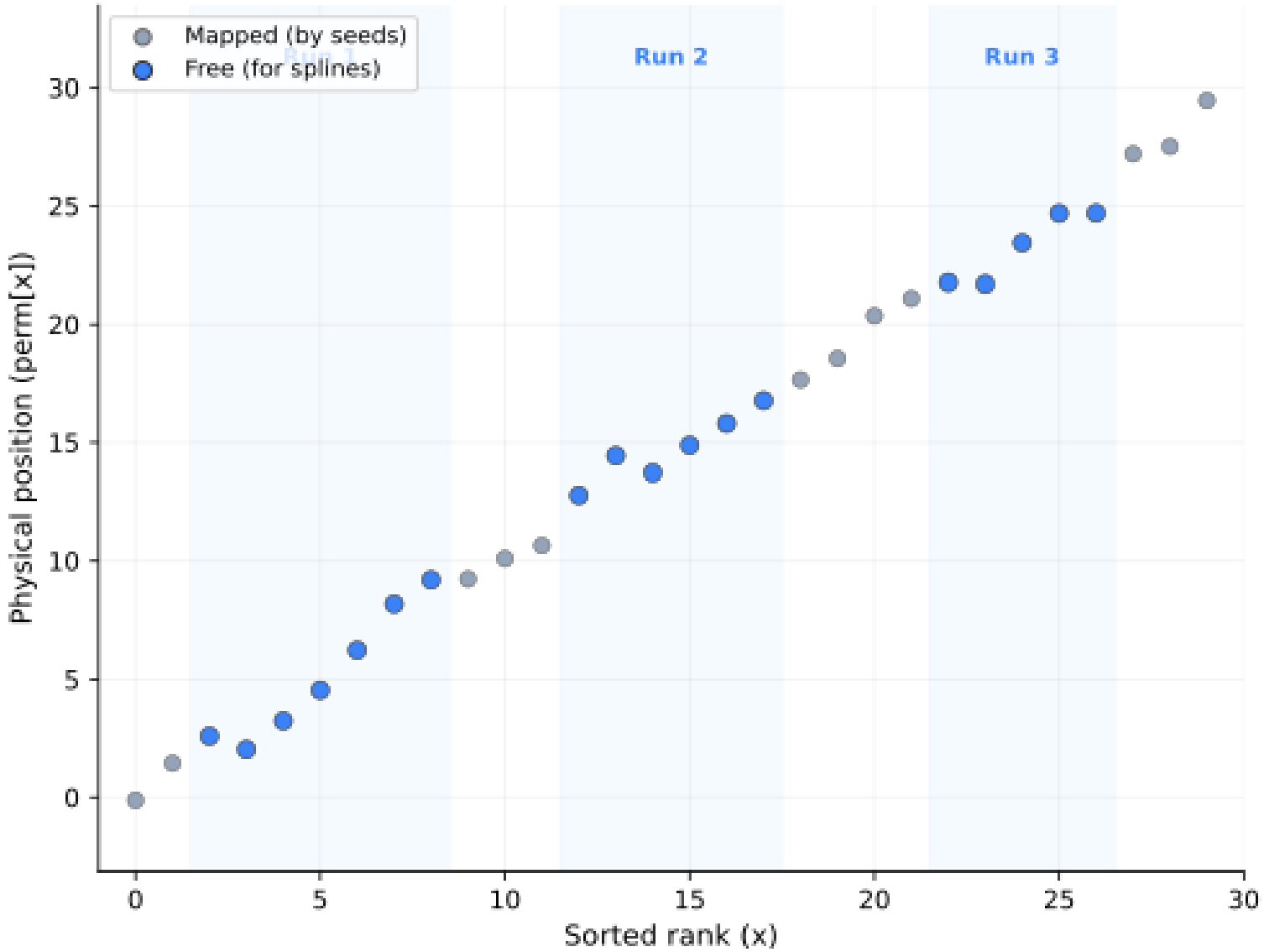
Find Local Pattern



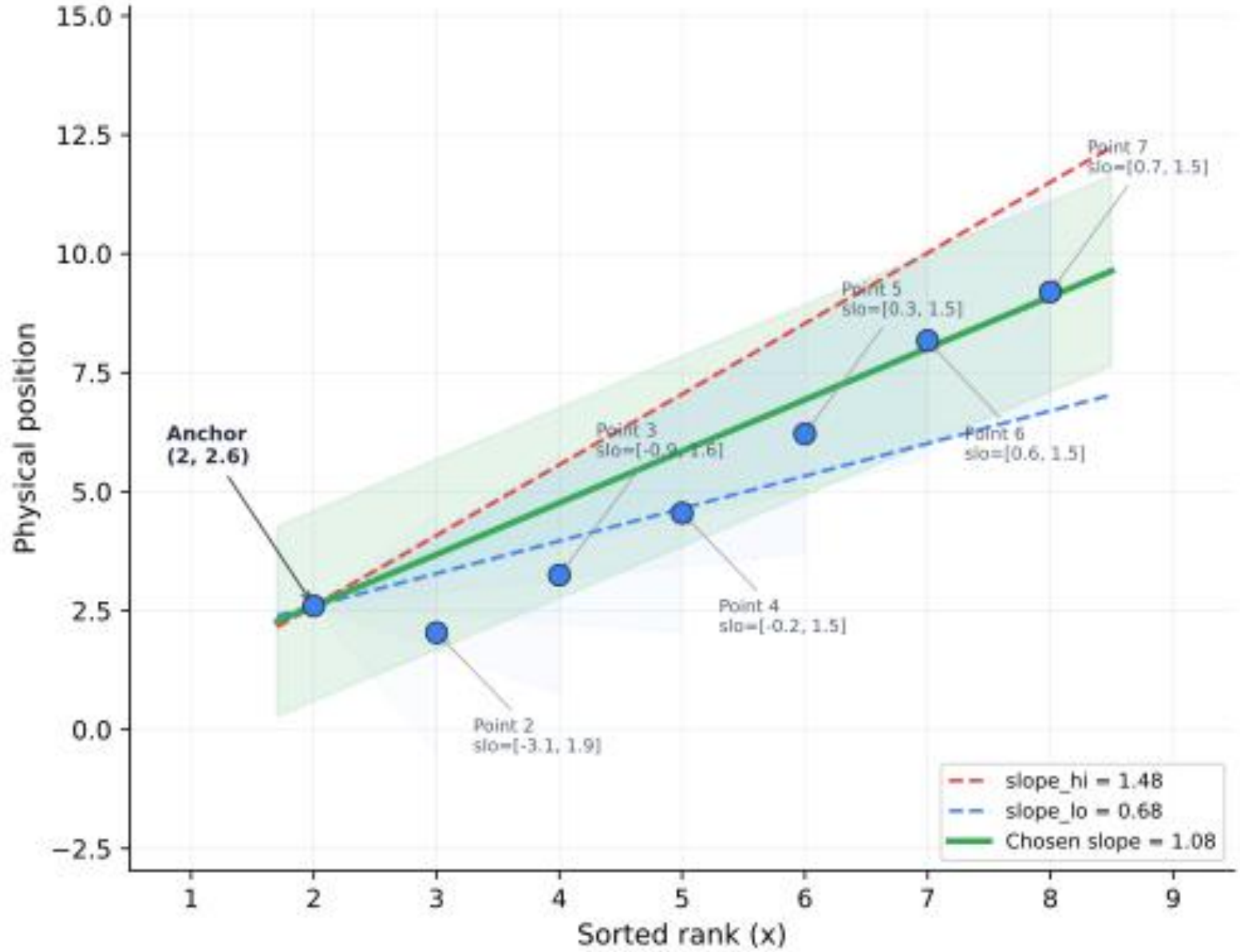
Find Local Pattern



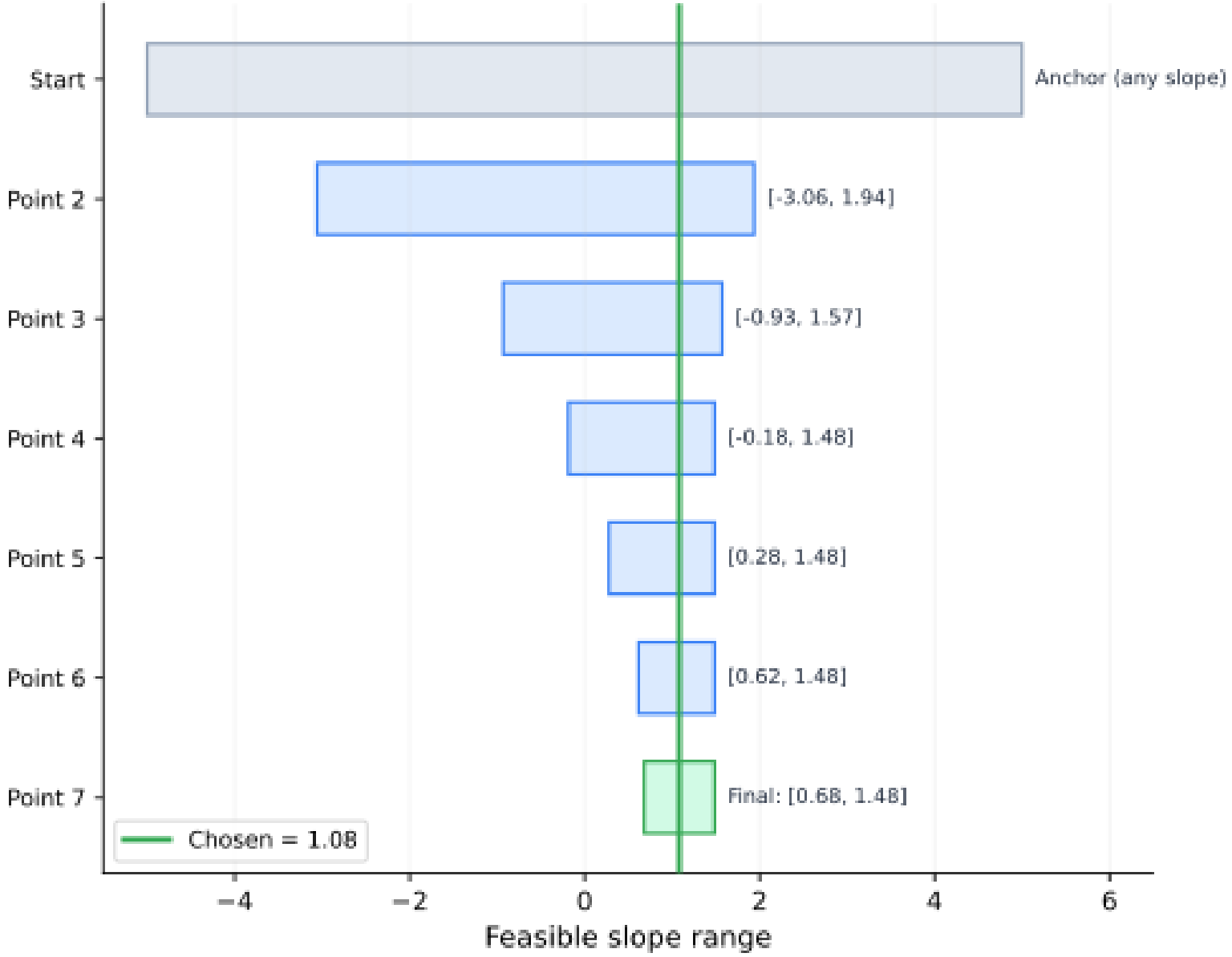
Find Local Pattern



Find Local Pattern

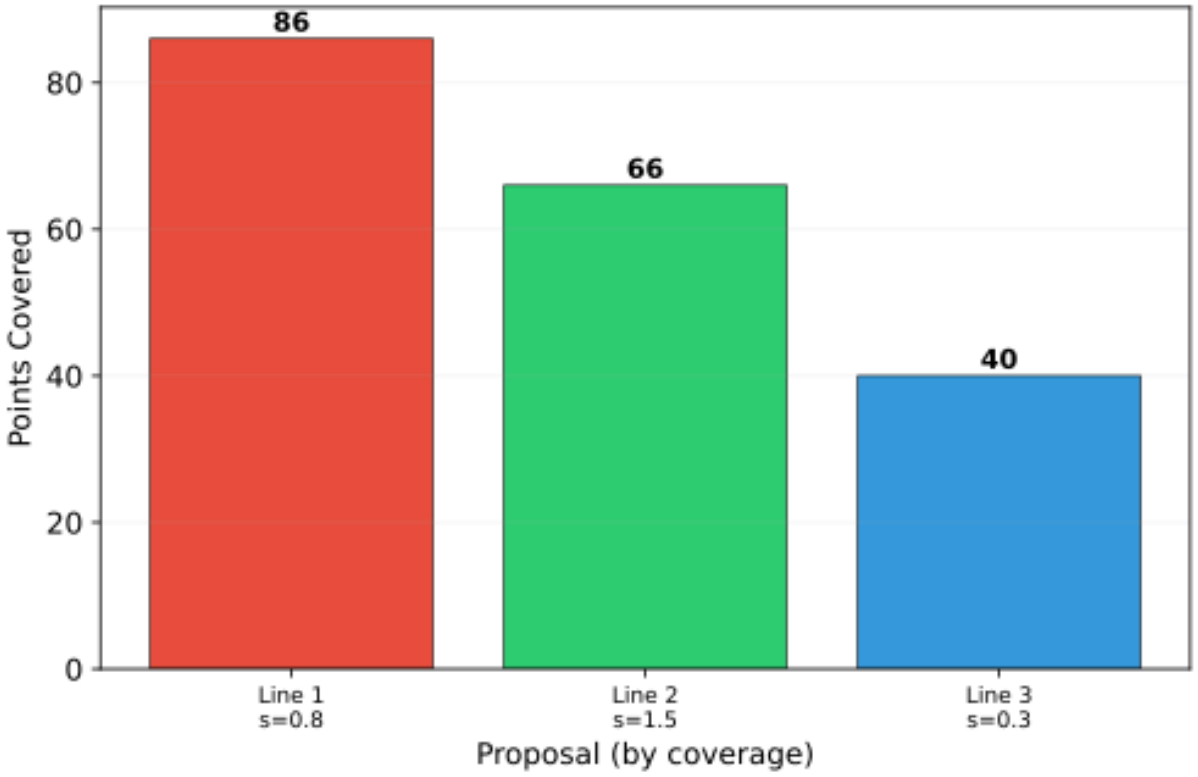


Convergence of slope, intercept



Is this method always good?

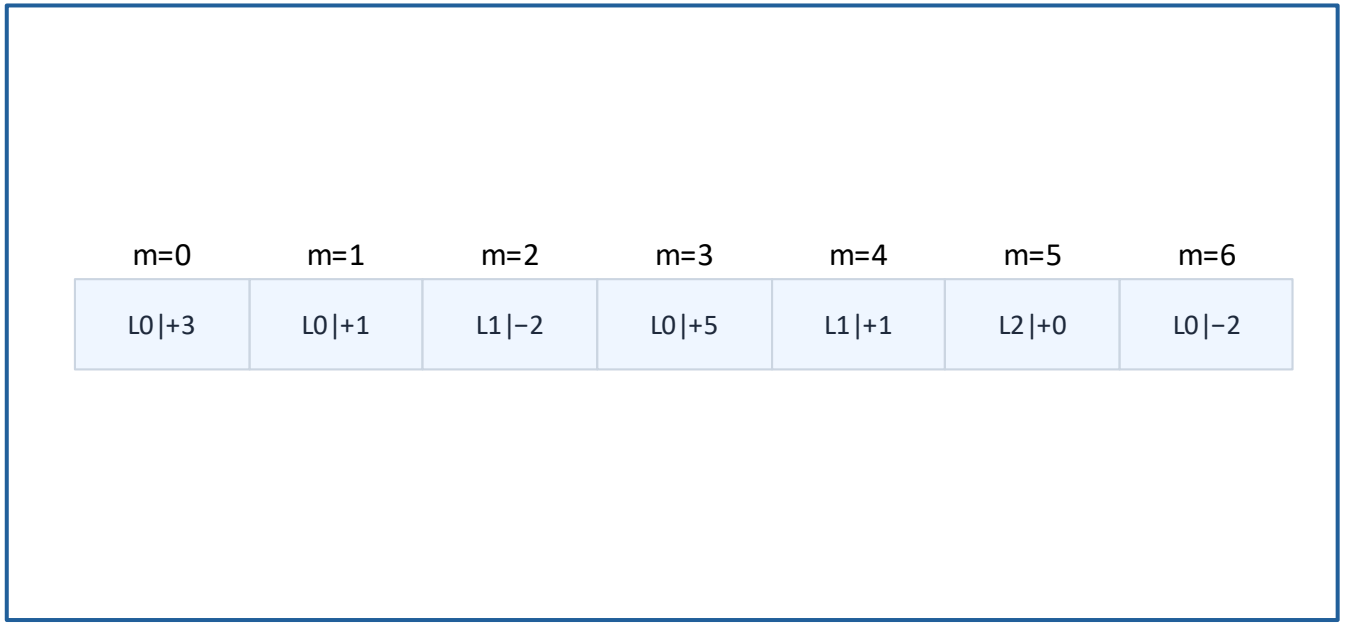
Lines are not always beneficial!



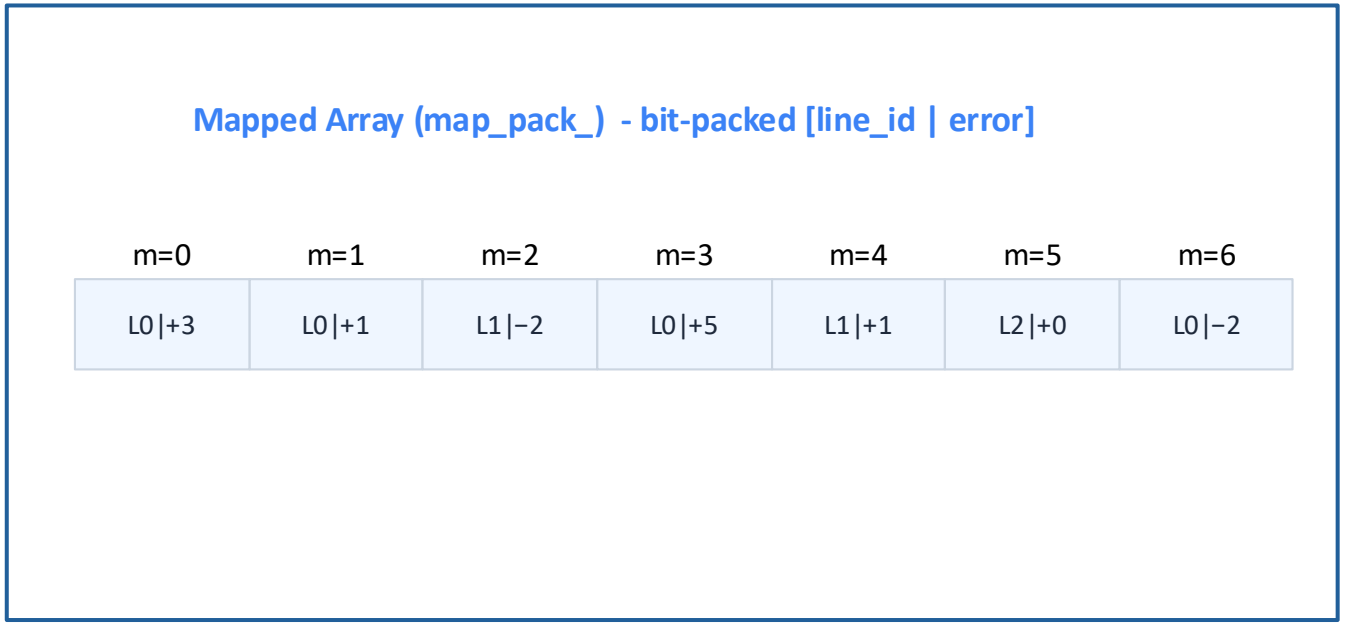
bits = (n_lines . 64 bits per line) + (mapped × (lid_bits + err_bits)) + (backup × ceil_log2(N)) + (rank overhead)

What does the entire system look like?

CMap Storage Structure



CMap Storage Structure



CMap Storage Structure

Mapped Array (map_pack_) - bit-packed [line_id | error]

m=0	m=1	m=2	m=3	m=4	m=5	m=6
L0|+3	L0|+1	L1|−2	L0|+5	L1|+1	L2|+0	L0|−2

lid	slope	intercept
L0	0.82	51.3
L1	1.47	−28.1
L2	0.31	182.0

CMap Storage Structure

Mapped Array (map_pack_) - bit-packed [line_id | error]

m=0	m=1	m=2	m=3	m=4	m=5	m=6
L0|+3	L0|+1	L1|−2	L0|+5	L1|+1	L2|+0	L0|−2

Lines Table (lines_[lid])

lid	slope	intercept
L0	0.82	51.3
L1	1.47	−28.1
L2	0.31	182.0

CMap Storage Structure

Mapped Array (map_pack_) - bit-packed [line_id | error]

m=0	m=1	m=2	m=3	m=4	m=5	m=6
L0|+3	L0|+1	L1|−2	L0|+5	L1|+1	L2|+0	L0|−2

Lines Table (lines_[lid])

lid	slope	intercept
L0	0.82	51.3
L1	1.47	−28.1
L2	0.31	182.0

Decode:

pos = round(slope.x+intercept) + error

CMap Storage Structure

Mapped Array (map_pack_) - bit-packed [line_id | error]

m=0	m=1	m=2	m=3	m=4	m=5	m=6
L0|+3	L0|+1	L1|−2	L0|+5	L1|+1	L2|+0	L0|−2

Lines Table (lines_[lid])

lid	slope	intercept
L0	0.82	51.3
L1	1.47	−28.1
L2	0.31	182.0

Decode:

pos = round(s×x+c) + err

b=0	b=1	b=2
42091	18874	99312

CMap Storage Structure

BOSTON UNIVERSITY

Mapped Array (map_pack_) - bit-packed [line_id | error]

m=0	m=1	m=2	m=3	m=4	m=5	m=6
L0|+3	L0|+1	L1|−2	L0|+5	L1|+1	L2|+0	L0|−2

Lines Table (lines_[lid])

lid	slope	intercept
L0	0.82	51.3
L1	1.47	−28.1
L2	0.31	182.0

Decode:

pos = round(s×x+c) + err

Backup Array (bak_pack_) - raw positions

b=0	b=1	b=2
42091	18874	99312

No line decode needed

DiSC lab

CMap Storage Structure

r=0	r=1	r=2	r=3	r=4	r=5	r=6	r=7	r=8	r=9
0	0	1	0	1	0	0	1	0	0

Mapped Array (map_pack_) - bit-packed [line_id | error]

m=0	m=1	m=2	m=3	m=4	m=5	m=6
L0|+3	L0|+1	L1|−2	L0|+5	L1|+1	L2|+0	L0|−2

Lines Table (lines_[lid])

lid	slope	intercept
L0	0.82	51.3
L1	1.47	−28.1
L2	0.31	182.0

Decode:

pos = round(s×x+c) + err

Backup Array (bak_pack_) - raw positions

b=0	b=1	b=2
42091	18874	99312

No line decode needed

CMap Storage Structure

Bitvector (1 bit/entry, SDSL rank support)

r=0	r=1	r=2	r=3	r=4	r=5	r=6	r=7	r=8	r=9
0	0	1	0	1	0	0	1	0	0

0 = mapped → lines + error

1 = backup → raw position

Mapped Array (map_pack_) - bit-packed [line_id | error]

m=0	m=1	m=2	m=3	m=4	m=5	m=6
L0|+3	L0|+1	L1|−2	L0|+5	L1|+1	L2|+0	L0|−2

Lines Table (lines_[lid])

lid	slope	intercept
L0	0.82	51.3
L1	1.47	−28.1
L2	0.31	182.0

Decode:

pos = round(s×x+c) + err

Backup Array (bak_pack_) - raw positions

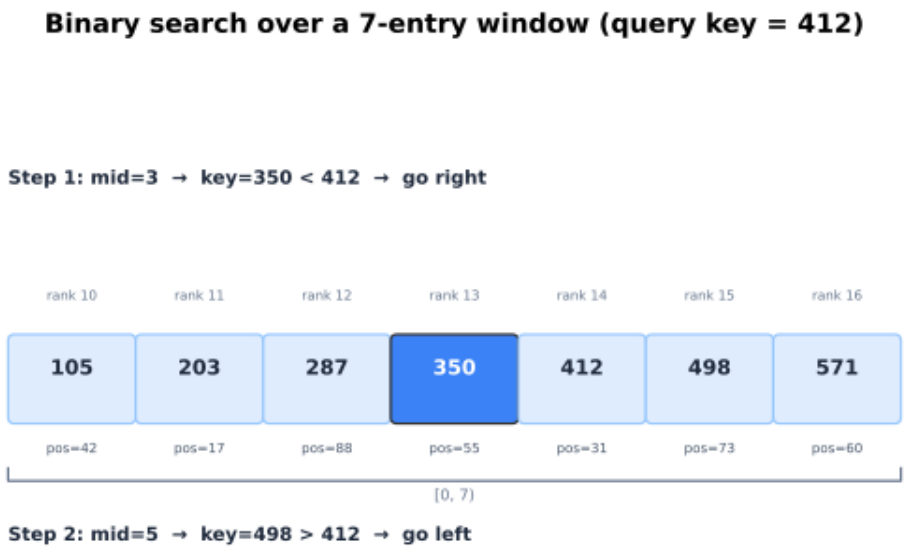
b=0	b=1	b=2
42091	18874	99312

No line decode needed.
Direct physical position.

data_[pos]

k0	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13	k14	…

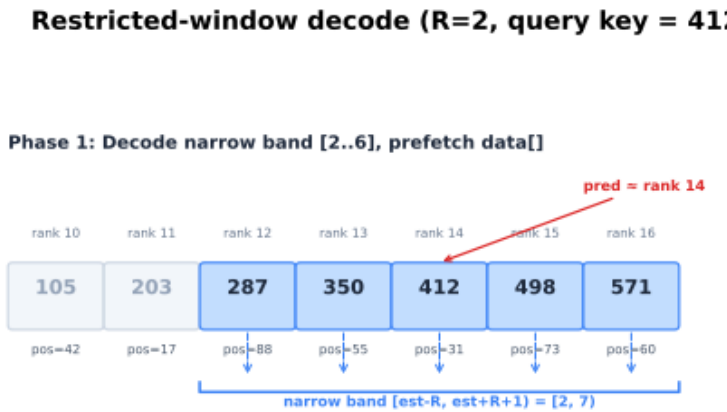
Single Query: Binary Search Path



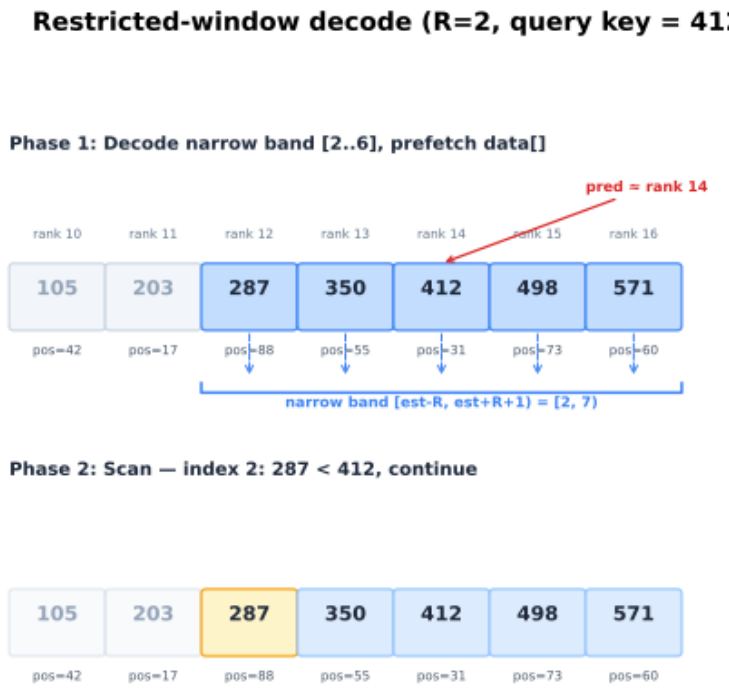
Single Query: Binary Search Path

Single Query: Binary Search Path

Single Query: Restricted Search



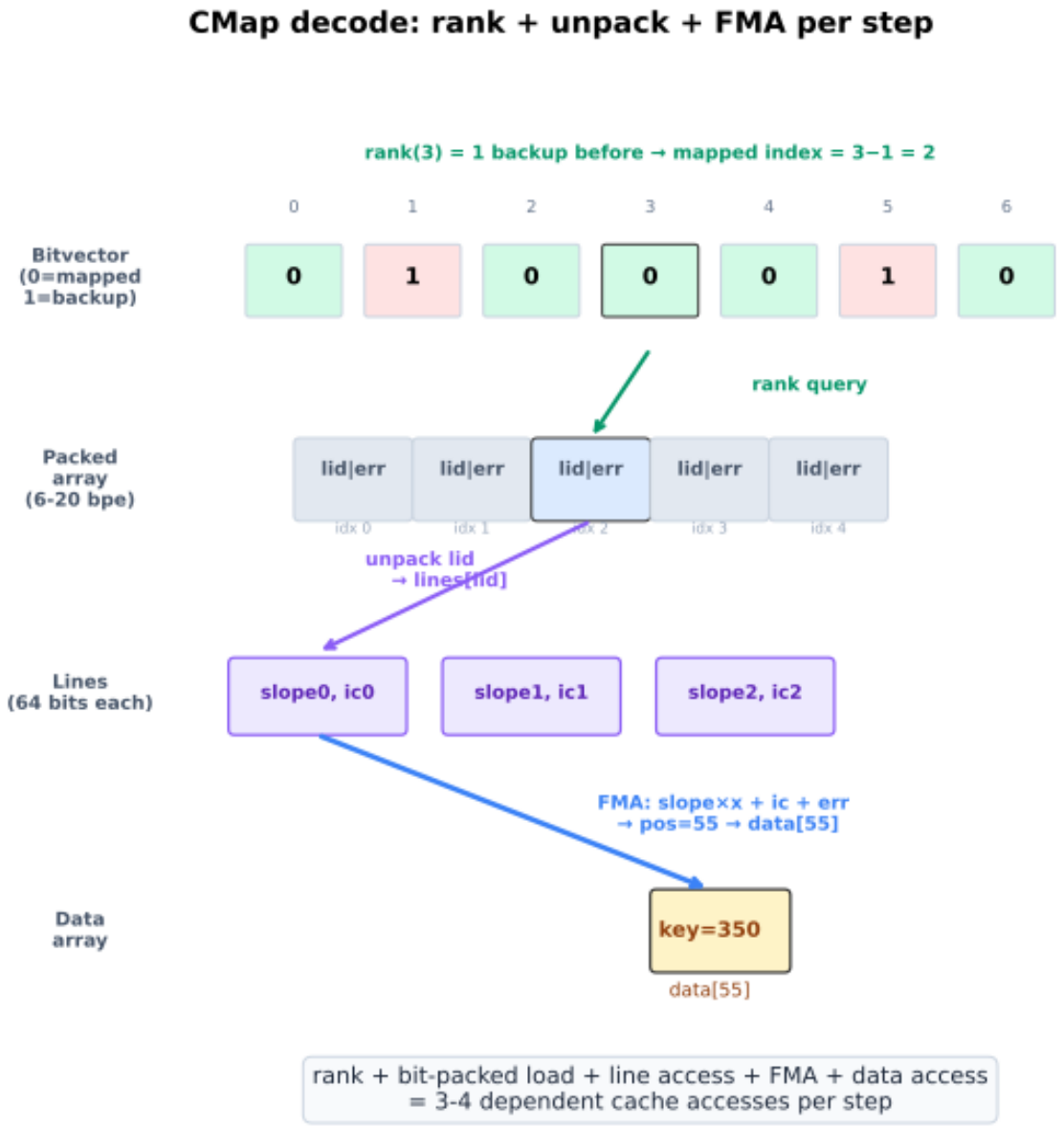
Single Query: Restricted Search



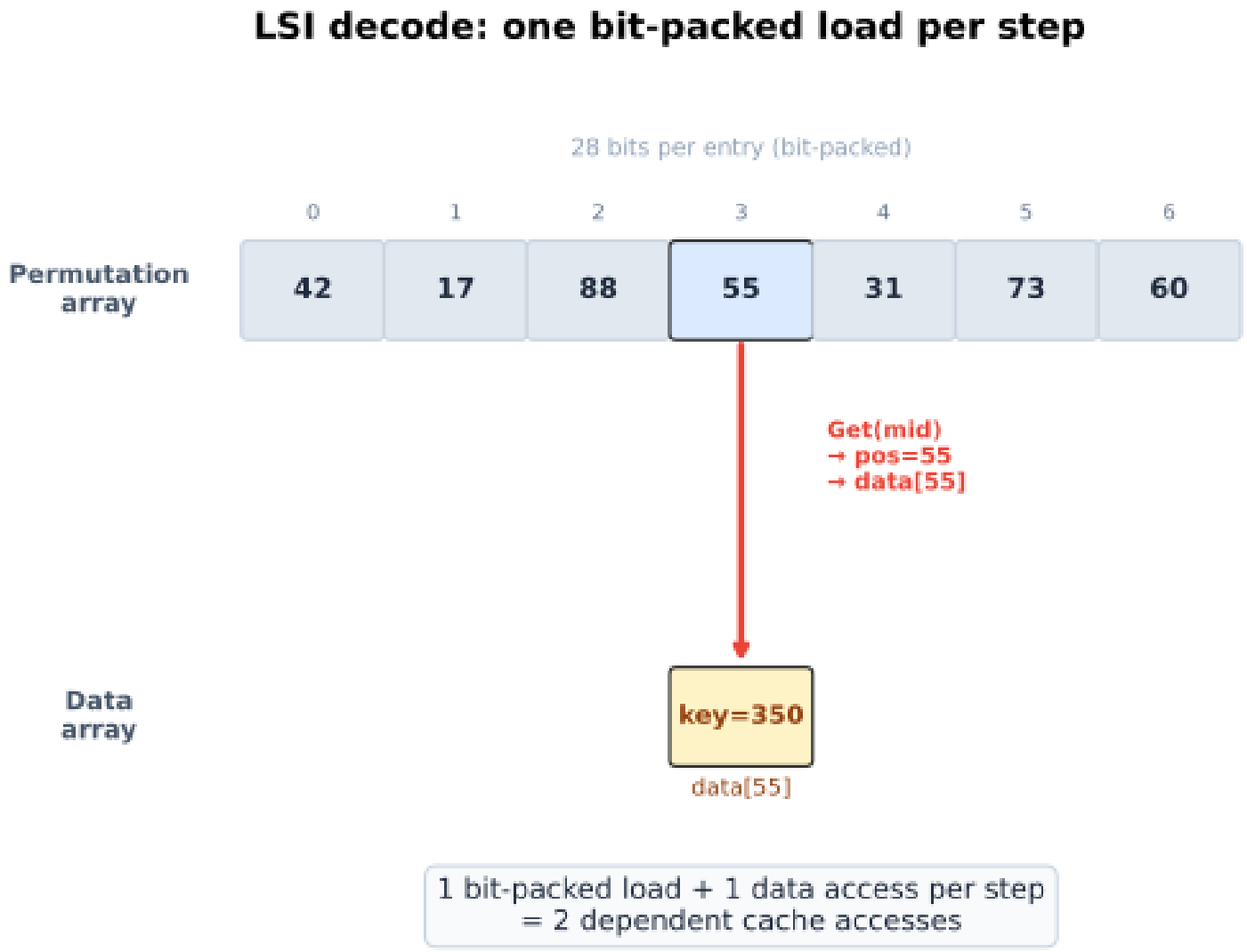
Single Query: Restricted Search

Single Query: Restricted Search

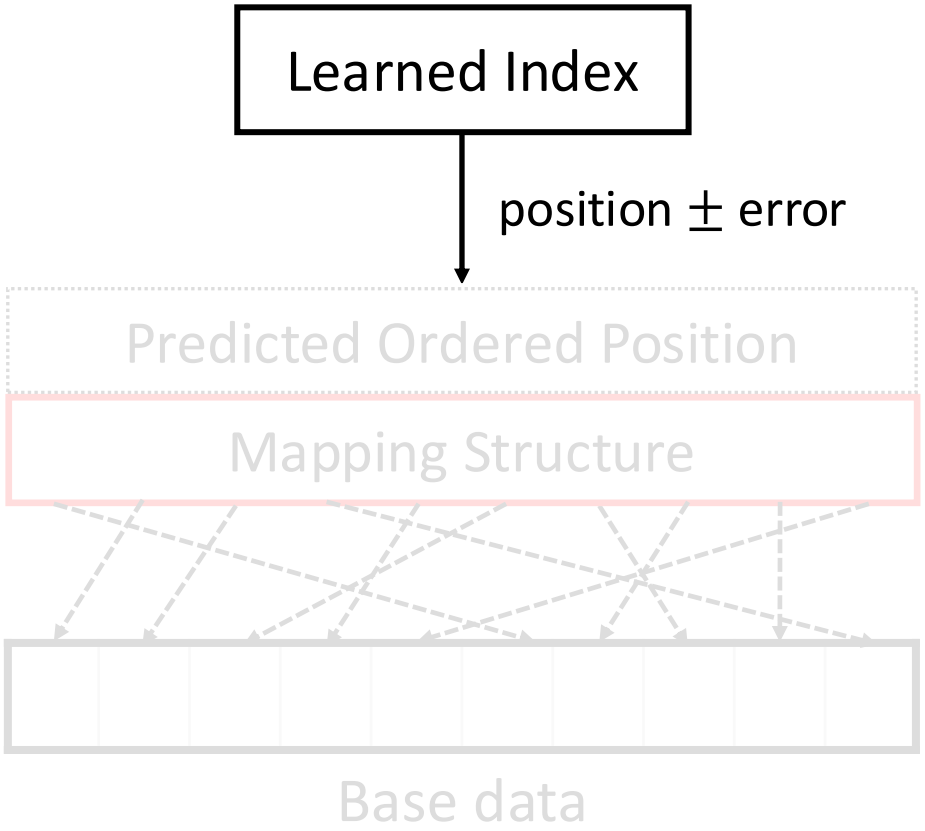
Single Query: Decode



Single Query: LSI Access Path

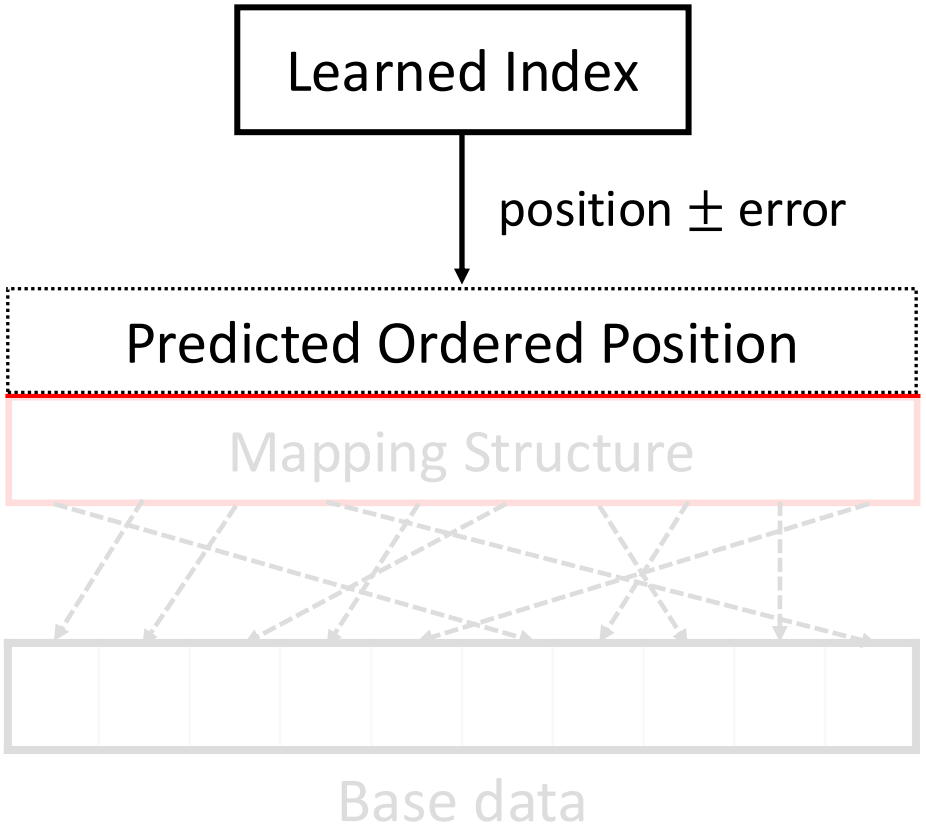


Single Query: Access Path



1 **Model Predict**

Single Query: Access Path



1 **Model Predict**

lo=pred−15, hi=pred+16

Single Query: Access Path

Single Query: Access Path

1 Model Predict

lo=pred−15, hi=pred+16

2 rank_window(lo) → base_rank + 32-bit window

0	0	1	0	0	0	0	1	0	0

↑est

3a Backup (bit=1)

bak_pack_ → raw pos

42091

3b Mapped (bit=0)

map_pack_ read:

L0|+3	L1|−2	L0|+5

↓ unpack

lines_[lid]:

L0	0.82	51.3

↓ FMA + SSE round

pos = round(slope × rank + intercept) + error

Single Query: Access Path

1 Model Predict

lo=pred−15, hi=pred+16

2 rank_window(lo) → base_rank + 32-bit window

0	0	1	0	0	0	0	1	0	0

↑est

3a Backup (bit=1)

bak_pack_ → raw pos

42091

prefetch(data_[pos])

3b Mapped (bit=0)

map_pack_ read:

L0|+3	L1|−2	L0|+5

↓ unpack

lines_[lid]:

L0	0.82	51.3

↓ FMA + SSE round

pos = round(slope × rank + intercept) + error

prefetch(data_[pos])

Single Query: Access Path

1 Model Predict

lo=pred−15, hi=pred+16

2 rank_window(lo) → base_rank + 32-bit window

0	0	1	0	0	0	0	1	0	0

↑est

3a Backup (bit=1)

bak_pack_ → raw pos

42091

prefetch(data_[pos])

3b Mapped (bit=0)

map_pack_ read:

L0|+3	L1|−2	L0|+5

↓ unpack

lines_[lid]:

L0	0.82	51.3

↓ FMA + SSE round

prefetch(data_[pos])

pos = round(slope × rank + intercept) + error

4 Scan: compare data_[pos] vs key

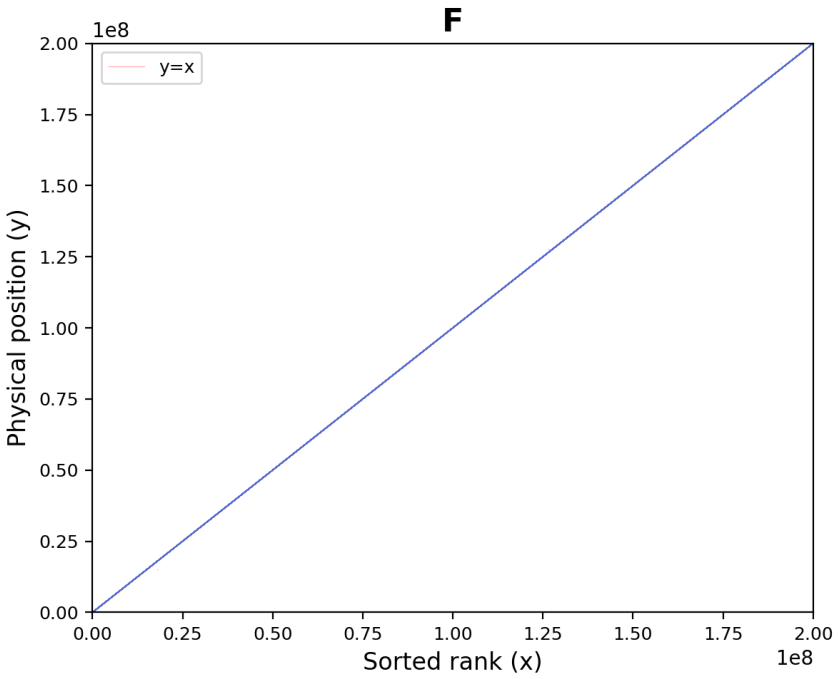
== key → return pos
> key → decode left, scan
< key → decode right, scan

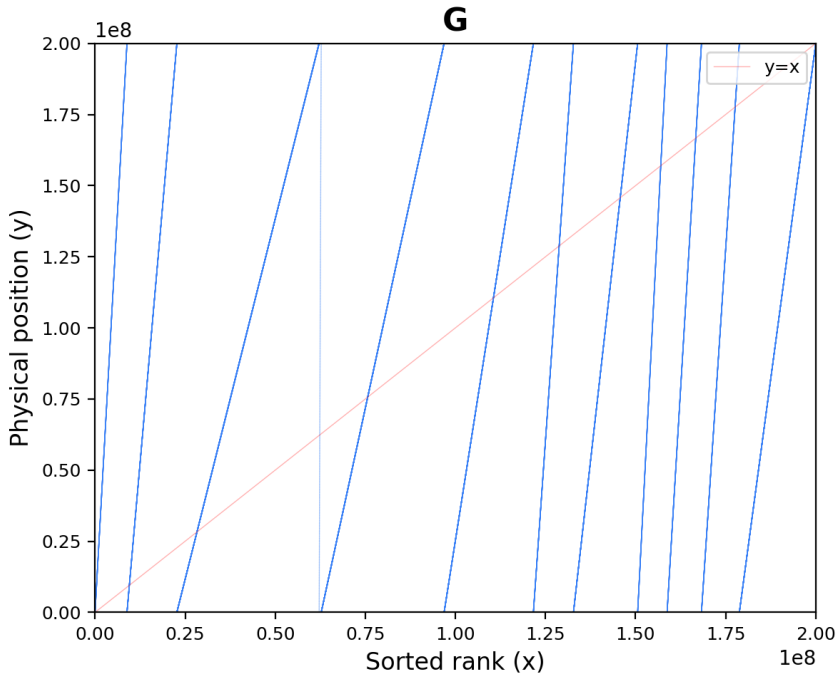
BOSTON UNIVERSITY

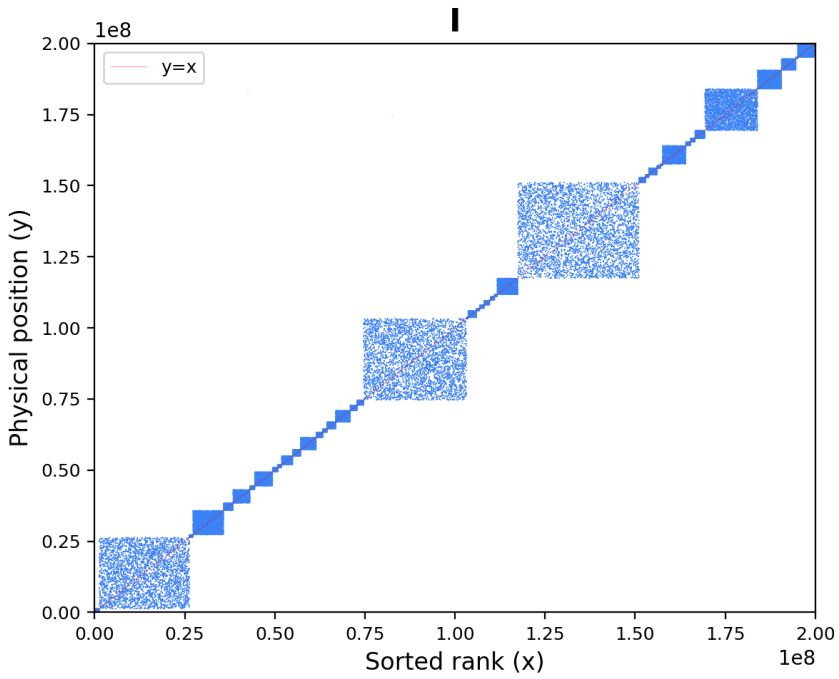
Does this Work?

DiSC lab

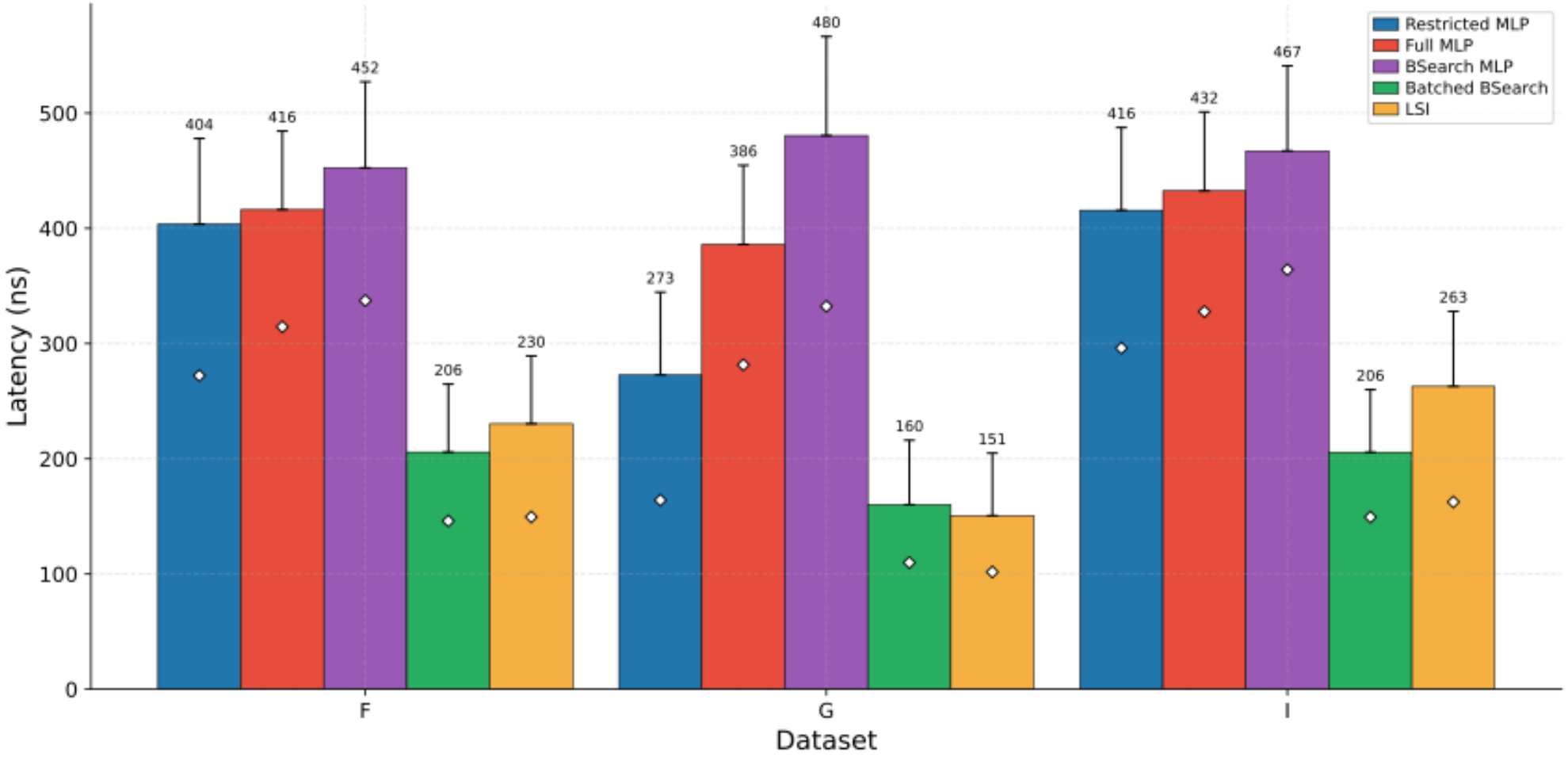
Evaluation Data Set



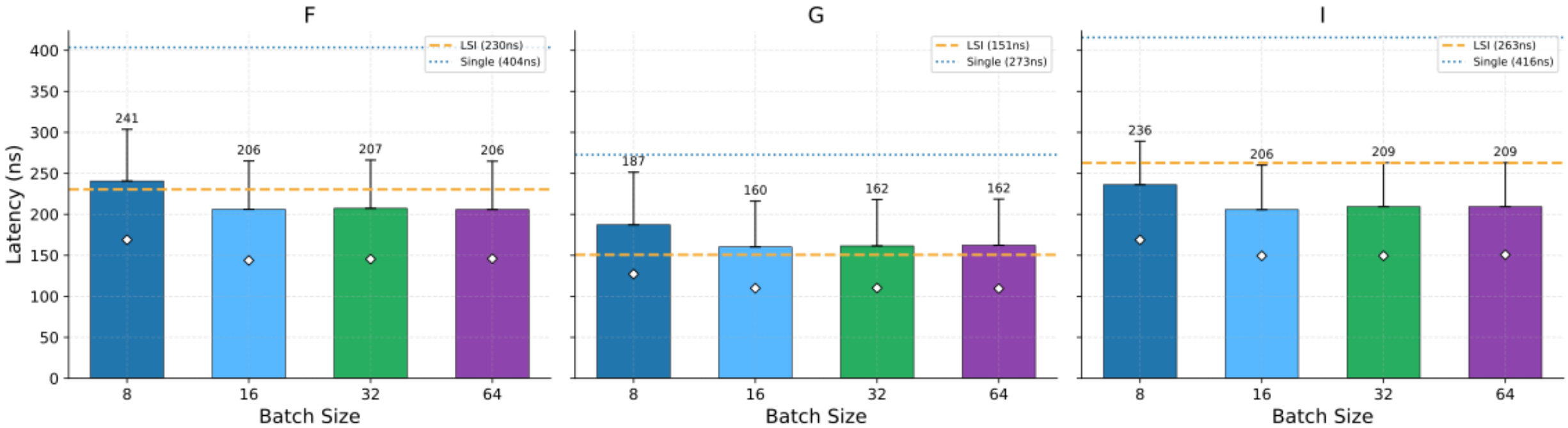




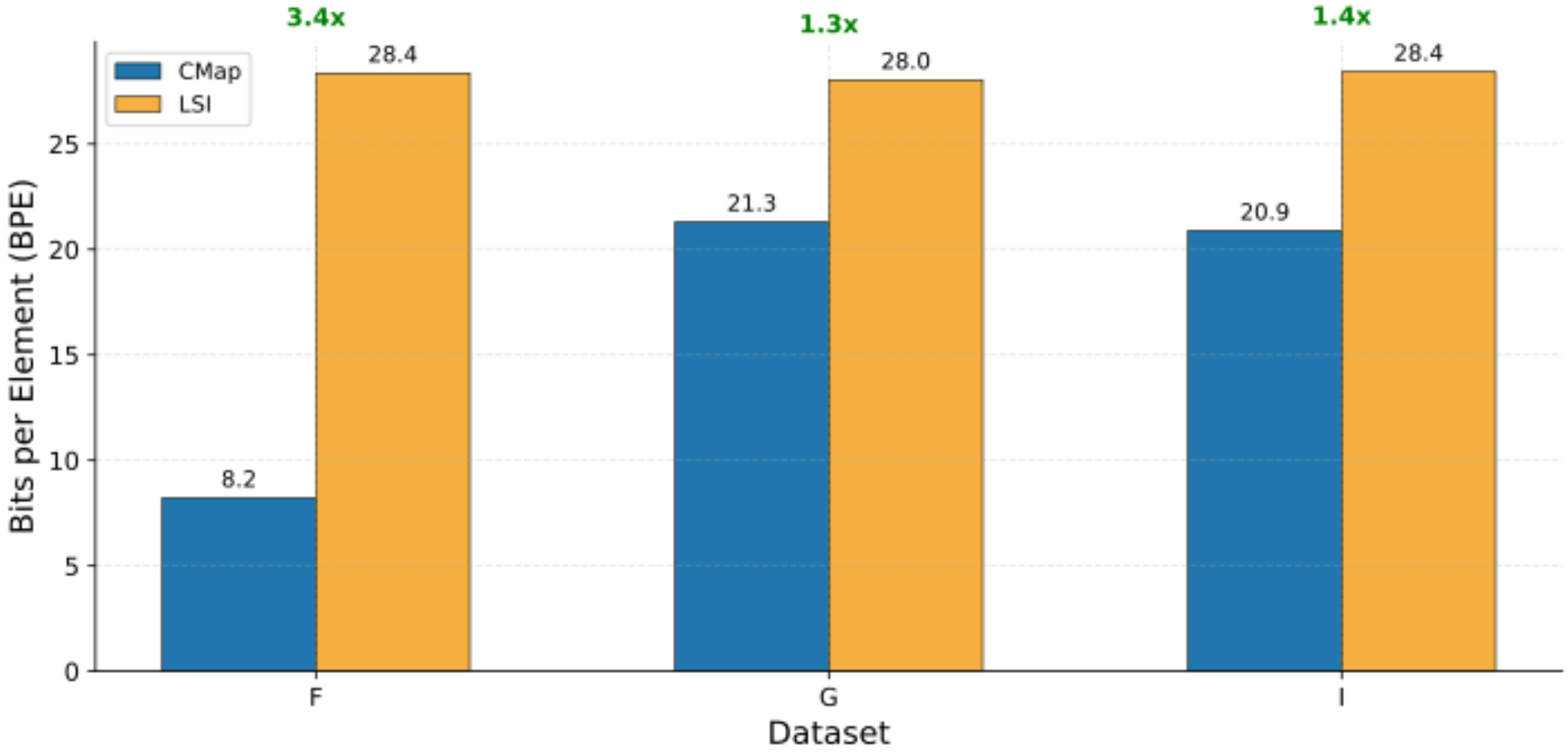
All Methods Combined



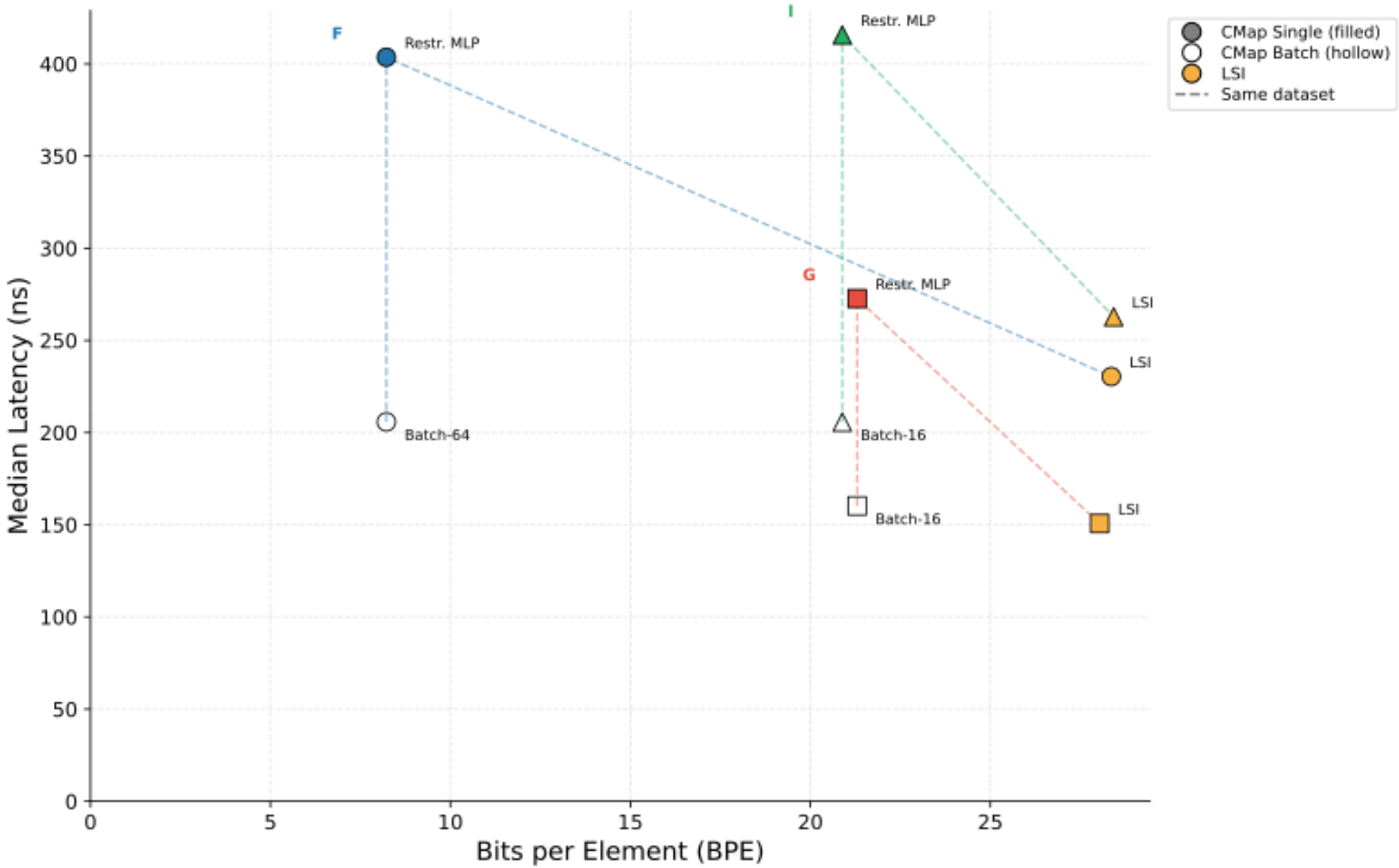
Batched Queries



Space Savings



Space-Time Tradeoff



Questions