Finding Matrix Multiplication Algorithms with Classical Planning



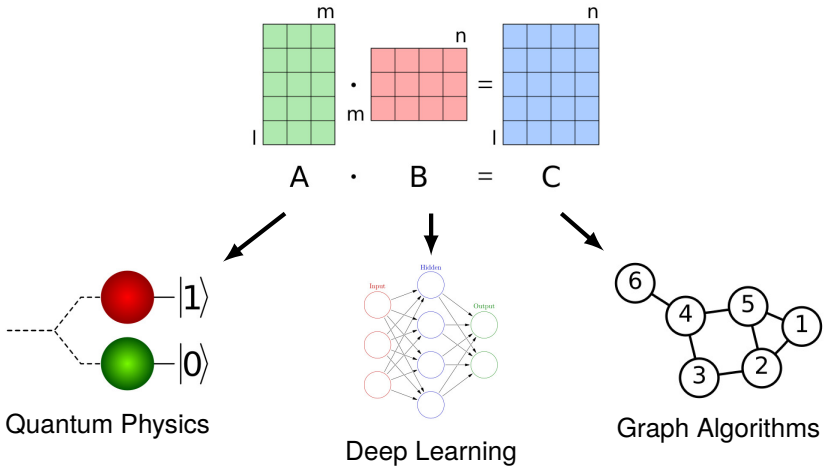

David Speck **Paul Höft** **Daniel Gnad** **Jendrik Seipp**

Linköping University

Representation, Learning and Planning Lab

LINKÖPING UNIVERSITY

Matrix Multiplication – What is it good for?



How to Improve Matrix Multiplication?

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} \end{pmatrix}$$

How to Improve Matrix Multiplication?

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} \end{pmatrix}$$

$\Longrightarrow$ 8 multiplications

How to Improve Matrix Multiplication?

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} \end{pmatrix}$$

$$\Longrightarrow \text{ 8 multiplications}$$

But: Strassen's Algorithm (Strassen 1973) $\Longrightarrow$ 7 multiplications! (-12.5%)

How to Improve Matrix Multiplication?

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} \end{pmatrix}$$

$$\implies \text{8 multiplications}$$

But: Strassen's Algorithm (Strassen 1973) $\implies$ 7 multiplications! (-12.5%)

And: Can compose multiplication algorithms for large matrices.

How to Improve Matrix Multiplication?

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} \end{pmatrix}$$

$$\implies \text{8 multiplications}$$

But: Strassen's Algorithm (Strassen 1973) $\implies$ 7 multiplications! (-12.5%)

And: Can compose multiplication algorithms for large matrices.

Q: How to find better algorithms?

How to Improve Matrix Multiplication?

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} \end{pmatrix}$$

$$\Longrightarrow \text{ 8 multiplications}$$

But: Strassen's Algorithm (Strassen 1973) $\Longrightarrow$ 7 multiplications! (-12.5%)

And: Can compose multiplication algorithms for large matrices.

Q: How to find better algorithms?

Rich space of matrix multiplication algorithms can be formalized as decompositions of a specific 3D-tensor (Strassen 1969)!

How to Improve Matrix Multiplication?

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} \end{pmatrix}$$

$$\implies \text{8 multiplications}$$

But: Strassen's Algorithm (Strassen 1973) $\implies$ 7 multiplications! (-12.5%)

And: Can compose multiplication algorithms for large matrices.

Q: How to find better algorithms?

Rich space of matrix multiplication algorithms can be formalized as decompositions of a specific 3D-tensor (Strassen 1969)!

AlphaTensor: Reinforcement learning agent that discovered new and faster algorithms using MCTS (Fawzi et al. 2022)

How to Improve Matrix Multiplication?

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} \end{pmatrix}$$

$$\Longrightarrow \text{ 8 multiplications}$$

But: Strassen's Algorithm (Strassen 1973) $\Longrightarrow$ 7 multiplications! (-12.5%)

And: Can compose multiplication algorithms for large matrices.

Q: How to find better algorithms?

Rich space of matrix multiplication algorithms can be formalized as decompositions of a specific 3D-tensor (Strassen 1969)!

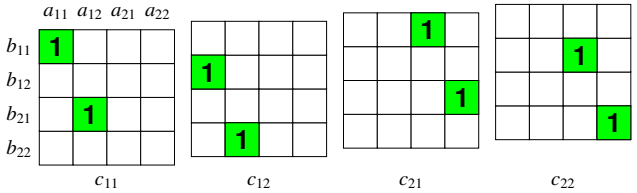
AlphaTensor: Reinforcement learning agent that discovered new and faster algorithms using MCTS (Fawzi et al. 2022) $\Longrightarrow$ Can we model it as a **classical planning task**?

Matrix Multiplication as Classical Planning

Strassen's Algorithm (in $\mathbb{Z}_2$)

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{pmatrix}$$

$$\Downarrow$$



State

$$\mathbf{U} = \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}$$

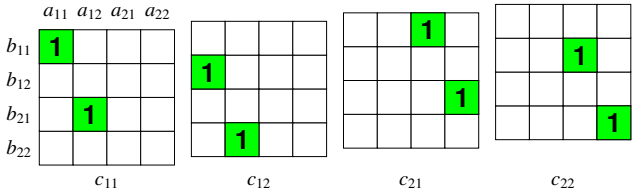
Actions

Matrix Multiplication as Classical Planning

Strassen's Algorithm (in $\mathbb{Z}_2$)

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{pmatrix}$$

$$\Downarrow$$



State

$$\mathbf{U} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

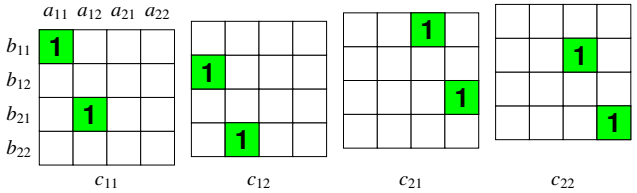
Actions

Matrix Multiplication as Classical Planning

Strassen's Algorithm (in $\mathbb{Z}_2$)

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{pmatrix}$$

$$\mathbf{U} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{V} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$



State

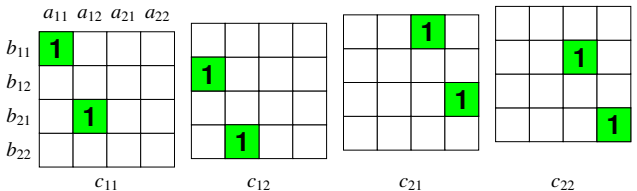
Actions

Matrix Multiplication as Classical Planning

Strassen's Algorithm (in $\mathbb{Z}_2$)

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{pmatrix}$$

$$\mathbf{U} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{V} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$



State

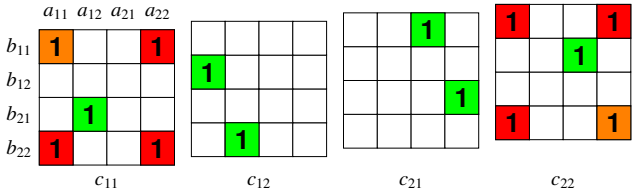
Actions

Matrix Multiplication as Classical Planning

Strassen's Algorithm (in $\mathbb{Z}_2$)

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{pmatrix}$$

$$\Downarrow$$



State

$$\mathbf{U} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{V} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

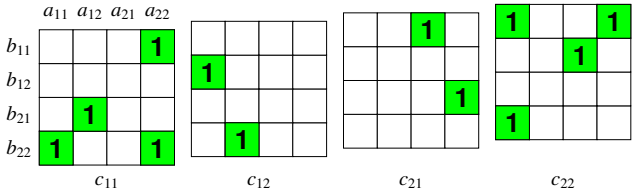
Actions

Matrix Multiplication as Classical Planning

Strassen's Algorithm (in $\mathbb{Z}_2$)

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{pmatrix}$$

$$\Downarrow$$



State

$$\mathbf{U} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

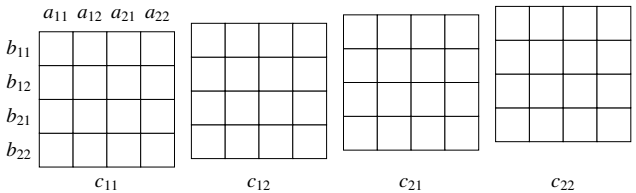
Actions

Matrix Multiplication as Classical Planning

Strassen's Algorithm (in $\mathbb{Z}_2$)

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{pmatrix}$$

$$\Downarrow$$



State

$$\mathbf{U} = \begin{bmatrix} 1\,0\,1\,0\,1\,1\,0 \\ 0\,0\,0\,0\,1\,0\,1 \\ 0\,1\,0\,0\,0\,1\,0 \\ 1\,1\,0\,1\,0\,0\,1 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} 1\,1\,0\,1\,0\,1\,0 \\ 0\,0\,1\,0\,0\,1\,0 \\ 0\,0\,0\,1\,0\,0\,1 \\ 1\,0\,1\,0\,1\,0\,1 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 1\,0\,0\,1\,1\,0\,1 \\ 0\,0\,1\,0\,1\,0\,0 \\ 0\,1\,0\,1\,0\,0\,0 \\ 1\,1\,1\,0\,0\,1\,0 \end{bmatrix}$$

Actions

Experiments

Setup

- 18 instances of MM problems
  - MM sizes: 1×1 to 3×3
  - Up to 134 million operators
- Optimal and satisficing planning
  - 4 planning systems
  - Different search techniques
- 10 hours and 80 GB on a single CPU core
- ${}^*X$: Concrete algorithm with X multiplications

Matrix Sizes	Rank			Satisficing				Optimal			
	L. Bound	U. Bound	Textbook	D–BFWS	LAMA	HC + h^{GC}	Lifted SAT	A* + h^{blind}	A* + h^{PDB}	Lifted SAT	Symbolic
111	1	1	1	*1	*1	*1	*101	*1	*1	*1	*1
⋮				⋮			⋮				⋮
132	6	6	6	*6	*6	*6	*8	4	*6	1	*6
213	6	6	6	*6	*6	*6	*6	4	*6	1	*6
222	7	7	8	*8	*8	*8	–	3	5	1	7
133	7	9	9	*9	*9	*9	–	3	4	1	7
⋮				⋮			⋮				⋮
323	7	15	18	–	*18	*18	–	2	2	–	3
333	19	23	27	–	–	*27	–	2	–	–	–

Experiments

Setup

- 18 instances of MM problems
  - MM sizes: 1×1 to 3×3
  - Up to 134 million operators
- Optimal and satisficing planning
  - 4 planning systems
  - Different search techniques
- 10 hours and 80 GB on a single CPU core
- ${}^*X$: Concrete algorithm with X multiplications

Matrix Sizes	Rank			Satisficing				Optimal			
	L. Bound	U. Bound	Textbook	D–BFWS	LAMA	HC + h^{GC}	Lifted SAT	A* + h^{blind}	A* + h^{PDB}	Lifted SAT	Symbolic
111	1	1	1	*1	*1	*1	*101	*1	*1	*1	*1
⋮				⋮			⋮				⋮
132	6	6	6	*6	*6	*6	*8	4	*6	1	*6
213	6	6	6	*6	*6	*6	*6	4	*6	1	*6
222	7	7	8	*8	*8	*8	–	3	5	1	7
133	7	9	9	*9	*9	*9	–	3	4	1	7
⋮				⋮			⋮				⋮
323	7	15	18	–	*18	*18	–	2	2	–	3
333	19	23	27	–	–	*27	–	2	–	–	–

Experiments

Setup

- 18 instances of MM problems
  - MM sizes: 1×1 to 3×3
  - Up to 134 million operators
- Optimal and satisficing planning
  - 4 planning systems
  - Different search techniques
- 10 hours and 80 GB on a single CPU core
- ${}^*X$: Concrete algorithm with X multiplications

Matrix Sizes	Rank			Satisficing				Optimal			
	L. Bound	U. Bound	Textbook	D–BFWS	LAMA	HC + h^{GC}	Lifted SAT	A* + h^{blind}	A* + h^{PDB}	Lifted SAT	Symbolic
111	1	1	1	*1	*1	*1	*101	*1	*1	*1	*1
⋮				⋮			⋮				⋮
132	6	6	6	*6	*6	*6	*8	4	*6	1	*6
213	6	6	6	*6	*6	*6	*6	4	*6	1	*6
222	7	7	8	*8	*8	*8	–	3	5	1	7
133	7	9	9	*9	*9	*9	–	3	4	1	7
⋮				⋮			⋮				⋮
323	7	15	18	–	*18	*18	–	2	2	–	3
333	19	23	27	–	–	*27	–	2	–	–	–

Experiments

Setup

- 18 instances of MM problems
  - MM sizes: 1×1 to 3×3
  - Up to 134 million operators
- Optimal and satisficing planning
  - 4 planning systems
  - Different search techniques
- 10 hours and 80 GB on a single CPU core
- ${}^*X$: Concrete algorithm with X multiplications

Matrix Sizes	Rank			Satisficing				Optimal			
	L. Bound	U. Bound	Textbook	D–BFWS	LAMA	HC + h^{GC}	Lifted SAT	A* + h^{blind}	A* + h^{PDB}	Lifted SAT	Symbolic
111	1	1	1	*1	*1	*1	*101	*1	*1	*1	*1
⋮				⋮			⋮				⋮
132	6	6	6	*6	*6	*6	*8	4	*6	1	*6
213	6	6	6	*6	*6	*6	*6	4	*6	1	*6
222	7	7	8	*8	*8	*8	–	3	5	1	7
133	7	9	9	*9	*9	*9	–	3	4	1	7
⋮				⋮			⋮				⋮
323	7	15	18	–	*18	*18	–	2	2	–	3
333	19	23	27	–	–	*27	–	2	–	–	–

Conclusions

- Matrix Multiplication can be modeled as classical planning
- Encoding preserves completeness and optimality
- Off-the-shelf planners can tackle non-trivial instances
  - Interesting & challenging domain
- Confirmation of known bounds
- **Future Work:** Domain-specific pruning functions / heuristics

LINKÖPING UNIVERSITY

Matrix Multiplication as Tensor Decomposition

Strassen's Algorithm (in $\mathbb{Z}_2$)

$$\mathbf{U} = \begin{bmatrix} 1&0&1&0&1&1&0 \\ 0&0&0&0&1&0&1 \\ 0&1&0&0&0&1&0 \\ 1&1&0&1&0&0&1 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} 1&1&0&1&0&1&0 \\ 0&0&1&0&0&1&0 \\ 0&0&0&1&0&0&1 \\ 1&0&1&0&1&0&1 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 1&0&0&1&1&0&1 \\ 0&0&1&0&1&0&0 \\ 0&1&0&1&0&0&0 \\ 1&1&1&0&0&1&0 \end{bmatrix}$$

$$m_1 = (a_{11} + a_{22})$$
$$m_2 = (a_{21} + a_{22})\, b_{11}$$
$$m_3 = a_{11}(b_{12} + b_{22})$$
$$m_4 = a_{22}(b_{21} + b_{11})$$
$$m_5 = (a_{11} + a_{12})b_{22}$$
$$m_6 = (a_{21} + a_{11})(b_{11} + b_{12})$$
$$m_7 = (a_{12} + a_{22})(b_{21} + b_{22})$$

$$c_{11} = m_1 + m_4 + m_5 + m_7$$
$$c_{12} = m_3 + m_5$$
$$c_{21} = m_2 + m_4$$
$$c_{22} = m_1 + m_2 + m_3 + m_6$$

Matrix Multiplication as Tensor Decomposition

Strassen's Algorithm (in $\mathbb{Z}_2$)

$$\mathbf{U} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\begin{aligned}
m_1 &= (a_{11} + a_{22})\,(b_{11} + b_{22}) \\
m_2 &= (a_{21} + a_{22})\,b_{11} \\
m_3 &= a_{11}(b_{12} + b_{22}) \\
m_4 &= a_{22}(b_{21} + b_{11}) \\
m_5 &= (a_{11} + a_{12})b_{22} \\
m_6 &= (a_{21} + a_{11})(b_{11} + b_{12}) \\
m_7 &= (a_{12} + a_{22})(b_{21} + b_{22}) \\
c_{11} &= m_1 + m_4 + m_5 + m_7 \\
c_{12} &= m_3 + m_5 \\
c_{21} &= m_2 + m_4 \\
c_{22} &= m_1 + m_2 + m_3 + m_6
\end{aligned}$$

Matrix Multiplication as Tensor Decomposition

Strassen's Algorithm (in $\mathbb{Z}_2$)

$$\mathbf{U} = \begin{bmatrix} 1&0&1&0&1&1&0 \\ 0&0&0&0&1&0&1 \\ 0&1&0&0&0&1&0 \\ 1&1&0&1&0&0&1 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} 1&1&0&1&0&1&0 \\ 0&0&1&0&0&1&0 \\ 0&0&0&1&0&0&1 \\ 1&0&1&0&1&0&1 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 1&0&0&1&1&0&1 \\ 0&0&1&0&1&0&0 \\ 0&1&0&1&0&0&0 \\ 1&1&1&0&0&1&0 \end{bmatrix}$$

$$m_1 = (a_{11} + a_{22})\,(b_{11} + b_{22})$$
$$m_2 = (a_{21} + a_{22})\,b_{11}$$
$$m_3 = a_{11}(b_{12} + b_{22})$$
$$m_4 = a_{22}(b_{21} + b_{11})$$
$$m_5 = (a_{11} + a_{12})b_{22}$$
$$m_6 = (a_{21} + a_{11})(b_{11} + b_{12})$$
$$m_7 = (a_{12} + a_{22})(b_{21} + b_{22})$$

$$c_{11} = m_1 + m_4 + m_5 + m_7$$
$$c_{12} = m_3 + m_5$$
$$c_{21} = m_2 + m_4$$
$$c_{22} = m_1 + m_2 + m_3 + m_6$$

References I

- Fawzi, Alhussein, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis und Pushmeet Kohli (2022). “Discovering faster matrix multiplication algorithms with reinforcement learning”. In: *Nature* 610.7930, S. 47–53.
- Strassen, Volker (1969). “Gaussian elimination is not optimal”. In: *Numerische Mathematik* 13.4, S. 354–356.
- Strassen, Volker (1973). “Vermeidung von Divisionen.”. ger. In: *Journal für die reine und angewandte Mathematik* 264, S. 184–202. URL: `http://eudml.org/doc/151394`.