arXiv:2603.02398v1 [cs.SC] 2 Mar 2026

小规模格式快速矩阵乘法：借助开源翻转图框架发现新方案

预印本

Andrew I.PerminovTAI研
究中心 系统编程研究所 莫斯科
perminov@ispras.ru

2026年3月4日

摘要

本文介绍一个开源 C++ 框架，用于通过翻转图方法发现快速矩阵乘法方案。该框架支持多种系数环——二元（$\mathbb{Z}_2$）、模三元（$\mathbb{Z}_3$）和整数三元（$\mathbb{Z}_T = \{-1, 0, 1\}$）——并实现了固定维度和元维度搜索算子。借助高效的系数向量位级编码和 OpenMP 并行，该工具能够在商用硬件上进行大规模探索。本研究涵盖了 680 个方案，范围从（$2 \times 2 \times 2$）到（$16 \times 16 \times 16$），其中 276 个方案现已采用 $\mathbb{Z}_T$系数，117 个采用整数系数。使用该框架后，79 个矩阵乘法方案的乘法复杂度（秩）得到改进。值得注意的是，研究发现了一种新的 $4 \times 4 \times 10$ 方案，仅需 115 次乘法即可实现 $\omega \approx 2.80478$，在这一特定规模下优于 Strassen 指数。此外，研究还重新发现了 93 个使用三元系数的方案，这些方案此前仅在有理数或整数范围内已知；以及 68 个使用整数系数的方案，这些方案此前需要使用分数。所有工具和发现的方案均已公开，以支持可复现研究。

关键词 快速矩阵乘法 ·翻转图 · 三元整数系数集 · 张量秩

1 引言

矩阵乘法是科学计算、机器学习及无数其他领域的核心基础运算。虽然朴素算法需要 $O(n^3)$ 次运算，但 Strassen 于 1969 年取得的突破 [1] 表明，2×2 矩阵只需进行 7 次乘法即可相乘，而非 8 次。这一发现确立了更快速算法的存在，并开启了数十年来对这类算法的探索。

对于较小的矩阵格式，例如 3×3 或 4×4，目标是找出所需乘法次数的精确最小值，即所谓的秩。这些小型算法是更大规模计算的构件，也具有直接的实际应用价值。从实现角度看，系数取自 $\{-1, 0, 1\}$（记作 $\mathbb{Z}_T$）非常理想，因为这意味着只需进行简单的加法和减法，无须乘以常数。相比之下，涉及大整数或分数的算法会在硬件中带来显著开销。
已开发出若干自动化方法来发现矩阵乘法算法：SAT求解器 [2, 3],数值方法 [4], 深度强化学习 [5], 约束编程 [6]以及翻转图方法 [7–10], ，该方法将搜索重新表述为对一张图的探索，其中顶点表示有效方案，边对应于局部变换。

本文提出了一个实现翻转图方法的开源C++ 框架。该框架支持多种系数环——$\mathbb{Z}_2$、 $\mathbb{Z}_3$和 $\mathbb{Z}_T$——并采用高效的位级编码以获得高性能。本文做出了三项主要贡献：

- 79 种矩阵乘法方案的秩得到改进，其中包括一种仅需 115 次乘法的新 $4 \times 4 \times 10$ 方案。对于这一特定规模，该方案的指数优于 Strassen 指数 $\omega \approx 2.80478$ 。
- 93 此前仅在有理数域或整数上已知的方案，如今通过三元整数系数重新发现（$\mathbb{Z}_T$）。
- 68此前需要分数的方案，如今通过整数系数重新发现（$\mathbb{Z}$）。

所有工具和发现的方案均以开源形式发布，以促进可复现研究和社区驱动的探索：

- 翻转图框架：https://github.com/dronperminov/ternary_flip_graph；
- 方案与结果：https://github.com/dronperminov/FastMatrixMultiplication。

本文结构如下。第 2 节回顾相关工作。第 3 节介绍矩阵乘法方案和翻转图方法的背景。第 4 节介绍框架的实现。第 5 节阐述构造新方案的技术。第 6 节展示实验结果。第 7 节讨论观察结果和未来方向。第 8 节总结全文。

2 相关工作

关于寻找矩阵乘法最优秩的问题，已有大量文献；相关结果由 Sedoglavic 维护的在线目录持续跟踪 [11]。该目录目前收录了 5426 种矩阵乘法格式，并为每种格式列出其已知的最佳秩，是本文中上界的主要参考依据。该目录侧重于秩而非系数最小化，因此与本文的目标相辅相成。

人们开发了各种自动化方法来解决发现矩阵乘法算法的问题。本节回顾其中的主要方法，重点介绍它们在实际系数环方面的优势与局限性。

2.1 SAT 求解

研究者通过将 Brent方程编码为布尔可满足性问题实例，应用 SAT求解器。对于 (2，2，2 : 7) 这类小型格式，SAT求解器可以在几秒内找到解。然而，对于 (3，3，3 : 23)，穷举式SAT求解会变得难以实现，运行数天仍无法找到解 [2]。

近期研究通过对称破缺技术改进了基于 SAT 的方法，这些技术通过消除冗余解来缩小搜索空间 [3]。实际方案还将 SAT 与局部搜索启发式相结合。研究人员从随机初始化的系数出发，发现了超过 17000 种非等价的 (3，3，3 : 23) 方案 [2, 12]。这种混合方法表明，尽管纯 SAT 求解面临可扩展性限制，但结合 SAT 组件的引导式搜索仍然很有价值。

2.2 数值方法

Smirnov [4]，探索了另一种方法，将寻找矩阵乘法方案的问题表述为数值优化问题。通过将 Brent方程视为双线性方程组，可以使用最小二乘等方法寻找近似解，再将其舍入为精确系数。然而，这种方法找到的解往往需要分数或较大的整数，而将其转换为更实用的系数环并非总是简单。

2.3 强化学习

DeepMind 的 AlphaTensor [5] 证明，深度强化学习能够发现新颖的矩阵乘法算法。它将算法发现建模为一场单人游戏，目标是分解矩阵乘法张量。随后推出的 AlphaEvolve [13] 则利用语言模型进一步推动自动化发现。

然而，这些人工智能发现的算法所采用的系数往往不利于实际实现。大整数、分数或复数频繁出现，导致硬件实现产生显著的计算开销。在发现算法后，再将其转换为高效的 $\mathbb{Z}_T$ 形式并非易事。

2.4 约束编程

约束编程为寻找矩阵乘法方案提供了一种声明式方法。通过将 Brent方程建模为约束满足问题，求解器可以搜索有效的系数赋值。这种方法已用于从头发现小型格式的方案 [6], ，证明了其在受约束问题空间中进行穷尽搜索的有效性。

2.5 翻转图方法

Kauers 等人提出的翻转图方法 [7],将搜索重新表述为图探索问题。在这一模型中，每个顶点代表一个有效的矩阵乘法方案，边则对应翻转——即在保持方案正确性的同时对其进行修改的局部变换。这种组合式视角使得系统探索算法空间成为可能。

后续研究提出了一种自适应翻转图算法，并引入 aplus算子 [8], ，提升了搜索效率。最新进展——元翻转图方法 [9], ——通过支持不同矩阵维度之间的转换扩展了这一方法，显著扩大了可搜索空间。

Khoruzhii 等人近期的研究 [14] 表明，翻转图搜索也可应用于结构化矩阵乘法，并在 15 种结构化格式中的 13 种上降低了渐近常数。这表明该方法的适用范围不止于一般（非结构化）情形。

本研究直接基于元翻转图范式，贡献了一套开源实现，重点关注 $\mathbb{Z}_T$ 系数和实际可用性。此处介绍的框架着重优化位级性能，统一支持多种系数环，并向研究社区公开发布。这延续了此前关于三元元翻转图的研究方向 [10], ；在那些研究中， $\mathbb{Z}_T$系数的优势首次在翻转图框架中得到探索。

3 预备知识

本节介绍矩阵乘法方案的形式化描述，并回顾本研究中使用的翻转图算子。

3.1 矩阵乘法方案

在任意域 $\mathbb{F}$ 上，将矩阵 $A \in \mathbb{F}^{m\times n}$ 与 $B \in \mathbb{F}^{n\times p}$ 相乘的秩为 r 的矩阵乘法方案，由三组定义计算过程的系数组成—— $u_{ij}^{(l)}$ 、 $v_{ij}^{(l)}$ 和 $w_{ij}^{(l)}$ 。中间乘积（乘法）计算如下：

$$\begin{aligned} m_1 &= (u_{11}^{(1)}a_{11} + \cdots + u_{mn}^{(1)}a_{mn})\cdot(v_{11}^{(1)}b_{11} + \cdots + v_{np}^{(1)}b_{np}) \\ &\vdots \\ m_r &= (u_{11}^{(r)}a_{11} + \cdots + u_{mn}^{(r)}a_{mn})\cdot(v_{11}^{(r)}b_{11} + \cdots + v_{np}^{(r)}b_{np}), \end{aligned}$$

结果矩阵 $C = AB$ 中的元素计算如下：

$$c_{ij} = w_{ij}^{(1)}m_1 + \cdots + w_{ij}^{(r)}m_r.$$

在这种表述中，系数被组织成三个张量： $U \in \mathbb{F}^{r\times m\times n}$、 $V \in \mathbb{F}^{r\times n\times p}$ 和 $W \in \mathbb{F}^{r\times m\times p}$。该方案计算 r 个中间乘积，每个中间乘积都是由 A 中的元素构成的线性组合与由 B中的元素构成的线性组合相乘得到的，随后使用来自 W的系数重新组合，以生成最终结果。

比较方程 $c_{ij} = \sum_k a_{ik}b_{kj}$ 中所有项 $a_{i_1i_2}$、 $b_{j_1j_2}$、 $c_{k_1k_2}$ 的系数，可得到多项式方程，也称为 Brent方程 [15]:

$$\sum_{l=1}^{r} u_{i_1 i_2}^{(l)} v_{j_1 j_2}^{(l)} w_{k_1 k_2}^{(l)} = \delta_{i_2 j_1} \delta_{i_1 k_1} \delta_{j_2 k_2}$$

对于 $i_1, k_1 \in \{1, \cdots m\}$、$i_2, j_1 \in \{1, \cdots, n\}$ 和 $j_2, k_2 \in \{1, \cdots, p\}$。右侧的 δ_{ij} 是克罗内克δ，即当 $i = j$ 时为 $\delta_{ij} = 1$，否则为 $\delta_{ij} = 0$。

出于数学对称性和计算便利性的考虑，该表述采用 C^T 而非直接使用 C，从而在搜索过程中得到更对称的 Brent 方程。在这种表示中，张量的维度为$U \in \mathbb{F}^{r \times m \times n}$、$V \in \mathbb{F}^{r \times n \times p}$ 和 $W \in \mathbb{F}^{r \times p \times m}$，其元素计算如下：

$$c_{ji} = w_{ij}^{(1)} m_1 + \cdots + w_{ij}^{(r)} m_r.$$

这种表示符合矩阵乘法算法文献中的标准做法，并简化了约束满足问题 [2, 3, 6]。

3.2 翻转图算子

翻转图方法 [7] 将有效矩阵乘法方案的空间建模为一个图，其中顶点对应方案，边对应保持正确性的局部变换。为遍历该图，定义了若干算子。

3.2.1 核心算子

- **翻转**：对于满足 $u_{(i)} = u_{(j)}$ 的两个秩一张量，该变换定义为：

$$\begin{aligned} u^{(i)} \otimes v^{(i)} \otimes w^{(i)} \quad &+ \quad u^{(j)} \otimes v^{(j)} \otimes w^{(j)} \to \\ u^{(i)} \otimes (v^{(i)} + v^{(j)}) \otimes w^{(i)} \quad &+ \quad u^{(j)} \otimes v^{(j)} \otimes (w^{(j)} - w^{(i)}) \end{aligned}$$

  此操作在修改方案结构的同时保持其秩不变。

- **加法**：对于满足 $u_{(i)} \neq u_{(j)}$、$v_{(i)} \neq v_{(j)}$和 $w_{(i)} \neq w_{(j)}$的两个秩一张量，该变换将方案扩展为：

$$\begin{aligned} u^{(i)} \otimes v^{(i)} \otimes w^{(i)} \quad &+ \quad u^{(j)} \otimes v^{(j)} \otimes w^{(j)} \quad \to \\ u^{(i)} \otimes (v^{(i)} + v^{(j)}) \otimes w^{(i)} \quad &+ \quad u^{(i)} \otimes v^{(j)} \otimes (w^{(j)} - w^{(i)}) \quad + \quad (u^{(j)} - u^{(i)}) \otimes v^{(j)} \otimes w^{(j)} \end{aligned}$$

  此操作在保持正确性的同时提高方案的秩。

- **Split**：对于满足 $u^{(i)} \neq u^{(j)}$ 的两个秩一张量，该变换将方案扩展 a s:

$$\begin{aligned} u^{(i)} \otimes v^{(i)} \otimes w^{(i)} \quad &+ \quad u^{(j)} \otimes v^{(j)} \otimes w^{(j)} \quad \to \\ u^{(j)} \otimes v^{(i)} \otimes w^{(i)} \quad &+ \quad u^{(j)} \otimes v^{(j)} \otimes w^{(j)} \quad + \quad (u^{(i)} - u^{(j)}) \otimes v^{(i)} \otimes w^{(i)} \end{aligned}$$

  这同样会提高秩，同时保持正确性。

- **约简**：对于满足 $u_{(i)} = u_{(j)}$ 和 $v_{(i)} = v_{(j)}$的两个秩一张量，变换将它们组合如下：

$$u^{(i)} \otimes v^{(i)} \otimes w^{(i)} \quad + \quad u^{(j)} \otimes v^{(j)} \otimes w^{(j)} \quad \to \quad u^{(i)} \otimes v^{(i)} \otimes (w^{(i)} + w^{(j)})$$

  此操作通过消除冗余分量来降低方案的秩。

所有这些算子都可以应用于 u、v 和 w 张量的任意排列，从而全面涵盖各种可能的局部变换。根据 [8], 的功能，加法算子和拆分算子的功能合并为一个名为扩展的算子。执行时，它会均匀随机地选择应用加法变换或拆分变换。

除了这些局部操作外，还有一种全局变换，可同时作用于整个方案。**夹逼**取三个可逆矩阵 $S_U \in \mathbb{F}^{m \times m}$、$S_V \in \mathbb{F}^{n \times n}$ 和 $S_W \in \mathbb{F}^{p \times p}$，并对每个秩一分量应用以下变换：

$$u^{(i)} \otimes v^{(i)} \otimes w^{(i)} \to (S_U u^{(i)} S_V^{-1}) \otimes (S_V v^{(i)} S_W^{-1}) \otimes (S_W w^{(i)} S_U^{-1})$$

此操作在重组系数在各行之间的分布方式时，能够保持正确性和秩不变。作为一种特殊情况，选择标量矩阵 $S_U = \alpha I$, $S_V = \beta I$, $S_W = \gamma I$ 相当于重新缩放三元因子。夹逼提供了一种探索同一方案的等价表示、同时不改变其秩的方法。

3.2.2 元算子

除了局部变换之外，元翻转图框架 [9] 还引入了可改变方案维度的算子：

- **合并**：将两个方案（m，n，p_1：r_1）和（m，n，p_2：r_2）合并，得到一个方案（m，n，p_1+p_2：r_1+r_2）。
- **乘积**：将两个方案（m_1，n_1，p_1：r_1）和（m_2，n_2，p_2：r_2）组合，得到（m_1m_2，n_1n_2，p_1p_2：r_1r_2）。
- **扩展**：通过与一个平凡的（m，n，1）方案合并，朴素地将一个方案（m，n，p：r）扩展为（m，n，$p+1$：$r+mn$）。
- **投影**：通过移除一个维度并消去零系数，将一个方案（m，n，p）缩减为（m，n，$p-1$）。

这些算子适用于维度的任意排列。它们能够在搜索过程中实现不同矩阵格式之间的转换，显著扩展可达方案的空间。

3.3 系数环

本文考虑了若干种系数环：

- $\mathbb{Z}_2$：二元系数 $\{0,1\}$，运算模 2。
- $\mathbb{Z}_3$：三元系数 $\{0,1,2\}$，运算模 3。
- $\mathbb{Z}_T$：整数三元系数 $\{-1,0,1\}$，采用标准整数算术。
- $\mathbb{Z}$：整数系数，其中至少有一个系数不属于 $\{-1,0,1\}$。
- $\mathbb{Q}$：有理数系数，其中至少有一个系数是非整数分数。

请注意，从技术上讲，$\mathbb{Z}_T$ 是 $\mathbb{Z}$的子集，但出于实际考虑，我们仍作此区分：标记为 $\mathbb{Z}$ 的方案至少包含一个绝对值大于 1 的系数，因此与纯三元方案相比，这类方案在硬件实现中的效率较低。

4 翻转图框架：设计与实现

三元_翻转_图框架是一套用于发现和变换快速矩阵乘法方案的 C++ 工具。本节将介绍其设计理念、高效的系数编码、核心工具、并行化策略、随机游走算法，以及针对模环的提升过程。

4.1 设计理念

该框架围绕以下几项指导原则构建：

- **纯 C++ 且无依赖项**：所有工具均使用标准 C 实现++，只需要 g++ 编译器。这确保了最大的可移植性，并便于研究人员安装。
- **支持多种系数环**：支持 $\mathbb{Z}_2$、$\mathbb{Z}_3$ 和 $\mathbb{Z}_T$。在 $\mathbb{Z}_T$ 中发现的方案无需任何提升步骤，即可直接适用于任意环。
- **通过位级编码提升性能**：所有运算均使用快速的位运算指令执行，而非逐元素循环，因此即使在消费级硬件上也能进行大规模探索。
- **可复现性**：随机种子完全可控，因此能够精确复现实验。
- **开源**：完整源代码公开提供，以促进可复现研究和社区驱动的发现。

4.2 高效位级编码

为提升性能，该框架将系数向量表示为无符号整数。它不会将每个系数单独存储在 int8_t 数组中（这会导致在翻转或加法等操作中必须遍历所有元素），而是将矩阵切片的整个系数向量打包成一个无符号整数。

对于最多包含 N 个元素的矩阵，每个系数向量都以 $\lceil \log_2 N \rceil$ 位表示形式存储。该框架支持四种整数类型——uint16_t、uint32_t、uint64_t 和 __uint128_t——并会自动选择能够容纳方案维度的最小类型。这将框架限制在每个矩阵最多包含 128 个元素的方案，即 $m \times n \leq 128$、$n \times p \leq 128$ 和 $p \times m \leq 128$。这涵盖了截至 $11 \times 11 \times 11$ 的所有具有实际意义的小型格式。

不同的系数环按如下方式编码：

- $\mathbb{Z}_2$：用一个无符号整数表示该向量，每一位对应一个系数（0 或 1）。
- $\mathbb{Z}_T$：使用两个无符号整数。第一个编码符号（负系数对应的位设为 1），第二个编码数值（ ± 1 对应的位设为 1，0 对应的位清零）。这种符号-数值表示法可高效处理 $\{-1, 0, 1\}$ 系数。
- $\mathbb{Z}_3$：同样使用两个无符号整数，分别表示值$\{0, 1, 2\}$的双位编码中的高位和低位。直接在这种表示法上实现模 3 运算。

采用这种编码后，原本需要扫描多达 128 个元素的操作可简化为少量位运算。这显著提升了性能，也使得在商用硬件上进行大规模随机游走搜索成为可能。

4.3 核心工具概览

该框架 提供多种命令行工具，每种工具在发现流程中各有特定用途 ：

- 翻转_图：针对固定维度（m，n，p）在翻转图中执行随机游走搜索。可从朴素方案（根据维度生成）或从文件加载的一个或多个现有方案开始初始化。
- 元_翻转_图：通过可更改方案维度的元操作扩展翻转_图。支持翻转_图的所有操作，以及投影、扩展、合并和乘积。每个随机游走阶段结束后，都可以按可配置的概率调用元操作。
- 查找_替代_方案：使用翻转、扩展和夹心操作，根据输入方案生成维度相同的不同方案。可用于扩展现有方案，以便独立分析，或作为新搜索的初始化方案。
- 提升：通过亨塞尔提升，将模环（$\mathbb{Z}_2$ 或 $\mathbb{Z}_3$）提升到一般环（$\mathbb{Z}_T$、$\mathbb{Z}$或 $\mathbb{Q}$）。对于 $\mathbb{Z}_2$ 方案，经过 $\mathbb{Z}_4 \to \mathbb{Z}_8 \to \mathbb{Z}_{16} \to \cdots$ 进行提升；对于 $\mathbb{Z}_3$ 方案，则经过 $\mathbb{Z}_9 \to \mathbb{Z}_{27} \to \mathbb{Z}_{81} \to \cdots$ 进行提升。每完成一个提升步骤后，都会尝试进行有理重构，以获得有理数或整数系数。
- 方案_优化器：使用翻转操作，根据选定的指标优化方案。默认情况下，优化目标是使朴素加法复杂度最小（使零系数数量最多），也可以将可用翻转次数作为优化目标。

4.4 并行性与可复现性

所有工具均支持通过 OpenMP 并行执行。方案循环会将各次迭代分配给可用线程，每个线程都在分配给自己的方案上执行随机游走。这种方法可随 CPU 核心数线性扩展。所有线程完成迭代后，将汇总并报告结果。

通过显式控制随机种子来确保可复现性。系统会提供一个主种子，每个运行器都会使用由种子 + i 确定性生成的序列，其中 i 表示第 i 个运行器。这样，无论使用多少个并行线程，都能精确复现任何搜索实验，这对科学验证至关重要。

该框架已在各种硬件配置上经过测试，从 12 核笔记本电脑到 96 核机构服务器。其轻量级设计且无需外部依赖，因此易于部署到不同环境中。

4.5 随机游走算法

该框架的核心是在翻转图中进行并行随机游走。多个独立的运行器执行相同的算法，各自探索搜索空间中的不同区域。

每个运行器都会维护一个当前方案，并执行一系列局部变换。每次迭代时，首先尝试翻转操作，因为翻转不会改变秩。如果翻转失败，则可以尝试扩展操作，以暂时

提高秩，从而摆脱局部最优解。翻转成功时，会以概率方式应用约简和夹心操作，以降低秩或重构方案。

该算法会跟踪两个计数器。第一个是 flipsCount，用于统计连续执行且未降低秩的翻转次数。第二个是 iterationsCount，用于统计自该运行器上次改进其已知最佳秩以来的总迭代次数。

如果 flipsCount 超过重置时从某个分布中采样得到的阈值，就会强制进行扩展（加法或拆分），以提高秩并探索图中的新区域。这可以防止运行器在仅靠翻转无法取得进一步进展的区域中停滞。

如果 iterationsCount 达到可配置的重置上限，而最佳秩仍未得到改进，运行器就会重置为从近期发现的改进方案池中抽取的一个方案。该方案池是一个循环缓冲区，只存储最近的改进；其大小由一个参数配置（通常设为 10）。如果找到的改进少于缓冲区容量，方案池也会包含初始方案。这样既能舍弃无成果的搜索路径，又能确保重置从搜索空间中有希望的区域开始。
这种方法使随机游走能够有效地遍历翻转图，而不会长时间困在某个特定区域。

4.6 从模环提升

在 $\mathbb{Z}_2$ 或 $\mathbb{Z}_3$ 中发现的方案分别仅在特征为 2 或 3 的域上有效。为了获得适用于任意域的算法，必须将这些方案提升到零特征。该框架实现了亨塞尔提升，随后进行有理重构。

在每一步从模数 p^k 提升到 p^{k+1}时，都必须求解一个线性方程组：

$$Jx \equiv (T - U \otimes V \otimes W) \pmod{p^k}$$

其中，J 是在当前方案处计算得到的 Brent方程的雅可比矩阵，而 T 是真正的矩阵乘法张量。该解 x提供修正项，以获得一个模 p^{k+1}有效的方案。
由于 Brent方程是双线性的，该方程组是不定的，通常有许多解。所选解会显著影响有理重构后的系数——某些解会产生紧凑的 $\mathbb{Z}_T$ 或整数系数，而另一些则会产生较大的分数。

对于从 $\mathbb{Z}_2$ 进行提升，每个提升步骤都会采用引导策略。对于每个系数位置，当前值 v_i（来自上一步提升）是已知的。两种可能的更新方式（$v_i + x_i \cdot p^k$）（模 p^{k+1}）分别为 $v_i + 0$ 和（$v_i + p^k$）（模 p^{k+1}），对应于将修正量设为 $x_i = 0$ 或 $x_i = 1$。如果这两个候选值中只有一个可以进行有理重构，则将修正量 x_i 固定为相应的值。这在$\mathbb{Z}_2 \to \mathbb{Z}_4$ 步骤尤为有效：$\mathbb{Z}_4$ 中的值 2 无法进行有效的有理重构（它们对应于 $1/2$），因此零往往会保持为零，而 $\mathbb{Z}_4$ 中的 1 和 3 则分别对应于 $\mathbb{Q}$中的 1 和 -1。
如果受约束系统有解，提升过程将采用固定系数继续进行，通常可直接得到一个 $\mathbb{Z}_T$ 方案。如果无解，则会丢弃所有约束，在不受限制的情况下求解系统，并继续进行更高阶的提升步骤，希望最终能够成功进行有理重构。

这种权衡在于，对解施加约束可能会剪除一些路径，而这些路径在经过更多提升步骤后本可导向有效方案。不过在实践中，受约束模式的成功率通常足够高，因此值得将其作为默认的首次尝试。这一策略仅应用于 $\mathbb{Z}_2$ 提升；目前尚未为 $\mathbb{Z}_3$ 实现此类约束。

4.7 可用性

完整源代码可在 GitHub 上获取：https://github.com/dronperminov/ternary_flip_graph。该代码仓库包含上述所有工具，以及文档和用法示例。发现的方案——包括所有改进的秩，以及重新发现的 $\mathbb{Z}_T$和整数方案——保存在单独的结果代码仓库中：https://github.com/dronperminov/FastMatrixMultiplication。这两个代码仓库均为开源项目，并在积极维护。

5 从现有方案构建新方案

虽然翻转图搜索针对固定维度的单个方案进行，但可以通过组合较小的方案来构造更大的方案。本节介绍两种互补的方法：通过元操作进行代数组合，以及

分块矩阵乘法。由此得到的构造可作为进一步进行翻转图优化的起点；对于超出框架编码限制的维度，这些构造则代表最终发现的方案。

5.1 通过元操作进行组合

使用第 3.2.2 节所述的合并、扩展和乘积操作，可以通过对现有方案进行简单的代数组合来获得适用于更大维度的新方案。这些操作具有结合律，并且可以重复应用。文献中的许多方案都源于此类组合。例如：

$$
\begin{aligned}
(3,3,7:49) &= (3,3,1:9) + (3,3,6:40) \quad \text{(extend)} \\
(3,3,9:63) &= (3,3,3:23) + (3,3,6:40) \quad \text{(merge)} \\
(6,6,8:203) &= (2,2,2:7) \times (3,3,4:29) \quad \text{(product)}
\end{aligned}
$$

其中，合并通过对秩求和来构建更大的输出维度，而乘积则以秩相乘为代价，构建指数级增大的维度。扩展操作提供了一种受控的方式，用于逐步增加维度。

5.2 分块矩阵乘法

对于超过一定规模的维度，合并和乘积通常会产生远非最优的秩。采用精心选择的基础方案进行分块矩阵乘法，可以构造出更灵活的方案。

给定目标维度（m, n, p），每个维度被划分为若干分块。分块数量（b_n, b_m, b_p）决定基础方案的维度。该基础方案定义了分块如何组合：其每一次乘法都指定涉及 A 的哪个分块、 B 的哪个分块以及 C 的哪个分块。

对于基础方案中的每一次乘法，实际分块大小由分块划分决定。由于所得分块乘法的并非所有元素都对最终矩阵 C 有贡献，可以选择使零出现在分块边界处的方案，从而在适当位置使用更小的快速矩阵乘法方案。总体秩是基础方案所有乘法中所用全部方案的秩之和。

例如，考虑为 (4, 7,15) 构造一个方案。将维度按 $[2,2] \times [3,4] \times [7,8]$ 划分，得到分块维度 (2, 2,2)。以 Strassen 原始的 $2 \times 2 \times 2$ 方案作为基础方案，需要进行七次分块乘法，每次都有各自的维度：

$$
\begin{aligned}
m_1 &: (2,4,8:51) \\
m_2 &: (2,3,7:35) \\
m_3 &: (2,4,7:45) \\
m_4 &: (2,3,8:40) \\
m_5 &: (2,4,8:51) \\
m_6 &: (2,4,7:45) \\
m_7 &: (2,3,8:40)
\end{aligned}
$$

总秩是用于各个块乘法的方案的秩之和： $51+35+45+40+51+45+40 =$307 次乘法，适用于 (4, 7,15) 格式。

5.3 与翻转图搜索的集成

通过合并、扩展、乘积和块乘法构造得到的方案均为有效方案，可作为翻转图搜索的起点，且符合该框架的 128 个元素限制。超出此限制的方案则直接报告为已发现的上界。

6 结果

该框架用于在维度最高达$16 \times 16 \times 16$的范围内，对快速矩阵乘法方案进行广泛搜索。本节介绍主要发现：79 个方案的秩得到改进、在更实用的系数环中重新发现已知方案，以及一种渐近效率优于 Strassen 的新 $4 \times 4 \times 10$ 方案。

6.1 实验设置

所有实验均在三个系统上进行 s:

- 一台配备 12 核英特尔酷睿 i7-9750H 的笔记本电脑，用于快速实验，以及使用全新初始化进行频繁重启。该设置可实现快速迭代，并测试新的搜索策略。
- 一台配备 20 核英特尔至强铂金 8358 的工作站，用于执行较长时间的搜索。通常，meta_翻转_图会一次运行一周，每天平均发现一至五次秩改进。
- JKU 的一套 96 核机构集群，由 Isaac Wood 慷慨提供[1]。他的运行发现了若干个 $\mathbb{Z}_2$ 系数方案，随后将其提升至 $\mathbb{Z}_T$。作者对这项贡献深表感谢。

Ov在该项目期间，这些系统总共执行了数十亿次翻转图迭代 .

6.2 改进的秩

表1列出了框架发现其秩严格优于此前已知上界的矩阵乘法格式。对于每种格式，表中并列显示此前最佳秩及其系数环，以及新发现的秩和系数环。文中还给出了指数 $\omega = 3\log(r)/\log(mnp)$，以便与 Strassen 指数 $\log_2 7 \approx 2.807$进行比较。

值得注意的是，具有 115 次乘法的 $4\times4\times10$方案实现了指数 $\omega \approx 2.80478$，严格优于 Strassen 指数。虽然对于固定的小型格式，渐近指数无法直接比较，但这表明，对于这一特定规模，该方案比递归应用 Strassen 算法更高效。

6.3 在三元系数中重新发现的方案

对于 93 种矩阵乘法格式，此前只有系数为有理数或整数的方案能够达到已知的最佳秩。使用该框架的搜索工具，还发现了秩相同、但系数属于 $\{-1,0,1\}$ 的其他方案。表2列出了在系数为 $\mathbb{Z}_T$ 时找到相同秩的替代方案所对应的格式。将这些方案表示为三元形式后，便可直接用于高效实现，因为所有乘法都可替换为加法和减法。

6.4 以整数系数重新发现的方案

另有68种此前需要分数的方案，仅用整数系数便重新发现了。虽然效率不如三元系数，但在实际实现中，整数系数仍优于分数。表3列出了以整数系数重新发现方案的格式。

6.5 总体统计数据

本研究追踪了680种值得关注的矩阵乘法方案。经过这项工作后，系数环上的分布如下：

- **三元**（$\mathbb{Z}_T$）：276种方案（40.6%）；
- **整数**（$\mathbb{Z}$）：117种方案（17.2%）；
- **有理数 (**$\mathbb{Q}$**)**：287 个方案（42.2%）。

在这 680 个方案中，有 29 个的秩低于 Strassen 算法针对其维度所能达到的秩——这意味着在特定规模下应用时，它们的效率更高。其中一个方案（ $4\times4\times10$ ）是在本研究中发现的，其余 28 个此前已为人所知。

7 讨论

本研究 本节回顾搜索过程中的观察结果，并概述未来研究方向 h.

[1]Isaac Wood，约翰内斯·开普勒大学代数研究所，奥地利林茨 A4040

表1：矩阵乘法格式的改进的秩

格式	Prev		New		ω	格式	Prev		New		ω
$m \times n \times p$	Rank	Ring	Rank	Ring		$m \times n \times p$	Rank	Ring	Rank	Ring	
$4 \times 4 \times 10$	120	$\mathbb{Q}$	115	$\mathbb{Z}_T$	**2.80479**	$9 \times 10 \times 14$	820	$\mathbb{Z}$	819	$\mathbb{Q}$	2.81897
$4 \times 4 \times 12$	142	$\mathbb{Q}$	141	$\mathbb{Z}_T$	2.82383	$9 \times 11 \times 11$	725	$\mathbb{Q}$	715	$\mathbb{Q}$	2.81951
$4 \times 4 \times 14$	165	$\mathbb{Q}$	163	$\mathbb{Q}$	2.82377	$9 \times 11 \times 12$	760	$\mathbb{Q}$	754	$\mathbb{Q}$	2.80736
$4 \times 4 \times 15$	177	$\mathbb{Q}$	176	$\mathbb{Z}_T$	2.83023	$9 \times 11 \times 13$	849	$\mathbb{Z}$	835	$\mathbb{Q}$	2.81873
$4 \times 4 \times 16$	189	$\mathbb{Q}$	188	$\mathbb{Z}_T$	2.83297	$9 \times 11 \times 14$	904	$\mathbb{Z}$	889	$\mathbb{Q}$	2.81584
$4 \times 5 \times 9$	136	$\mathbb{Q}$	132	$\mathbb{Z}_T$	2.82082	$9 \times 11 \times 15$	981	$\mathbb{Q}$	960	$\mathbb{Z}$	2.82080
$4 \times 5 \times 10$	151	$\mathbb{Z}$	146	$\mathbb{Z}_T$	2.82181	$9 \times 11 \times 16$	1030	$\mathbb{Z}$	1023	$\mathbb{Q}$	2.82197
$4 \times 5 \times 11$	165	$\mathbb{Z}$	160	$\mathbb{Z}_T$	2.82287	$9 \times 12 \times 13$	900	$\mathbb{Q}$	884	$\mathbb{Q}$	2.80849
$4 \times 5 \times 12$	179	$\mathbb{Z}_T$	174	$\mathbb{Z}_T$	2.82397	$9 \times 12 \times 16$	1080	$\mathbb{Q}$	1072	$\mathbb{Q}$	2.80786
$4 \times 5 \times 13$	194	$\mathbb{Z}$	191	$\mathbb{Z}_T$	2.83361	$9 \times 13 \times 13$	996	$\mathbb{Z}$	981	$\mathbb{Q}$	2.82044
$4 \times 5 \times 14$	208	$\mathbb{Z}$	206	$\mathbb{Q}$	2.83660	$9 \times 13 \times 14$	1063	$\mathbb{Z}$	1041	$\mathbb{Q}$	2.81626
$4 \times 5 \times 15$	226	$\mathbb{Z}$	221	$\mathbb{Z}_T$	2.83925	$9 \times 13 \times 15$	1135	$\mathbb{Q}$	1119	$\mathbb{Z}$	2.81927
$4 \times 5 \times 16$	240	$\mathbb{Q}$	235	$\mathbb{Z}_T$	2.83943	$9 \times 13 \times 16$	1210	$\mathbb{Z}$	1183	$\mathbb{Q}$	2.81727
$4 \times 6 \times 13$	228	$\mathbb{Z}$	227	$\mathbb{Z}_T$	2.83386	$9 \times 14 \times 14$	1136	$\mathbb{Z}$	1121	$\mathbb{Q}$	2.81806
$4 \times 7 \times 11$	227	$\mathbb{Z}$	226	$\mathbb{Z}_T$	2.83793	$9 \times 15 \times 15$	1290	$\mathbb{Q}$	1284	$\mathbb{Z}$	2.82048
$4 \times 9 \times 11$	280	$\mathbb{Z}_T$	279	$\mathbb{Z}_T$	2.82435	$9 \times 15 \times 16$	1350	$\mathbb{Z}$	1341	$\mathbb{Q}$	2.81374
$5 \times 5 \times 9$	167	$\mathbb{Z}$	161	$\mathbb{Z}_T$	2.81461	$9 \times 16 \times 16$	1444	$\mathbb{Z}_T$	1431	$\mathbb{Q}$	2.81546
$5 \times 5 \times 10$	184	$\mathbb{Q}$	178	$\mathbb{Z}_T$	2.81544	$10 \times 11 \times 15$	1067	$\mathbb{Q}$	1055	$\mathbb{Z}_T$	2.81890
$5 \times 5 \times 11$	202	$\mathbb{Q}$	195	$\mathbb{Z}_T$	2.81639	$10 \times 13 \times 16$	1332	$\mathbb{Z}$	1326	$\mathbb{Z}_T$	2.82322
$5 \times 5 \times 12$	220	$\mathbb{Z}$	208	$\mathbb{Z}_T$	2.80737	$11 \times 11 \times 15$	1170	$\mathbb{Z}_T$	1169	$\mathbb{Z}_T$	2.82412
$5 \times 5 \times 13$	237	$\mathbb{Z}$	228	$\mathbb{Z}_T$	2.81614	$11 \times 12 \times 13$	1102	$\mathbb{Z}$	1092	$\mathbb{Q}$	2.81794
$5 \times 5 \times 14$	254	$\mathbb{Z}$	248	$\mathbb{Z}_T$	2.82357	$11 \times 12 \times 15$	1264	$\mathbb{Q}$	1240	$\mathbb{Q}$	2.81505
$5 \times 5 \times 15$	271	$\mathbb{Q}$	266	$\mathbb{Z}_T$	2.82617	$11 \times 13 \times 13$	1210	$\mathbb{Z}$	1205	$\mathbb{Z}_T$	2.82722
$5 \times 5 \times 16$	288	$\mathbb{Q}$	284	$\mathbb{Z}_T$	2.82851	$11 \times 13 \times 14$	1298	$\mathbb{Z}$	1292	$\mathbb{Z}_T$	2.82717
$5 \times 6 \times 9$	197	$\mathbb{Z}$	193	$\mathbb{Z}_T$	2.82009	$11 \times 13 \times 16$	1472	$\mathbb{Z}$	1452	$\mathbb{Q}$	2.82364
$5 \times 7 \times 8$	205	$\mathbb{Q}$	204	$\mathbb{Z}_T$	2.83140	$11 \times 14 \times 14$	1388	$\mathbb{Z}$	1376	$\mathbb{Z}_T$	2.82449
$5 \times 9 \times 9$	294	$\mathbb{Q}$	293	$\mathbb{Z}_T$	2.83825	$11 \times 14 \times 15$	1471	$\mathbb{Z}$	1460	$\mathbb{Z}$	2.82228
$6 \times 7 \times 7$	215	$\mathbb{Z}_T$	212	$\mathbb{Z}_T$	2.82740	$11 \times 14 \times 16$	1571	$\mathbb{Q}$	1548	$\mathbb{Q}$	2.82144
$6 \times 7 \times 8$	239	$\mathbb{Z}_T$	238	$\mathbb{Z}_T$	2.82216	$12 \times 12 \times 14$	1250	$\mathbb{Q}$	1240	$\mathbb{Q}$	2.80838
$6 \times 7 \times 10$	296	$\mathbb{Z}$	293	$\mathbb{Q}$	2.82116	$12 \times 13 \times 16$	1556	$\mathbb{Q}$	1548	$\mathbb{Z}_T$	2.81679
$7 \times 7 \times 10$	346	$\mathbb{Z}$	345	$\mathbb{Q}$	2.83008	$13 \times 13 \times 13$	1426	$\mathbb{Q}$	1421	$\mathbb{Q}$	2.83012
$7 \times 8 \times 15$	571	$\mathbb{Q}$	570	$\mathbb{Q}$	2.82723	$13 \times 13 \times 14$	1524	$\mathbb{Z}$	1511	$\mathbb{Z}_T$	2.82684
$7 \times 9 \times 11$	480	$\mathbb{Q}$	478	$\mathbb{Z}_T$	2.82965	$13 \times 13 \times 16$	1713	$\mathbb{Q}$	1704	$\mathbb{Q}$	2.82471
$7 \times 9 \times 15$	639	$\mathbb{Z}$	634	$\mathbb{Q}$	2.82523	$13 \times 14 \times 14$	1625	$\mathbb{Z}$	1614	$\mathbb{Z}_T$	2.82535
$7 \times 14 \times 15$	976	$\mathbb{Z}$	969	$\mathbb{Z}$	2.82857	$13 \times 14 \times 15$	1714	$\mathbb{Z}$	1698	$\mathbb{Z}_T$	2.81995
$8 \times 8 \times 16$	672	$\mathbb{Q}$	671	$\mathbb{Q}$	2.81705	$13 \times 14 \times 16$	1825	$\mathbb{Q}$	1806	$\mathbb{Q}$	2.82033
$8 \times 9 \times 11$	533	$\mathbb{Q}$	531	$\mathbb{Z}_T$	2.82030	$13 \times 15 \times 16$	1932	$\mathbb{Z}$	1908	$\mathbb{Q}$	2.81663
$8 \times 9 \times 14$	669	$\mathbb{Z}$	666	$\mathbb{Z}_T$	2.82022	$14 \times 14 \times 16$	1939	$\mathbb{Q}$	1938	$\mathbb{Q}$	2.82065
$9 \times 10 \times 10$	600	$\mathbb{Z}$	597	$\mathbb{Z}_T$	2.81897	$15 \times 15 \times 16$	2173	$\mathbb{Q}$	2155	$\mathbb{Q}$	2.81201
$9 \times 10 \times 13$	772	$\mathbb{Z}$	765	$\mathbb{Q}$	2.81958						

7.1 翻转潜力作为质量指标

搜索过程中，我们发现了一个有趣的规律：方案中潜在翻转的数量——满足 $1 \le i < j \le r$，使得 $u_i = u_j$、$v_i = v_j$ 或 $w_i = w_j$ 的配对数。观察发现，这个数量越大，方案就越有可能离最优秩越远。反过来则不成立：潜在翻转很少的方案不一定是最优的，因为有些方案尽管重合行很少，却经受住了所有改进尝试。然而，具有较高翻转潜力的方案（通常有 60–120 对）在进一步搜索过程中几乎总是表现出稳定的秩约简。

这一观察与以下事实一致：扩展操作会提高秩，同时也会增加潜在翻转的数量。尽管还需要进一步研究才能确立严格的关联，但翻转潜力可以作为一种有用的启发式方法：潜在翻转较多的方案是值得进行更深入搜索的候选方案，因此应为其分配更多运行器。

表2：使用三元系数重新发现的方案（$\mathbb{Z}_T$）

格式 $m \times n \times p$	Rank	已知 ring	格式 $m \times n \times p$	Rank	已知 ring	格式 $m \times n \times p$	Rank	已知 ring
$2 \times 7 \times 8$	88	$\mathbb{Z}$	$6 \times 8 \times 10$	329	$\mathbb{Z}$	$10 \times 10 \times 12$	770	$\mathbb{Z}$
$2 \times 8 \times 15$	188	$\mathbb{Z}$	$6 \times 8 \times 11$	357	$\mathbb{Q}$	$10 \times 10 \times 13$	838	$\mathbb{Z}$
$3 \times 3 \times 7$	49	$\mathbb{Q}$	$6 \times 8 \times 12$	378	$\mathbb{Q}$	$10 \times 10 \times 14$	889	$\mathbb{Z}$
$3 \times 3 \times 9$	63	$\mathbb{Q}$	$6 \times 9 \times 9$	342	$\mathbb{Z}$	$10 \times 10 \times 15$	957	$\mathbb{Q}$
$3 \times 4 \times 6$	54	$\mathbb{Z}/\ \mathbb{Q}$	$6 \times 9 \times 10$	373	$\mathbb{Z}$	$10 \times 10 \times 16$	1008	$\mathbb{Q}$
$3 \times 4 \times 9$	83	$\mathbb{Q}$	$6 \times 11 \times 15$	661	$\mathbb{Z}$	$10 \times 11 \times 11$	793	$\mathbb{Z}$
$3 \times 4 \times 10$	92	$\mathbb{Q}$	$6 \times 12 \times 15$	705	$\mathbb{Z}$	$10 \times 11 \times 12$	850	$\mathbb{Z}$
$3 \times 4 \times 11$	101	$\mathbb{Q}$	$6 \times 12 \times 16$	746	$\mathbb{Q}$	$10 \times 11 \times 13$	924	$\mathbb{Z}$
$3 \times 4 \times 12$	108	$\mathbb{Q}$	$6 \times 13 \times 15$	771	$\mathbb{Z}$	$10 \times 11 \times 14$	981	$\mathbb{Z}$
$3 \times 4 \times 16$	146	$\mathbb{Q}$	$7 \times 8 \times 10$	385	$\mathbb{Z}$	$10 \times 12 \times 12$	910	$\mathbb{Z}$
$3 \times 5 \times 10$	115	$\mathbb{Z}$	$7 \times 8 \times 11$	423	$\mathbb{Q}$	$10 \times 12 \times 13$	990	$\mathbb{Z}$
$3 \times 6 \times 8$	108	$\mathbb{Z}/\ \mathbb{Q}$	$7 \times 8 \times 12$	454	$\mathbb{Q}$	$10 \times 12 \times 14$	1050	$\mathbb{Z}$
$3 \times 8 \times 12$	216	$\mathbb{Q}$	$7 \times 9 \times 10$	437	$\mathbb{Z}$	$10 \times 13 \times 13$	1082	$\mathbb{Z}$
$4 \times 4 \times 11$	130	$\mathbb{Q}$	$7 \times 12 \times 15$	831	$\mathbb{Z}$	$10 \times 13 \times 14$	1154	$\mathbb{Z}$
$4 \times 6 \times 9$	159	$\mathbb{Q}$	$7 \times 13 \times 15$	909	$\mathbb{Z}$	$10 \times 13 \times 15$	1242	$\mathbb{Z}$
$4 \times 6 \times 11$	194	$\mathbb{Q}$	$8 \times 8 \times 11$	475	$\mathbb{Q}$	$10 \times 14 \times 14$	1232	$\mathbb{Z}$
$4 \times 6 \times 15$	263	$\mathbb{Z}$	$8 \times 8 \times 13$	559	$\mathbb{Q}$	$10 \times 14 \times 15$	1327	$\mathbb{Z}$
$4 \times 7 \times 7$	144	$\mathbb{Z}/\ \mathbb{Q}$	$8 \times 9 \times 13$	624	$\mathbb{Z}$	$10 \times 14 \times 16$	1423	$\mathbb{Z}$
$4 \times 7 \times 12$	246	$\mathbb{Z}$	$8 \times 9 \times 15$	705	$\mathbb{Z}$	$10 \times 15 \times 15$	1395	$\mathbb{Z}$
$4 \times 7 \times 15$	307	$\mathbb{Q}$	$8 \times 9 \times 16$	746	$\mathbb{Q}$	$10 \times 15 \times 16$	1497	$\mathbb{Z}$
$4 \times 8 \times 13$	297	$\mathbb{Z}$	$8 \times 10 \times 11$	588	$\mathbb{Z}$	$11 \times 11 \times 11$	873	$\mathbb{Z}$
$4 \times 9 \times 14$	355	$\mathbb{Z}$	$8 \times 10 \times 12$	630	$\mathbb{Z}$	$11 \times 11 \times 12$	936	$\mathbb{Z}$
$4 \times 9 \times 15$	375	$\mathbb{Z}$	$8 \times 10 \times 13$	686	$\mathbb{Z}$	$11 \times 11 \times 13$	1023	$\mathbb{Z}$
$5 \times 6 \times 16$	340	$\mathbb{Q}$	$8 \times 10 \times 14$	728	$\mathbb{Z}$	$11 \times 11 \times 14$	1093	$\mathbb{Z}$
$5 \times 7 \times 10$	254	$\mathbb{Z}$	$8 \times 11 \times 14$	804	$\mathbb{Z}$	$11 \times 12 \times 14$	1182	$\mathbb{Z}$
$5 \times 7 \times 11$	277	$\mathbb{Z}$	$8 \times 11 \times 15$	859	$\mathbb{Z}$	$11 \times 13 \times 15$	1377	$\mathbb{Z}$
$5 \times 7 \times 13$	325	$\mathbb{Q}$	$8 \times 12 \times 14$	861	$\mathbb{Z}$	$13 \times 13 \times 15$	1605	$\mathbb{Z}$
$5 \times 8 \times 12$	333	$\mathbb{Q}$	$8 \times 13 \times 14$	945	$\mathbb{Z}$	$13 \times 15 \times 15$	1803	$\mathbb{Z}$
$5 \times 9 \times 10$	322	$\mathbb{Q}$	$8 \times 14 \times 14$	1008	$\mathbb{Z}$	$14 \times 14 \times 15$	1813	$\mathbb{Z}$
$5 \times 9 \times 15$	474	$\mathbb{Z}$	$10 \times 10 \times 10$	651	$\mathbb{Z}$	$14 \times 15 \times 15$	1905	$\mathbb{Z}$
$6 \times 6 \times 7$	183	$\mathbb{Z}/\ \mathbb{Q}$	$10 \times 10 \times 11$	719	$\mathbb{Z}$	$15 \times 15 \times 15$	2058	$\mathbb{Q}$

表3：具有整数系数的重新发现的方案（$\mathbb{Z}$）

格式 $m \times n \times p$	Rank	格式 $m \times n \times p$	Rank	格式 $m \times n \times p$	Rank	格式 $m \times n \times p$	Rank
$2 \times 5 \times 7$	55	$2 \times 8 \times 14$	175	$4 \times 10 \times 13$	361	$6 \times 8 \times 16$	511
$2 \times 5 \times 8$	63	$2 \times 11 \times 12$	204	$4 \times 10 \times 14$	385	$6 \times 9 \times 11$	407
$2 \times 5 \times 13$	102	$2 \times 11 \times 13$	221	$4 \times 10 \times 15$	417	$6 \times 9 \times 12$	434
$2 \times 5 \times 14$	110	$2 \times 11 \times 14$	238	$4 \times 10 \times 16$	441	$6 \times 10 \times 13$	520
$2 \times 5 \times 15$	118	$2 \times 13 \times 15$	300	$4 \times 11 \times 14$	429	$6 \times 10 \times 14$	553
$2 \times 5 \times 16$	126	$2 \times 13 \times 16$	320	$4 \times 11 \times 16$	489	$6 \times 10 \times 15$	597
$2 \times 6 \times 8$	75	$2 \times 15 \times 16$	368	$4 \times 14 \times 14$	532	$6 \times 10 \times 16$	630
$2 \times 6 \times 13$	122	$3 \times 4 \times 8$	73	$5 \times 7 \times 9$	229	$6 \times 13 \times 14$	730
$2 \times 6 \times 14$	131	$3 \times 5 \times 7$	79	$5 \times 8 \times 9$	260	$6 \times 13 \times 16$	819
$2 \times 7 \times 7$	76	$3 \times 5 \times 13$	147	$5 \times 8 \times 16$	445	$6 \times 14 \times 14$	777
$2 \times 7 \times 9$	99	$3 \times 5 \times 14$	158	$5 \times 9 \times 11$	353	$6 \times 14 \times 15$	825
$2 \times 7 \times 12$	131	$3 \times 5 \times 15$	169	$5 \times 9 \times 12$	377	$6 \times 14 \times 16$	880
$2 \times 7 \times 13$	142	$3 \times 7 \times 7$	111	$5 \times 10 \times 13$	451	$7 \times 8 \times 16$	603
$2 \times 7 \times 14$	152	$3 \times 8 \times 9$	163	$5 \times 10 \times 14$	481	$7 \times 10 \times 13$	614
$2 \times 7 \times 15$	164	$3 \times 8 \times 11$	198	$5 \times 10 \times 15$	519	$7 \times 10 \times 14$	653
$2 \times 7 \times 16$	175	$3 \times 8 \times 16$	288	$5 \times 10 \times 16$	549	$7 \times 13 \times 14$	852
$2 \times 8 \times 9$	113	$3 \times 10 \times 16$	360	$5 \times 11 \times 16$	609	$9 \times 14 \times 15$	1185

7.2 搜索难度的变化

并非所有方案在翻转图搜索中的表现都相同。有些格式尽管规模较大，却能迅速收敛到低秩解。例如，朴素的 $4 \times 4 \times 4$ 方案通常能在相对

搜索时间较短。相比之下，看似更小的格式（如 $2\times4\times5$ 或 $3\times3\times6$）可能需要经过显著更多次迭代才能取得进展。

这种多变性意味着，关于翻转图行为的任何假设——无论涉及搜索策略、系数环还是质量启发式——在得出一般性结论之前，都必须在多种不同方案上进行检验。适用于一种格式的结果未必适用于另一种格式，而系统性的评估需要广泛的实验覆盖。

7.3 指数趋势

矩阵维度与所得指数 $\omega = 3\log(r)/\log(mnp)$ 之间的关系远非简单直接。图 1 展示了这种复杂性：在前两个维度 $m \le n \le p$固定时，指数随最后一个维度 p 增大而变化。每个图都对应一个固定的 m（范围为 3 至 8），其中多条曲线分别对应不同的 n 值，范围从 m 到 14。水平虚线表示 Strassen 指数 $\log_2 7 \approx 2.807$。

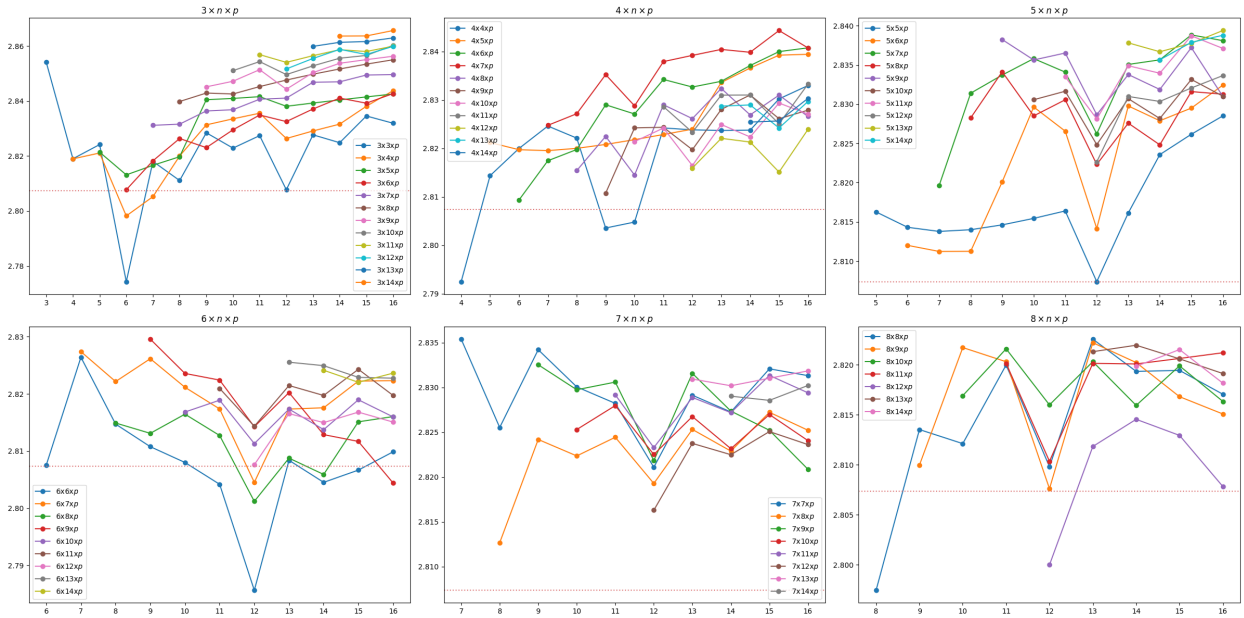


图 1：固定 m 和 n 时，指数 ω 随最后一个维度 p 的变化（沿 x 轴递增）。上排：$m = 3$，4，5；下排：$m = 6$，7，8。虚线表示 Strassen 指数。

从这些图中可以观察到几点。首先，不同维度下的变化趋势有显著差异。例如，$6\times9\times p$ 的指数随着 p 增大总体呈下降趋势（尽管偶尔会小幅上升），而对于 $5\times5\times p$，指数总体呈上升趋势。这表明，与其他格式相比，某些格式从最后一个维度的增加中获益更多。

其次，可以看到明显的不连续现象，即指数会急剧下降至总体趋势以下。最突出的例子是 $4\times4\times p$：$p = 9$ 和 $p = 10$ 的指数低于 Strassen 界，但随着 p 进一步增大，指数又会回升。

第三，许多格式在 $p = 12$ 处出现向下的尖峰，在 $5\times n\times12$ 和 $8\times n\times12$ 曲线中尤为明显。这表明，对于 $m\times n\times12$ 格式，可能存在尚未得到充分理解的高效构造。该尖峰可能受到高效的 $3\times3\times6$ 和 $3\times4\times6$ 方案通过分块构造传播的影响，但完整的解释仍有待探究。

这些观察表明，快速矩阵乘法的研究领域远非一帆风顺。维度的微小变化可能带来不成比例的巨大改进，而看似相似的格式可能呈现截然不同的扩展行为。为何某些维度能实现效率极高的方案，仍是一个悬而未决的问题。

7.4 系数环之间的差距

关于 $3 \times 3 \times 6$ 格式，有一个耐人寻味的观察。虽然已知的最佳有理方案的秩为 40 [4],，但在 $\mathbb{Z}_T$、 $\mathbb{Z}_2$中的搜索，甚至 $\mathbb{Z}_3$，都一再止步于秩 42。该有理方案涉及系数 $\pm 1/8$，这些系数可以转换回 $\mathbb{Z}_3$（因为 $8 \equiv 2$（模 3））， 但秩为 40 的整数方案或三元方案仍难以找到。
对于 $2 \times 4 \times 5$，也出现了同样的现象：已知的最佳有理秩为 32 [13]，但在受限系数环中进行搜索只能达到秩 33。这引出了一个根本问题：对于最优秩仅通过有理方案得知的格式，是否存在秩相同的整数方案或三元方案? 近期研究 [16] 提出了一种方法，用于确定具有无理系数的方案能否转换为有理方案，并提供了一种检查有理方案是否无法通过夹心操作转换为整数系数方案的方法。迄今为止，针对 $3 \times 3 \times 6$ 得到的有理替代方案都无法通过该方法转换为整数方案。

7.5 亨塞尔提升中的挑战

将模环中的方案提升到零特征需要在每一步求解欠定线性系统。由于雅可比矩阵不是方阵，这些系统通常有许多解。解的选择会显著影响有理重构后系数的大小——有些解会得到紧凑的 $\mathbb{Z}_T$ 或整数方案，另一些则会产生较大的分数。

第 4.6 节所述的约束提升策略有助于引导搜索朝着紧凑型解发展，但它远非完美。剪枝解空间可能会截断一些路径，而这些路径在经过额外的提升步骤后本可以导向有效方案。开发更好的提升空间导航方法——或许可以枚举多个解，或使用优化技术——能够显著提高获得最小系数方案的能力。

7.6 对规模的需求

翻转图搜索具有天然的并行性：每个运行器独立探索，运行器越多，对搜索空间的覆盖范围就越广。鉴于该空间规模极其庞大，最具挑战性格式的进展最终受限于可用的计算资源。

该框架的轻量化设计使其非常适合部署在大型计算平台上。在超级计算基础设施上运行数千乃至数百万个并发运行器，能够显著加快发现进程。即使仅使用几十个核心，本文所呈现的结果也发现了 79 个秩改进。扩展到机构集群或云平台，可能会取得多得多的成果。

开源版本欢迎大家开展这类工作，作者也希望社区中的其他人能在更大规模的系统上使用这些工具，拓展人们对快速矩阵乘法已知成果的边界。

8 结论

本文介绍了一个开源 C++ 框架，利用翻转图方法发现快速矩阵乘法方案。该框架支持多种系数环—— $\mathbb{Z}_2$、$\mathbb{Z}_3$ 和 $\mathbb{Z}_T$ ——并实现了固定维度和元维度搜索算子。通过对系数向量进行高效的位级编码以及采用 OpenMP 并行，该工具可在商用硬件上开展大规模探索。

借助这一框架，本文取得了三项主要成果：

- 在多个维度上，79 种矩阵乘法方案的秩得到改进。值得注意的是，研究发现了一种新的$4 \times 4 \times 10$ 方案，仅需 115 次乘法，对于这一特定规模达到了 $\omega \approx 2.80478$，效率高于递归应用 Strassen 算法。
- 通过有针对性地搜索秩相同的替代方案，重新发现了 93 种此前仅在有理数或整数上已知的方案，其系数为 $\mathbb{Z}_T$。这些方案现在可直接用于高效实现，无需乘以常数。
- 另有 68 个此前需要分数的方案，仅使用整数系数便重新发现了它们，从而消除了进行有理数运算的必要。

所有工具和发现的方案均以开源形式发布，以促进可复现研究和社区驱动的发现。该框架设计轻量——采用纯 C++ 编写，无外部依赖——因此任何拥有标准编译器的研究人员都可以参与搜索。

搜索过程中的观察表明，翻转潜力——U、V 或 W 中重合行的数量——与进一步改进的可能性相关。尽管这并非严格保证，但这种启发式方法已被证明有助于识别有前景的候选方案，也可能为未来的搜索策略提供参考。

未来工作仍有若干方向。将搜索扩展到分布式环境——使用数千乃至数百万个核心——有望大幅加快在128元素限制内最大格式上的进展。在广泛方案上对翻转潜力假说进行系统研究，将有助于判断其能否作为引导搜索投入的可靠启发式方法。深入探究系数环与最优秩之间的关系，或可揭示为何某些格式允许三元解，而另一些格式似乎需要分数。最后，改进引导提升策略，以便更好地探索可能提升的空间，或许能为在模环中发现的方案得到更紧凑的系数。

开源发布——框架见https://github.com/dronperminov/ternary_flip_graph，结果见 https://github.com/dronperminov/FastMatrixMultiplication——邀请研究界合作，共同推进这些方向，并发现越来越快的矩阵乘法算法。

参考文献

[1] Volker Strassen。高斯消元并非最优。*Numerische mathematik*，13(4):354–356，1969。[2] Marijn JH Heule、Manuel Kauers 和 Martina Seidl。在快速矩阵乘法中进行局部搜索。载于国际可满足性测试理论与应用会议，第 155–163 页。施普林格，2019。[3] Jason Yang。通过布尔可满足性问题排除具有对称性的低秩矩阵乘法张量分解。*arXiv* 预印本 *arXiv:2402.01011*，2024。[4] Alexey V Smirnov。矩阵乘法的双线性复杂度与实用算法。计算数学与数学物理，53(12):1781–1795，2013。[5] Alhussein Fawzi、Matej Balog、Aja Huang、Thomas Hubert、Bernardino Romera-Paredes、Mohammadamin Barekatain、Alexander Novikov、Francisco J R. Ruiz、Julian Schrittwieser、Grzegorz Swirszcz 等。利用强化学习发现更快的矩阵乘法算法。《自然》，610(7930):47–53，2022。[6] Arnaud Deza、Chang Liu、Pashootan Vaezipoor 和 Elias B Khalil。轻松实现快速矩阵乘法：一种约束编程方法。*arXiv* 预印本 *arXiv:2306.01097*，2023。[7] Manuel Kauers 和 Jakob Moosbauer。矩阵乘法的翻转图。载于*2023* 年国际符号与代数计算研讨会论文集，第 381–388 页，2023。[8] Yamato Arai、Yuma Ichikawa 和 Koji Hukushima。用于矩阵乘法的自适应翻转图算法。载于*2024* 年国际符号与代数计算研讨会论文集，第 292–298 页，2024。[9] Manuel Kauers 和 Isaac Wood。探索矩阵乘法的元翻转图。*arXiv* 预印本 *arXiv:2510.19787*，2025。[10] Andrew I Perminov。通过三元元翻转图实现快速矩阵乘法。*arXiv* 预印本 *arXiv:2511.20317*，2025。[11] Alexandre Sedoglavic。又一份快速矩阵乘法算法目录，2019。网址：*http://fmm. univ-lille. fr*，访问日期：*3* 月，2025。[12] Marijn JH Heule、Manuel Kauers 和 Martina Seidl。矩阵乘法的新方法 3×3。《符号计算杂志》，104:899–916，2021。[13] Alexander Novikov、Ngân V˜u、Marvin Eisenberger、Emilien Dupont、Po-Sen Huang、Adam Zsolt Wagner、Sergey Shirobokov、Borislav Kozlovskii、Francisco JR Ruiz、Abbas Mehrabian 等。AlphaEvolve：用于科学与算法发现的编程智能体。*arXiv* 预印本 *arXiv:2506.13131*，2025。[14] Kirill Khoruzhii、Patrick Gelß 和 Sebastian Pokutta。通过翻转图搜索实现更快的结构化矩阵乘法算法。*arXiv* 预印本 *arXiv:2511.10786*，2025。[15] Richard P Brent。矩阵乘法算法。技术报告，斯坦福大学，1970。[16] Yoav Moran、Oded Schwartz 和 Shuncheng Yuan。从复数快速矩阵乘法到有理数快速矩阵乘法。*arXiv* 预印本 *arXiv:2602.13171*，2026。

指数低于 Strassen 界的方案

表 4 列出了所有维度不超过 $16 \times 16 \times 16$ 、且指数$\omega = 3\log(r)/\log(mnp)$严格小于 Strassen 指数 $\log_2 7 \approx 2.807$ 的矩阵乘法格式。表中列出了每种格式的秩、系数环和相应指数。具有 115 次乘法的 $4 \times 4 \times 10$ 方案是在本研究中发现的；其余 28 种方案此前已知，列于此处以供参考。其中， $3 \times 4 \times 6$方案作为本研究的一部分，以 $\mathbb{Z}_T$ 系数重新发现，因此， $6 \times 8 \times 12$ 方案（由 $(3 \times 4 \times 6) \otimes (2 \times 2 \times 2)$ 得到）也可采用三元形式。

表 4：矩阵　乘法格式，其指数低于 Strassen 界（维度 $\leq 16 \times 16 \times 16$）。

格式 $m \times n \times p$	Rank	Ring	ω	格式 $m \times n \times p$	Rank	Ring	ω
$3 \times 3 \times 6$	40	$\mathbb{Q}$	2.77430	$8 \times 12 \times 12$	720	$\mathbb{Q}$	2.79998
$3 \times 4 \times 6$	54	$\mathbb{Z}_T$	2.79820	$8 \times 16 \times 16$	1248	$\mathbb{Q}$	2.80511
$3 \times 4 \times 7$	63	$\mathbb{Q}$	2.80522	$9 \times 9 \times 11$	576	$\mathbb{Q}$	2.80733
$4 \times 4 \times 4$	48	$\mathbb{Q}$	2.79248	$9 \times 9 \times 12$	600	$\mathbb{Q}$	2.78962
$4 \times 4 \times 9$	104	$\mathbb{Q}$	2.80356	$9 \times 10 \times 12$	684	$\mathbb{Q}$	2.80382
$4 \times 4 \times 10$	115	$\mathbb{Z}_T$	2.80479	$9 \times 12 \times 12$	800	$\mathbb{Q}$	2.79806
$6 \times 6 \times 11$	268	$\mathbb{Q}$	2.80418	$9 \times 12 \times 15$	1000	$\mathbb{Q}$	2.80416
$6 \times 6 \times 12$	280	$\mathbb{Q}$	2.78563	$12 \times 12 \times 12$	1040	$\mathbb{Q}$	2.79567
$6 \times 6 \times 14$	336	$\mathbb{Q}$	2.80452	$12 \times 12 \times 13$	1152	$\mathbb{Q}$	2.80669
$6 \times 6 \times 15$	360	$\mathbb{Q}$	2.80666	$12 \times 12 \times 15$	1280	$\mathbb{Q}$	2.79555
$6 \times 7 \times 12$	336	$\mathbb{Q}$	2.80452	$12 \times 12 \times 16$	1392	$\mathbb{Q}$	2.80475
$6 \times 8 \times 12$	378	$\mathbb{Z}_T$	2.80119	$12 \times 15 \times 15$	1600	$\mathbb{Q}$	2.80132
$6 \times 8 \times 14$	441	$\mathbb{Q}$	2.80590	$12 \times 16 \times 16$	1824	$\mathbb{Q}$	2.80525
$6 \times 9 \times 16$	556	$\mathbb{Q}$	2.80442	$16 \times 16 \times 16$	2304	$\mathbb{Q}$	2.79248
$8 \times 8 \times 8$	336	$\mathbb{Q}$	2.79744				