Finding Matrix Multiplication Algorithms with Classical Planning Extended Abstract

David Speck, Paul Höft, Daniel Gnad, and Jendrik Seipp
Department of Computer and Information Science, Linköping University
{david.speck, paul.hoft, daniel.gnad, jendrik.seipp}@liu.se

***Abstract*— Matrix multiplication is a fundamental operation of linear algebra, with applications ranging from quantum physics to artificial intelligence. Given its importance, enormous resources have been invested in the search for faster matrix multiplication algorithms. Recently, this search has been cast as a single-player game. By learning how to play this game efficiently, the newly-introduced AlphaTensor reinforcement learning agent discovers many new faster algorithms. In this paper, we show that finding matrix multiplication algorithms can also be cast as a classical planning problem. Based on this observation, we introduce a challenging benchmark suite for classical planning and evaluate state-of-the-art planning techniques on it. We analyze the strengths and limitations of different planning approaches in this domain and show that we can use classical planning to find lower bounds and concrete algorithms for matrix multiplication.**

I. Introduction

In the age of big data and deep learning the demand for efficient computation is higher than ever. One particularly important operation that is crucial for many applications is the multiplication of matrices. Many fields in industry and research depend on matrix multiplication, ranging from weather simulations, via quantum physics, to computer graphics and machine learning. Given the ubiquity of matrix multiplication (MM), great effort has been spent on deriving more efficient algorithms [1], [2], [3], [4], [5]. In this context, *more efficient* means that an algorithm uses fewer multiplications, which are the critical operations. Even a minor reduction of these operations for the multiplication of small matrices will result in huge savings of compute time and energy. This is because (1) the multiplication of large matrices can be composed of algorithms for smaller ones, and (2) given the sheer number of multiplications needed for example to train a neural network, boosting this basic operation can tremendously speed up the overall process.

Finding more efficient MM algorithms is extremely challenging, since minimizing the number of multiplications is an NP-complete problem [6]. Recently, [5] presented a novel approach to MM algorithm discovery based on casting the problem as a single-player game. The action space of this game is vast, with over 2^{27} actions for two 3×3 matrices, which is orders of magnitude larger than that of the game Go with hundreds of actions. By training a reinforcement learning agent, AlphaTensor, they established faster algorithms for several matrix sizes.

In this work, we propose to model the search for MM algorithms as a classical planning problem. Similarly to previous work on MM discovery [7], [5], we restrict the solution space to MM over modular arithmetic in $\mathbb{Z}_2$, so shorter algorithms could possibly exist for $\mathbb{Z}$ that cannot be discovered using our encoding. We show the correctness of our model and evaluate a variety of planning techniques on the MM discovery problem. In contrast to [5], today's off-the-shelf planners fail to uncover *novel* algorithms (with a tiny fraction of [5]'s compute power). However, already with the current (non-tailored) state of the art in planning, we can derive non-trivial lower bounds and find MM algorithms that are structurally different from the textbook algorithm.

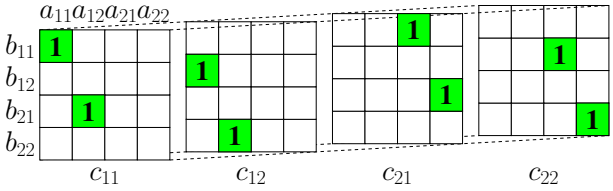

Fig. 1. Initial tensor $\mathcal{T}_{2,2,2}$ of a matrix multiplication task $\mathbf{C} = \mathbf{AB}$ of two 2×2 matrices. The highlighted cells indicate which entries to read from the input matrices $\mathbf{A}$ and $\mathbf{B}$ and where to write the result into the $\mathbf{C}$ matrix. For example, $c_{11} = a_{11}b_{11} + a_{12}b_{21}$.

For full details, we refer to our main paper [8].

II. Matrix Multiplication as Classical Planning

Similar to [5], the underlying idea of our encoding is that a state represents a tensor $\mathcal{T}_{m,n,p}$ and actions describe how to update that tensor by selecting the bits that are swapped. We can then specify a planning problem $\Pi_{m,n,p}$ that describes the search for MM algorithms of two matrices with sizes $m \times n$ and $n \times p$, respectively. Such a problem can be described by a tensor $\mathcal{T}_{m,n,p}$ with $|\mathcal{T}_{m,n,p}| = m^2 \cdot n^2 \cdot p^2$ entries. Hence, we use $|\mathcal{T}_{m,n,p}|$ propositional state variables $\mathcal{V}$ in the planning model to represent every possible assignment of zeros and ones to the tensor. The initial state describes which entries to read from the input matrices and where to write the result (cf. Fig. 1). This is specified by true state variables. In the unique goal state, all variables are false, which means that exactly the required calculations have been performed. To encode the algorithm steps, we define an operator o for each possible choice of the three vectors $\mathbf{u} = \mathbf{u}(o)$, $\mathbf{v} = \mathbf{v}(o)$, and $\mathbf{w} = \mathbf{w}(o)$, of which $\mathcal{T}_{m,n,p}$ is composed. The sizes of $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ are $m \cdot n$, $n \cdot p$, and $m \cdot p$ respectively, which is reflected in the number of operators. The operators update the current state s, i.e., the tensor T_s, in such a way that the successor state s' represents the tensor $T_{s'} = T_s - \mathbf{u}(o) \otimes \mathbf{v}(o) \otimes \mathbf{w}(o)$.

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

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

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

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

Fig. 2. Decomposition (left) for a matrix multiplication problem $\mathcal{T}_{2,2,2}$ of two 2×2 matrices and the corresponding $\mathbb{Z}_2$-Strassen algorithm (right).

Consider the MM task $\mathcal{T}_{2,2,2}$. The planning problem $\Pi_{2,2,2}$ modeling this task has 64 binary variables, where the initial state represents the tensor shown in Figure 1. A possible plan $\pi = \langle o_1, \ldots, o_7 \rangle$ for $\Pi_{2,2,2}$ is represented by the matrices of Figure 2 (left). Each operator o_i corresponds to the vectors $\mathbf{u} = \mathbf{u}(o)$, $\mathbf{v} = \mathbf{v}(o)$, and $\mathbf{w} = \mathbf{w}(o)$ of the i-th column of the matrix $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$. The shown plan is optimal and describes Strassen's algorithm, illustrated in Figure 2 (right).

Similar to the TensorGame described by [5], which expresses a reinforcement learning problem, the planning problem models the decomposition of a tensor in a sequential manner. Every solution to one of our planning problems describes a concrete matrix multiplication algorithm.

III. Experiments

We evaluated a variety of planning approaches on the MM benchmark suite to investigate which techniques perform well on the challenging problem of finding algorithms for MM. This includes the dual best-first width search planner (Dual-BFWS) from the International Planning Competition 2018 [9], the lifted SAT-based planner LiSAT [10], the symbolic search planner Symk [11], and the heuristic search planner Scorpion [12]. For each MM planning task, we ran the considered planning algorithms on a single CPU core with a time limit of 10 hours and a memory limit of 80 GB. All our source code and data are available online [13].

The results for different matrix sizes are shown in Figure 3. We distinguish between satisficing planners (which find any solution) and optimal planners (which guarantee shortest plans). The satisficing planners are capable of finding MM algorithms up to size $\Pi_{3,3,3}$, but none that is shorter than the textbook algorithm. The optimal planners can prove non-trivial lower bounds for many instances. Most remarkably is the symbolic search planner Symk, which proves that indeed no algorithm exists for $\Pi_{2,2,2}$ that is better than Strassen's.

IV. Conclusions

We showed that finding algorithms for matrix multiplication can be modeled as a classical planning problem in a sound and complete way. Our results show that we can use planning to find concrete algorithms and prove non-trivial bounds. This is remarkable considering the small resources of ten hours we used, compared to several years of CPU and GPU time of other works [4], [5]. We believe that this work lays a new foundation for research in this domain and hope that it will encourage researchers from the planning community to join us in this quest.

	Satisficing				Optimal				Rank	
Matrix Sizes	D–BFWS	LAMA	HC + h^{GC}	Lifted SAT	A* + h^{blind}	A* + h^{PDB}	Lifted SAT	Symbolic	Bounds	Textbook
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

Fig. 3. Solution lengths of MM tasks found by various planning algorithms. Entries with asterisks indicate the length of a found plan describing an algorithm, all other entries show proved lower bounds. With "–" we indicate that no solution or lower bound could be found. The two right columns serve as a reference for the rank of the decomposition problem with the lower and upper bounds, and the Textbook algorithm.

References

[1] V. Strassen, "Vermeidung von divisionen." *Journal für die reine und angewandte Mathematik*, vol. 264, pp. 184–202, 1973. [Online]. Available: http://eudml.org/doc/151394

[2] J. D. Laderman, "On algorithms for minimizing the number of multiplications in matrix products," Ph.D. dissertation, New York University, USA, 1976.

[3] M. Bläser, "On the complexity of the multiplication of matrices of small formats," *Journal of Complexity*, vol. 19, no. 1, pp. 43–60, 2003.

[4] M. J. H. Heule, M. Kauers, and M. Seidl, "Local search for fast matrix multiplication," in *Proc. SAT 2019*, 2019, pp. 155–163.

[5] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. J. R. Ruiz, J. Schrittwieser, G. Swirszcz, D. Silver, D. Hassabis, and P. Kohli, "Discovering faster matrix multiplication algorithms with reinforcement learning," *Nature*, vol. 610, no. 7930, pp. 47–53, 2022.

[6] J. Håstad, "Tensor rank is NP-complete," in *International Colloquium on Automata, Languages, and Programming*. Springer, 1989, pp. 451–460.

[7] M. J. Heule, M. Kauers, and M. Seidl, "New ways to multiply 3×3-matrices," *Journal of Symbolic Computation*, vol. 104, pp. 899–916, 2021.

[8] D. Speck, P. Höft, D. Gnad, and J. Seipp, "Finding matrix multiplication algorithms with classical planning," in *Proc. ICAPS 2023*, 2023.

[9] G. Francès, H. Geffner, N. Lipovetzky, and M. Ramiréz, "Best-first width search in the IPC 2018: Complete, simulated, and polynomial variants," in *IPC-9 Planner Abstracts*, 2018, pp. 23–27.

[10] D. Höller and G. Behnke, "Encoding lifted classical planning in propositional logic," in *Proc. ICAPS 2022*, 2022, pp. 134–144.

[11] D. Speck, R. Mattmüller, and B. Nebel, "Symbolic top-k planning," in *Proc. AAAI 2020*, 2020, pp. 9967–9974.

[12] J. Seipp, T. Keller, and M. Helmert, "Saturated cost partitioning for optimal classical planning," *JAIR*, vol. 67, pp. 129–167, 2020.

[13] D. Speck, P. Höft, D. Gnad, and J. Seipp, "Code and data for the ICAPS 2023 paper "Finding Matrix Multiplication Algorithms with Classical Planning"," https://doi.org/10.5281/zenodo.7731696, 2023.