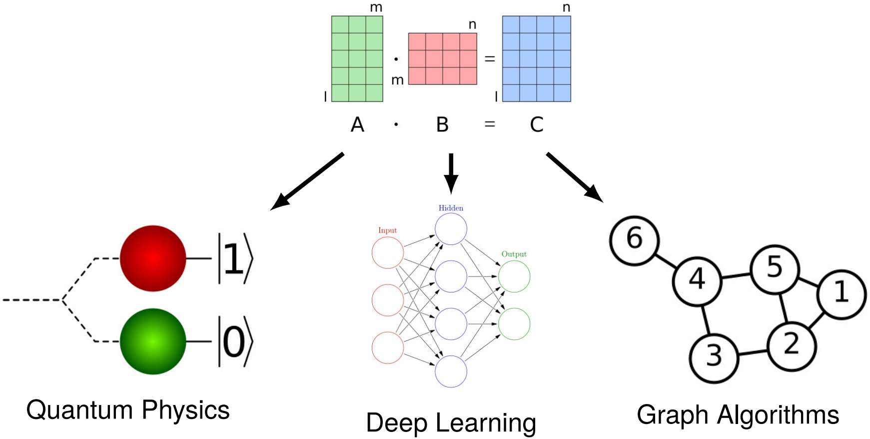
Finding Matrix Multiplication Algorithms with Classical Planning

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

Linköping University, RLPLab

Matrix Multiplication – What is it good for?



How to Improve Matrix Multiplication?

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

LINKÖPING UNIVERSITY

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?

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

$\Longrightarrow$ 8 multiplications

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

LINKÖPING UNIVERSITY

How to Improve Matrix Multiplication?

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

$\implies$ 8 multiplications

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

And: can compose multiplication algorithms for large matrices.

Strassen's Algorithm

$$m_1 = (a_{11} + a_{22})$$

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

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

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

Strassen's Algorithm

$$m_1 = (a_{11} + a_{22})(b_{11} + b_{22})$$

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

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

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

Strassen's Algorithm

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

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

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

$$m_1 = (a_{11} + a_{22})(b_{11} + b_{22})$$

$$c_{11} = m_1$$

$$c_{22} = m_1$$

LINKÖPING UNIVERSITY

Strassen's Algorithm

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

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

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

$$m_1 = (a_{11} + a_{22})\,(b_{11} + b_{22})$$
$$m_2 = (a_{21} + a_{22})\,b_{11}$$

$$c_{11} = m_1$$

$$c_{21} = m_2$$
$$c_{22} = m_1 + m_2$$

Strassen's Algorithm

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

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

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

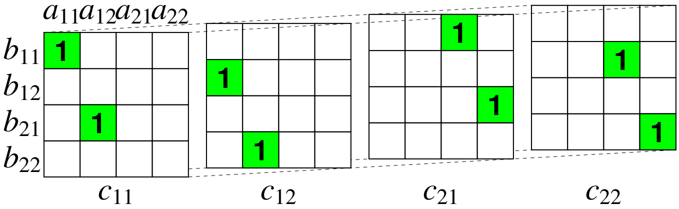
$$m_1 = (a_{11} + a_{22})\ (b_{11} + b_{22})$$
$$m_2 = (a_{21} + a_{22})\ b_{11}$$
$$m_3 = a_{11}(b_{12} + b_{22})$$
$$m_4 = a_{22}(b_{21} + b_{11})$$
$$m_5 = (a_{11} + a_{12})b_{22}$$
$$m_6 = (a_{21} + a_{11})(b_{11} + b_{12})$$
$$m_7 = (a_{12} + a_{22})(b_{21} + b_{22})$$

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

Matrix Multiplication as Tensor Operations

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

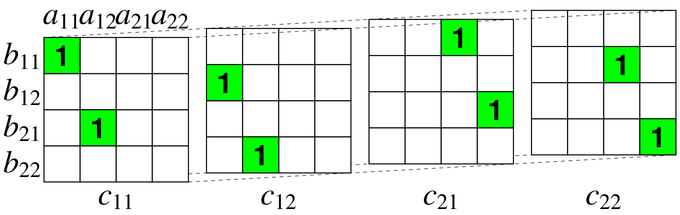
$$\Downarrow$$



Matrix Multiplication as Tensor Operations

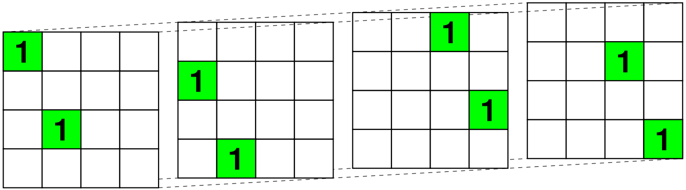
$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{pmatrix}$$

$$\Downarrow$$

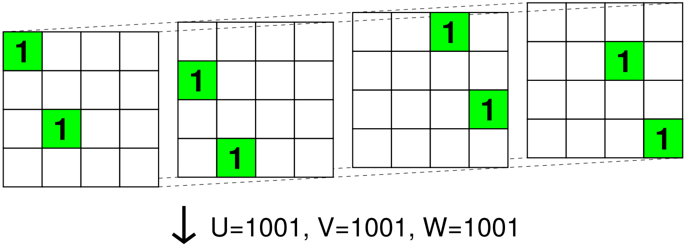


For example: $c_{1,1} = a_{1,1} b_{1,1} + a_{1,2} b_{2,1}$

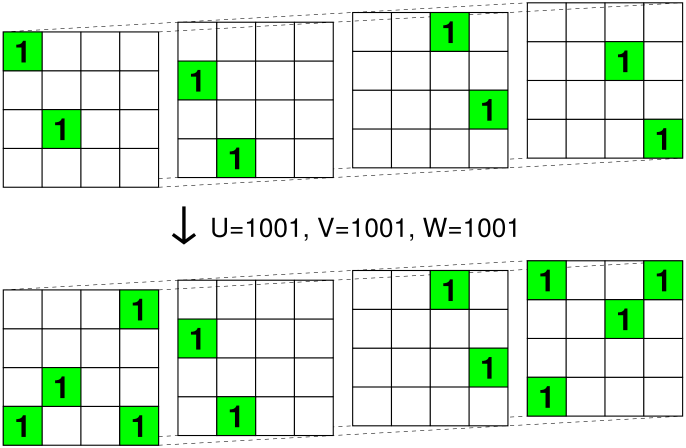
Strassen #1 as Tensor Operation



Strassen #1 as Tensor Operation



Strassen #1 as Tensor Operation

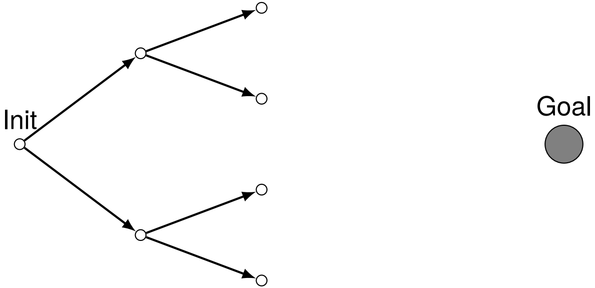


Planning in a Nutshell

Set of variables describe world states; actions specify dynamics via preconditions + effects.

Planning in a Nutshell

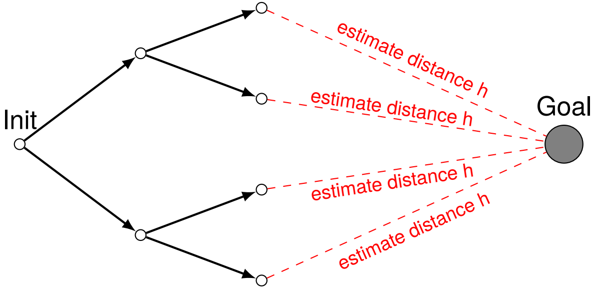
Set of variables describe world states; actions specify dynamics via preconditions + effects.



$\rightarrow$ State-space search

Planning in a Nutshell

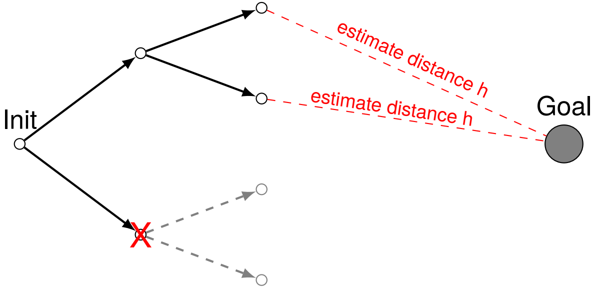
Set of variables describe world states; actions specify dynamics via preconditions + effects.



→ State-space search with domain-independent (1) **heuristics**

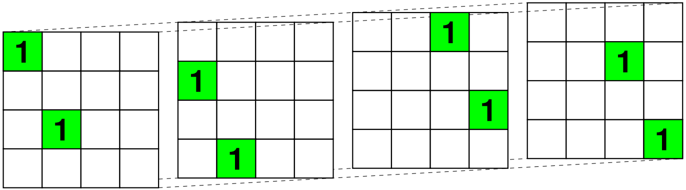
Planning in a Nutshell

Set of variables describe world states; actions specify dynamics via preconditions + effects.



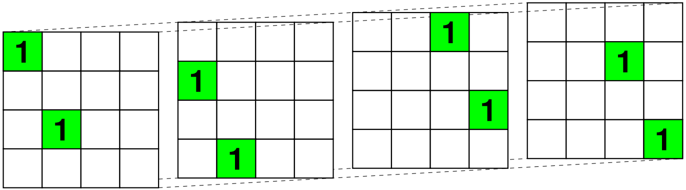
→ State-space search with domain-independent (1) **heuristics** or (2) **pruning methods**.

Matrix Multiplication as Planning



LINKÖPING UNIVERSITY

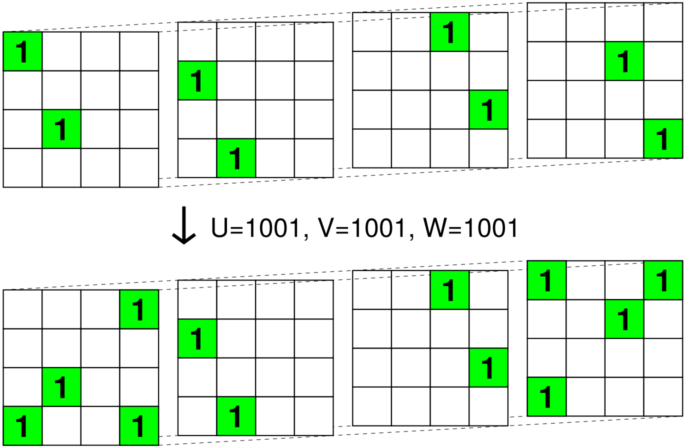
Matrix Multiplication as Planning



↓ U=1001, V=1001, W=1001

LINKÖPING UNIVERSITY

Matrix Multiplication as Planning



Experiments

Setup

- 18 instances of MM problems
  - MM sizes: 1×1 to 3×3
  - Up to 134 million operators
- Optimal and satisfying planning
  - 4 planning systems
  - Different search techniques
- 10 hours and 80 GB on a single CPU core

Matrix Sizes	Rank: L. Bound	Rank: U. Bound	Rank: Textbook
1 1 1	1	1	1
⋮			
1 3 2	6	6	6
2 1 3	6	6	6
2 2 2	7	7	8
1 3 3	7	9	9
⋮			
3 2 3	7	15	18
3 3 3	19	23	27

LINKÖPING UNIVERSITY

Experiments

Setup

- 18 instances of MM problems
  - MM sizes: 1×1 to 3×3
  - Up to 134 million operators
- Optimal and satisfying planning
  - 4 planning systems
  - Different search techniques
- 10 hours and 80 GB on a single CPU core

Matrix Sizes	Rank: L. Bound	Rank: U. Bound	Rank: Textbook
1 1 1	1	1	1
⋮			
1 3 2	6	6	6
2 1 3	6	6	6
2 2 2	7	7	8
1 3 3	7	9	9
⋮			
3 2 3	7	15	18
3 3 3	19	23	27

Experiments

Setup

- 18 instances of MM problems
  - MM sizes: 1×1 to 3×3
  - Up to 134 million operators
- Optimal and satisfying planning
  - 4 planning systems
  - Different search techniques
- 10 hours and 80 GB on a single CPU core

Matrix Sizes	Rank: L. Bound	Rank: U. Bound	Rank: Textbook
1 1 1	1	1	1
⋮			
1 3 2	6	6	6
2 1 3	6	6	6
2 2 2	7	7	8
1 3 3	7	9	9
⋮			
3 2 3	7	15	18
3 3 3	19	23	27

Experiments

Setup

- 18 instances of MM problems
  - MM sizes: 1×1 to 3×3
  - Up to 134 million operators
- Optimal and satisfying planning
  - 4 planning systems
  - Different search techniques
- 10 hours and 80 GB on a single CPU core

Matrix Sizes	Rank			Satisficing			
	L. Bound	U. Bound	Textbook	D–BFWS	LAMA	HC + h^{GC}	Lifted SAT
1 1 1	1	1	1	*1	*1	*1	*101
⋮				⋮			⋮
1 3 2	6	6	6	*6	*6	*6	*8
2 1 3	6	6	6	*6	*6	*6	*6
2 2 2	7	7	8	*8	*8	*8	–
1 3 3	7	9	9	*9	*9	*9	–
⋮				⋮			⋮
3 2 3	7	15	18	–	*18	*18	–
3 3 3	19	23	27	–	–	*27	–

LINKÖPING UNIVERSITY

Experiments

Setup

- 18 instances of MM problems
  - MM sizes: 1×1 to 3×3
  - Up to 134 million operators
- Optimal and satisfying planning
  - 4 planning systems
  - Different search techniques
- 10 hours and 80 GB on a single CPU core

Matrix Sizes	Rank: L. Bound	Rank: U. Bound	Rank: Textbook	Satisficing: D–BFWS	Satisficing: LAMA	Satisficing: HC + h^{GC}	Satisficing: Lifted SAT	Optimal: $A^* + h^{blind}$	Optimal: $A^* + h^{PDB}$	Optimal: Lifted SAT	Optimal: 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	–	–	–

LINKÖPING UNIVERSITY

History / Related Work

- Strassen 1969: Multiplication of 2x2 matrices with 7 multiplications.
- Laderman 1976: Show schemes with 23 multiplications for 3x3 matrices.
- Håstad 1989: Minimizing the number of multiplications is **NP**-complete.
- Heule et al. 2019: Matrix multiplication as satisfiability, new 23-schemes for 3x3 matrices.
- Fawzi et al. 2022: Matrix multiplication as single-player game tackled using reinforcement learning, several new bounds up to size 5x5 (98 $\rightarrow$ 96).
- Deza et al. 2023: Matrix multiplication as constraint programming.

LINKÖPING UNIVERSITY

Conclusions

- Matrix Multiplication can be modeled as classical planning (or CSP, SAT, ...)
- Encoding preserves completeness and optimality
- Off-the-shelf planners can tackle non-trivial instances
- Confirmation of known bounds

Conclusions

- Matrix Multiplication can be modeled as classical planning (or CSP, SAT, ...)
- Encoding preserves completeness and optimality
- Off-the-shelf planners can tackle non-trivial instances
- Confirmation of known bounds
- **Future Work:** Domain-specific pruning functions / heuristics

Thank you!

Questions?

LINKÖPING UNIVERSITY

References I

- Deza, Arnaud, Chang Liu, Pashootan Vaezipoor und Elias B. Khalil (2023). *Fast Matrix Multiplication Without Tears: A Constraint Programming Approach*. arXiv: `2306.01097 [cs.AI]`.
- 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.
- Håstad, Johan (1989). “Tensor rank is NP-complete”. In: *International Colloquium on Automata, Languages, and Programming*. Springer, S. 451–460.
- Heule, Marijn J. H., Manuel Kauers und Martina Seidl (2019). “Local Search for Fast Matrix Multiplication”. In: *Proc. SAT 2019*, S. 155–163.
- Laderman, Julian David (1976). “On Algorithms for Minimizing the Number of Multiplications in Matrix Products”. Diss. New York University, USA.
- 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`.