UNIVERSIDADE DA CORUÑA

# Sparse Tetris

## Reconstructing Sparse Matrices with Polyhedra

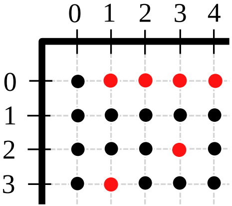
**Gabriel Rodríguez** Louis-Noël Pouchet

13th International Workshop on Polyhedral Compilation Techniques, IMPACT 2023
Toulouse, FR, January 16th, 2023

# Motivation



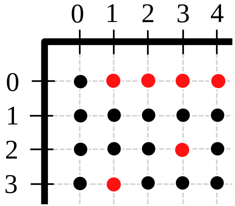
- Sparse data structures are central to scientific computing:
  - Graph processing, neural net inference after weight pruning, etc.
- In order to save storage space and computations, sparse representation formats have been introduced, e.g., COO:




```
row_idx  = [0,0,0,0,2,3]
col_idx = [1,2,3,4,3,1]
data     = [ ... ]
```

# Motivation

- Index arrays can be compressed to reduce storage footprint, e.g., CSR:

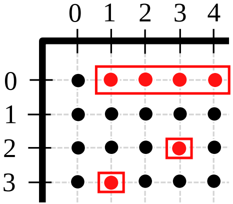



```
row_ptr = [0,4,4,5,6]
col_idx = [1,2,3,4,3,1]
data    = [ ... ]
```

- Many different compression schemes are possible, e.g., DIA, CSC, BCRS, CDS, etc.

# Motivation

- Index arrays can be compressed to reduce storage footprint, e.g., CSR:




```
row_ptr = [0,4,4,5,6]
col_idx = [1,2,3,4,3,1]
data    = [ ... ]
```

- Many different compression schemes are possible, e.g., DIA, CSC, BCRS, CDS, etc.

## Core idea

- **Encode sparsity structure as convex polyhedra.**

$$\{i, j | (i = 0 \wedge 1 \leq j \leq 4); (i = 2, j = 3); (i = 3, j = 1)\}$$

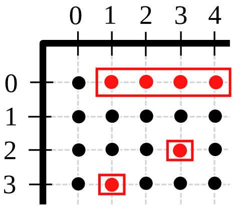
# An application: SpMV



- Typically, code is generic for any sparsity structure, e.g., CSR:

```
for (i = 0; i < nrows; i++)
  for (j = pos[i]; j <= pos[i+1]; j++)
    y[i] += A[j] * x[cols[j]];
```

- Data-specific codes (DSCG)[1]: program specialized for sparse structure:




```
for (j = 1; j <= 4; j++)
  y[0] += A[j-1] * x[j];
y[2] += A[4] * x[3];
y[3] += A[5] * x[1];
```

[1] [T. Augustine et al. Generating Piecewise-Regular Code from Irregular Structures. PLDI19]

# An Application: SpMV



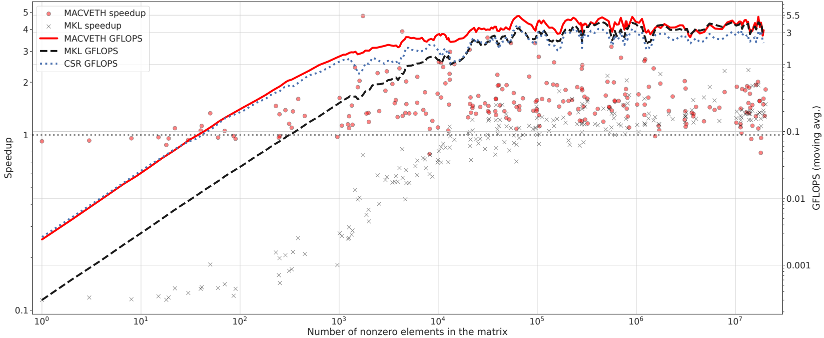



Figure: M. Horro et al. Custom High-Performance Vector Code Generation for Data-Specific Sparse Computations. PACT 2022.

- Good performance demonstrated for SpMV-DSCG codes.
- Ad-hoc vectorization can be used to push the limits of the technique.

# Sparse-Polyhedral Format



- In previous work, the polyhedral description was attached to a CSR file.
- The CSR data vector was **NOT reordered.**
- Need to find regularity over three streams:

```
row_idx = [0,0,0,0,2,3]
col_idx = [1,2,3,4,3,1]
data_idx = [0,1,2,3,4,5]
```

```
for (j = 1; j <= 4; j++)
  y[0] += A[j-1] * x[j];
y[2] += A[4] * x[3];
y[3] += A[5] * x[1];
```

# Sparse-Polyhedral Format

## Informal specification

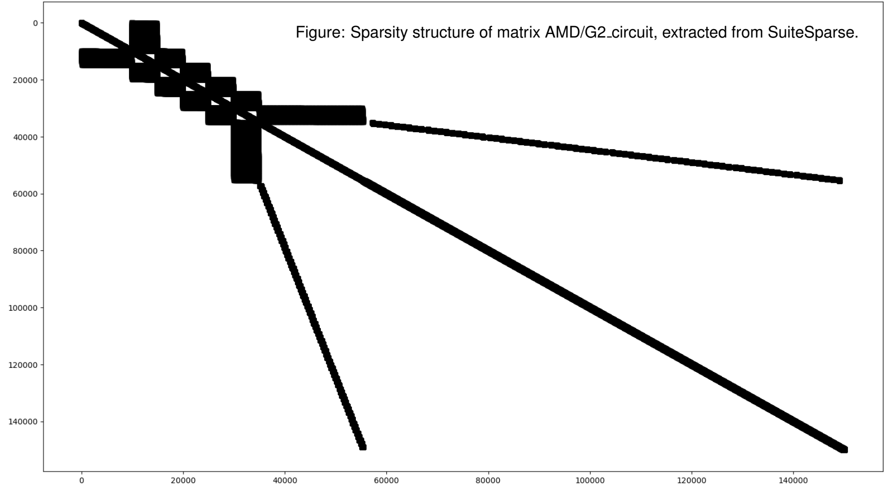
- Encode sparsity as polyhedra.
- A matrix is a dictionary of shapes, including:
  - Dimensionality.
  - Shape encoding (rectangle, vertices, ISL).
  - List of origins.
  - Pointer to start of data in **reordered** data array.

- Need to find regularity over **two streams** only.

```
row_idx = [0,0,0,0,2,3]
col_idx = [1,2,3,4,3,1]
data_idx = [0,1,2,3,4,5]
```

# Finding polyhedra in sparsity





# Finding polyhedra in sparsity



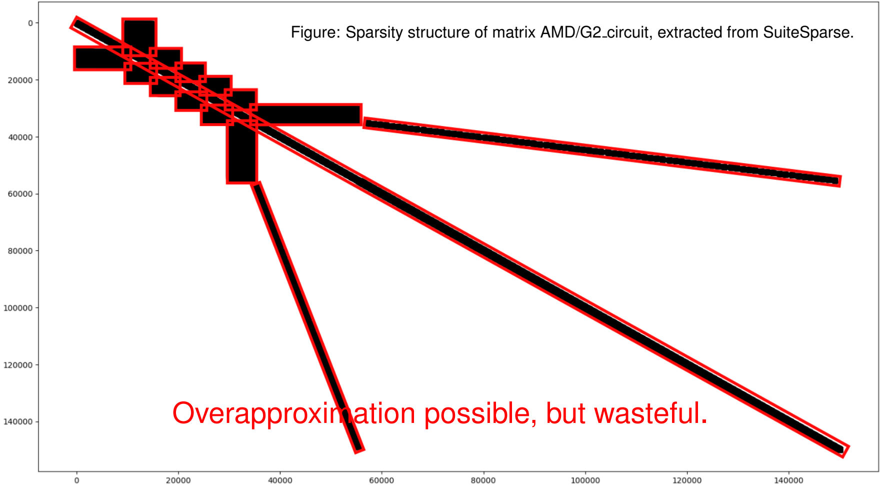
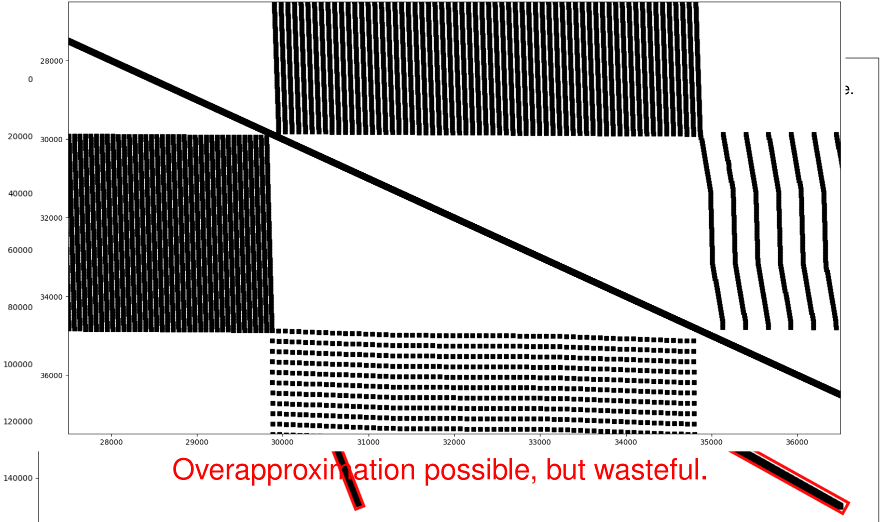



Figure: Sparsity structure of matrix AMD/G2_circuit, extracted from SuiteSparse.

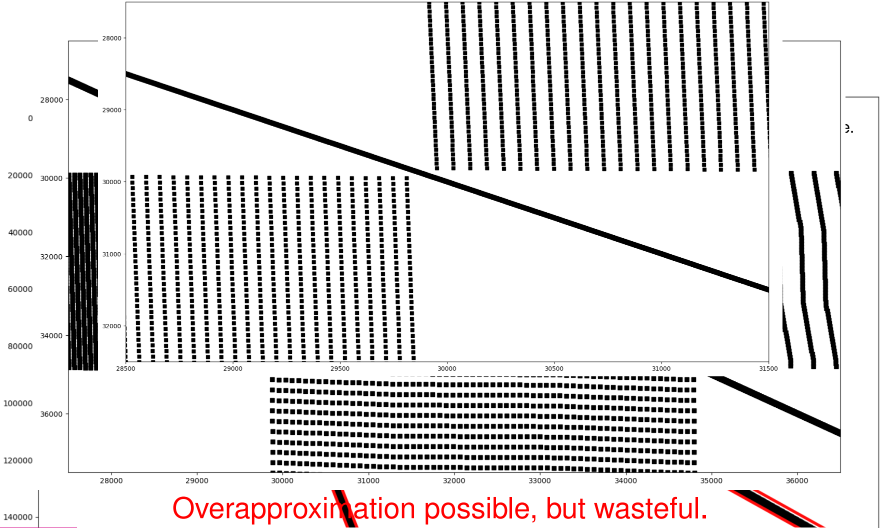
# Finding polyhedra in sparsity






Overapproximation possible, but wasteful.

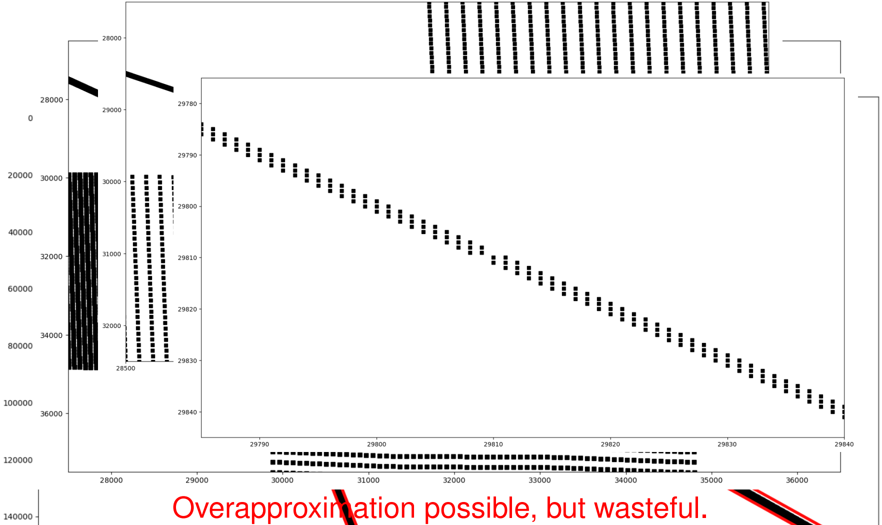
# Finding polyhedra in sparsity




Overapproximation possible, but wasteful.

# Finding polyhedra in sparsity






Overapproximation possible, but wasteful.

# Finding polyhedra in sparsity

## Two complementary approaches

### Trace compression

- Find multidimensional polyhedra that generate a sequence of points.
- Fewer pieces, higher dimensionality, irregular.
- Leads to smaller codes with inefficient loops.

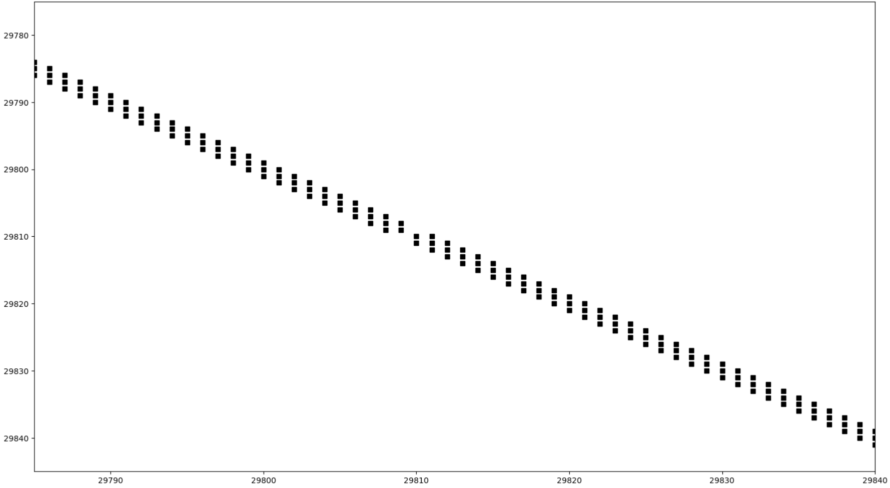
e.g., Ketterlin & Clauss, CGO08; Rodríguez et al., CGO16.

### Pattern-matching

- Pre-defined set of patterns of interest.
- Matching applied over entire sparse structure.
- Many pieces, regular, predefined dimensionality.
- Leads to larger codes with no loops, just SIMD (for 1-d pieces).
- **Can be fused into higher dimensional, regular pieces.**

# Finding polyhedra in sparsity
## Pattern-matching

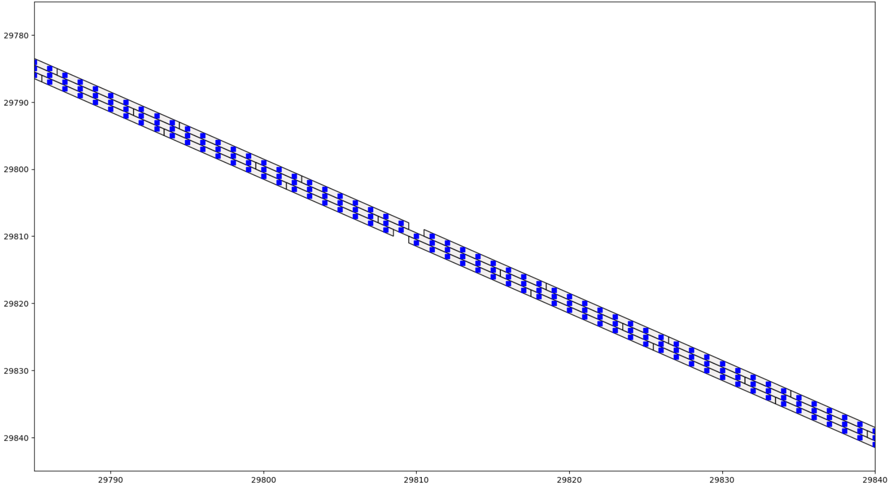




# Finding polyhedra in sparsity

## Pattern-matching

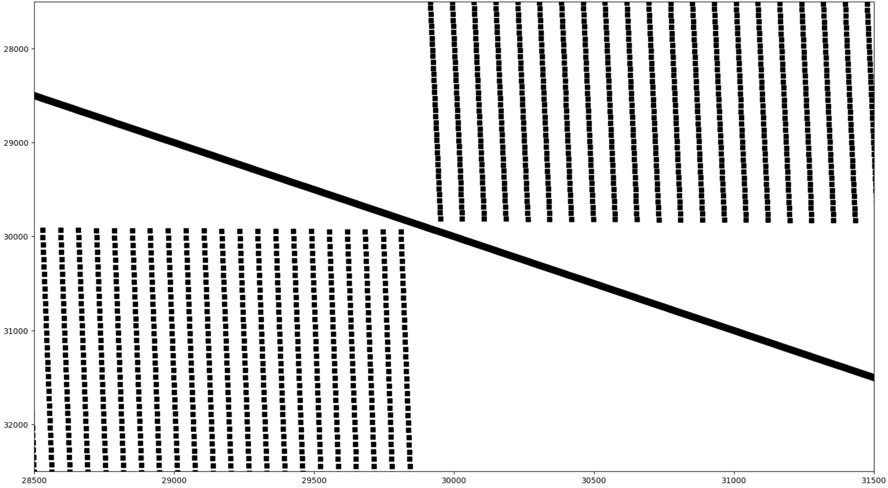




# Finding polyhedra in sparsity

## Trace reconstruction

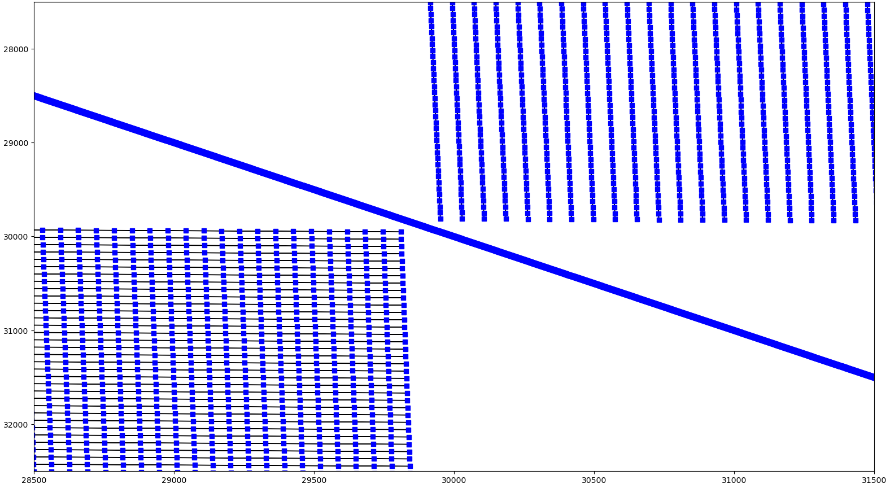




# Finding polyhedra in sparsity

## Trace reconstruction

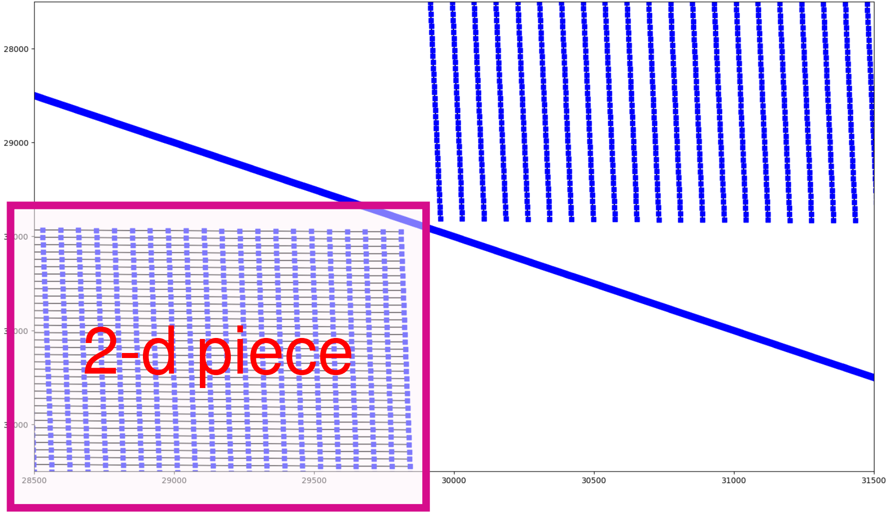




# Finding polyhedra in sparsity

## Trace reconstruction



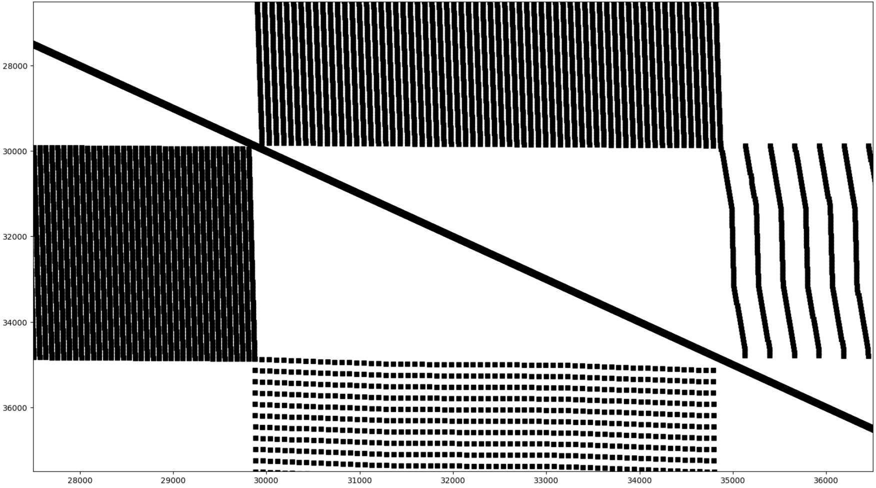


# Finding polyhedra in sparsity

## Hybrid approach

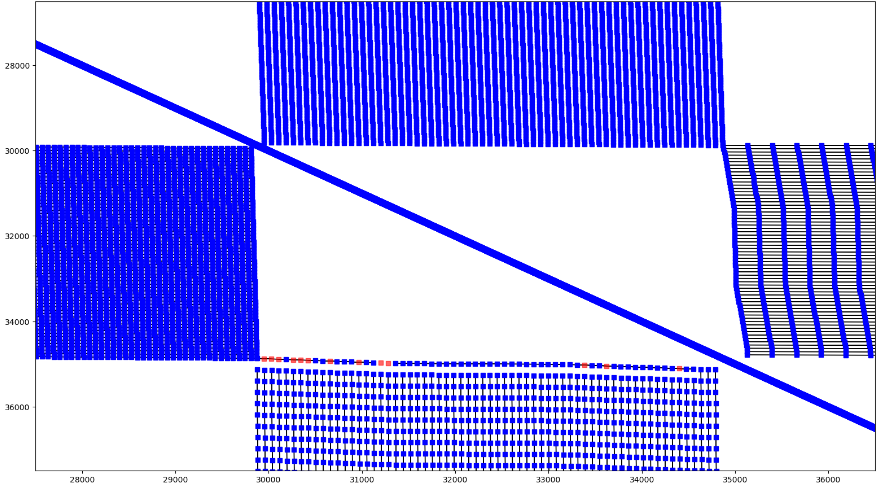
Mining for regularity

Mining for regularity



# Finding polyhedra in sparsity

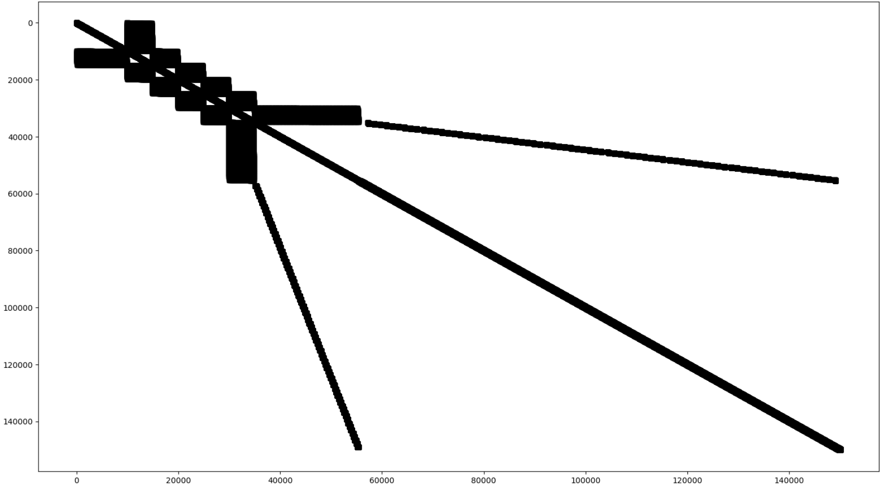
## Hybrid approach



# Finding polyhedra in sparsity

## Hybrid approach

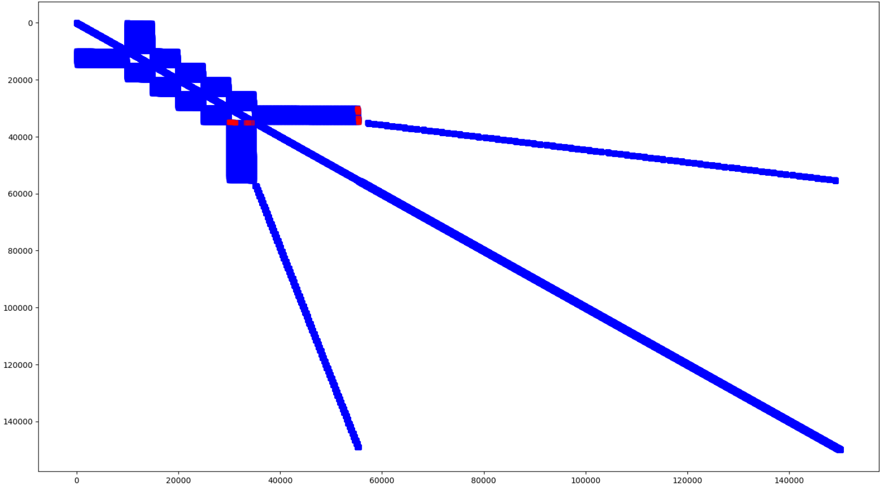




# Finding polyhedra in sparsity

## Hybrid approach

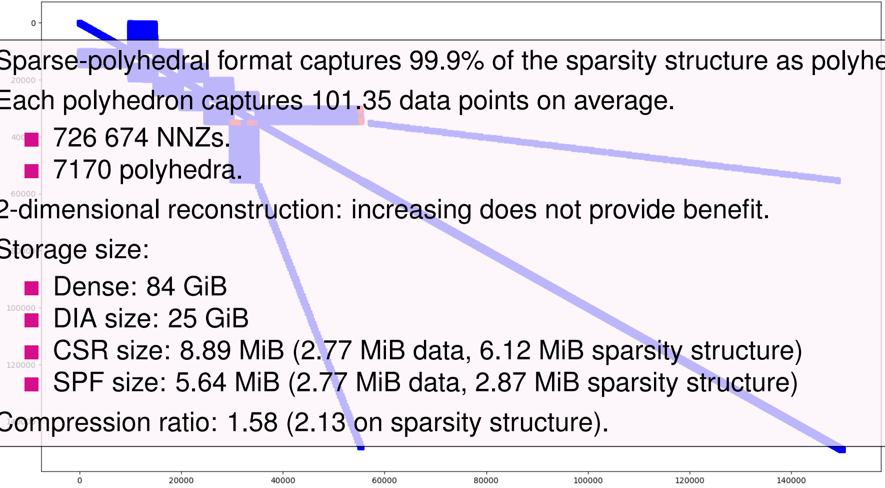




# Finding polyhedra in sparsity

## Hybrid approach

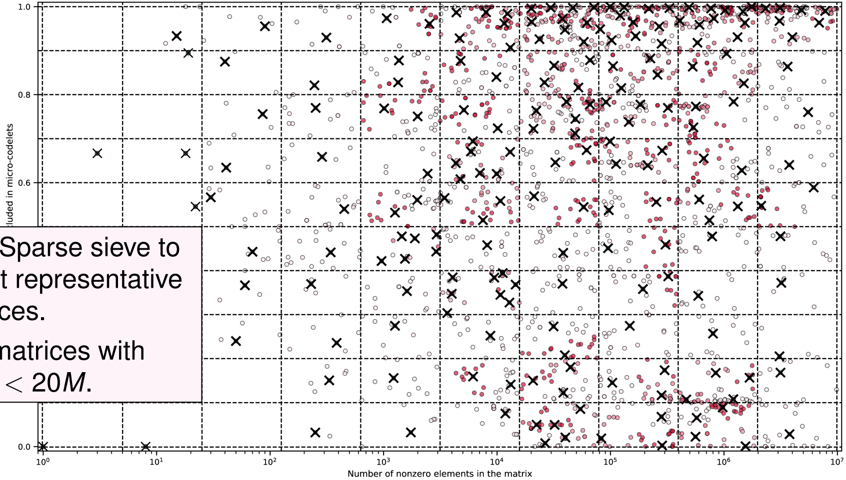




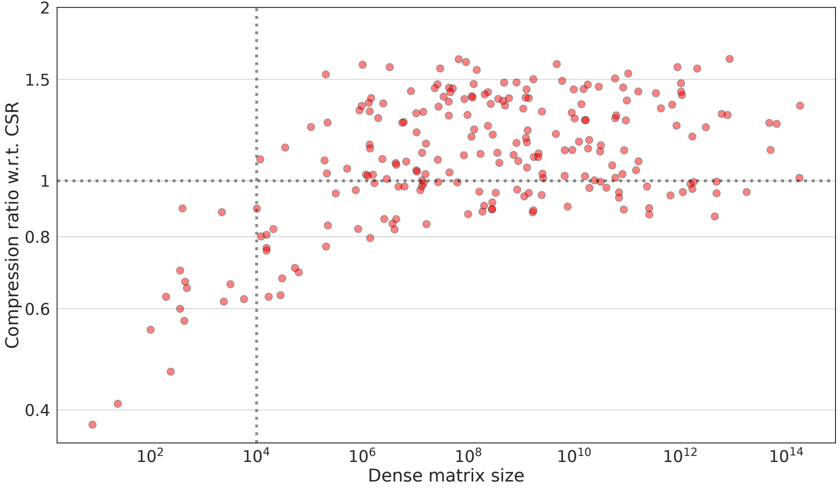

- Sparse-polyhedral format captures 99.9% of the sparsity structure as polyhedra.
- Each polyhedron captures 101.35 data points on average.
  - 726 674 NNZs.
  - 7170 polyhedra.
- 2-dimensional reconstruction: increasing does not provide benefit.
- Storage size:
  - Dense: 84 GiB
  - DIA size: 25 GiB
  - CSR size: 8.89 MiB (2.77 MiB data, 6.12 MiB sparsity structure)
  - SPF size: 5.64 MiB (2.77 MiB data, 2.87 MiB sparsity structure)
- Compression ratio: 1.58 (2.13 on sparsity structure).

# Experimental Setup

## Matrix selection



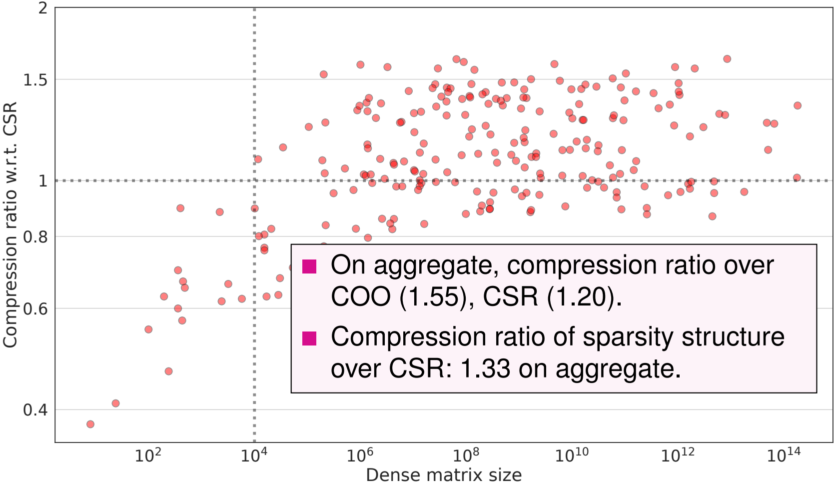

- SuiteSparse sieve to select representative matrices.
- 230 matrices with $NNZ < 20M$.



Compression ratio w.r.t. CSR
2
1.5
1
0.8
0.6
0.4
$10^2$
$10^4$
$10^6$
$10^8$
$10^{10}$
$10^{12}$
$10^{14}$
Dense matrix size

# Experimental results

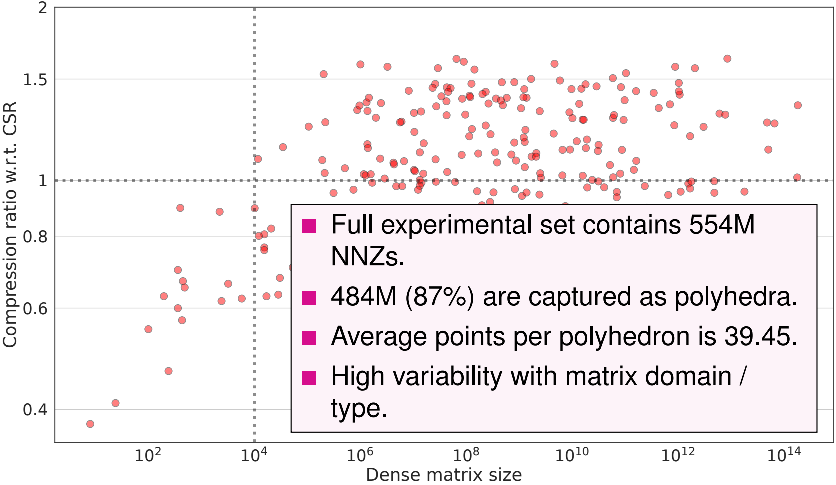
## Compression




- On aggregate, compression ratio over COO (1.55), CSR (1.20).
- Compression ratio of sparsity structure over CSR: 1.33 on aggregate.

# Experimental results

## Compression




- Full experimental set contains 554M NNZs.
- 484M (87%) are captured as polyhedra.
- Average points per polyhedron is 39.45.
- High variability with matrix domain / type.

# Challenges and remarks



- Problem is (most likely :-)) NP-complete: merging points into a piece affects the selection of other pieces.
- Shape selection driven by target:
  - Compression.
  - Kernel-specific approaches (e.g., data locality).
  - Domain-specific approaches (e.g., typical shapes).
  - Hardware-specific approaches (e.g., SIMD).
- Code generation approach for sparse linear algebra.
- Extensive study on 200+ matrices demonstrates potential benefits.
- Applications of the polyhedra-over-CSR representation have demonstrated good performance.

UNIVERSIDADE DA CORUÑA

Colorado State University

NSF

# Sparse Tetris

Thank you for your time!

**Gabriel Rodríguez** Louis-Noël Pouchet

13th International Workshop on Polyhedral Compilation Techniques, IMPACT 2023
Toulouse, FR, January 16th, 2023