Compression Algorithms for Geometries Supporting Operations

Simon Erlandsson & Leo Westerberg

Problem Formulation

Problem statement:

- To reduce the data size, compression algorithms can be applied to remove redundancies in the data.
- If using conventional compression algorithms, the data has to be decompressed before it can be operated on.

Research questions:

- Is it possible to perform operations on compressed geometric data without decompressing the entire geometries?
- How can domain-specific constraints and structures, in the context of maps, be exploited to improve the performance of operations and geometry compression?

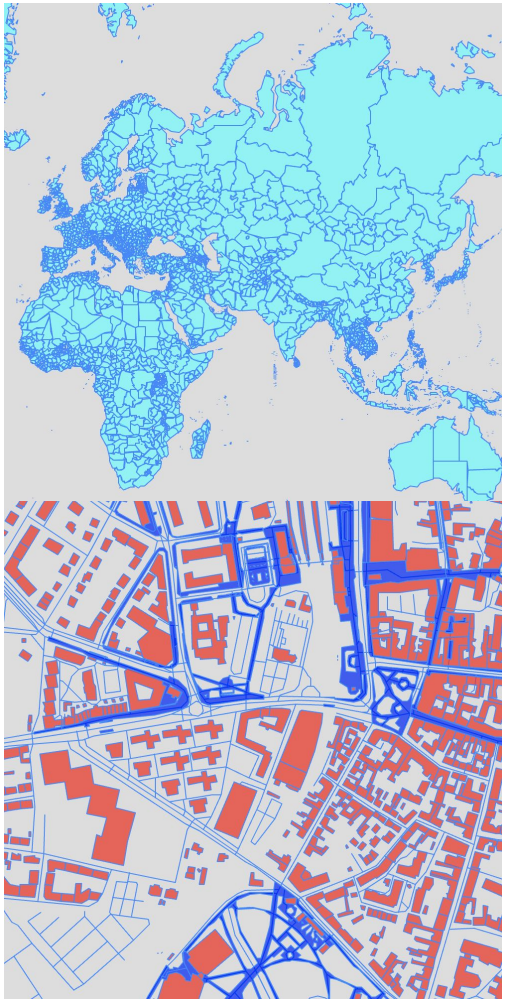
Reduce size + Fast operations!

Domain

- Structure of geometries:
  - Consists of an ordered sequence of coordinates (vertices)
  - Can have different types (see image)
  - Additional data for the geometric structure

	`POINT (30 10)`
	`LINESTRING (30 10, 10 30, 40 40)`
	`POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))`
	`POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10), (20 30, 35 35, 30 20, 20 30))`

- Data source: Open Street Map



Operations

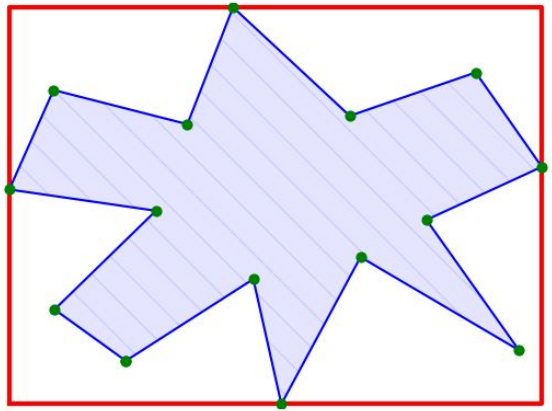
Operations for individual geometries:

- **Add vertex**
- **Bounding box**
- Compress
- Decompress

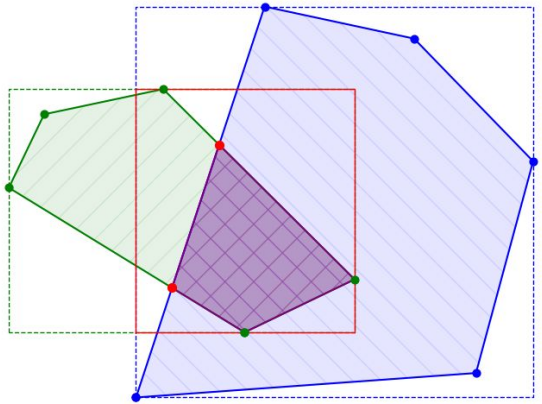
Operations between two geometries:

- **Is Intersecting**: True/False
- **Intersection:** Overlapping geometry

Example: Intersection is used to validate data.



Bounding Box



Intersecting Geometries

Compression: Integer Delta Encoding

- Avoid storing the X and Y in full.
- Store coordinate difference (delta).
- Fewer bits needed for smaller deltas.

Example:
77, 82, 72, 72 → 77, 5, -10, 0

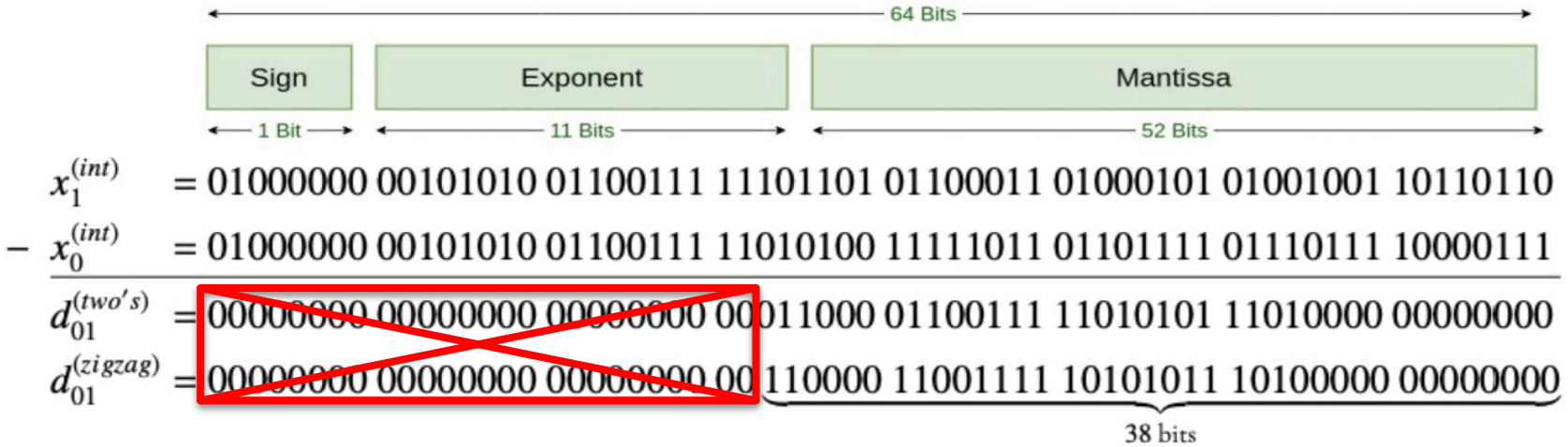
PROBLEM: Coordinates are in floating-point format!

Floating-Point Delta Encoding

- Interpret floating-points as an integer bit sequence

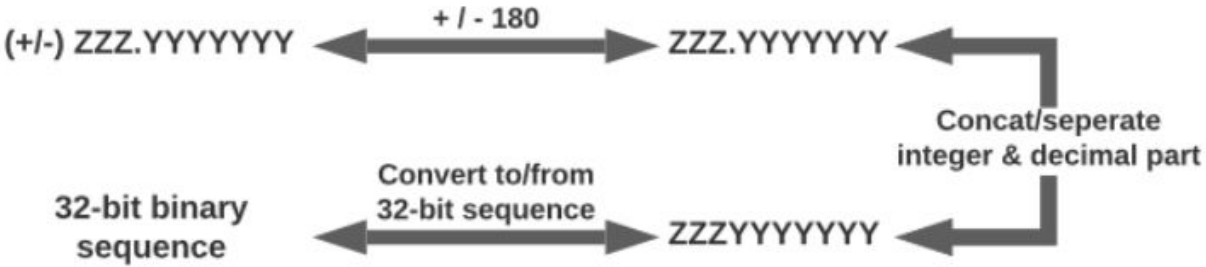
Example:

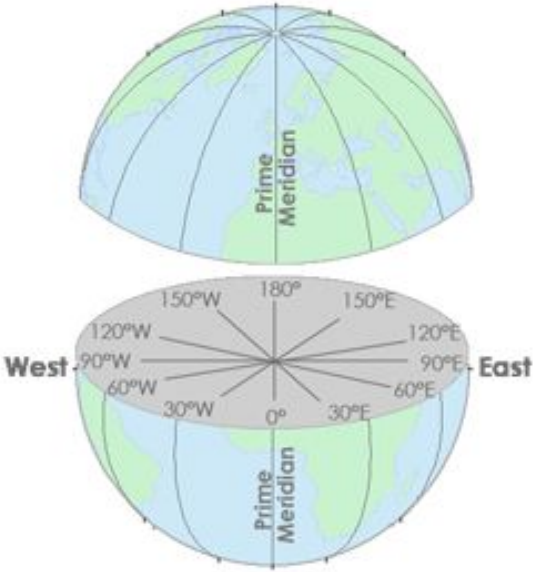
- (x0, x1) = (13.2027968, 13.2029830)
- The exponent, sign and prefix of mantissa cancel out.



32-bit Integer Decomposition Representation

- For coordinates with MAX 7 decimals of precision
- Possible values: 2 * 180 * 10^7 < 2^(32) - 1
- Fits in a 32-bit integer!

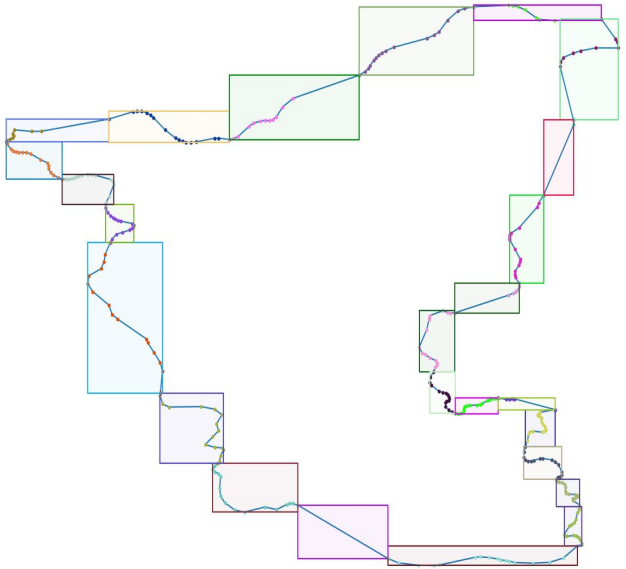




- Calculate delta, as previously.

Baseline: Floating-Point Delta

- Segment the coordinate sequence into **chunks** with the **first** coordinate **represented in full** while the rest are **delta encoded.**



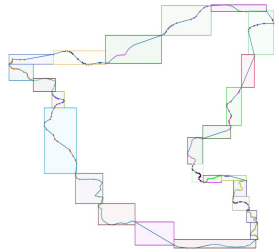
Baseline: Floating-Point Delta

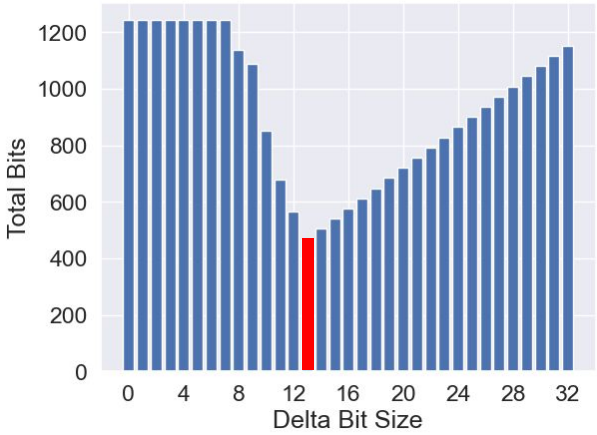
- Segment the coordinate sequence into **chunks** with the **first** coordinate **represented in full** while the rest are **delta encoded.**

- Delta bit-length is **optimally** selected for each geometry by calculating resulting total length for each setting.

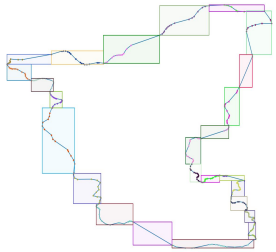
    - If a delta **can not fit** in the specified size, a **new chunk** is created

    - Tradeoff:
        More deltas, with **higher** bit-length
        Fewer deltas, with **lower** bit-length





Baseline: Floating-Point Delta

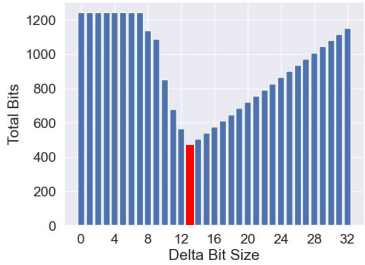


- Segment the coordinate sequence into **<u>chunks</u>** with the **first** coordinate **represented in full** while the rest are **delta encoded.**

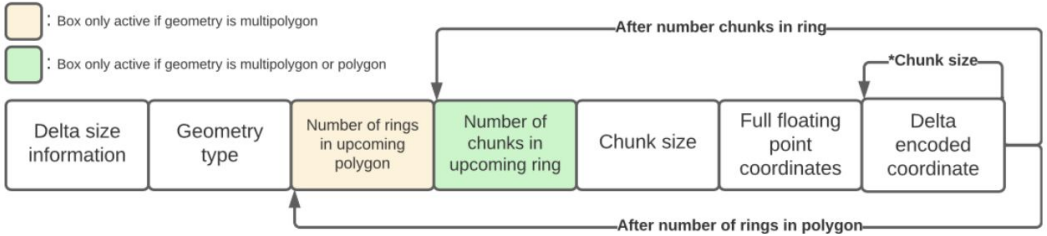
- Delta bit-length is **optimally** selected for each geometry by calculating resulting total length for each setting.

    - If a delta **can not fit** in the specified size, a **new chunk** is created

    - Tradeoff:
        More deltas, with **higher** bit-length
        Fewer deltas, with **lower** bit-length

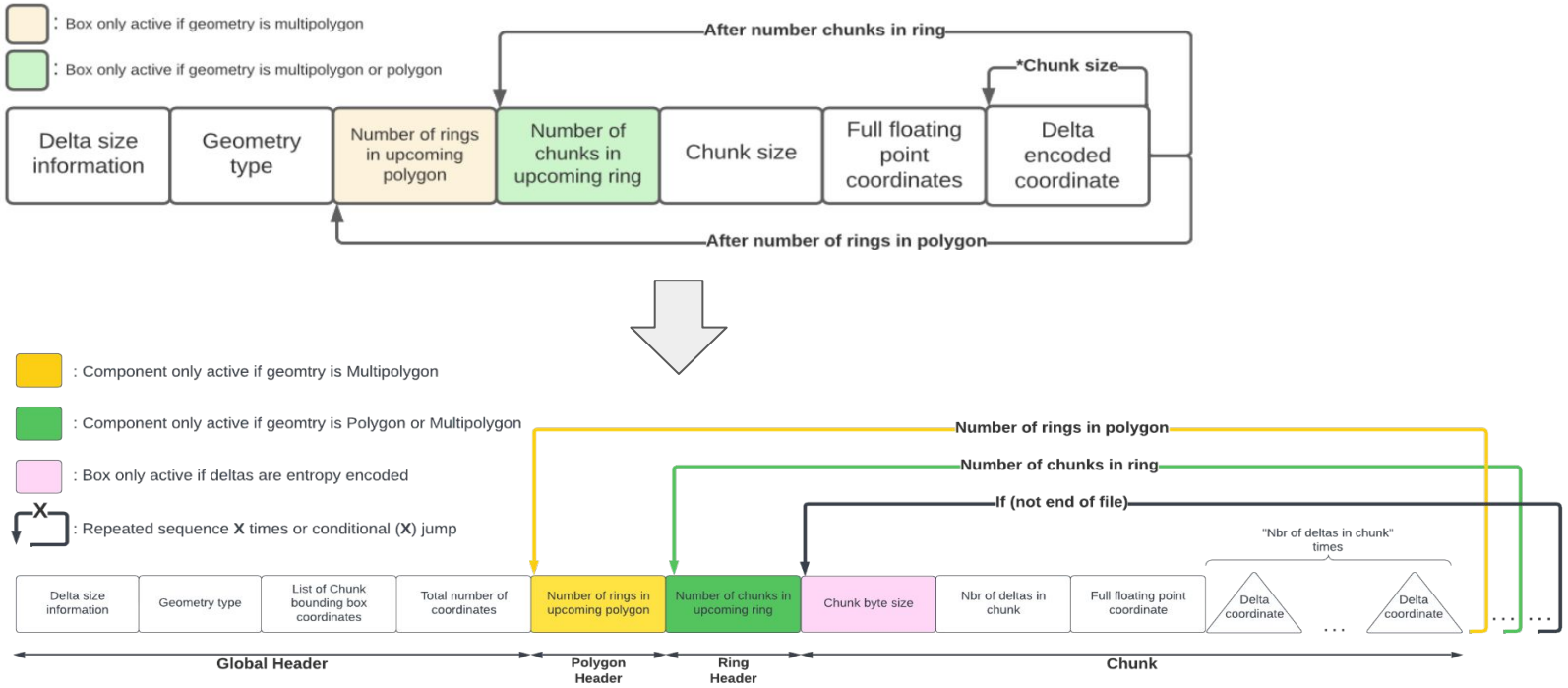


- Resulting format:



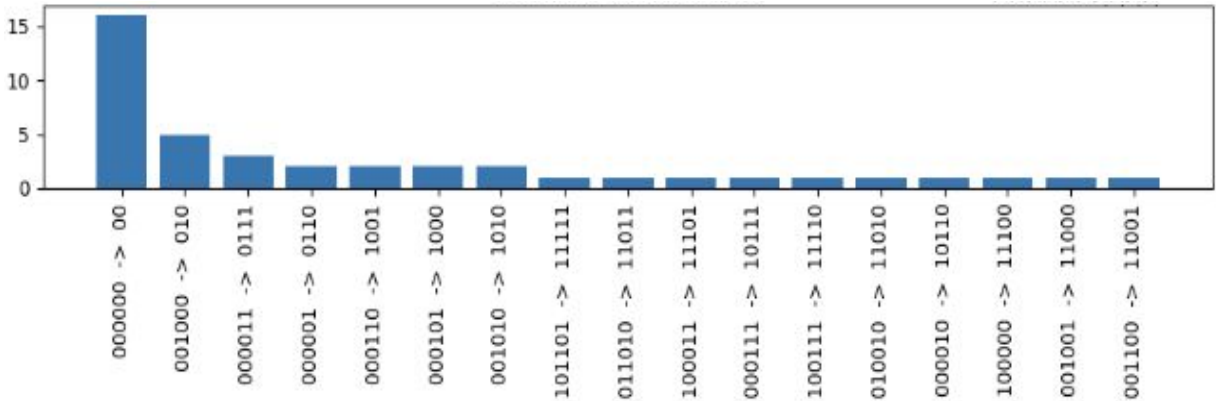
Floating-Point Delta Extended (FPDE)

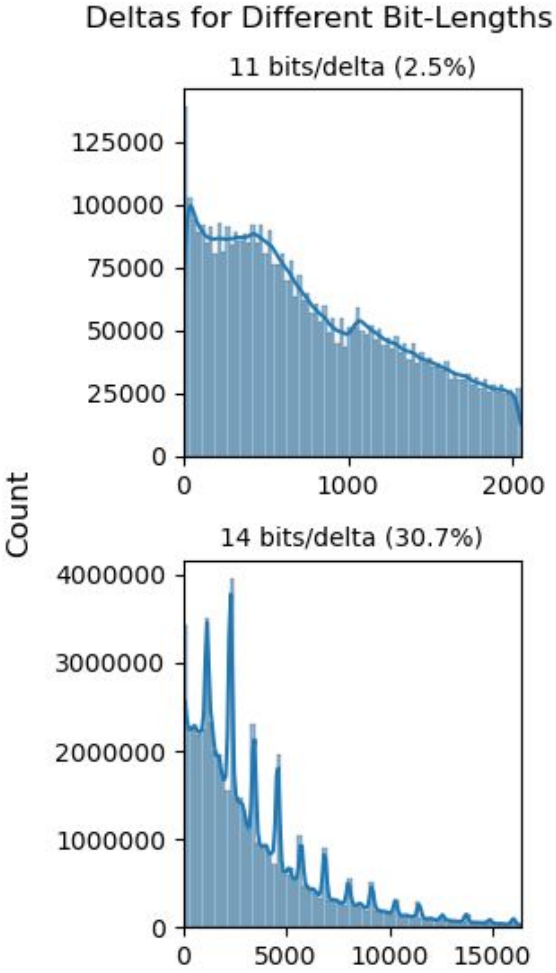
- Add integrated operability support.



Entropy Encoding (FPDE)

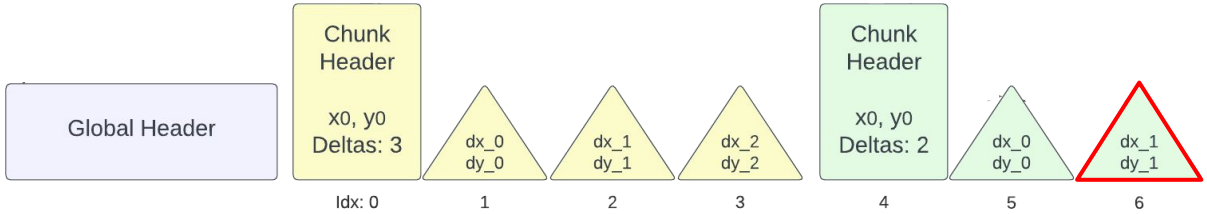
- Reduce size further with **Huffman, Golomb, etc.** encoding of deltas.
- Encode frequently occurring deltas with fewer bits, and infrequent ones with more.





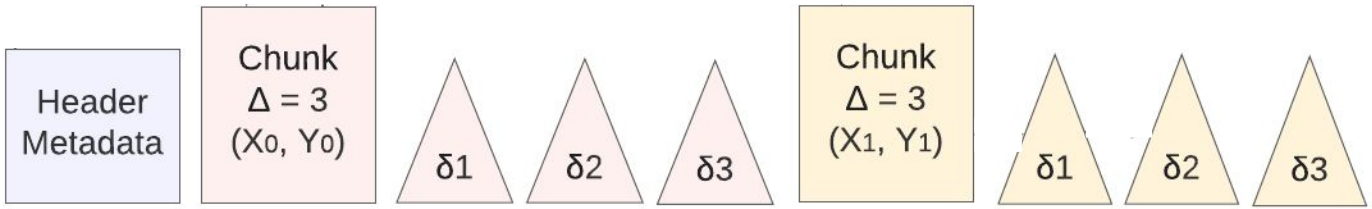
Operation Implementations

- **Random access**:



Add Vertex (*modifying*)

Add(6, (x_a, y_a)):

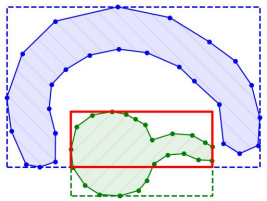


BYTES

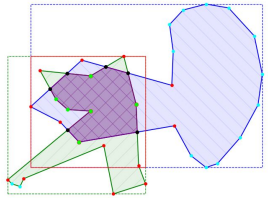
Intersection + IsIntersection (*binary*)

- Decompress chunks which have a bounding box overlapping with any of the other geometries chunk bounding box
- **Only** chunks where intersection can occur

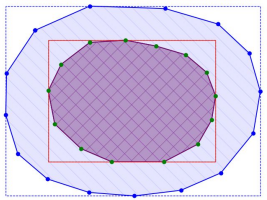
- 3 cases of intersection when bounding boxes overlap



Disjoint



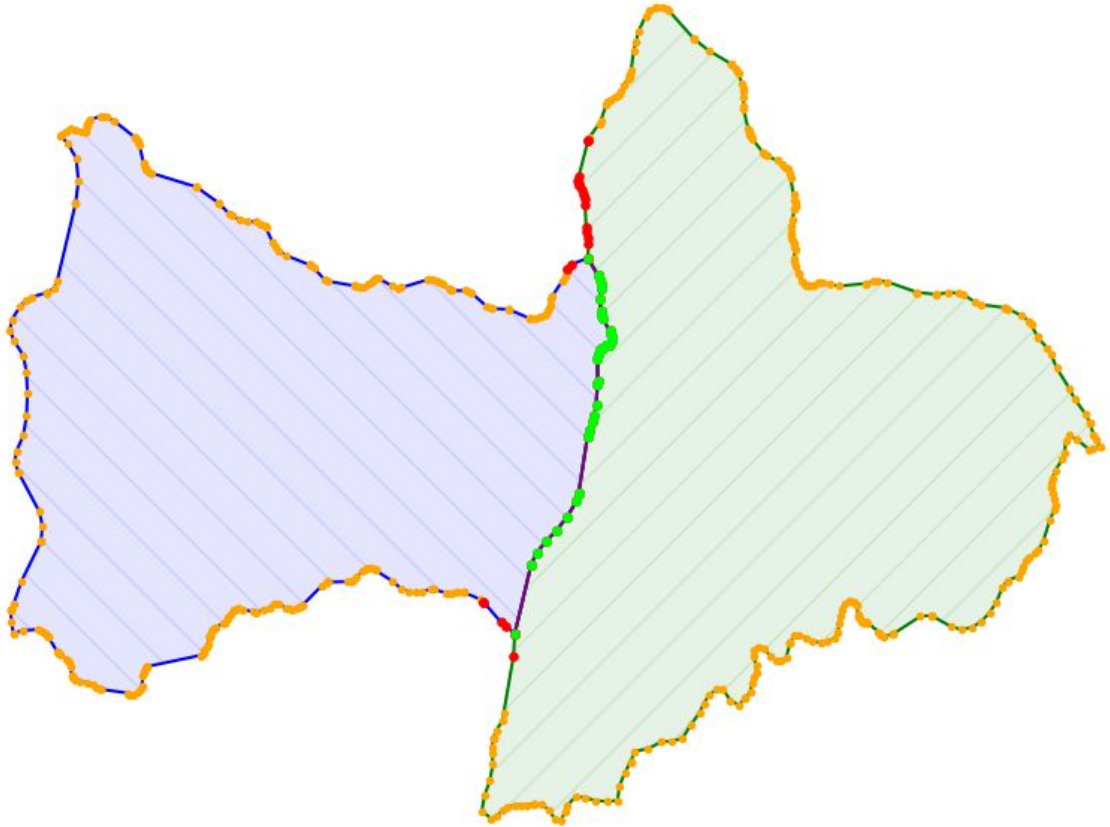
Crossing



Contained

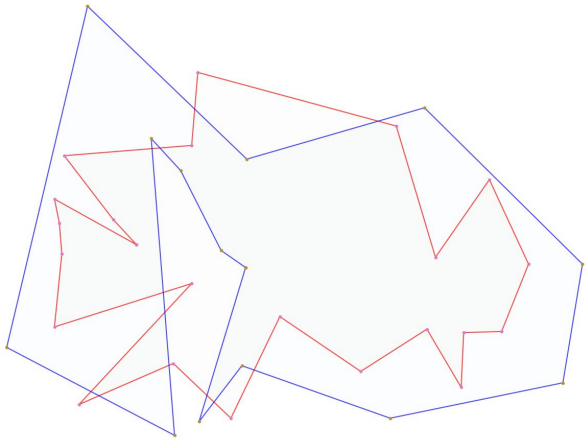
Intersection Filtering Example

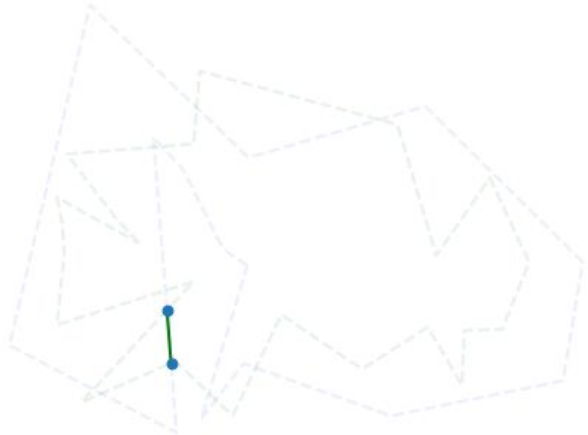
1. Get common bbox
2. Find chunks in common bbox.
3. Remove non-intersecting chunks.
4. Decompress remaining chunks.

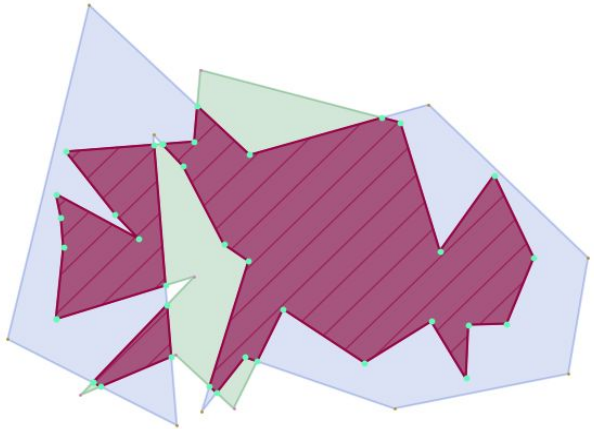


Intersection Example (Crossing)

Input → FPDE processing → Output





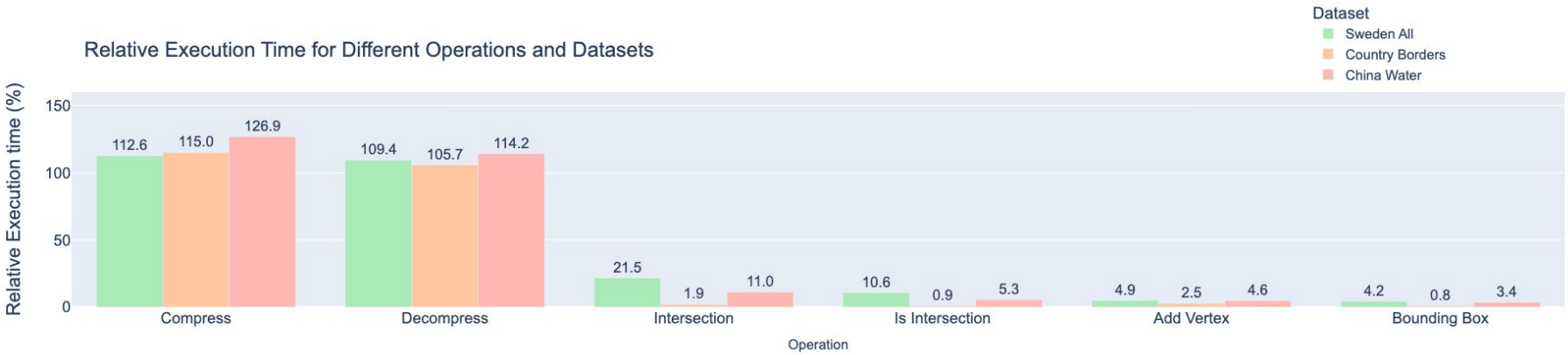


Benchmarking - Our Implementation (Speed)

- The baseline is Floating-Point Delta without optimizations for operations (Decompression + operations on full geometry)

- Random pairs for binary operation
- FPDE performs operations much faster than baseline for various datasets

- Misleading for (Is) Intersection

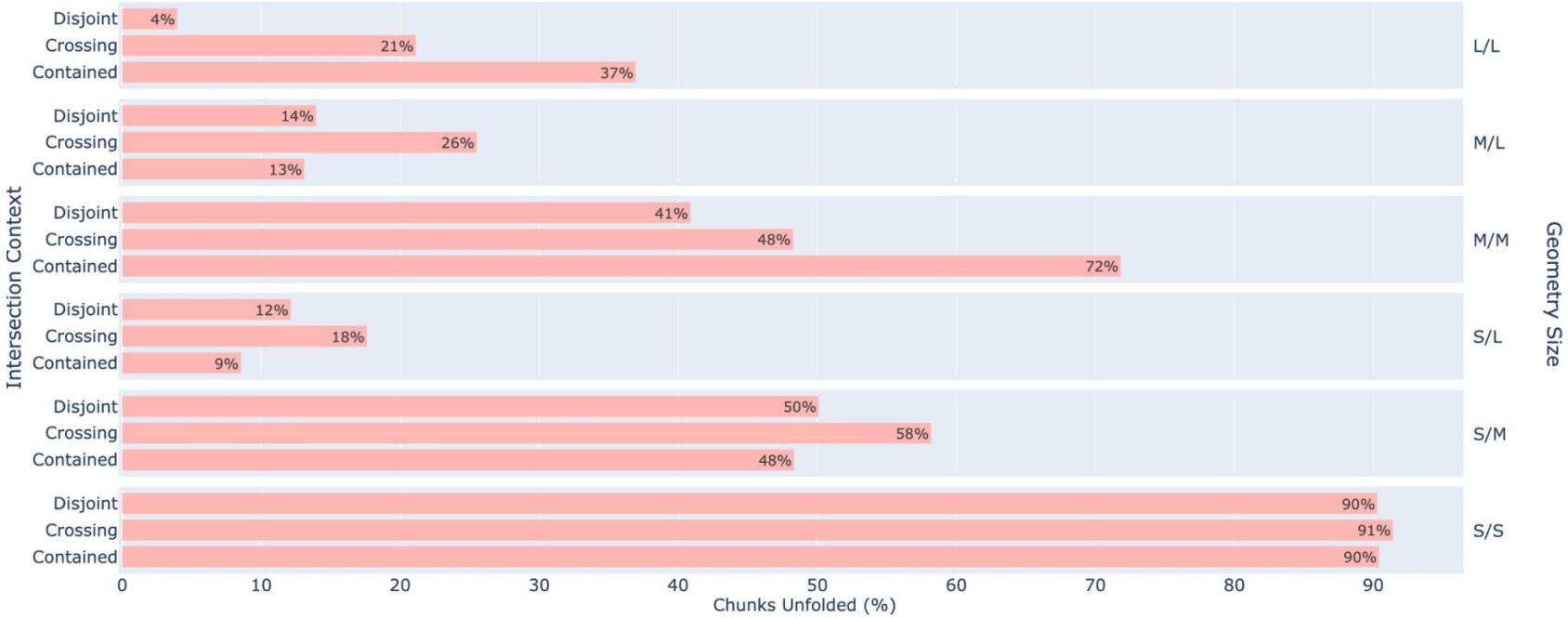


Benchmarking - Our Implementation (Intersection)

- Divide benchmarks for different geometry contexts
- More effective when including large geometries
- S = Small M = Medium L = Large

Benchmarking - Our Implementation (Intersection)

- Fraction of chunks partially decompressed for each intersection context



Benchmarking - Our Implementation (Size)

- WKB (Well-Known Binary) is a common standard format for representing geometries.

Conclusion

- In general, our format performs operations faster as compared to *full decompression* + library operation.
- Comp. factor and speed *depends heavily* on the geometries. Different solutions for different sizes.
- Performance comes at the expense of compressibility.
- Maps data compresses well by simple schemes (delta encoded integer coordinates) + entropy encoding.

Extra big thanks to:

Björn Pedersen
Hampus Londögård
Patrick Cording
Per Svensson
Martin Lindberg

THANK YOU FOR LISTENING

QUESTIONS?