Improving Matrix-vector Multiplication via Lossless Grammar Compressed Matrices

Paolo Ferragina, Travis Gagie, Dominik Köppl, Giovanni Manzini, Gonzalo Navarro, Manuel Striani, Francesco Tosoni

Francesco Tosoni, MSc
PhD Student in Computer Science
L.go B. Pontecorvo 3, 56127 Pisa PI, Italy
Informatica, ed. C, piano 2°
francesco.tosoni@phd.unipi.it

Main Results

A new **lossless compression scheme for real-valued matrices**

- good compression ratios
- reducing time for linear-algebra operations.
- achieve compression ratio bounded in terms of the **k-th order empirical entropy** of input matrix; and
- **the cost of matrix-vector multiplications is proportional to the size of the *compressed* matrix.**

Experiments performed on Machine Learning matrices show our approach is faster and more lightweight than competing methods.

Very Large Data Bases

VLDB 2022

Background (1/4)

Very large matrices present **scalability challenges**:

- storage and operations
- bandwidth resources in server-to-client transmissions
- CPU/GPU memory communications

⇒ matrix compression appears as an attractive choice.

Background (2/4)

***Lossy* compression** impairs the accuracy of the subsequent operations.

- low-precision storage
- sparsification & quantisation

⇒ attentive and manual application.

***Lossless* compression** is a better "*automated*" alternative:

- data-independent
- No need of *a priori* knowledge about the input data

Background (3/4)

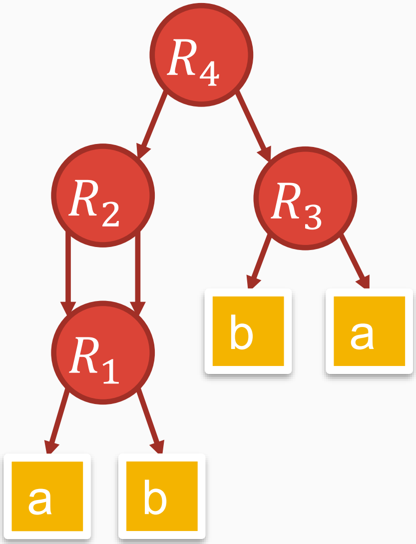
Traditional tools (Huffman, Lempel-Ziv, bzip, RLE) perform poorly on matrices:

- Not always able to unfold the (hidden) **dependencies** between rows and columns
- Require the **full-matrix decompression** for the linear-algebra operations

Background (4/4)

In the following, as (lossless) compressor we use `RePair`, which is a **grammar compressor** based on **straight line programs (SLP).**

$$S = ababba$$



$$R_1 \rightarrow ab$$
$$R_2 \rightarrow R_1R_1$$
$$R_3 \rightarrow ba$$
$$R_4 \rightarrow R_2R_3$$

Our Approach

The CSRV Representation

Compute **the CSRV representation of a matrix**, a modification of the CSR representation.

- V is the set of distinct nonzeros.
- S is a sequence of pairs. The first element is the index of the nonzero in V; the second is the column index.

The special symbol $ is the row delimiter.

CSRV: an example.

$$\begin{array}{c} \\ \\ \end{array}\begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix}$$

$$\begin{bmatrix} 1.2 & 3.4 & 5.6 & 0 & 2.3 \\ 2.3 & 0 & 2.3 & 4.5 & 1.7 \\ 1.2 & 3.4 & 2.3 & 4.5 & 0 \\ 3.4 & 0 & 5.6 & 0 & 2.3 \\ 2.3 & 0 & 2.3 & 4.5 & 0 \\ 1.2 & 3.4 & 2.3 & 4.5 & 3.4 \end{bmatrix}$$

$$V = [\overset{1}{1.2} \quad \overset{2}{1.7} \quad \overset{3}{2.3} \quad \overset{4}{3.4} \quad \overset{5}{4.5} \quad \overset{6}{5.6}]$$

$$\begin{aligned} S = {} & \langle 1,1\rangle\ \langle 4,2\rangle\ \langle 6,3\rangle\ \langle 3,5\rangle\ \$\ \langle 3,1\rangle\ \langle 3,3\rangle\ \langle 5,4\rangle\ \langle 2,5\rangle\ \$ \\ & \langle 1,1\rangle\ \langle 4,2\rangle\ \langle 3,3\rangle\ \langle 5,4\rangle\ \$\ \langle 4,1\rangle\ \langle 6,3\rangle\ \langle 3,5\rangle\ \$ \\ & \langle 3,1\rangle\ \langle 3,3\rangle\ \langle 5,4\rangle\ \$\ \langle 1,1\rangle\ \langle 4,2\rangle\ \langle 3,3\rangle\ \langle 5,4\rangle\ \langle 4,5\rangle\ \$ \end{aligned}$$

CSRV: an example.

$$
\begin{array}{ccccc}
1 & 2 & 3 & 4 & 5
\end{array}
$$

$$
\begin{bmatrix}
1.2 & 3.4 & 5.6 & 0 & 2.3 \\
2.3 & 0 & 2.3 & 4.5 & 1.7 \\
1.2 & 3.4 & 2.3 & 4.5 & 0 \\
3.4 & 0 & 5.6 & 0 & 2.3 \\
2.3 & 0 & 2.3 & 4.5 & 0 \\
1.2 & 3.4 & 2.3 & 4.5 & 3.4
\end{bmatrix}
$$

$$
V = [\overset{1}{1.2} \quad \overset{2}{1.7} \quad \overset{3}{2.3} \quad \overset{4}{3.4} \quad \overset{5}{4.5} \quad \overset{6}{5.6}]
$$

$$
\begin{aligned}
S = {} & \langle 1,1\rangle \ \langle 4,2\rangle \ \langle 6,3\rangle \ \langle 3,5\rangle \ \$ \ \langle 3,1\rangle \ \langle 3,3\rangle \ \langle 5,4\rangle \ \langle 2,5\rangle \ \$ \\
& \langle 1,1\rangle \ \langle 4,2\rangle \ \langle 3,3\rangle \ \langle 5,4\rangle \ \$ \ \langle 4,1\rangle \ \langle 6,3\rangle \ \langle 3,5\rangle \ \$ \\
& \langle 3,1\rangle \ \langle 3,3\rangle \ \langle 5,4\rangle \ \$ \ \langle 1,1\rangle \ \langle 4,2\rangle \ \langle 3,3\rangle \ \langle 5,4\rangle \ \langle 4,5\rangle \ \$
\end{aligned}
$$

CSRV: an example.

$$
\begin{array}{ccccc}
1 & 2 & 3 & 4 & 5
\end{array}
$$

$$
\begin{bmatrix}
1.2 & 3.4 & 5.6 & 0 & 2.3 \\
2.3 & 0 & 2.3 & 4.5 & 1.7 \\
1.2 & 3.4 & 2.3 & 4.5 & 0 \\
3.4 & 0 & 5.6 & 0 & 2.3 \\
2.3 & 0 & 2.3 & 4.5 & 0 \\
1.2 & 3.4 & 2.3 & 4.5 & 3.4
\end{bmatrix}
$$

$$
V = \overset{1}{[1.2} \quad \overset{2}{1.7} \quad \overset{3}{2.3} \quad \overset{4}{3.4} \quad \overset{5}{4.5} \quad \overset{6}{5.6]}
$$

$$
\begin{aligned}
S = {} & \langle 1,1\rangle \langle 4,2\rangle \langle 6,3\rangle \langle 3,5\rangle \$ \langle 3,1\rangle \langle 3,3\rangle \langle 5,4\rangle \langle 2,5\rangle \$ \\
& \langle 1,1\rangle \langle 4,2\rangle \langle 3,3\rangle \langle 5,4\rangle \$ \langle 4,1\rangle \langle 6,3\rangle \langle 3,5\rangle \$ \\
& \langle 3,1\rangle \langle 3,3\rangle \langle 5,4\rangle \$ \langle 1,1\rangle \langle 4,2\rangle \langle 3,3\rangle \langle 5,4\rangle \langle 4,5\rangle \$
\end{aligned}
$$

Grammar Compression

$$S = \langle 1,1\rangle\ \langle 4,2\rangle\ \langle 6,3\rangle\ \langle 3,5\rangle\ \$\ \langle 3,1\rangle\ \langle 3,3\rangle\ \langle 5,4\rangle\ \langle 2,5\rangle\ \$$$
$$\langle 1,1\rangle\ \langle 4,2\rangle\ \langle 3,3\rangle\ \langle 5,4\rangle\ \$\ \langle 4,1\rangle\ \langle 6,3\rangle\ \langle 3,5\rangle\ \$$$
$$\langle 3,1\rangle\ \langle 3,3\rangle\ \langle 5,4\rangle\ \$\ \langle 1,1\rangle\ \langle 4,2\rangle\ \langle 3,3\rangle\ \langle 5,4\rangle\ \langle 4,5\rangle\ \$$$

$$\mathcal{R} = \{N_1 \rightarrow \langle 3,3\rangle \langle 5,4\rangle \quad N_2 \rightarrow \langle 1,1\rangle \langle 4,2\rangle \quad N_3 \rightarrow \langle 3,1\rangle N_1$$
$$N_4 \rightarrow \langle 6,3\rangle \langle 3,5\rangle \quad N_5 \rightarrow N_2\, N_4 \quad N_6 \rightarrow N_3 \langle 2,5\rangle$$
$$N_7 \rightarrow N_2\, N_1 \quad N_8 \rightarrow \langle 4,1\rangle N_4 \quad N_9 \rightarrow N_7 \langle 4,5\rangle \}$$
$$C = N_5 \$ N_6 \$ N_7 \$ N_8 \$ N_3 \$ N_9 \$$$

Considering only the first line of S (smaller example)

$$S = \langle 1,1\rangle\ \langle 4,2\rangle\ \langle 6,3\rangle\ \langle 3,5\rangle\ \$$$
$$\mathcal{R} = \{ N_2 \rightarrow \langle 1,1\rangle \langle 4,2\rangle$$
$$N_4 \rightarrow \langle 6,3\rangle \langle 3,5\rangle$$
$$N_5 \rightarrow N_2\, N_4 \}$$
$$C = N_5 \$$$

Compressed Representations of $\mathcal{R}$ and $\mathcal{C}$.

	C	R	V
re_32	32 bit	32 bit	64 bit
re_iv	packed arrays	packed arrays	64 bit
re_ans	ANS (entropy coder)	packed arrays	64 bit

$\Rightarrow$ different time-space trade-offs

Data Parallelism

1.2	3.4	5.6	0	2.3
2.3	0	2.3	4.5	1.7
1.2	3.4	2.3	4.5	0
3.4	0	5.6	0	2.3
2.3	0	2.3	4.5	0
1.2	3.4	2.3	4.5	3.4

$$b = 3$$

We finally take advantage of multicore architectures with a **data-parallel version of our code**. We divide each matrix into b row blocks

$\Rightarrow y = Mx$ and $x^t = y^t M$ consist of b independent multiplications over a single block.

Experiments

Data Sets

matrix	rows	cols	*nonzeros*	#|*nonzeros*|
Susy	5 000 000	18	98.82%	20 352 142
Higgs	11 000 000	28	92.11%	8 083 943
Airline78	14 462 943	29	72.66%	7 794
Covtype	581 012	54	22.00%	6 682
Census	2 458 285	68	43.03%	45
Optical	325 834	174	97.50%	897 176
Mnist2m	2 000 000	784	25.25%	255
ImageNet	1 262 102	900	30.99%	824

Compression & Multiplications

As for the **compression ratios**

- superior to `gzip`
- within 20% of `xz`
- support for matrix-to-vector multiplications directly over the compressed file (`gzip` and `xz` cannot).

As for the **matrix-multiplication algorithms**, the peak memory usage for our multi-threaded algorithms is for most inputs between 6% and 50% of the size of the uncompressed matrix

CLA

We compare our matrix compressor to the one of Compressed Linear Algebra (CLA) system (state-of-the-art).

- As for **compression**, our approach is up to 10% more effective than CLA over 7 (out of 8) data sets
- The space improvement is even greater if we consider **the peak memory usage** at run time, being a factor between 3.14 and 19.12.
- **CLA is always at least two times slower** than our compressors, although we use just 16 threads while CLA uses all the available ones (80).

Performance Comparison

Table 5: Performance comparison considering compressed space, peak memory usage (PM), and average running time in seconds for matrix-vector multiplication; see text for details. Sizes and PMs are expressed as percentages.

	re_iv 16 threads			re_ans 16 threads			CLA multithread			gzip 16 threads			uncompressed 16 threads	
matrix	size (%)	PM (%)	time (s)	size (%)	PM (%)	time (s)	size (%)	PM (%)	time (s)	size (%)	PM (%)	time (s)	PM (%)	time (s)
Susy	68.99	77.53	0.35	65.99	82.77	0.45	76.14	—	—	53.27	63.09	2.22	106.14	0.17
Higgs	41.63	46.68	0.58	37.44	44.63	0.71	32.74	146.68	2.09	48.38	54.56	5.48	103.74	0.51
Airl.	9.35	16.06	0.17	8.13	16.43	0.23	12.34	120.27	1.17	13.27	17.53	6.27	103.57	0.75
Covt.	4.78	16.25	0.01	4.17	16.11	0.01	4.55	70.15	0.05	6.25	10.26	0.41	103.51	0.03
Census	2.00	5.70	0.01	1.55	7.25	0.02	3.77	108.96	0.16	5.54	7.92	1.89	101.77	0.12
Optical	36.05	44.50	0.06	34.93	56.39	0.09	40.44	176.90	0.20	53.55	57.26	1.00	101.47	0.04
Mn.2m	6.24	8.19	0.64	5.88	8.30	0.82	6.22	47.09	1.98	6.46	6.76	24.96	100.16	0.57
Im.Net	4.70	6.59	0.48	4.28	6.59	0.48	6.67	56.80	0.97	5.52	5.89	10.91	100.16	0.46

Column Reordering

Column Reordering

Sometimes matrices exhibit **correlations between columns**.

We propose *four* **column-reordering algorithms** hinging on a novel column-similarity score

⇒ further reduction of

- up to 16% in terms of space occupancy on disks
- up to 16% in the peak memory usage during matrix-vector multiplications.

Thank you!

Transparency & Reproducibility. All source files of our algorithms, as well as the scripts to reproduce the experimental results, are available at the repository https://gitlab.com/manzai/mm-repair .
Data sets are available at a public Kaggle repository.

Francesco Tosoni, MSc
PhD Student in Computer Science
L.go B. Pontecorvo 3, 56127 Pisa PI, Italy
Informatica, ed. C, piano 2°
francesco.tosoni@phd.unipi.it