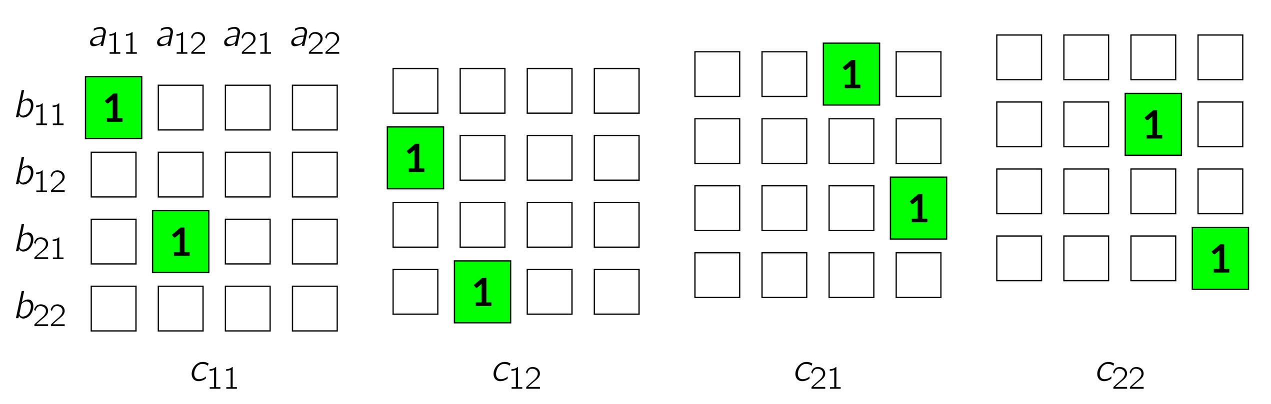
Finding Matrix Multiplication Algorithms with Planning

David Speck, Paul Höft, Daniel Gnad, Jendrik Seipp
Linköping University

Matrix Multiplication in Tensor Space

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

$$\downarrow$$



$$c_{11} = a_{11}b_{11} + a_{12}b_{21}$$

$$\dots$$

Classical Planning in a Nutshell

Finite-domain variables describe world states; a set of discrete actions specifies the world dynamics. Which actions to apply to get from initial state to goal?

Matrix Multiplication as Classical Planning

MM problem: $\mathbf{C} = \mathbf{AB}$, two matrices with sizes $m \times n$ and $n \times p$.
Planning Problem: $\Pi_{m,n,p} = (\mathcal{V}, \mathcal{I}, \mathcal{G}, \mathcal{O})$
Each variable v_i corresponds to an entry of a 3-D tensor:

$$\mathcal{V} = \{v_i \mid 1 \leq i \leq m^2 \cdot n^2 \cdot p^2\}$$

$$\mathcal{I}(v[a_{ij}, b_{jk}, c_{ik}]) = 1 \text{ for all } i \in [m], j \in [n], \text{ and } k \in [p]; \text{ else } \mathcal{I}(v) = 0$$

$$\mathcal{G}(v) = 0 \text{ for all } v \in \mathcal{V}$$

Operator for all binary vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ that update the current tensor state.

$$\mathcal{O} = \{o_i \mid 1 \leq i \leq 2^{m \cdot n + n \cdot p + m \cdot p}\}$$

The operators update a state s, i.e., the tensor T_s, in such a way that the successor state $s' = s[\![o]\!]$ represents the tensor $T_{s'} = T_s - \mathbf{u}(o) \otimes \mathbf{v}(o) \otimes \mathbf{w}(o)$.

Applications

Weather Simulations, Quantum Physics, Computer Graphics, Deep Learning, …

On Solving Hard Problems Without Machine Learning: a Case for Traditional Reasoners.

Full paper.

Strassen's Algorithm

$$\mathbf{U} = \begin{bmatrix} 1010110 \\ 0000101 \\ 0100010 \\ 1101001 \end{bmatrix} \quad \mathbf{V} = \begin{bmatrix} 1101010 \\ 0010010 \\ 0001001 \\ 1010101 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} 1001101 \\ 0010100 \\ 0101000 \\ 1110010 \end{bmatrix}$$

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

Experimental Results

	Satisficing				Optimal				Rank	
Matrix Sizes	D–BFWS	LAMA	HC + h^{GC}	Lifted SAT	$A^* + h^{blind}$	$A^* + h^{PDB}$	Lifted SAT	Symbolic	Bounds	Textbook
1 1 1	*1	*1	*1	*101	*1	*1	*1	*1	1	1
1 1 2	*2	*2	*2	*21	*2	*2	*2	*2	2	2
1 2 1	*2	*2	*2	*21	*2	*2	*2	*2	2	2
1 1 3	*3	*3	*3	*12	*3	*3	*3	*3	3	3
1 3 1	*3	*3	*3	*12	*3	*3	*3	*3	3	3
1 2 2	*4	*4	*4	*12	*4	*4	*4	*4	4	4
2 1 2	*4	*4	*4	*9	*4	*4	*4	*4	4	4
1 2 3	*6	*6	*6	*6	4	*6	1	*6	6	6
1 3 2	*6	*6	*6	*8	4	*6	1	*6	6	6
2 1 3	*6	*6	*6	*6	4	*6	1	*6	6	6
2 2 2	*8	*8	*8	–	3	5	1	7	7	8
1 3 3	*9	*9	*9	–	3	4	1	7	7/9	9
3 1 3	*9	*9	*9	–	3	4	1	7	7/9	9
2 2 3	*12	*12	*12	–	3	3	1	3	7/11	12
2 3 2	*12	*12	*12	–	3	4	1	3	7/11	12
2 3 3	–	*18	*18	–	2	2	–	3	7/15	18
3 2 3	–	*18	*18	–	2	2	–	3	7/15	18
3 3 3	–	–	*27	–	2	–	–	–	19/23	27

$^*X$: Concrete algorithm with X multiplications.