arXiv:2303.17352v1 [cs.DM] 30 Mar 2023

矩阵链的基本算法*

FRANCISCO L ´OPEZ†、Lars Karlsson†、和 Paolo Bientinesi†

摘要。 对于给定的 n 个矩阵的乘积，矩阵链乘法问题要求找到一种加括号方式，使算术运算次数最少。1973年，Godbole 提出了一种如今已成为经典的动态规划方法，其时间复杂度关于链长为三次。已知的最佳算法运行时间为线性对数级，而已知的最佳近似算法运行时间为线性，且近似因子小于二。所有解的共同点是，它们都从 C_{n-1} 种不同的加括号方式（卡特兰数 $n-1$）中选出一种最优加括号方式。我们研究了这些加括号方式，发现：(a) 所有数量呈指数级增长的加括号方式都有用，因为它们在输入空间的某个无限子集上是最优的；(b) 只有 $n+1$ 种加括号方式是必要的，因为它们在输入空间的某个无限子集上能任意优于次优方式；(c) 最佳必要加括号方式的成本绝不会超过最佳非必要加括号方式的两倍。通过对输入空间进行随机采样，我们进一步发现，在绝大多数输入中，必要加括号方式集合都包含一种最优加括号方式；而且，最佳必要加括号方式的表现平均而言远比最坏情况界限所暗示的更接近最优。研究结果对线性代数表达式编译器的开发有直接影响，尤其是在编译时矩阵规模未知的情况下。

关键词。 矩阵乘法、矩阵链、近似算法、线性代数编译器 - 编译器

数学主题分类代码。 68N20, 68Q25, 68W25

1. 引言。 经典的矩阵链乘法问题 旨在寻找将一系列 $n \geq 3$ 矩阵相乘的最低成本方式

$$M := M_1 M_2 \cdots M_n. \tag{1.1}$$

由于矩阵乘法满足结合律，(1.1) 中的 $n-1$ 次乘法可以按任意顺序执行。共有 $(n-1)!$ 种排序，其中一些是等价的，即它们执行完全相同的乘法集合，但顺序不同。例如，$(M_1M_2)(M_3M_4)$ 有两种等价的排序，因为乘法 $M_1 \times M_2$ 和 $M_3 \times M_4$ 彼此独立。每一类等价排序都由 (1.1) 的一种特定加括号方式 确定。下文中，我们使用术语 排序 来指代由加括号方式施加于乘法的部分 排序。排序会影响矩阵链求值的算术成本（从而影响执行时间）。因此，矩阵链乘法问题要求找到一种排序（加括号方式），使计算M的成本最小。

排列的数量为 $C_{n-1} = \frac{(2n-2)!}{n!(n-1)!}$，即第 $n-1$ 个卡特兰数，它以指数级速度增长 n。尽管如此，仍可使用教科书中的动态规划表述，在 $\Theta(n^3)$ 时间内精确求解该问题 [10], ; 而目前已知最快的算法运行时间为 $O(n \log n)$ [12, 13]。此外，还存在具有良好（乘法）保证的近似算法，其中一些算法的运行时间为 $O(n)$ [4, 5]。相关工作的概述请参见第 2 节。

在这项工作中，我们从不同的角度研究矩阵链乘法问题，探讨以下问题。

*提交至编辑部的日期。**经费支持：**

†瑞典于默奥大学计算机科学系（flopz@cs.umu.se, larsk@cs.umu.se, pauldj@cs.umu.se）

- 问题 1：所有排列都有用吗？如果某种排列对于至少一组矩阵规模而言，其成本严格低于任何其他排列的成本，我们就称该排列是有用的。直观地说，每种有用的排列都具有价值，因为它至少在该问题的某些实例中表现最佳。
- Q2：所有有用的排序都是必要排序吗？如果存在一种矩阵规模组合，使某种排列达到最优，且其成本与次优排列的成本之比可以任意大，那么这种排列就是必要排序。直观而言，每种必要算法都尤为重要，因为对于某些问题实例，它都能任意程度地优于次优算法。
- Q3：那些有用但非必要的排序（如果存在）的重要性有多大？如果存在某个实例，使某种排列达到最优，而次优排列的成本与之相差不超过一个常数因子，那么这种排列就是有用但非必要的。直观而言，如果能够证明这个因子有所降低，那么在不对性能造成重大负面影响的情况下，就可以不再考虑非必要排序。我们要探究有用但非必要的排序是否存在，以及如果舍弃它们需要付出多大代价。

我们作出如下贡献。我们在第 4 节回答 Q1，证明所有排序确实都有用，而且更重要的是，对于该问题的无穷多个实例，所有排序都各自有用。我们在第 5 节回答 Q2 和 Q3，证明 $n+1$ 特定的排序是必要排序，而其他所有排序都是非必要排序（尽管它们有用）。此外，我们证明，将排列的选择限制在必要排序集合内，会得到一种此前发表的近似算法，其近似因子为 2。最后，在第 6 节中，我们给出实验性证据，表明：i）对于大多数问题实例，一个必要排序是最优的；ii）对于大多数其他实例，移除所有非必要排序所造成的惩罚远低于最优成本两倍这一理论上界。

我们的结果直接影响线性代数表达式编译器（例如 MATLAB、Julia、Linnea [2] 以及其他编译器）的构建，尤其是在部分或全部矩阵规模在编译时未知这一常见场景下。矩阵链是所有线性代数表达式的一个子集。因此，如果编译器排除任何一个必要排序，那么对于无穷多种矩阵规模组合，生成代码与最优方案之间的差距都可能任意大。另一方面，如果编译器排除数量呈指数增长的所有非必要排序，只保留数量呈线性增长的必要排序，那么为矩阵链生成的代码的成本就可以保证不超过最优成本的两倍。此外，在大多数实例中，生成的代码要么是最优的，要么其性能会比该界限所暗示的更接近最优。换言之，对于规模未知且目标为矩阵链或包含矩阵链的任意表达式的线性代数表达式编译器，必须生成能够实现所有 矩阵链排序的代码，才能保证每个矩阵链实例都达到最优成本。然而，如果可以接受一定的性能下降，那么编译器只需为数量呈线性增长的必要算法生成代码。

2. 相关工作。 矩阵链在以往的研究中得到了广泛探讨。1973年，Godbole 发表了关于矩阵链乘法问题的首篇论文，提出了如今已成为经典的动态规划算法，可在 $\Theta(n^3)$ 时间内求得最优排序 [10]。1982年，Yao 对 Godbole 的算法进行了改进，提出了时间复杂度为 $O(n^2)$ 的算法。此后不久，Hu 和 Shing 发表了一种相当复杂的算法，可在 $O(n\log\ n)$

时间内求得最优排序 [12, 13]。后来有人发现，他们其中一个引理的证明有误，但在修正证明后，结论仍然成立 [18]。Hu 和 Shing 的研究还表明，矩阵链乘法问题可以转化为将一个有 $n+1$ 条边的凸多边形划分为 $n-1$ 个不相交三角形（这一事实在该领域的已发表文献中屡次被引用）。其他研究则侧重于针对具有特定性质的实例寻找最优排序，例如 Santoro 的工作 [17]（一种用于稀疏方阵的 $O(n)$ 算法），以及 Wang 等人的工作 [19]（一种 $O(n\log\ m)$ 算法，其中 m 是维度序列中的局部极小值数量）。

近似算法在 Godbole 发表动态规划算法 [10] 后不久便开始出现。1975年，Chandra 提出了一种近似因子为 2 的近似算法 [4]。（遗憾的是，这项工作除了以一份如今极难获取的技术报告形式发表外，再未正式发表。）此后不久，1978年，Chin 提出了一种具有 1.25 近似因子的 $O(n)$ 近似算法 [5]。1981年，Hu 和 Shing 提出了对 Chin 近似算法的重新表述，不过他们采用了其他方法（对一个 n 边形进行三角剖分），并将近似因子改进为 1.155 [11]。

多处理器和多核架构多年来变得越来越重要。因此，一些已发表的研究 [6, 16, 3] 着重于并行化算法，以便在多处理器架构上找到最优排序。另一些研究 [6, 7] 则专注于近似算法（通过并行化 Chin 以及 Hu 和 Shing 提出的算法），同样面向多处理器架构。还有一些研究关注在并行架构上实际求值矩阵链 [15]。较新的研究致力于优化矩阵链的求值，同时将处理器间通信和不同的矩阵乘法算法纳入考量 [20]。此外，值得一提的是，文献 [1]，旨在将具有特定性质（例如对角、三角和对称矩阵）及运算（例如求逆和转置）的矩阵链映射到基础线性代数子程序 [14, 9, 8] 内核。

所有这些工作的共同点在于，它们都旨在解决矩阵链乘法问题。在本文中，我们从不同的角度研究这一问题。我们不再针对特定实例寻找最优（或近似最优）的排列，而是考察在全部问题实例上，各种排列之间的相对表现。我们发现，所有排列都有其贡献，其中少数几种更是绝对不可或缺。

3. 预备知识。本节将介绍一些基本术语。表达式 (1.1) 是一个长度为 n 的矩阵链。令 $\mathcal{S}_n$ 表示排列（加括号方式）的集合。则 $|\mathcal{S}_n| = C_{n-1}$，即第 $(n-1)$ 个卡特兰数。对于所有 $i \in \{1, 2, \ldots, n\}$，矩阵 M_i 的大小为 $k_{i-1} \times k_i$。矩阵链乘法问题（问题 1）的一个实例由元组 $\boldsymbol{k} = (k_0, k_1, \ldots\ldots, k_n)$ 指定。

Definition 3.1（矩阵乘法的代价）。给定三个整数 a、b、c，将 $(M_{a+1} \cdots M_b)$ 与 $(M_{b+1} \cdots M_c)$ 相乘的成本由函数 $\phi_{\boldsymbol{k}}(a, b, c) := \phi(k_a, k_b, k_c) = k_a k_b k_c$ 给出。ϕ 关于其参数单调不减且具有旋转对称性，即 $\phi(p, q, r) = \phi(r, p, q)$。因此，$\phi_{\boldsymbol{k}}$ 的参数也具有旋转对称性。

$\mathcal{S}_n$ 中某种排列的成本由若干项 $n-1$ 相加得到 $\phi_{\boldsymbol{k}}(a, b, c)$，每一项都有其自身的三元组 $(a, b, c) \in \{0, 1, \ldots, n\}^3$。一种排列可以由其成本函数中出现的索引三元组集合 $\mathcal{A}$ 唯一表示 (a, b, c)。因此，成本

一个实例 $\boldsymbol{k}$ 上的排列 $\mathcal{A}$ 可以表示为

$$T(\mathcal{A}, \boldsymbol{k}) = \sum_{(a,b,c)\in\mathcal{A}} \phi_{\boldsymbol{k}}(a,b,c) = \sum_{(a,b,c)\in\mathcal{A}} \phi(k_a, k_b, k_c).$$

例如，排列 $\mathcal{A}$ 采用加括号方式 $M = ((M_1M_2)M_3)M_4$ 时，其成本函数为 $T(\mathcal{A}, \boldsymbol{k}) = k_0k_1k_2 + k_0k_2k_3 + k_0k_3k_4$。索引三元组表示为 $\mathcal{A} = \{(0, 1, 2), (0, 2, 3), (0, 3, 4)\}$。

矩阵链乘法问题可形式化如下。

PROBLEM 1 （矩阵链乘法问题）。给定一个矩阵链 (1.1)，其维数序列为 $\boldsymbol{k} = (k_0, k_1, \ldots, k_n)$，求一个排列 $\mathcal{A} \in \mathcal{S}_n$，使得$T(\mathcal{A}, \boldsymbol{k}) = \min_{\mathcal{B}\in\mathcal{S}_n} T(\mathcal{B}, \boldsymbol{k})$。

如果某个实例中一种排列的成本属于最低成本之列，我们就称其为（该实例下的）最优排列。此外，如果没有其他排列具有相同的成本，我们就称该排列为唯一最优排列。

DEFINITION 3.2（最优顺序与唯一最优顺序）。在实例 $\boldsymbol{k}$上，若排列 $\mathcal{A}$ 满足 $T(\mathcal{A}, \boldsymbol{k}) \leq \min_{\mathcal{B}\in\mathcal{S}_n\setminus\{\mathcal{A}\}} T(\mathcal{B}, \boldsymbol{k})$，则称该排列为最优排列。若不等式为严格不等式，则该排列是唯一最优的。

假设移除某些排列子集 $\mathcal{Q} \subset \mathcal{S}_n$，即不允许使用这些排列来计算矩阵链。实例 $\boldsymbol{k}$ 上的最优成本，在可选排列从 $\mathcal{S}_n$ 限制为 $\mathcal{S}_n \setminus \mathcal{Q}$时，将会增加或保持不变。成本的相对增幅量化了不允许使用 $\mathcal{Q}$中的排列所造成的性能惩罚。

DEFINITION 3.3（惩罚）。移除排列子集 $\mathcal{Q} \subset \mathcal{S}_n$所造成的惩罚，是最优成本的相对增幅（若有），由函数

$$P(\mathcal{Q}, \boldsymbol{k}) = \frac{\min_{\mathcal{B}\in\mathcal{S}_n\setminus\mathcal{Q}} T(\mathcal{B}, \boldsymbol{k})}{\min_{\mathcal{A}\in\mathcal{S}_n} T(\mathcal{A}, \boldsymbol{k})} - 1 \in [0, \infty).$$

分子是在所有剩余排列中取得的最优成本。分母是在所有排列中取得的最优成本。

当且仅当剩余排序集合 $\mathcal{S}_n \setminus \mathcal{Q}$ 包含一个最优排序时，惩罚才为零。直观地说，如果在实例 $\boldsymbol{k}$ 上 $\mathcal{Q}$ 的惩罚为 x%，那么当不允许使用 $\mathcal{Q}$ 中的排序时，最优成本将增加 x%。

本文所回答的问题围绕“有用顺序与关键顺序”这一概念展开，该概念已在引言中作了非正式介绍。

DEFINITION 3.4（有用顺序与关键顺序）。如果存在一个实例 $\boldsymbol{k}$，使得 $\mathcal{A}$ 是唯一最优的，则称排序 $\mathcal{A}$ 有用。如果对于任意 $r \in \mathbb{R}^+$，都存在一个实例 $\boldsymbol{k}$，使得 $P(\{\mathcal{A}\}, \boldsymbol{k}) > r$，则称排序 $\mathcal{A}$ 关键。换言之，可以选择一个实例，使得移除排序 $\mathcal{A}$的惩罚任意增大。

4. 每个排序都是有用的。长度为 n 的矩阵链实例具有 $n + 1$ 个参数，并且可以使用 $\mathcal{S}_n$中数量关于 n呈指数级增长的任意排序进行求值。排序数量之多表明，其中一些排序可能毫无用处，也就是说，它们不可能在任何实例上成为唯一最优排序。然而，事实并非如此；下文的定理 4.3表明，每个排序都是有用的（定义 3.4）。

为了证明这一结果，我们首先证明引理 4.1。

LEMMA 4.1. 在某个实例上唯一最优的排列，在无限多个实例上也唯一最优。

证明。设 $\boldsymbol{k}=(k_0,\ k_1,\ \ldots\ldots,\ k_n)$ 是一个实例，在该实例中 $\mathcal{A}$ 是唯一最优的，也就是说，对于所有 $\mathcal{B}\in\mathcal{S}_n\setminus\{\mathcal{A}\}$，

$$\frac{T(\mathcal{B},\boldsymbol{k})}{T(\mathcal{A},\boldsymbol{k})}>1.$$

元组 $\boldsymbol{k}$ 可以乘以一个正整数 α 进行缩放，使得

$$\boldsymbol{k}'=\alpha\boldsymbol{k}=(\alpha k_0,\alpha k_1,\ldots,\alpha k_n).$$

因此，$\phi_{\boldsymbol{k}'}(a,\ b,\ c)=(\alpha k_a)(\alpha k_b)(\alpha k_c)=\alpha^3\phi_{\boldsymbol{k}}(a,\ b,\ c)$。由此可知，对于所有 $\mathcal{B}\in\mathcal{S}_n\setminus\{\mathcal{A}\}$，

$$\frac{T(\mathcal{B},\boldsymbol{k}')}{T(\mathcal{A},\boldsymbol{k}')}=\frac{\alpha^3T(\mathcal{B},\boldsymbol{k})}{\alpha^3T(\mathcal{A},\boldsymbol{k})}=\frac{T(\mathcal{B},\boldsymbol{k})}{T(\mathcal{A},\boldsymbol{k})}>1.$$

换句话说，在 $\boldsymbol{k}'$ 上，$\mathcal{A}$ 也具有唯一最优性。由于 α 有无限多种取值，且每种取值都产生一个不同的实例，我们得出结论：存在无限多个实例，使得 $\mathcal{A}$ 具有唯一最优性。 □

我们需要 [5, 定理 1]（此处重新陈述并改写为引理 4.2），该定理给出了一个充分条件，确保某个特定的矩阵乘法出现在所有最优次序中。

LEMMA 4.2. 令 $m=\operatorname{argmin}_i k_i$ 为 $\boldsymbol{k}$ 中最小元素的索引。若对某个 $i\in\{0,1,\ldots,n\}$（取模（mod $n+1$））$k_i>k_{i-1}$、$k_i>k_{i+1}$，且条件

$$\frac{1}{k_i}<\frac{1}{k_{i-1}}+\frac{1}{k_{i+1}}-\frac{1}{k_m} \tag{4.1}$$

成立，则乘法 M_iM_{i+1} 必须出现在每一种最优排列中。

证明。见 [5, 定理 1]。 □

现在我们可以证明本节的主要结果了。

THEOREM 4.3. $\mathcal{S}_n$ 中的每一种排列，都在无限多个实例上是唯一最优的。

证明。$n\le 2$ 的情形显然，因此设 $n\ge 3$。考虑任意一个排列$\mathcal{A}\in\mathcal{S}_n$。我们将构造一个实例 $\boldsymbol{k}$，使得 $\mathcal{A}$ 是唯一最优的。我们分三步完成。步骤 1 中，我们构造一个递增的正整数序列。步骤 2 中，我们将该序列中的值赋给 $\boldsymbol{k}$。此时，我们已构造出一个 实例，使得 $\mathcal{A}$ 是唯一最优的。步骤 3 中，我们应用引理 4.1，证明每一种排列在无限 多个实例上都是唯一最优的。

第 1 步。构造一个正整数序列。令 $1=R_0<R_1<\ldots<R_{n-1}$ 为由以下各项组成的递增正有理数有限序列：

$$R_\ell=\frac{2^n+\ell-n-1}{2^n+\ell-n-2^\ell},$$

其中 $\ell\in\{0,1,\ldots,n-1\}$。对于固定的 n，该数列随 ℓ 增大，因为分子增大而分母减小。此外，由于 $R_0=1$ 和 $R_{n-1}=2$，该数列为正。如果 $\ell_1,\ \ell_2<\ell$，那么

$$\frac{1}{R_\ell}<\frac{1}{R_{\ell_1}}+\frac{1}{R_{\ell_2}}-\frac{1}{R_0}. \tag{4.2}$$

为验证这一点，请注意 $R_0 = 1$ 和 $R_{\ell_1},\ R_{\ell_2} \leq R_{\ell-1}$，因此

$$\frac{1}{R_\ell} < \frac{2^n + \ell - n - 2^\ell}{2^n + \ell - n - 2} = \frac{2}{R_{\ell-1}} - \frac{1}{R_0} \leq \frac{1}{R_{\ell_1}} + \frac{1}{R_{\ell_2}} - \frac{1}{R_0}.$$

令 $c = L_0 < L_1 < \cdots < L_{n-1}$（其中 $L_\ell = cR_\ell$）为通过将 R_ℓ 乘以 c得到的递增整数有限序列； c是序列 R_0、R_1、.....、R_{n-1}中所有分母的最小公倍数。注意，性质 (4.2) 对整数序列 L_0、 L_1、 ..... 、 L_{n-1} 也成立。

步骤 2。为 k_i 赋值，其中 $i \in \{0, 1, \ldots, n\}$。 为使 $\mathcal{A}$ 成为唯一最优，我们将序列 L_0、L_1、.....、L_{n-1} 中的整数分配给 $\boldsymbol{k}$ 的各个元素，使得根据引理 4.2， $\mathcal{A}$ 中的每次矩阵乘法都会出现在每种最优排列中。换言之， $\mathcal{A}$ 在 $\boldsymbol{k}$上必须是唯一最优的。

识别由 $\mathcal{A}$ 确定并关联在一起的矩阵乘法 M_iM_{i+1}。 ˆ 设定 $k_i = L_{n-1}$。将大小为 $k_{i-1} \times k_{i+1}$ 的 M_iM_{i+1} 替换为 $M_i := M_iM_{i+1}$，得到长度为 $n-1$ 的约化链。重复此约化过程，直到所有乘法都被消除。

在第一次迭代中，将 L_{n-1} 赋给某个 k_{i_1}。在第二次迭代中，将 L_{n-2} 赋给某个 k_{i_2}。最后，在最后一次迭代中，将 L_1 赋给某个 $k_{i_{n-1}}$。此时，所有维度 $k_1, k_2, \ldots, k_{n-1}$ 都已从序列 $L_1, \ldots, L_{n-1}$ 中分配到一个整数。最后，设定 $k_0, k_n = L_0$。由于 L 的元素按从大到小的顺序分配，因此对于 $\mathcal{A}$ 中每次矩阵乘法涉及的任意三个维度 k_{i-1}, k_i, k_{i+1}，不等式 (4.2) 都成立。因此，根据引理 4.2， $\mathcal{A}$ 中的每次乘法都必须出现在每一种最优排序中。由于这样的排序只有一种， $\mathcal{A}$ 在 $\boldsymbol{k}$ 上必定是唯一最优的。

步骤 3：从一个实例推广到无穷多个实例。 由引理 4.1 可知， $\mathcal{A}$ 在无穷多个实例中是唯一最优的。 □

5. 必要排序寥寥无几。 本节将通过确定长度为 n 的矩阵链的必要排序集合，说明只有少数排序是必要的。

令 $\mathcal{E}_n$ 表示如下形式的排序集合

$$(M_1(M_2 \cdots (M_{h-1}M_h) \cdots)) \,\big|\, ((\cdots (M_{h+1}M_{h+2}) \cdots M_{n-1})M_n),$$

其中 $h \in \{0, 1, \ldots, n\}$。对于 $h = 0$，该排序是从左到右的；对于 $h = n$，则是从右到左的。由于关联模式从维度 k_h 处分界线向两个方向展开，我们称这些排序从维度 k_h展开 。令 $\mathcal{E}_{n,h} \in \mathcal{E}_n$ 表示从 k_h 处展开的排序。这类排序共有 $n+1$ 个。对于 $n \geq 4$，它们彼此不同；然而，由于对称性，对于 $n \leq 3$，只有 $n-1$ 个不同的排序。更准确地说，

$$|\mathcal{E}_n| = \begin{cases} n-1 & \text{if } n \leq 3, \\ n+1 & \text{otherwise.} \end{cases}$$

我们将证明， $\mathcal{E}_n$ 中的所有排列都是必要的（定义 3.4），且不存在其他必要排列。因此，对于长度为 n 的矩阵链， $\mathcal{E}_n$ 就是 必要排序的集合。

下面的引理表明， $\mathcal{E}_n$ 中的所有排列都是必要的。

LEMMA 5.1. 对于任意 $r \in \mathbb{R}^+$ 和 $\mathcal{E}_{n,\ h} \in \mathcal{E}_n$，都存在无穷多个实例$\boldsymbol{k}$，使得 $P(\{\mathcal{E}_{n,h}\}, \boldsymbol{k}) > r$。

证明。考虑形如 $\boldsymbol{k} = (\alpha, \ldots, \alpha, \mathbf{1}, \alpha, \ldots, \alpha)$ 的实例，其中 α 是一个大于一的正整数，且 $k_h = 1$。$\mathcal{E}_{n,h}$ 在 $\boldsymbol{k}$ 上的成本为

$$T(\mathcal{E}_{n,h}, \boldsymbol{k}) = \sum_{1}^{n-1} 1 \cdot \alpha \cdot \alpha = (n-1)\alpha^2.$$

只有 $\mathcal{E}_{n,\ h}$ 在成本函数的每个 $n-1$ 项中都包含 k_h。任何其他排列的成本函数中，至少有一项、至多有 $n-2$ 项包含 k_h。由于 $\alpha > 1$，要使成本最小化，就应最大化包含 k_h 的项数。因此，令 $\mathcal{B} \neq \mathcal{E}_{n,\ h}$ 为一个排列，其中 k_h 出现在成本函数的 $n-2$ 项中。它在 $\boldsymbol{k}$ 上的成本为

$$T(\mathcal{B}, \boldsymbol{k}) = \alpha \cdot \alpha \cdot \alpha + \sum_{1}^{n-2} 1 \cdot \alpha \cdot \alpha = \alpha^3 + (n-2)\alpha^2.$$

这样的排列总是存在。移除 $\mathcal{E}_{n,\ h}$ 对 $\boldsymbol{k}$ 的惩罚为

$$\begin{aligned} P(\{\mathcal{E}_{n,h}\}, \boldsymbol{k}) &= \frac{\min_{\mathcal{A} \in \mathcal{S}_n \setminus \{\mathcal{E}_{n,h}\}} T(\mathcal{A}, \boldsymbol{k})}{T(\mathcal{E}_{n,h}, \boldsymbol{k})} - 1 = \frac{T(\mathcal{B}, \boldsymbol{k})}{T(\mathcal{E}_{n,h}, \boldsymbol{k})} - 1 \\ &= \frac{\alpha^3 + (n-2)\alpha^2}{(n-1)\alpha^2} - 1 = \frac{\alpha}{n-1} + \frac{n-2}{n-1} - 1 = \Theta(\alpha). \end{aligned}$$

由于惩罚会随着 $\alpha \to \infty$ 增大而无界增长，对于任意 $r \in \mathbb{R}^+$，我们都可以选择足够大的 α，使惩罚大于 r。根据引理 4.1，存在无穷多个具有相同惩罚的实例。

□

引理 5.1 表明，$\mathcal{E}_n$ 中的每个排列都是必要的。然而，这并不意味着其他所有排列都不是必要的。下面的引理表明，排列 $\mathcal{E}_{n,\ h}$ 的成本严格小于最优成本的两倍，其中 h 的选择使得 k_h 是 $\boldsymbol{k}$ 中最小的元素。该结果出自 Chandra [4, 定理 3]。我们改进了证明的表述。

LEMMA 5.2. 设 $\mathcal{A}$ 是实例 $\boldsymbol{k}$ 上的一个最优排列，且 $m = \operatorname{argmin}_i k_i$，则 $T(\mathcal{E}_{n,m}, \boldsymbol{k}) < 2T(\mathcal{A}, \boldsymbol{k})$。

证明。</style}情形 是平凡的，因此假设 。令 :(, k)。我们证明 $T(\mathcal{E}_{n,m}, \boldsymbol{k}) < D \leq 2T_{\text{opt}}$，其中 D 是一个经过适当选择的和

$$D := \sum_{i \in \{0,1,\ldots,n\}} \phi_{\boldsymbol{k}}(i-1, i, m).$$

这里以及证明的其余部分中，$\phi_{\boldsymbol{k}}$ 的所有参数均按模 $n+1$ 计算。例如，$i = 0$ 对应的 D 中的项为 $\phi_{\boldsymbol{k}}(n, 0, m)$。我们先证明 $D \leq 2T_{\text{opt}}$（步骤 1），然后证明 $T(\mathcal{E}_{n,m}, \boldsymbol{k}) < D$（步骤 2）。我们先作一个观察。

观察 1. 每个矩阵 M_i 都恰好参与一次乘法。如果 M_i 作为左操作数出现（对于某个 $X = \prod_{j=i+1}^{z} M_j$，$M_i X$），那么成本中就会出现项 $\phi_{\boldsymbol{k}}(i-1,\ i,\ z)$。在这种情况下，维度 k_i 会被消去，因此其他项都不能包含参数 i。反之，如果 M_i 作为右操作数出现（对于某个 $X = \prod_{j=z}^{i-1} M_j$，$X M_i$），那么成本中就会出现项 $\phi_{\boldsymbol{k}}(z,\ i-1,\ i)$。在这种情况下，维度 k_{i-1} 会被消去，因此其他项都不能包含参数 $i-1$。总之，对于任意排列，其成本函数中恰好有一项同时包含 $i-1$ 和 i（对于 $i \in \{1, 2, \ldots, n\}$）。

任意排列中的最后一次乘法形式为 $(M_1 \cdots M_z)(M_{z+1} \cdots M_n)$，其中 $1 \leq z \leq n-1$。这次乘法会在成本中增加项 $\phi_{\boldsymbol{k}}(0,\ z,\ n) = \phi_{\boldsymbol{k}}(n,\ 0,\ z)$（注意参数中的模运算和旋转）。因此，对于每个$i \in \{0, 1, \ldots, n\}$，$T_{\text{opt}}$ 恰好有一项形如 $\phi_{\boldsymbol{k}}(i-1,\ i,\ z)$。

步骤 1. $D \leq 2T_{\text{opt}}$. </style>证明基于将 D 中的每一项与 $2T_{\text{opt}}$ 中的某一项进行匹配，并使 D 中的每一项都小于或等于其匹配项。由此可得 $D \leq 2T_{\text{opt}}$。

设 σ 是这样一个函数：它将每个 $i \in \{0, 1, \ldots, n\}$ 中形如 $\phi_{\boldsymbol{k}}(i-1, i, m)$ 的项映射到 T_{opt} 中形如 $\phi_{\boldsymbol{k}}(i-1, i, z)$ 的项。根据观察 1，该函数是良定义的，因为 T_{opt} 中指定形式的项有且仅有一个。

考虑 T_{opt} 中的任意项 t。我们证明，关于 σ 如何将 D 中的项映射到 T_{opt} 中的项，只有三种情况：

- σ 将 D 中的无项映射到 t。当且仅当 t中不存在一对连续指标时，才会出现这种情况。也就是说，若 $t = \phi_{\boldsymbol{k}}(a, b, c)$，则 $a+1 \neq b$(模 $n+1$)、$b+1 \neq c$ (模 $n+1$)，且 $c+1 \neq a$ (模 $n+1$)。
- σ 将 D 中的一项映射到 t。当且仅当 t中恰好有一对连续指标时，才会出现这种情况。
- σ 将 D 中的两项映射到 t。当且仅当对于某个 i（模 $n+1$），t 的形式为$\phi_{\boldsymbol{k}}(i-1, i, i+1)$时，才会出现这种情况。

当 $n \geq 3$ 时，函数 σ 无法将 D 中的超过两项映射到 t，因为同余方程组 $a+1 \equiv b$（模 $n+1$）、$b+1 \equiv c$（模 $n+1$）和 $c+1 \equiv a$（模 $n+1$）在 $n \geq 3$时无解。（若 $n = 2$，则 $a, b, c = 0, 1, 2$满足这些条件，并且有三对连续指标：(0,1)、(1, 2)和(0,2)。）因此，σ 至多将 D 中的两项映射到 T_{opt}中的每一项。由于 σ定义良好，D 中的每一项都会映射到 T_{opt}中的某一项，但 T_{opt} 中的每一项不一定都被 D中的某一项映射到。

由于 $2T_{\text{opt}} = T_{\text{opt}} + T_{\text{opt}}$ 中每个项都有两个副本，并且 σ 将 D 中的项映射到 T_{opt} 中同一项的数量不超过两个，因此 D 中的每一项都可以与 $2T_{\text{opt}}$ 中的一项匹配，且后者的每一项至多匹配一次。由此建立的每一对都具有如下形式：$\phi_{\boldsymbol{k}}(i-1,\ i,\ m)$ 属于 D，而 $\phi_{\boldsymbol{k}}(i-1,\ i,\ z)$ 属于 T_{opt}。由于k_m 是 $\boldsymbol{k}$ 中最小的元素，且函数 ϕ 单调不减（回顾定义 3.1），因此 $\phi_{\boldsymbol{k}}(i-1,\ i,\ m) \leq \phi_{\boldsymbol{k}}(i-1,\ i,\ z)$。因此，$D \leq 2T_{\text{opt}}$。

步骤 2. $T(\mathcal{E}_{n,\ m},\ \boldsymbol{k}) < D$。$\mathcal{E}_{n,\ m}$ 在 $\boldsymbol{k}$ 上的成本为

$$T(\mathcal{E}_{n,m}, \boldsymbol{k}) = \sum_{i \in \{0,1,\ldots,n\} \setminus \{m, m+1\}} \phi_{\boldsymbol{k}}(i-1, i, m).$$

$T(\mathcal{E}_{n,\ m},\ \boldsymbol{k})$ 中的每一项也都出现在 D 中。由于 D 的项数严格更多，$T(\mathcal{E}_{n,\ m},\ \boldsymbol{k}) < D$。

结合步骤 1 和 2，可得 $T(\mathcal{E}_{n,\ m},\ \boldsymbol{k}) < 2T_{\text{opt}}$。 □

请注意，只要 ϕ 是单调不减的，并且其参数具有旋转对称性，引理 5.2 就成立，因为证明基于对不同成本表达式中的项进行匹配，而不是基于对 ϕ的实际求值。

图 1 展示了在 $m = 3$ 情况下，引理 5.2 证明中对项进行匹配的两层结构，以及最优排序 $((M_1M_2)((M_3(M_4M_5))M_6))(M_7M_8)$。$2T_{\text{opt}}$ 的项位于最上面的两行，D 的项位于中间一行，而 $T(\mathcal{E}_{n,\ m},\ \boldsymbol{k})$ 的项则显示在最下方。D 的每一项都已通过 σ 与 $2T_{\text{opt}}$ 中一个相等或更大的项进行了一对一匹配。类似地，$T(\mathcal{E}_{n,\ m},\ \boldsymbol{k})$ 的每一项都已与 D 中对应的相同项匹配。

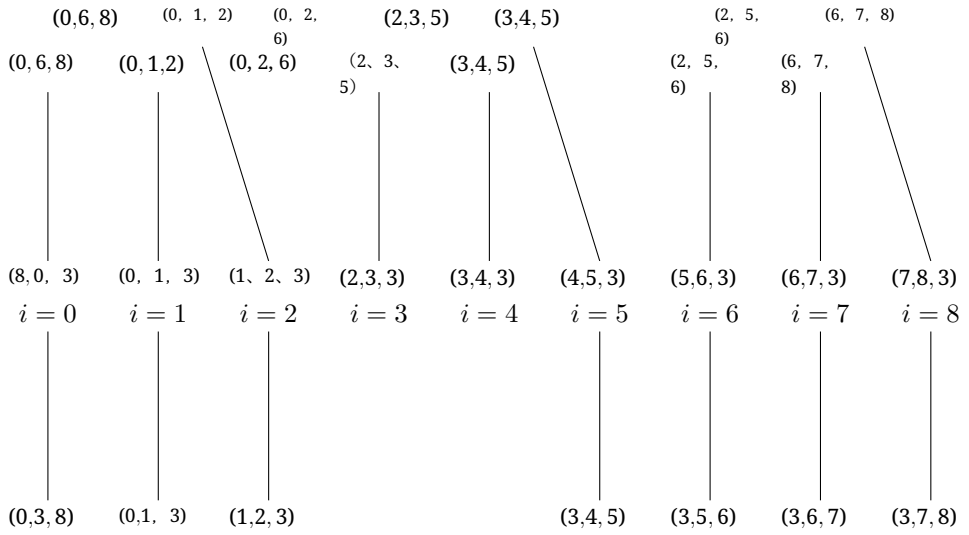


FIG. 1. *Example of term matching between* $2T_{\text{opt}}$ *(above),* D *(middle), and* $T(\mathcal{E}_{n,\,m},\ \boldsymbol{k})$ *(bottom).*

下面的定理表明，移除 $\mathcal{E}_n$ 中每个非排序所造成的惩罚不超过 100%。换言之，只有 $\mathcal{E}_n$ 中的排序是必不可少的。尽管根据定理 4.3，所有其他排序都有用，但即使在最坏情况下，移除它们造成的惩罚也很小（最差也不会超过最优成本的两倍）。

THEOREM 5.3. 移除所有不属于 $\mathcal{E}_n$ 的排列所产生的最大惩罚低于 100%，即 $\max_{\boldsymbol{k}} P(\mathcal{S}_n \setminus \mathcal{E}_n, \boldsymbol{k}) < 100\%$。

证明。令 $m = \operatorname{argmin}_i k_i$。则

$$P(\mathcal{S}_n \setminus \mathcal{E}_n, \boldsymbol{k}) = \frac{\min_{\mathcal{B}\in\mathcal{S}_n\setminus(\mathcal{S}_n\setminus\mathcal{E}_n)} T(\mathcal{B},\boldsymbol{k})}{\min_{\mathcal{A}\in\mathcal{S}_n} T(\mathcal{A},\boldsymbol{k})} - 1 = \frac{\min_{\mathcal{B}\in\mathcal{E}_n} T(\mathcal{B},\boldsymbol{k})}{\min_{\mathcal{A}\in\mathcal{S}_n} T(\mathcal{A},\boldsymbol{k})} - 1$$
$$\leq \frac{T(\mathcal{E}_{n,m},\boldsymbol{k})}{\min_{\mathcal{A}\in\mathcal{S}_n} T(\mathcal{A},\boldsymbol{k})} - 1 < 2 - 1 = 100\%.$$

最后一个不等式由引理 5.2 得出。 □

以下定理确立了我们的主要结果。

THEOREM 5.4. $\mathcal{E}_n$ 中的所有排序都是必要的，除此之外没有其他排序是必要的。

证明。根据引理 5.1，$\mathcal{E}_n$ 中的所有排序都是必要的。根据定理 5.3，$\mathcal{S}_n \setminus \mathcal{E}_n$ 中没有排序是必要的。 □

6. 实验。前两节给出的理论结果确立了 $\mathcal{E}_n$ 是必要排序集合，并且移除所有其他排序所导致的惩罚上限为 100%。然而，理论分析并未说明必要算法成为最优算法的频率，也没有揭示移除所有非必要排序所造成的平均惩罚。为了解答这些定量问题，我们进行了以下实验。

对于所有 $n \in \{2, 3, \ldots, 9\}$，我们随机抽取了 100 万个（有放回）长度为 n 的矩阵链乘法问题实例。每个维度k_i（ $i = 0$，1，.....， n）均从 1 到 1000 的均匀分布中随机选取。对于每个实例 $\boldsymbol{k}$，我们记录了移除所有非必要排序所造成的惩罚，即 $P(\mathcal{S}_n \setminus \mathcal{E}_n,\ \boldsymbol{k})$。令 $\mathcal{P}_n$ 表示样本集合（$|\mathcal{P}_n| = 10^6$），令$\mathcal{P}_n^+ \subseteq \mathcal{P}_n$ 表示惩罚非零的样本子集。

n	2	3	4	5	6	7	8	9
$\lvert\mathcal{S}_n\rvert$	1	2	5	14	42	132	429	1430
$\lvert\mathcal{E}_n\rvert$	1	2	5	6	7	8	9	10
$\lvert\mathcal{P}_n^+\rvert/\lvert\mathcal{P}_n\rvert$	0%	0%	0%	7.9%	10.1%	11.2%	11.8%	12.2%

TABLE 1

Fraction of sampled instances where removing every non-essential ordering resulted in a non-zero penalty.

表1最后一行显示，移除所有非必要排序所产生的惩罚非零的样本比例。对于其他所有样本，惩罚均为零，这意味着一个或多个必要排序是最优的。这些结果表明，对于大多数实例（当 $n \leq 9$ 时为 $> 87\%$），某个必要排序是最优的。也就是说，这些排序不仅是必要的，而且其中一个（不一定总是同一个）几乎总是最优的。惩罚非零的样本比例会随 n 增加，并且似乎渐近有界于某个较小的常数。然而，我们尚无法证明随着 $n \to \infty$ 增大，情况确实如此。

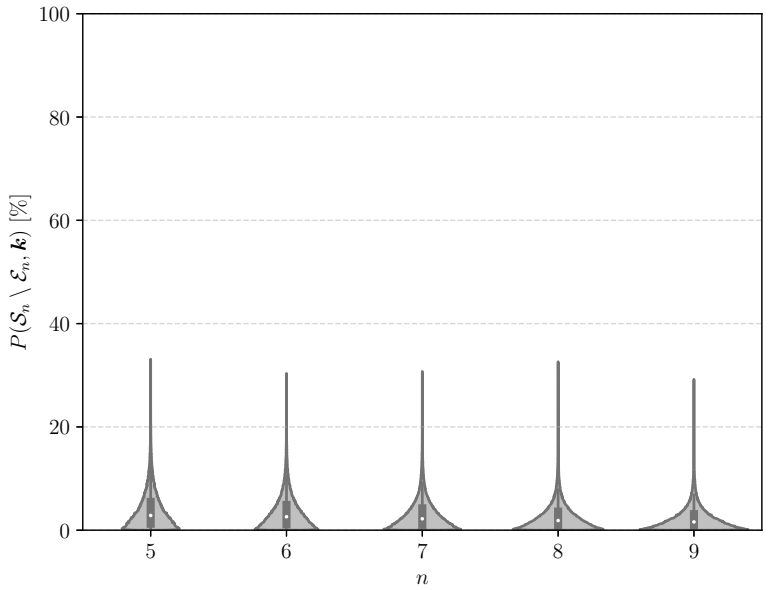


FIG. 2。 *Estimated distributions of the penalty when every non-essential ordering is removed for uniform random sampling and* $n \in \{5, 6, \ldots, 9\}$. *The cases with a penalty of* 0 *are not included.* %

图 2 展示了受限惩罚的样本分布小提琴图

对于 $n \in \{5, 6, \ldots, 9\}$，范围从 $\mathcal{P}_n^+$ 到 $\mathcal{P}_n^+$（对于 $n \leq 4$，每种排列都是必要的，因此 $\mathcal{P}_n^+$ 为空）。定理 5.3 表明，移除所有非必要排序所产生的惩罚上限为 100%；但正如这些结果所示，在实践中，惩罚往往远低于 0%。图 2 中每个 n 的平均（非零）惩罚为 $< 4\%$。在 $\mathcal{P}_n^+$ 的样本中，$> 99\%$ 的样本惩罚为 $< 20\%$（每个 n）。在少数惩罚非零的情况（约 10%）中，惩罚几乎总是 $< 20\%$，远优于理论上限 100%（定理 5.3）。如果考虑所有样本（即用 $\mathcal{P}_n$ 而非 $\mathcal{P}_n^+$），那么 $> 99\%$ 的样本中，惩罚为 $< 10\%$。

7. 结论。 本文研究了将一条由 n 个矩阵组成的矩阵链相乘的问题，并考察了这些 C_{n-1} 排列在输入空间上的表现。我们提出了一系列相互关联的问题，其答案对于这样一种常见场景下的线性代数编译器具有重要意义：矩阵的大小在编译时是未知的。首先，我们证明了指数数量级的所有排列中的每一个都是有用的，即在无限多个实例上达到唯一最优。因此，旨在为所有实例生成最优代码的线性代数编译器不能舍弃指数数量级的任何一种排列。

假设无法进行这样的生成，或者可以接受性能上的某种损失，则可以且必须减少排列的集合。所选的精简排列集合应包含这样一些排列：如果存在的话，它们对于某个实例能够达到唯一最优，并且比任何其他排列便宜任意多。我们证明，只有 $n + 1$ 排列属于这一类别，即属于必要排列；同时，舍弃所有非必要排列所带来的惩罚上界为 100%。因此，如果谨慎选择排列子集，保留 $n+1$ 排列并舍弃其他所有指数数量级的排列，编译器便可以生成这样的代码：其执行的操作数永远不会超过最优操作数的两倍。

此外，我们还提供了实验性证据表明，对于 $n \leq 9$，在大多数实例（$> 87\%$）中，某种必要排序是最优的。此外，在少数不存在最优必要排序的实例中，实际惩罚远比其 100% 的理论上限接近 0%，平均惩罚为 4%。这些结果表明，将排序集合从 C_{n-1} 缩减为数量仅线性增长的必要排序，并从中选择一种执行顺序，对于大多数操作数大小组合而言都是最优的，或者其成本仅略高于最优成本。

总之，尽管矩阵链的每种排序在某个实例中都是唯一最优的，但线性代数表达式的编译器可以将代码生成限制为仅采用 $n + 1$ 必要排序，并保证求值几乎总是以接近最优的成本进行。

致谢。

参考文献

[1] H. Barthels, M. Copik, and P. Bientinesi, *The generalized matrix chain algorithm*, 载于《2018年代码生成与优化国际研讨会论文集》, ACM, 2018年, 第138–148页。[2] H. Barthels, C. Psarras, and P. Bientinesi, *Linnea: Automatic generation of efficientlinear algebra programs*, 《ACM数学软件汇刊》(TOMS), 47 (2021), 第1–26页。[3] P. G. Bradford, G. J. Rawlins, and G. E. Shannon, *Efficient matrix chain ordering in polylog time* 27 1998 466 490 , 《SIAM计算学报》, ( ), 第–页。[4] A. Chandra, *Computing matrix chain products in near-optimal time*, 技术报告RC-5625

(#24393)，IBM Thomas J. Watson Research Center，美国纽约州约克敦高地，邮政信箱218号，1975年。[5] F. Y. Chin, *An o(n) algorithm for determining a near-optimal computation order of matrix chain products* 21 1978 544 549 ，《ACM通信》，( )，第–页。[6] A. Czumaj, *Parallel algorithm for the matrix chain product and the optimal triangulation problems* 1993 294 294 ，《计算机科学讲义》，( )，第–页。[7] A. Czumaj, *Very fast approximation of the matrix chain product problem*，《算法学报》，21 (1996)，第71–79页。[8] J. J. Dongarra, J. Du Croz, S. Hammarling, and I. S. Duff, *A set of level 3 basic linearalgebra subprograms*，《ACM数学软件汇刊》(TOMS)，16 (1990)，第1–17页。[9] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, *A proposal for an extended set of fortran basic linear algebra subprograms* 20 1985 2 ，《ACM Signum通讯》，( )，第–18页。[10] S. S. Godbole, *On efficient computation of matrix chain products*，《IEEE计算机汇刊》，100 (1973)，第864–866页。[11] T. Hu and M. Shing, *An o (n) algorithm to find a near-optimum partition of a convex polygon*，《算法学报》，2 (1981)，第122–138页。[12] T. Hu and M. Shing, *Computation of matrix chain products. part i*，《SIAM计算学报》，11 (1982)，第362–373页。[13] T. C. Hu and M. Shing, *Computation of matrix chain products. part ii*，《SIAM计算学报》，13 (1984)，第228–251页。[14] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, *Basic linear algebra subprograms for fortran usage*，《ACM数学软件汇刊》(TOMS)，5 (1979)，第308–323页。[15] H. Lee, J. Kim, S. J. Hong, and S. Lee, *Processor allocation and task scheduling of matrix chain products on parallel systems*，《IEEE并行与分布式系统汇刊》，14 (2003)，第394–407页。[16] P. Ramanan, *An efficient parallel algorithm for the matrix-chain-product problem*，《SIAM计算学报》，25 (1996)，第874–893页。[17] N. Santoro, *Chain mulitplication of matrices of approximately or exactly the same size*，《ACM通信》，27 (1984)，第152–156页。[18] O. Schwartz and E. Weiss, *Revisiting "computation of matrix chain products"*，《SIAM计算学报》，48 (2019)，第1481–1486页。[19] X. Wang, D. Zhu, and J. Tian, *Efficient computation of matrix chain*，载于《2013年第8届国际计算机科学与教育会议》，IEEE，2013年，第703–707页。[20] E. Weiss and O. Schwartz, *Computation of matrix chain products on parallel machines*，载于《2019年IEEE国际并行与分布式处理研讨会》(IPDPS)，IEEE，2019年，第491–500页。