arXiv:2603.16327v1 [math.AT] 17 Mar 2026

持久同调计算的最坏情况示例

UZAY ÇETIN 和 ERGÜN YALÇIN

摘要。我们构造了计算持久同调的标准约化算法的最坏情况示例。我们的构造与 Morozov 引入的最坏情况示例相似，但我们用由基三角形和鳍三角形组成的条带取代了单三角形排列。这种结构使我们能够给出构造这些示例的显式算法，并进行实验以比较不同版本的约化算法的运行时间。我们进一步证明，经过适当的边和三角形细分后，这些条带示例仍然是最坏情况，并且可以实现为过滤图的团复形，因此，对于一系列尺度参数，它们也是有限点云的 Vietoris–Rips 复形。

1. 引言

随着人口增长以及互联网和数字技术的发展，数据已成为我们生活中最重要的组成部分之一。统计方法和聚类技术等经典数学方法有助于数据分析，而拓扑数据分析（TDA）在过去20年中已成为一种重要的数据分析新工具 [2]。目前，TDA 已应用于许多领域，包括基因组学、机器人学、医学影像、癌症研究、传染病传播、金融网络和自然语言分析（例如，参见 [3], [6], [13], 和 [5]）。

拓扑数据分析利用代数拓扑中的方法来分析数据。TDA 中的主要方法之一是持久同调，它由 Edelsbruner、Letscher、Zomorodin、Carlsson 等人开发（见 [8], [15]）。在大多数应用中，通过持久同调进行的 TDA 以嵌入于 $\mathbb{R}^m$ 中的有限度量空间为输入，并生成持久图作为输出。持久同调之所以如此有效，原因之一是它可以通过一种称为约化算法的简单算法来计算。许多计算机软件包采用不同版本的约化算法，用于计算过滤单纯复形的持久图（见 [12]，其中比较了不同软件包的运行时间表现）。

在 [10], 中，Morozov 构造了一个过滤单纯复形族，为约化算法提供了最坏情况示例，其运行时间为 $\Omega(N^3)$，其中 N 表示单形的数量。本文的主要目的是给出一组与 Morozov 的示例类似的新最坏情况示例，但构造方法是将基三角形和鳍三角形逐步添加到一条三角形条带中，而不是将三角形添加到单个三角形内部。我们将这组新的最坏情况示例称为条带示例。由于其条带结构，更容易给出构造它们的显式算法，也更容易开展实验，比较不同版本约化算法的运行时间。

在定理 3.2 中，我们证明了约化算法在条带最坏情况示例上的运行时间仍为 $\Omega(N^3)$，其中 N 是单形的数量。在附录 A 中，我们给出

日期：2026年3月18日。2020 数学主题分类。55N31（主要），68W40、68R10（次要）。关键词和短语。拓扑数据分析、持久同调、约化算法、算法复杂度、团复形。

了一段简单的伪代码，用于构造任意数量三角形的条带示例。在第4节中，我们使用这段代码进行计算实验，并报告比较约化算法不同版本运行时间的结果，例如标准约化算法 [8], 带扭转的约化算法 [4], 以及约化算法的前瞻变体 [11]。

在第 5 节中，我们证明，将条带示例中的一些三角形和边细分后，这些示例可以实现为过滤图的团复形。随后，我们证明，这些过滤复形可以实现为与某个欧几里得空间中的有限数据云相关联的过滤图。因此，我们得出结论：这些修改后的条带示例可以实现为某个实数序列对应的有限点云的 Vietoris–Rips 复形。

2. 矩阵约化算法

关于持久同调的基本定义，请参见 [6], [13], 和 [5] 。本文中，我们仅给出与过滤单纯复形相关的定义，以便说明约化算法。

定义 2.1. 一个单纯复形 K 由一对 (V, S) 构成，其中 V 是 K的顶点集，而 S 是 V的子集构成的集合，这些子集称为 K的单形，并满足以下性质：

(1) 对所有 $v \in V$, $\{v\} \in S$，(2) 若
$\sigma \in S$ 且 $\tau \subseteq \sigma$，则 $\tau \in S$。

若单形 σ 属于 K，则称其为 k-单形 或 k-维单形，如果 $|\sigma| = k+1$。对于 K中的单形 σ，称子集 $\tau \subseteq \sigma$ 为 σ的面 。如果 $\dim \tau = \dim \sigma - 1$，则称 τ为 σ的棱面 。如果单纯复形 K 的顶点集 V 是有限集，则称该单纯复形是有限的。本文中，我们假设所有单纯复形都是有限的。 K 的维数是满足以下条件的最大d： K 含有一个 d维单形。如果单纯复形 $K' = (V', S')$ 是单纯复形 $K = (V, S)$ 的子复形，则称之为子复形。在这种情况下，我们记作 $K' \subseteq K$。

定义 2.2. 一列嵌套的单纯复形

$$\emptyset \subseteq K_1 \subseteq K_2 \subseteq \cdots \subseteq K_m = K$$

称为 K 的过滤。带有过滤的单纯复形称为过滤单纯复形。

单纯复形 $K = (V,\ S)$ 上的过滤函数是一个函数 $f\colon S \to \mathbb{R}$，满足当 $\tau \subseteq \sigma$ 时，$f(\tau) \leq f(\sigma)$。对于每个过滤复形 K，我们可以定义一个过滤函数$f\colon K \to \mathbb{Z}^+ = \{1, 2, \dots\}$，使得对于每个单形 σ，$f(\sigma)$ 是满足$\sigma \in K_i$的最小 $i \in \mathbb{Z}$。反之，给定复形 K上的过滤函数 f，我们可以选取有限多个实数 $r_1, \dots, r_m$，并通过令 $K_i \subseteq K$ 为由满足 $f(\sigma) \leq r_i$ 的单形 $\sigma \in K$ 构成的子复形，来定义 K 上的过滤。

设 $K = (V,\ S)$ 是一个带有关联过滤函数 $f\colon K \to \mathbb{Z}^+$ 的过滤单纯复形 $\emptyset \subseteq K_1 \subseteq K_2 \subseteq \cdots \subseteq K_m = K$。考虑 $\mathbb{F}_2$-系数下 K 的链复形 $C_*(K)$。我们可以将链复形 $C_*(K)$ 视为分次 $\mathbb{F}_2$-向量空间 $C_*(K) = \bigoplus_{p=0}^{d} C_p(K)$，其中 $d = \dim(K)$，并将边界映射族 $\partial_* = \{\partial_p : C_p(K) \to C_{p-1}(K)\}_{p=1}^{d}$ 视为分次向量空间之间次数为 -1 的线性变换。

为了定义表示边界映射 $\partial_*\colon C_*(K) \to C_*(K)$ 的矩阵，我们假设 K 中的单形已完全排序，即按列表 σ_1、.....、σ_N 排列，并满足以下性质：

(1) 如果 σ_i 是 σ_j 的面，则 $i < j$。
(2) 如果 $f(\sigma_i) < f(\sigma_j)$，则 $i < j$。

K 中单形的这种排序称为与过滤相容的排序 。根据条件 (1)，边界矩阵是严格上三角矩阵。

定义 2.3. 设 K 是一个过滤单纯复形，$S = \{\sigma_1, \sigma_2, \ldots, \sigma_N\}$ 是 K 中按照与过滤相容的顺序排列的单形集合。K 的边界矩阵是一个 $N \times N$ 矩阵 D，其元素属于 $\mathbb{F}_2$，满足

$$D_{i,j} = \begin{cases} 1 & \text{if } \sigma_i \text{ is a facet of } \sigma_j \\ 0 & \text{otherwise.} \end{cases}$$

对于每个 j，边界矩阵 D 的第 j 列记为 D_j。

2.1. **标准归约算法**。文献 [8] 表明，持久图的计算可以通过一种简单的算法完成，该算法将边界矩阵化为约化形式。要定义此算法，我们需要引入一个称为 low 函数的函数。

定义 2.4. 对于每个元素属于 $\mathbb{F}_2$ 的非零向量$\mathbf{v}$，$\mathbf{v}$ 的*low* 值 定义为

$$\text{low}(\mathbf{v}) = \max(\{i \,|\, \mathbf{v}[i] = 1\})$$

其中 $\mathbf{v}[i]$ 表示$\mathbf{v}$的第 i 个元素。我们令 $\text{low}(\mathbf{0}) = 0$。

标准约化算法类似于高斯消元，但它是对列进行操作。对于从 1 到 N 依次取值的每个 j，按如下方式修改第 j 列 D_j：如果对于某个 $i < j$，$\text{low}(D_i) = \text{low}(D_j)$，则将向量 D_j 替换为 $D_i + D_j$。这里的加法在域 $\mathbb{F}_2$ 中进行。算法 1 给出了标准算法的伪代码。

算法 1 标准约化算法

输入： 边界矩阵 $D = [D_1|D_2|\cdots|D_N] \in \mathbb{F}_2^{N\times N}$**输出：** 约化矩阵 $R \in \mathbb{F}_2^{N\times N} R \leftarrow D$;**对于** $j = 1$ 到 N，**执行**
while $\exists i < j$ 满足 $\text{low}(D_i) = \text{low}(D_j) \neq 0$ **时执行** ▷ $\Omega(N)$
$D_j \leftarrow D_i + D_j$**end while end for** ▷ $\Omega(N)$
▷ $\Omega(N)$

对于每个 $k = 1, \ldots, d$，维数为 d 的过滤复形 K 的持久图 $\text{dgm}_k(K)$ 是所有持久对 $(f(\sigma_i), f(\sigma_j))$ 的集合，其中 $f(\sigma_i)$ 是维数为 k 的上同调类出生时的过滤层级，而 $f(\sigma_j)$ 是它消亡时的层级（详情见[6] ）。注意，如果 R 是对 K 的边界矩阵应用约化算法后得到的约化矩阵，那么当且仅当 σ_i 是一个 k 维单形且 $i = \text{low}(R_j) \neq 0$ 时，持久对 $(f(\sigma_i), f(\sigma_j))$ 属于持久图 $\text{dgm}_k(K)$。

对于所有具有 $R_i = 0$ 的 R 列，我们可以得出结论：在 $f(\sigma_i)$ 处发生了出生。如果对于某个 i，$R_i = 0$，且在任意 j 的情况下，R_i 处都没有低值，j，那么我们称 $(f(\sigma_i), \infty)$ 为一个持久性对。

例 2.5. 设 K 为过滤单纯复形，其单形按出现顺序如下：

$$K = \{[1], [2], [3], [1,2], [1,3], [2,3], [1,2,3]\}.$$

K 的边界矩阵为：

$$D = \begin{array}{c} \\ [1] \\ [2] \\ [3] \\ [1,2] \\ [1,3] \\ [2,3] \\ [1,2,3] \end{array} \begin{array}{c} \begin{array}{ccccccc} [1] & [2] & [3] & [1,2] & [1,3] & [2,3] & [1,2,3] \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

然后，对边界矩阵 D 应用标准约化算法，步骤如下：

$$D = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{C5+C6 \to C6} \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{C4+C6 \to C6} \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = R$$

如果我们将过滤函数设为把 σ_i 指派给索引 i 的函数，那么根据约化矩阵中低位元素的索引，我们可以得出，过滤单纯复形 K 的持久对集合为

$$\mathrm{dgm}_0(K) = \{(1, \infty), (2, 4), (3, 5)\} \text{ and } \mathrm{dgm}_1(K) = \{(6, 7)\}.$$

2.2. **带扭转的约化算法。** Chen 和 Kerber [4]提出了约化算法的一种变体，称为带扭转的约化算法。这种新算法背后的思路是，如果先对高维单形进行约化，那么在处理第i列之前，对 j列进行约化后，我们就会找到一个持久对（$f(\sigma_i)$ $f(\sigma_j)$）。由于这个持久对表明单形 σ_i 生成了一个同调类，因此约化矩阵中的第 i列必为零。于是我们可以清空这一列，也就是直接将其置零，而无需进行任何列加法操作。这能省去许多不必要的列操作，让约化算法运行得快得多。

为了介绍该算法的伪代码，首先注意到，边界矩阵可以分解为针对线性映射 $\partial_p : C_p(K) \to C_{p-1}(K)$ for$p = 1, \ldots, d$ 的更小边界矩阵。约化算法可以利用这些更小的边界矩阵进行描述。带扭转的约化算法的伪代码见算法 2。

例 2.6. 令 D 表示例 2.5 中过滤单纯复形的边界矩阵。然后，应用一步 Chen 和 Kerber 的算法即可得到约化矩阵 R：

$$D = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{\text{clear the 6-th column}} \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = R$$

算法 2 带扭转的约化算法

输入： 边界矩阵 $D = [D_1|D_2|\cdots|D_N] \in \mathbb{F}_2^{N\times N}$
输出： 约化矩阵 $R \in \mathbb{F}_2^{N\times N}$
$R \leftarrow D$; L： $[0, 0, \ldots, 0]$
对于 $\delta = d, \ldots, 1$ **do**
  对于 $j = N, \ldots, 1$ **执行**
    如果 $\dim \sigma_j = \delta$ **则**
      当 $R_j \neq 0$ 且 $L[\mathrm{low}(R_j)] \neq 0$ **时执行**
        $R_j \leftarrow R_j + R_{L[\mathrm{low}(R_j)]}$
      结束循环
      如果 $R_j \neq 0$ **，则**
        $i \leftarrow \mathrm{low}(R_j)$
        $L[i] \leftarrow j$
        $R_i \leftarrow 0$
      结束 if
    结束 if
  结束循环
结束循环

扭转算法是一种非常高效的算法，当前所有持久同调计算软件包（如 Gudhi、Phat 和 Ripser）都在使用它。我们将在第 2.4 节讨论扭转算法的运行时间。

2.3. **带前瞻的约简算法**。 在最近的一篇论文中，Morozov 和 Skraba[11] 提出了一种能够以矩阵乘法时间计算持久同调的算法。在该论文中，他们还讨论了标准约化算法的一种前瞻变体。该算法的主要思想是：如果列 R_j 的 low 值为 $i \neq 0$，那么对于每个满足 $j' > j$ 和 $R_{i,\,j'} = 1$ 的列 $R_{j'}$，我们将列 R_j 加到 $R_{j'}$ 中，使其满足 $R_{i,\,j'} = 0$。算法 3 给出了该算法的伪代码。

算法 3 带前瞻的约简算法

输入： 边界矩阵 $D = [D_1|D_2|\cdots|D_N] \in \mathbb{F}_2^{N\times N}$
输出： 约化矩阵 $R \in \mathbb{F}_2^{N\times N}$
$R \leftarrow D$;
对于 $j = 1$ 至 N **执行** ▷ $\Omega(N)$
  如果 $\mathrm{low}(R_j) \neq 0$ **，则**
    $i \leftarrow \mathrm{low}(R_j)$
    对于 $j' > j$ 且 $R[i, j'] = 1$ **执行** ▷ $\Omega(N)$
      $R[\cdot, j'] \leftarrow R[\cdot, j'] + R[\cdot, j]$ ▷ **Ω(N)**
    结束循环
  如果结束
循环结束

例 2.7. 令 D 表示例 2.5 中的边界矩阵。然后，使用前瞻式约化算法，可得到约化矩阵 R，如下所示：

$$D = \begin{bmatrix} 0&0&0&1&1&0&0\\ 0&0&0&1&0&1&0\\ 0&0&0&0&1&1&0\\ 0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0 \end{bmatrix} \xrightarrow{C4+C6\to C6} \begin{bmatrix} 0&0&0&1&1&1&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&1&1&0\\ 0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0 \end{bmatrix} \xrightarrow{C5+C6\to C6} \begin{bmatrix} 0&0&0&1&1&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0 \end{bmatrix} = R$$

请注意，在这个示例中，约化矩阵恰好使用与标准算法相同数量的操作得到。这通常并非如此。有些示例中，前瞻约化算法的表现优于约化算法。我们将在第4节讨论将约化算法的前瞻变体应用于条带最坏情况示例时的运行时间。

2.4. 约化算法的运行时间。 结果表明，对于约化算法的三种变体，其最坏情况下的运行时间均为 $\Omega(N^3)$，其中 N 是单形的数量。

对于标准约化算法，我们可以通过计算算法各个步骤的运行时间得出这一点。在该算法中，我们有一个包含 N 个元素的**for** 循环，因此其运行时间为 $\Omega(N)$。**while**循环的运行时间也为 $\Omega(N)$。在这些循环中，两个大小为 N 的向量相加，所需时间为 $\Omega(N)$。因此，标准约化算法的运行时间为 $\Omega(N^3)$。

带扭转的约化算法的运行时间计算与此类似。在该算法中，前两个**for**循环的运行时间为 $\Omega(N)$，**while**循环的运行时间为 $\Omega(N)$，加法操作所需时间为 $\Omega(N)$。因此，带扭转的约化算法的复杂度也为 $\Omega(N^3)$。

实践表明，扭转算法的效率要高得多，因为边界矩阵通常是稀疏矩阵。可以根据约化矩阵中持久对的数量，对该算法的运行时间进行明确计算（见 [4]）。在使用规模为 10 的随机数据点进行的实验中，我们发现，采用带扭转的约化算法后，执行的向量加法次数大幅减少。我们针对最坏情况条带示例所做的实验也支持这一结论（见第4节）。

在标准算法的前瞻变体中，有两个**for**循环，每个循环的运行时间均为 $\Omega(N)$，此外还有一次运行时间为 $\Omega(N)$ 的向量加法。因此，前瞻算法的运行时间也为 $\Omega(N^3)$。与扭转算法类似，我们可以进行一些计算，将前瞻约化算法的效率与其他算法进行比较。在使用规模为 10 的随机数据集进行的实验中，我们发现，前瞻约化算法的运行时间优于标准约化算法。

3. 另一种最坏情况示例：条带

本节将构造一个针对约化算法的最坏情况示例，称为条带示例；它与 Morozov 在 [10] 中提出的最坏情况示例类似。

对于每个正整数 n，**条带最坏情形复形** $X(n)$是一个过滤的单纯复形，由 n 个位于平面上的基三角形和 n 个鳍三角形组成；每个鳍三角形有一条边位于平面上，另一个顶点位于平面外。我们按照图1所示放置这些基三角形和鳍三角形。基三角形标记为 i，鳍三角形标记为-i，其中 $i = 1, \ldots\ldots, n$。这些三角形的顶点标记为 $v_1, \ldots\ldots, v_{n+2}$和 $f_1, \ldots\ldots, f_n$。基三角形从中心开始标记，然后左右交替。

我们将基三角形的水平边标记为 $1'$，……，n'，使得 i' 是基三角形 i 的水平边。垂直底边标记为 1，……，$n+1$，其中1是连接 v_1 和 v_2 的边；对于所有 $i=2$，……，$n+1$，边 i 连接 v_{i-1} 和 v_{i+1}，如图1所示。标记为 $n+1$ 的底边也标记为 $0'$，以便更容易说明它在过滤中的位置。

鳍三角形的边标记为 -1、...、$-n$ 和 -1^*、...、$-n^*$。边 -1 位于顶点 f_1 和 v_2 之间，边 -1^* 位于 f_1 和 v_1 之间。对于所有 $i=2$、...、n，边 $-i$ 位于顶点 f_i 和 v_{i+1} 之间，边 $-i^*$ 位于 f_i 和 v_{i-1} 之间。注意，除标签为 $n+1=0'$ 的边外，每条垂直基边上方都有一个鳍三角形。

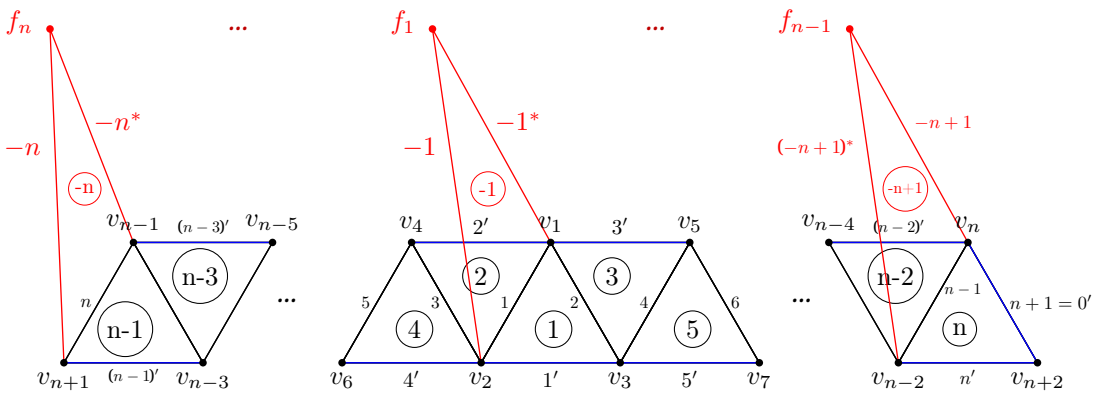
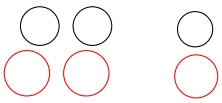


图1. 对于 $n\equiv 1$ (模 4)，单纯复形 $X(n)$。

条带最坏情况示例通过添加三角形逐步构造。从 $X(n)$ 到 $X(n+1)$，只需在边 $n+1$ 上方添加一个鳍三角形，并在标记为 n 的边旁添加一个底部三角形。注意，单纯复形 $X(n)$ 有 $2n+2$ 个顶点、$4n+1$ 条边和 $2n$ 个三角形。$X(n)$ 的单形数量为 $N=8n+3$。$X(n)$ 的欧拉示性数等于 1，因为它的同调与一个点的同调同构。要定义 $X(n)$ 上的过滤，我们按照以下顺序排列该复形的单形：

(1) 鳍部顶点 $f_1,\ldots,f_n$, (2) 底部顶点 $v_1,\ldots,v_{n+2}$, (3) 消除连通分支的水平基边 $n',(n-1)',\ldots,1'$, (4) 底边 $n+1$，也标记为 $0'$, (5) 消除鳍连通分支的鳍边 $-n^*,\ldots,-1^*$, (6) 产生圈的鳍边 $-n,-n+1,\ldots,-1$, (7) 垂直基边 $n,n-1,\ldots,1$, (8) 底三角形 1 , 2 , . . . , n , 以及 (9) 鳍三角形 -1 , -2 , . . . , -n。



为简化计算，在定理 3.2 的证明中，我们仅对 ∂_2 的边界矩阵应用约化算法：$C_2(X(n))\to C_1(X(n))$。我们将该矩阵记为 $B(n)$，将对应的约化矩阵记为 $R(n)$。

例 3.1。 </style}对于 $n = 2$，边界矩阵 $B(2)$ 和约化矩阵 $R(2)$ 如下：

$$B(2) = \begin{array}{c} \\ 2' \\ 1' \\ 0' \\ -2^* \\ -1^* \\ -2 \\ -1 \\ 2 \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} \textcircled{1} & \textcircled{2} & \textcircled{-1} & \textcircled{-2} \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array} \xrightarrow[C_1+C_3 \to C_3]{\substack{C_1+C_2 \to C_2 \\ C_2+C_3 \to C_3 \\ C_2+C_4 \to C_4}} \begin{array}{c} \begin{array}{cccc} \textcircled{1} & \textcircled{2} & \textcircled{-1} & \textcircled{-2} \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \end{array} = R(2)$$

矩阵 $B(2)$ 各列的处理过程可以用稀疏矩阵表示法来解释：

$$\textcircled{1} \to (1, 2, 1')$$

请注意，列 1 的低点是对应于边 1的行。在这种情况下，我们称三角形 1 与边 1配对。对于三角形 2 ，我们有

$$\textcircled{2} \to (1, 0', 2')$$
$$\textcircled{1} + \textcircled{2} \to (1, 2, 1') + (1, 0', 2') = (2, 0', 1', 2')$$

这使三角形 2 与边 2配对。对于鳍三角形，我们有类似的配对过程。

$$\textcircled{-1} \to (1, -1, -1^*)$$
$$\textcircled{-1} + \textcircled{1} \to (1, -1, -1^*) + (1, 2, 1') = (2, -1, -1^*, 1')$$
$$\textcircled{-1} + \textcircled{1} + (\textcircled{1} + \textcircled{2}) \to (2, -1, -1^*, 1') + (2, 0', 1', 2') = (-1, -1^*, 0', 2')$$

这使鳍三角形 -1 与边 -1配成一对。最后，对于鳍三角形 -2 ，我们有

$$\textcircled{-2} \to (2, -2, -2^*)$$
$$\textcircled{-2} + (\textcircled{1} + \textcircled{2}) \to (2, -2, -2^*) + (2, 0', 1', 2') = (-2, -2^*, 0', 1', 2')$$

这使三角形 -2 与边 -2配成一对。因此，一维同调的持久对 H_1 为

$$\mathrm{dgm}_1(X(2)) = \{(1, \textcircled{1}), (2, \textcircled{2}), (-1, \textcircled{-1}), (-2, \textcircled{-2})\}.$$

这里我们用单形列出了持久对，因为在上面的矩阵中，我们用单形标记列和行。注意，这些对对应于约化矩阵 $R(2)$ 中低点的索引。

现在我们证明本节的主要结果。

定理 3.2。 令 $B(n)$ 表示边界映射 $\partial_2 : C_2(X(n)) \to C_1(X(n))$的矩阵。将标准约化算法应用于 $B(n)$的运行时间复杂度为$\Omega(n^3)$。因此，针对$X(n)$ 的边界矩阵，标准约化算法的复杂度为$\Omega(N^3)$，其中 $N = 8n + 3$ 是 $X(n)$的单形数量。

证明。 将上述针对 $n = 2$的配对算法推广后，我们可以看到，底部三角形i 将与以下底边 i 配对，其代表元列表如下：

$$
\begin{aligned}
1 &\to (1, 2, 1') \\
1 + 2 &\to (1, 2, 1') + (1, 3, 2') = (2, 3, 1', 2') \\
1 + 2 + 3 &\to (2, 3, 1', 2') + (2, 4, 3') = (3, 4, 1', 2', 3') \\
&\vdots \\
1 + 2 + \cdots + \text{n-1} &\to (n-1, n, 1', 2', \cdots, (n-1)') \\
1 + 2 + \cdots + \text{n} &\to (n, 0', 1', 2', \cdots, n')
\end{aligned}
$$

该表应按以下方式阅读。1 的边界是由边构成的有序三元组 (1, 2, 1′)。因此，1 与 1 配对。三角形 2 的边界是 (1, 3, 2′)。由于 1 已与 1 配对，我们需要将 1 的列加到 2 的列上。这会生成一个边界为 (2, 3, 1′, 2′) 的链 $1 + 2$。因此，三角形 2 与 2 配对。我们以此类推，直到对所有 i，三角形i 都与边 i 配对。从表中可以看出，在约化矩阵 $R(n)$ 中，i 对应的列有 $i + 2$ 个非零项。对于每个 i，我们将链 $1 + \cdots + \text{i} = (i, i + 1, 1', 2', \ldots, i')$ 记为 c_i。

处理基三角形的复杂度可按如下方式计算：由于将两个分别含有 a_1 和 a_2 个非零项的向量相加，其复杂度为 $a_1 + a_2$，因此执行上述所有加法操作的复杂度为

$$(3 + 3) + (4 + 3) + (5 + 3) + \cdots + ((n + 1) + 3) = \frac{1}{2}(n^2 + 9n - 10).$$

现在我们计算处理鳍三角形的复杂度。对于每个 i，鳍三角形 -1 的处理过程如下表所示：

$$
\begin{aligned}
\text{-1} &\to (1, -1, -1^*) \\
\text{-1} + c_1 &\to (1, -1, -1^*) + (1, 2, 1') = (2, -1, -1^*, 1') \\
(\text{-1} + c_1) + c_2 &\to (2, -1, -1^*, 1') + (2, 3, 1', 2') = (3, -1, -1^*, 2') \\
(\text{-1} + c_1 + c_2) + c_3 &\to (3, -1, -1^*, 2') + (3, 4, 1', 2', 3') = (4, -1, -1^*, 1', 3') \\
(\text{-1} + c_1 + c_2 + c_3) + c_4 &\to (4, -1, -1^*, 1', 3') + (4, 5, 1', 2', 3', 4') = (5, -1, -1^*, 2', 4') \\
&\vdots \\
(\text{-1} + c_1 + c_2 + \cdots + c_{n-1}) + c_n &\to \begin{cases} (-1, -1^*, 0', 1', 3', 5', \cdots, n') & \text{if } n \text{ is odd} \\ (-1, -1^*, 0', 2', 4', 6', \ldots, n') & \text{if } n \text{ is even.} \end{cases}
\end{aligned}
$$

这使鳍三角形 -1 与边 -1配对。每个鳍三角形都有类似的处理过程。如果我们假设鳍三角形 -1 是通过执行加法操作 $\text{-1} + c_1 + \cdots + c_n$ 来处理的，那么非零项上的域加法总次数为

等于

$$3+\sum_{i=1}^{n}(i+2)=\frac{1}{2}(n^2+5n+6).$$

以类似的方式，如果我们处理第 j 个鳍三角形 -j ，可以看到我们需要执行 -j $+c_j+\cdots+c_n$ 次加法操作，因此加法操作的总次数为

$$3+\sum_{i=j}^{n}(i+2)=\frac{1}{2}(n^2+5n+-j^2-3j+10).$$

将这些结果对所有 $j=1$、……、 n 求和，可得加法操作的次数等于

$$\frac{1}{6}n(2n^2+9n+25).$$

将此结果加到底部三角形处理的复杂度上，可得加法操作的总次数为

$$S(n)=\frac{1}{3}(n^3+6n^2+26n-15).$$

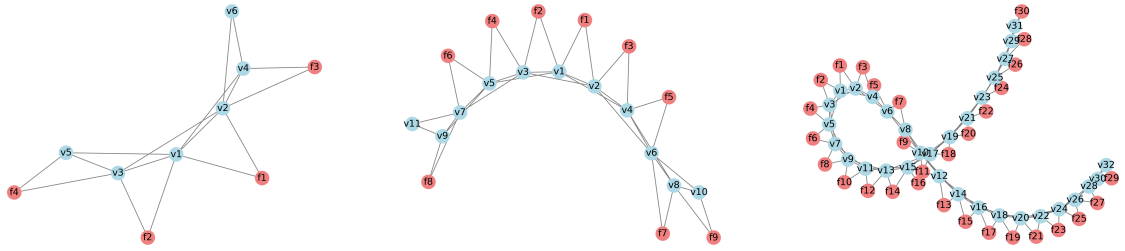
特别地， $S(n)=\Omega(n^3)$。由于复形中的单形数量为 $N=8n+3$，因此标准约化算法的最坏情况运行时间为 $\Omega(N^3)$。□

注 3.3。 在我们的实现中，每个中间步骤都会执行加法操作，因此实际执行的加法操作次数通常大于 $T(n)$。

4. 条带示例实验

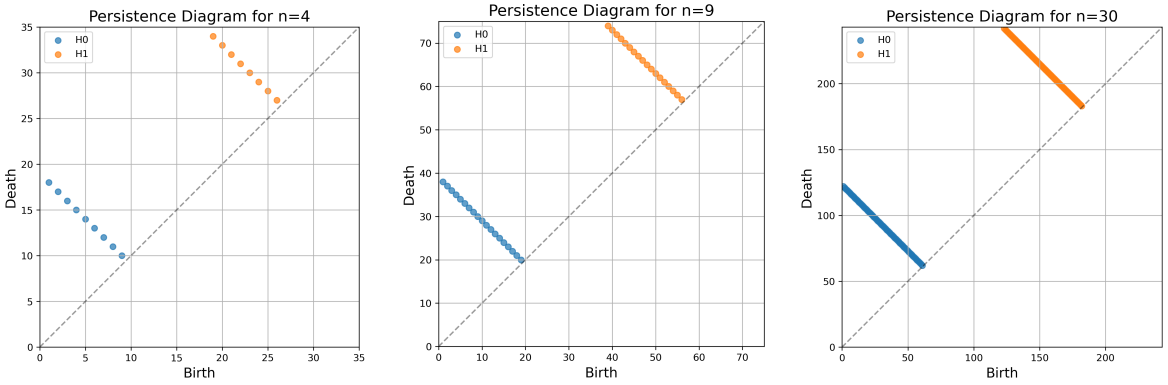
我们提出另一种最坏情况示例的动机之一，是能够编写使用最坏情况示例进行性能测试的计算机程序。条带示例的线性结构使我们能够给出一个简单的显式算法，为每个 $n\geq 1$ 生成复形 $X(n)$。该算法见附录 A。使用该算法编写的 Python 代码，以及我们所进行实验的结果，可在 GitHub 页面上找到：https://github.com/Ergun1234/Morozov_type_Examples。

运行用于构造条带示例 $X(n)$的 Python 代码，并对 $X(n)$ 的边界矩阵运行标准约化算法后，我们得到了条带示例的持久图。图 2 展示了若干 n 取值下的过滤单纯复形 $X(n)$，图 3 则展示了相应的持久图。



（a） $n=4$　　（b） $n=9$　　（c） $n=30$

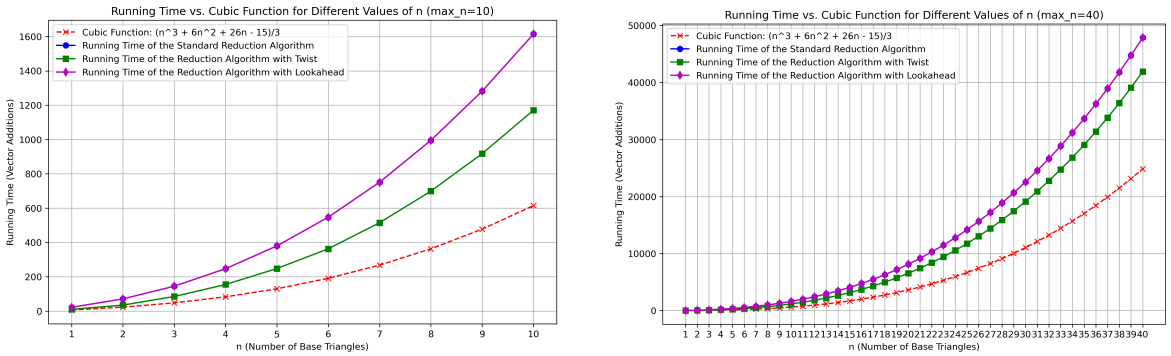
图 2。对于不同的 n 值，条带最坏情形单纯复形 $X(n)$ 的 1-骨架。



(a) $n = 4$ (b) $n = 9$ (c) $n = 30$

图 3. 条带最坏情况示例 $X(n)$在 n取不同值时的持久图。蓝色点对应 H_0 特征，橙色点对应 H_1 特征。虚线表示对角线 $y = x$。

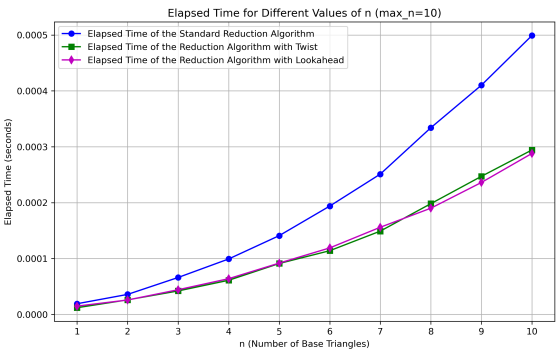
我们进一步对分裂最坏情况示例进行了实验，并计算了约化算法、标准约化算法、带扭转的约化算法以及带前瞻的约化算法的运行时间。图 4 展示了这三种算法的运行时间，其中运行时间通过一个计数器计算，该计数器会跟踪计算约化边界矩阵过程中执行的加法操作次数。图 5 展示了这些算法在配备 M5 处理器的 MacBook 上的实际运行时间。



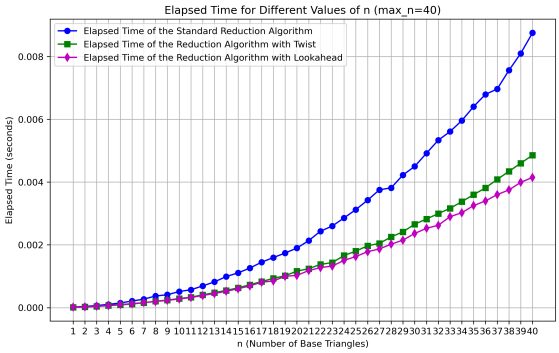
(a) $\max_n = 10$ (b) $\max_n = 40$

图 4. 各种约化算法的运行时间与三次函数的比较，针对 $\max_n$ 的两个取值（即实验中使用的 n 的最大值）。

这些图表明，标准约化算法与前瞻约化算法的运行时间相同。带扭转的约化算法的运行时间比另外两种算法短。我们将在例 4.1 中解释其中的原因。这三种算法的耗时表明，将约简算法应用于条带示例时，扭转变体和前瞻变体的表现都远优于标准算法。



(a) $\max_n = 10$



(b) $\max_n = 40$

图5. 对于两个 $\max_n$ 值，不同约化算法变体的运行耗时随参数 n 的变化。

例 4.1. 对于条带示例 $X(2)$，考虑边界映射$\partial_1 : C_1(X(2)) \to C_0(X(2))$对应的矩阵 $B(1)$。

$$B(1) = \begin{array}{c} \\ f_1 \\ f_2 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 2' & 1' & 0' & -2^* & -1^* & -2 & -1 & 2 & 1 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

为了约化边界矩阵 $B(1)$ 的前五列，我们执行列操作 $C_1 + C_3 \to C_3$ 和 $C_4 + C_5 \to C_5$，需要进行 8 次域加法。随后再进行 9 次列加法，以清除最后四列。这些操作还需要额外进行 36 次域加法。如果改用带扭转的约简算法，由于我们先处理矩阵 $B(2)$，因此将 $B(1)$ 的最后四列设为零。这样一来，域加法的次数便从 44 次减少到 8 次。由于处理 $B(2)$ 需要进行 27 次域加法，约化边界矩阵所执行的域加法总次数便从 71 次减少到 35 次。这就是图 4 中第一幅图里 $n = 2$ 观察到的下降。下表展示了约简算法的各个变体对 $n \leq 5$执行的域加法次数。

表 1. 标准约化算法及其变体执行的域加法次数。

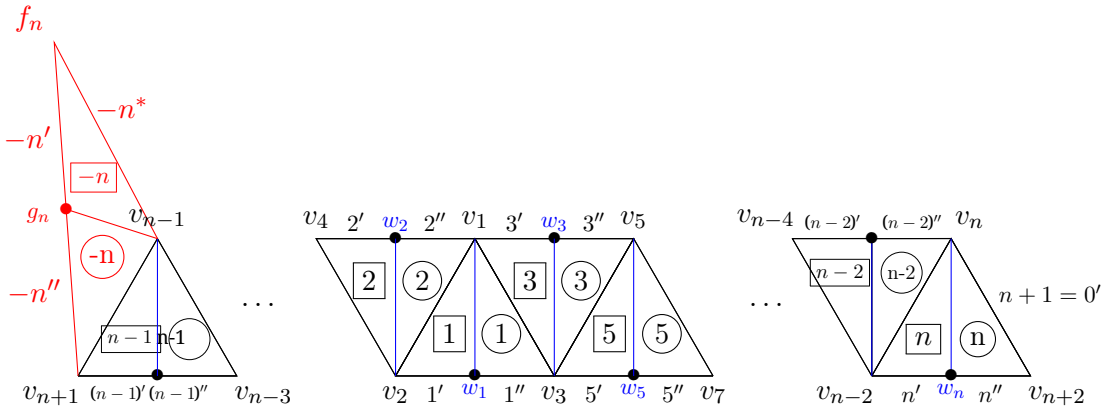
n	标准约化	带扭转的前瞻	
1	22	10	22
2	71	35	71
3	145	85	145
4	247	155	247
5	380	248	380

5. 一个作为旗复形的最坏情况示例

本节将说明，对分裂复形中的一些边和三角形进行细分并加以修改后，得到的过滤复形是某个过滤图的旗复形。在本节末尾，我们还将说明，这些修改后的条带示例可以实现为某个有限数据云的 Vietoris–Rips 复形，其中实数按递增序列排列。

定义 5.1. 设 $G = (V,\ E)$ 是一个无向无环图。 G 的团复形 是一个单纯复形 $Cl(G)$，其顶点集为 V；当且仅当对于每个 $i < j$， $\{v_i, v_j\}$ 都是 G 中的一条边时，顶点的子集 $\{v_0, \ldots, v_k\}$ 才是 $Cl(G)$ 中的一个单形。

注意，如果 G 是一个过滤图，其过滤为 $\emptyset \subseteq G_1 \subseteq G_2 \subseteq \cdots \subseteq G_m$，那么与之关联的团复形序列 $\emptyset \subseteq Cl(G_1) \subseteq Cl(G_2) \subseteq \cdots \subseteq Cl(G_m)$ 给出一个过滤单纯复形。按照定义，分裂最坏情况示例无法实现为过滤图的团复形，因为在过滤过程中，三角形的出现时间比其边界边晚了若干步。为了将它们实现为团复形，我们对分裂示例作如下修改：



该图与前一个例子类似；不同之处在于，我们添加顶点$w_1, \ldots, w_n$，作为水平基边 $1', \ldots, n'$ 的中点，并将由此得到的半边记为 i' 和 i''，分别对应 $i = 1, \ldots, n$。类似地，我们在鳍三角形的边 $-1, \ldots, -n$ 的中点处添加顶点 $g_1, \ldots, g_n$，并将由此得到的半边记为$-i'$和 $-i''$，分别对应 $i = 1, \ldots, n$。

我们还添加新边，将基三角形和鳍三角形各自平分。对于基三角形，这些新边记为 $e_1^b, \ldots, e_n^b$；对于鳍三角形，则记为 $e_1^f, \ldots, e_n^f$。它们以被细分的三角形为索引。对于被分割的三角形，在基三角形中，我们将左侧的记为方框标记的 i，右侧的记为圆圈标记的 i。对于鳍三角形，它们分别记为方框标记的 $-i$ 和圆圈标记的$-i$ 。这个修改后的单纯复形具有如下过滤：

(1) 鳍顶点 f_1、……、 f_n 和 g_1、……、 g_n；(2) 底部顶点 v_1、……、 v_{n+2} 和 w_1、……、 w_n；(3) 水平基边 n''、……、 $1''$ 和 n'、……、 $1'$；(4) 同时标记为 $0'$ 的基边 $n+1$；(5) 破坏鳍连通分支的鳍边 $-n^*$、……、 -1^*；(6) 产生圈的鳍边 $-n''$、……、 $-1''$ 和 $-n'$、……、 $-1'$；(7) 垂直底边 n、 $n-1$、……、 1；(8) 底部中线和底部三角形 e_1^b、 1、 1、……、 e_n^b、 n、 n；(9) 鳍中线和鳍三角形 e_1^f、 -1、 -1、……、 e_n^f、 $-n$、 -n。

注意，对于所有 i，边 e_i^b 以及底部三角形 i 和 i 会在过滤中同时出现。鳍三角形的情况类似：边 e_i^f 以及鳍三角形 $-i$ 和 i会在同一过滤层级出现。我们将这个修改后的示例称为 修改后的分裂示例，并记为 $Y(n)$。

对于每个 $k = 0$，1，2，令 $B'(k)$ 表示 ∂_k 的边界矩阵：$C_k(Y(n)) \to C_{k-1}(Y(n))$。对于边界矩阵，我们采用上面列表给出的线性顺序。

例 5.2。本例展示如何使用约化算法对修改后的分裂示例 $Y(2)$ 的边界矩阵 $B'(2)$ 进行约化。我们用稀疏矩阵表示法说明各列的处理过程：

$$\begin{aligned}
\boxed{1} &\to (e_1^b, 1, 1')\\
d_1 = \boxed{1} + \textcircled{1} &\to (e_1^b, 1, 1') + (e_1^b, 2, 1'') \to (1, 2, 1', 1'')
\end{aligned}$$

因此，三角形 1 与 e_1^b 配对，而三角形 1 与边 1配对。对于三角形 2 和 2，我们有

$$\begin{aligned}
\boxed{2} &\to (e_2^b, 0', 2')\\
\boxed{2} + \textcircled{2} &\to (e_2^b, 0', 2') + (e_2^b, 1, 2'') \to (1, 0', 2', 2'')\\
d_2 = (\boxed{2} + \textcircled{2}) + d_1 &\to (1, 0', 2', 2'') + (1, 2, 1', 1'') = (2, 0', 1', 1'', 2', 2'').
\end{aligned}$$

这表明三角形 2 与 e_2^b 配对，三角形 2 与 2配对。对于鳍三角形，我们有类似的配对过程。

$$\begin{aligned}
\boxed{-1} &\to (e_1^f, -1', -1^*)\\
\boxed{-1} + \textcircled{-1} &\to (e_1^f, -1', -1^*) + (e_1^f, 1, -1'') \to (1, -1', -1'', -1^*)\\
(\boxed{-1} + \textcircled{-1}) + d_1 &\to (1, -1', -1'', -1^*) + (1, 2, 1', 1'') \to (2, -1', -1'', -1^*, 1', 1'')\\
(\boxed{-1} + \textcircled{-1} + d_1) + d_2 &\to (2, -1', -1'', -1^*, 1', 1'') + (2, 0', 1', 1'', 2', 2'')\\
&\to (-1', -1'', -1^*, 0', 2', 2'').
\end{aligned}$$

这使鳍三角形 -1 与 e_1^f 配对，并使鳍三角形 -1 与边 $-1'$配对。最后，我们有

$$\begin{aligned}
\boxed{-2} &\to (e_2^f, -2', -2^*)\\
\boxed{-2} + \textcircled{-2} &\to (e_2^f, -2', -2^*) + (e_2^f, 2, -2'') \to (2, -2', -2'', -2^*)\\
(\boxed{-2} + \textcircled{-2}) + d_2 &\to (2, -2', -2'', -2^*) + (2, 0', 1', 1'', 2', 2'')\\
&\to (-2', -2'', -2^*, 0', 1', 1'', 2', 2'').
\end{aligned}$$

这将三角形 -2 与 e_2^f 配对，而三角形 -2 则与边 $-2'$配对。因此，$Y(2)$ 的一维同调 H_1 的持久对为

$$\{(e_1^b, \boxed{1}), (1, \textcircled{1}), (e_2^b, \boxed{2}), (2, \textcircled{2}), (e_1^f, \boxed{-1}), (-1', \textcircled{-1}), (e_2^f, \boxed{-2}), (-2', \textcircled{-2})\}.$$

由于有些持久对由在同一过滤层级出现的单形构成，因此它们会在持久图中显示为对角线上的点。

现在我们可以证明以下命题：

命题 5.3。 修改后的条带示例 $Y(n)$ 是一个过滤图的团复形，其时间复杂度为 $\Omega(M^3)$ ，其中 M 表示单形的数量。

证明。 在每个过滤层级上，修改后的条带示例都是其一维骨架的团复形，因为复形中的所有三角形都与其边界边出现在同一过滤层级。

要证明将约化算法应用于 $B'(n)$具有三次运行时间复杂度，注意复形 $Y(n)$ 有 $4n+2$ 个顶点、 $8n+1$ 条边和 $4n$ 个三角形。因此，复形 $Y(n)$ 中的单形数量为 $M = 16n+3$。注意，在约化算法中，框出的三角形总是与将三角形一分为二的那条边配对。为了得到链 $t_i = i + i$ 和 $t_{-i} = -i + -i$ ，我们执行 $2n$ 次向量加法，而每次向量加法都使用$12n$ 次域加法。

链 t_i 和 t_{-i} 的处理方式与原始条带示例中三角形的处理方式非常相似。事实上，处理这些链所用的向量加法次数相同。唯一的区别在于，由于修改后的情形引入了更多边，因此相加的向量含有更多非零项。这增加了约化过程中域加法的次数。现在，我们计算处理边界矩阵 $B'(n)$ 各列所用的域加法的确切次数。

令 $d_i = t_1 + \cdots + t_i$。注意， d_i的列有 $2i+2$ 个非零项。由于 d_i 的列是通过将 t_i 和 d_{i-1}的列相加得到的，因此处理 t_1、……、 t_n的所有列所需的域加法次数为

$$(4+4) + (6+4) + (8+4) + \cdots + (2n+4) = n^2 + 5n - 6.$$

处理 t_{-j}的列时，我们执行加法操作 $t_{-j} + d_j + \cdots + d_n$，这需要

$$4 + \sum_{i=j}^{n} (2i+2) = n^2 + 3n - j^2 - j + 6$$

次域加法。将这些加法操作对所有 $j = 1$、……、 n求和，可知处理鳍三角形的列需要执行 $\frac{2}{3}n(n^2+3n+8)$ 次加法。因此，约化矩阵 $B'(2)$所需的域加法总次数为

$$S'(n) = \frac{1}{3}(2n^3 + 9n^2 + 37n - 18).$$

由于 $M = 16n+3$，我们有 $S'(n) = \Omega(M)$。注意，与条带示例 $X(n)$的约化过程相比，修改后的复形 $Y(n)$ 的单形数量和约化过程中的域加法次数都大致增加了一倍。□

接下来我们将展示，修改后的条带示例可以实现为一列实数 $0 < r_1 < \cdots < r_s$ 作为尺度参数时某个有限数据云的 Vietoris–Rips 复形。我们首先回顾数据云的 Vietoris–Rips 复形的定义。

定义 5.4。 设 X 是 $\mathbb{R}^l$ 中的有限点集。对于实数 $r \geq 0$，*Vietoris– Rips* 复形 $VR(X, r)$ 定义为如下单纯复形：其顶点集为 X，其 n-单形是 $\{x_0, \ldots, x_n\}$ 中满足对于每个 i， j 都有 $d(x_i, x_j) \leq r$ 的 X 的子集。

对于递增的正实数序列 $0 < r_1 < r_2 < \cdots < r_m$，我们可以将单纯复形 K_i 定义为 Vietoris–Rips 复形 $VR(X, r_i)$，从而得到一个过滤单纯复形。以下命题隐含于经典距离几何和 Vietoris–Rips 复形理论之中，但我们未能找到直接的参考文献。为完整起见，我们在此给出证明。

命题 5.5。 设 $\emptyset \subseteq G_1 \subseteq \cdots \subseteq G_m$ 是定义在有限顶点集 $V = \{1, \ldots, n\}$ 上的一个过滤图，并令 $\mathrm{Cl}(G_t)$ 表示 G_t 对于 $t = 1$、.. .、 m 的团复形。那么存在一些点

$$X = \{p_1, \ldots, p_n\} \subset \mathbb{R}^l$$

以及实数的递增序列 $0 < r_1 < \cdots < r_m$，使得对于每个 $t = 1, \ldots, m$，都存在一个单纯复形的同构

$$VR(X, r_t) \cong \mathrm{Cl}(G_t).$$

证明。 众所周知，Vietoris–Rips 复形是旗复形，因此也是其底层一维子复形的团复形（例如参见 [7, 第三节.2]）。因此，只需将过滤图 G_i 实现为嵌套的距离阈值图即可。

由于 $n = 1$ 情形显然成立，我们可以假设 $n \geq 2$。对于每个无序对 $\{i, j\}$，定义其出现层级

$$\ell(i, j) = \min\{t : \{i, j\} \in E(G_t)\},$$

若该边从未出现，则令其为 $\ell(i, j) = \infty$。注意，对于所有 i, j，都有 $\ell(i, j) \geq 1$。固定

$$\varepsilon := \frac{1}{n-1}, \qquad \delta_t := 2^{-t}\varepsilon$$

对于 $t = 1, \ldots, m$，令 $\delta_{m+1} := 0$。于是我们有 $\frac{1}{2} \geq \delta_1 > \cdots > \delta_m > \delta_{m+1} = 0$，且对于每个 i,

$$\sum_{j \neq i} \delta_{\ell(i,j)} \leq (n-1)\delta_1 = (n-1)\frac{\epsilon}{2} = \frac{1}{2}.$$

定义一个 $n \times n$ 对称矩阵 $K = (K_{ij})$，使得

$$K_{ii} = 1, \qquad K_{ij} = \delta_{\ell(i,j)} \text{ for all } i \neq j.$$

根据格尔施戈林圆盘定理（例如参见 [14]）， K 的每个特征值都位于区间 $[1 - \frac{1}{2}, 1 + \frac{1}{2}]$ 内。特别地， K 是正定的。

根据线性代数中的一个标准结论，每个实对称正定矩阵都是一个格拉姆矩阵。因此，存在向量 p_1、.....、 $p_n \in \mathbb{R}^l$，使得

$$K_{ij} = \langle p_i, p_j \rangle$$

（参见 [1, 定理 7.35]）。特别地，对于所有 i， $\|p_i\| = 1$ ，并且

$$\|p_i - p_j\|^2 = \|p_i\|^2 + \|p_j\|^2 - 2\langle p_i, p_j \rangle = 2 - 2\delta_{\ell(i,j)}.$$

对于每个 $t = 1$、.....、 $m + 1$，定义

$$d_t := \sqrt{2 - 2\delta_t}.$$

对于所有 i、 j，我们有 $\|p_i - p_j\| = d_{\ell(i,j)}$。由于序列 δ_t 严格递减，序列 d_t 严格递增。我们有

$$1 \leq d_1 < d_2 < \cdots < d_m < d_{m+1} = \sqrt{2}.$$

对于每个 $t = 1$, ..., m，选择一个半径 r_t，使得

$$d_t < r_t < d_{t+1}.$$

Then

$$\|p_i - p_j\| \leq r_t \iff d_{\ell(i,j)} \leq d_t \iff \ell(i, j) \leq t \iff \{i, j\} \in E(G_t).$$

因此，尺度为 r_t 时的距离阈值图恰好是 G_t。证明至此完成。 □

我们以如下推论结束本文。

推论 5.6. 修改后的条带示例 $Y(n)$ 可以实现为某个实数序列 $0 < r_1 < \cdots < r_m$ 作为尺度参数时有限数据集的 *Vietoris–Rips* 复形。

证明。 这由命题 5.3 和 5.5 得出。 □

文献 [9] 还给出了 Erdös-Rényi 和 Vietoris-Rips 过滤模型的最坏情况示例。对于这些模型，最坏情况运行时间为 $\Theta(n^7)$，因为还考虑了所有高维单形的处理。

致谢与经费声明

第二作者获 TÜBİTAK 2219 国际博士后研究奖学金计划（2023 年第 2 期）资助。我们衷心感谢 TÜBİTAK 对本研究的支持。感谢 Matthew Gelvin、Barış Coşkunüzer、Caroline Yalçın 和 Koray Karabina 阅读论文的早期版本并提出宝贵意见。论文部分内容是在第二作者休假访问滑铁卢大学期间撰写的。第二作者感谢滑铁卢大学在其休假期间提供的热情接待。

参考文献

[1] Sheldon Axler。《线性代数应该这样学》。施普林格，第 3 版，2015 年。[2] Gunnar Carlsson。拓扑学与数据。《美国数学会通报》，46:255–308，2009 年。[3] Frédéric Chazal 和 Bertrand Michel。拓扑数据分析导论：面向数据科学家的基础与实践。《人工智能前沿》，4:667963，2021 年。[4] Chao Chen 和 Michael Kerber。利用扭转进行持久同调计算。载于《第 27 届欧洲计算几何研讨会论文集（EuroCG 2011）》，第 1–4 页，2011 年。[5] Baris Coskunuzer 和 Cüneyt Gürcan Akçora。机器学习中的拓扑方法：面向实践者的教程，2024 年。[6] Tamal K. Dey 和 Yusu Wang。《数据分析计算拓扑学》。剑桥大学出版社，2022 年。[7] Herbert Edelsbrunner 和 John Harer。《计算拓扑学：导论》。美国数学会，2010 年。[8] Herbert Edelsbrunner、David Letscher 和 Afra Zomorodian。拓扑持久性与简化。《离散与计算几何》，28:511–533，2002 年。[9] Barbara Giunti、Guillaume Houry 和 Michael Kerber。团过滤的矩阵约化平均复杂度。载于《2022 年国际符号与代数计算研讨会论文集》，ISSAC ’22，第 187–196 页，纽约州纽约市，美国，2022 年。计算机协会。[10] Dmitriy Morozov。最坏情况下，持久性算法需要三次时间。《生物几何新闻》，2005 年 2 月。杜克大学计算机科学系。[11] Dmitriy Morozov 和 Primoz Skraba。矩阵乘法时间内的持久（上）同调，2024 年。[12] Nina Otter、Mason A. Porter、Ulrike Tillmann、Peter Grindrod 和 Heather A. Harrington。持久同调计算路线图。《EPJ 数据科学》，6(1):17，2017 年。[13] Raul Rabadán 和 Andrew J. Blumberg。《基因组学与进化的拓扑数据分析：生物学中的拓扑学》。剑桥大学出版社，剑桥和纽约，2020 年。[14] Thomas S. Shores。《应用线性代数与矩阵分析》。《数学本科生教材》。施普林格，第 2 版，2018 年。[15] Afra Zomorodian 和 Gunnar Carlsson。计算持久同调。《离散与计算几何》，33:249–274，2005 年。

附录 A. 构造条带示例的算法

构造过滤单纯复形条带 $X(n)$（其中 $n \geq 2$）的算法如下。复形 $X(1)$ 可以通过插入图中列出的单形轻松构造。

算法4 构建单纯复形的顶点、边和三角形的有序列表
$X(n)$ 对于 $n \geq 2$

函数构建顶点(n)
    返回 $[f_1, \ldots, f_n, v_1, \ldots, v_{n+2}]$
结束函数
函数构建边(n)
    $E \leftarrow [\,]$
    ▷ 水平基边 $n', \ldots, 1', 0'$
    对于 $i = n, n-1, \ldots, 3$ **do**
        将 (v_{i-2}, v_{i+2}) 追加到 E ▷边 i'
    结束循环
    将 $(v_1,\ v_4)$ 追加到 E ▷边 $2'$
    将 $(v_2,\ v_3)$ 追加到 E ▷边 $1'$
    将 $(v_n,\ v_{n+2})$ 追加到 E ▷边 $0'$
    ▷ 鳍边 $-n^*, \ldots\ldots, -1^*$
    对于 $i = n, n-1, \ldots\ldots, 2$ **do** ▷ 边 $-i^*$
        将 $(f_i,\ v_{i-1})$ 追加到 E
    结束循环
    将 (f_1, v_1) 添加到 E ▷ 边 -1^*
    ▷ 鳍边 $-n, \ldots, -1$
    对于 $i = n, n-1, \ldots, 1$ **do** ▷ 边 $-i$
        将 (f_i, v_{i+1}) 追加到 E
    结束循环
    ▷ 垂直底边 $n, \ldots, \mathbf{1}$
    对于 $i = n, n-1, \ldots, 2$ **do**
        将 (v_{i-1}, v_{i+1}) 追加到 E ▷边 i
    结束循环
    将 (v_1, v_2) 追加到 E ▷边 $\mathbf{1}$
    返回 E
结束函数
函数构建三角形(n)
    $T \leftarrow [\,]$
    ▷ 基三角形 ①, ②, ……, ⓝ
    将 (v_1, v_2, v_3) 追加到 T ▷ 三角形 ①
    将 (v_1, v_2, v_4) 追加到 T ▷ 三角形 ②
    对于 $i = 3, \ldots, n$ **do**
        将 (v_{i-2}, v_i, v_{i+2}) 追加到 T ▷ 三角形 ⓘ
    结束循环
    ▷ 鳍三角形 -1, -2, . . . , -n
    将 $(f_1,\ v_1,\ v_2)$ 添加到 T ▷三角形 -1
    对于 $i = 2, \ldots, n$ **do**
        将 $(f_i,\ v_{i-1},\ v_{i+1})$ 追加到 T ▷三角形 -i
    结束循环
    返回 T
结束函数

算法5 为 $n \geq 2$ 构建过滤单纯复形 $X(n)$

函数创建最坏情况复形(n)
    $V \leftarrow$构建顶点(n)
    $E \leftarrow$构建边(n)
    $T \leftarrow$构建三角形(n)
    $\mathcal{K} \leftarrow [\,]$
    $t \leftarrow 1$
    对每个 $v \in V$ **执行**
        将（$\{v\}$，t）追加到 $\mathcal{K}$
        $t \leftarrow t + 1$
    结束循环
    对每个 $e \in E$ **执行**
        将（sort(e) t）追加到 $\mathcal{K}$
        $t \leftarrow t + 1$
    结束循环
    对每个 $\sigma \in T$ **执行**
        将 (sort(σ) t) 追加到 $\mathcal{K}$
        $t \leftarrow t + 1$
    结束循环
    返回 $\mathcal{K}$
结束函数

Uzay Çetin，数学系，俄亥俄州立大学，哥伦布，俄亥俄州，43210- 1174，美国电子邮件地址：cetin.16@osu.edu

Ergün Yalçın，比尔肯特大学数学系，土耳其安卡拉 06800 Bilkent电子邮箱：yalcine@fen.bilkent.edu.tr