Alon-Tarsi 列表着色方法的高效实现与强化

Zdeněk Dvořák ！ 查理大学计算机科学研究所（CSI），布拉格，捷克共和国

摘要

作为多项式方法在组合学中的首批应用之一，Alon 和 Tarsi 提出了一种证明图具有可选择性的方法（即能够从任意给定大小的列表中着色）。我们描述了一种高效实现该方法的方式，使得测试具有约 70 条边的图的可选择性成为可能。我们还证明，当 Alon-Tarsi 方法无法证明图具有可选择性时，图多项式的其他系数会对无法为图着色的列表赋值提供约束。这通常使我们能够确认图在给定列表赋值下的可着色性，或仅通过测试少数几个列表赋值来判定其可选择性。该实现可在 https://gitlab.mff.cuni.cz/dvorz9am/alon-tarsi-method 找到。

2012 年 ACM 学科分类 计算理论 → 应用领域的理论与算法

关键词与短语 列表着色，可约性，多项式方法

资助 *Zdeněk Dvořák*：获捷克科学基金会项目 22-17398S（曲面图中的流与圈）资助。

arXiv:2301.06571v1 [cs.DM] 16 Jan 2023

1 引言

列表着色（最初由 Vizing [13]以及 Erdős 等人 [8]提出）是普通图的正常着色的一种推广，它自然地出现在可着色性的归纳证明中，同时自身也拥有丰富而有趣的理论。对于图 G，列表赋值是一个系统 $L = \{L_v : v \in V(G)\}$，它为 G 的各个顶点分配有限集合，指定可用于给每个顶点着色的颜色。一个 L-着色是一个函数 $\varphi \in \bigtimes_{v \in V(G)} L_v$，使得对于每条边 $uv \in E(G)$，都有 $\varphi(u) \neq \varphi(v)$。特别地，如果 L 为所有顶点赋予相同的、含有 k 种颜色的列表，那么 L-着色恰好就是使用 k 种颜色的正常着色。若图 G k-可选着色，即对于每个列表赋值 L，只要对每个 $v \in V(G)$ 都有$|L_v| = k$，G 就存在一个 L-着色。显然，k-可选着色的图是 k-可着色的，但反过来并不成立——对于每个 k，都存在一个不是 k-可选着色的二部图。此外，尽管所有平面图都可用 4 种颜色着色 [2, 3]，，但仍存在不可进行 4-可选着色的平面图 [14]（不过所有平面图都是 5-可选着色的 [12]）。

正如我们所提到的，通过可约构形方法进行的可着色性归纳证明中，自然会出现列表着色。在这种情形下，每个顶点的列表大小可以不同，这一点很有必要。对于函数 $s : V(G) \to \mathbb{N}$，s-列表赋值是指满足以下条件的列表赋值 L：对于每个顶点 $v \in V(G)$，都有 $|L_v| \geq s(v)$；如果对于每个 s-列表赋值 L，G 都是 L-可着色的，那么我们称 G 是 s-可选择的。可约构形方法可概括如下。设 G 是一个对子图封闭的图类，并且我们要证明 G 中的所有图都是 k-可着色的。假设 C 是图 $G \in$ G 的一个诱导子图，并且对于 $v \in C$，令 $s'(v) = k - (\deg_G v - \deg_C v)$。如果 C 是 s'-可选择的，那么要证明 G 是 k-可着色的，只需证明 $G - V(C)$ 是 k-可着色的。事实上，给定 $G - V(C)$ 的一个 k-着色 φ，对于每个 $v \in V(C)$，令 L_v 为在 v 的邻域上未使用的颜色集合。显然，对于每个 $v \in V(C)$，都有 $|L_v| \geq s(v)$，因此 C 是 L-可着色的。此外，L 的选取保证了 C 的任意 L-着色都能与 φ 合并，得到整个图 G 的一个适当的 k-着色。因此，如果我们证明 G 中的每个图都包含

一个具有此性质的诱导子图，这就表明 $\mathcal{G}$ 中的所有图都是 k-可着色的，实际上甚至是 k-可选择的。大量论文采用了这种方法或其变体；一般性介绍见 [6]。

要进行这类论证，需要能够验证对于给定的函数 $s\colon V(C) \to \mathbb{N}$，图 C 是否具有 s-可选择性。只要使用的可约构形不多，就可以通过临时的办法证明这一点。然而，在更复杂的证明中，可约构形的数量可能达到数百甚至数千，手动检查它们的可选择性并不可行。因此，有必要找到一种自动检查可选择性的方法。对于给定的列表赋值 L，判断给定的图 C 是否 L-可着色当然是 NP 困难，而且目前已知（或被认为存在）的算法都没有亚指数时间复杂度；不过，这实际上并不是什么大问题：对于含有 10–20 个顶点的典型构形，标准的指数时间着色算法（甚至通用的 CSP 或 SAT 求解器）足以在合理时间内找到一个 L-着色。然而，要验证 s-可选择性，就必须遍历所有可能的 s-列表赋值，其数量增长非常快[1]。例如，为 n 个顶点指定大小为 1 的列表，其方式数就等于贝尔数 $B_n \geq \exp(\Omega(n\log\, n))$；当 $n = 15$ 时，该数超过 10^9。因此，通常不会精确判定可选择性，而是采用保守启发式方法：这些方法要么正确判定图 C 具有 s-可选择性，要么判定失败，不提供任何有关 C 可选择性的信息。

一种流行的启发式方法基于多项式方法，由 Alon 和 Tarsi [1]提出。对于图 G，变量 x_v 上的图多项式 P_G（对应于 $v \in V(G)$）定义如下：任意选取 G 的一个定向 $\vec{G}$，并令

$$P_G = \prod_{(u,v)\in E(\vec{G})} (x_v - x_u).$$

注意，图多项式的定义唯一到一个符号为止。全文中，诸如“G 的图多项式是 p”这样的表述，意为“存在G 的一个定向，使得上述定义得到 p（而对于任何其他定向，所得结果要么是 p，要么是 $-p$）”。对于指标集 I 和函数 $f\colon I \to \mathbb{N}$，定义 $x^f = \prod_{i\in I} x_i^{f(i)}$。对于变量 x_i 上的多项式 q（对应于 $i \in I$），令 $[x^f]\, q$ 表示单项式x^f 在 q中的系数。变量 x_i 在 q 中的度 $\deg_{x_i} q$ 为 $\max\{f(i)\colon f \in \mathbb{N}^I,\ [x^f]\, q \neq 0\}$。对于$\varphi : I \to \mathbb{R}$，令 $q(\varphi)$表示 q 在 φ 处的取值，即

$$q(\varphi) = \sum_{f\in\mathbb{N}^I} [x^f]\, q \cdot \prod_{i\in I} \varphi(i)^{f(i)}.$$

图多项式通过以下简单的观察与列表着色相关联 .

▶ **观察 1**。设 $L = \{L_v \subset \mathbb{R}\colon\ v \in V(G)\}$ 是对 G 的顶点进行的列表分配。图 G 当且仅当存在 $\varphi \in \bigtimes_{v\in V(G)} L_v$ 使得 $P_G(\varphi) \neq 0$ 时是 L-可着色的。

Alon-Tarsi 可选择性方法基于以下代数命题，该命题称为组合零点定理。

[1] 我们要指出的是，当构型非常小时，遍历所有相关的 s-列表赋值可能是可行的，而我们只需证明 k-着色的可约性；也就是说，我们只需研究由 {1、.....、k} 的子集组成的列表赋值，并且可以利用颜色之间可互换这一事实来减少可能选择的数量。

▶ **定理 2**。 设 $q \neq 0$ 是变量 x_i 上的多项式，其中 $i \in I$，并设 $\{S_i \subset \mathbb{R} : i \in I\}$ 是有限集系统。如果对于每个 $i \in I$都有 $\deg_{x_i} q < |S_i|$，则存在 $s \in \bigtimes_{i\in I} S_i$ 使得 $q(s) \neq 0$。

稍作补充推导，便可得到以下结论。

▶ **定理 3.** 设 G 是一个图，且 $s, f\colon V(G) \to \mathbb{N}^{V(G)}$ 是函数，使得$f(v) < s(v)$ 对每个 $v \in V(G)$都成立。若 $[x^f]\, P_G \neq 0$，则 G 是 s-可选择的。

下面举一个简单的例子，说明如何应用定理 3。

▶ **例 4.** 当 n 为偶数时，顶点为 $0, \ldots, n-1$的 n-环的图多项式为 $2x_0 \cdots x_{n-1} + \ldots$；因此，对每个 $v \in V(C_n)$令 $f(v) = 1$，定理 3 表明 C_n是 2-可选择的。

当然，要利用定理 3 证明可选择性，我们需要能够计算图多项式的相关系数。直接计算需要遍历$2^{|E(G)|}$ 种可能，因此对于边数超过约 30 的图来说并不实际。作为我们的首项贡献，我们在第 2 节描述一种算法，用于查找相关系数；该算法在实际应用中效率更高。我们评估了这一算法的一个实现，并讨论了形成该算法的各种选择以及可能的改进。

定理 3 是一种保守的启发式方法，也就是说，如果 P_G的所有相关系数都为零，那么它无法告诉我们 G 是否可 s-选择着色。作为第二项贡献，我们利用图多项式的更多系数，来获取有关哪些列表赋值会导致 G 无法着色的信息。对于列表赋值$\{L_v : V \in V(G)\}$，颜色 c 的特征向量 是函数 $\chi_{c,L} : V(G) \to \{0,1\}$，满足

$$\chi_{c,L}(v) = \begin{cases} 1 & \text{if } c \in L_v \\ 0 & \text{if } c \notin L_v \end{cases}$$

对于每个 $v \in V(G)$。对于 $v \in V(G)$，令 $1_v : V(G) \to \{0,1\}$ 由以下方式定义：对每个 $u \in V(G) \setminus \{u\}$，令 $1_v(v) = 1$，且 $1_v(u) = 0$。

▶ 定理 *5.* 设 G 为一个图，设 $L = \{L_v \subset \mathbb{R} : v \in V(G)\}$ 是分配给 G各顶点的列表赋值，并假设 $f : V(G) \to \mathbb{Z}$ 满足：对每个 $v \in V(G)$，均有 $f(v) < |L_v|$，且$\sum_{v\in V(G)} f(v) = |E(G)| - 1$。如果 G 不可 L-着色，那么对于每种颜色 c，

$$\sum_{z\in V(G)} [x^{f+1_z}]\, P_G \cdot \chi_{L,c}(z) = 0.$$

定理 5 可通过对 Alon 和 Tarsi [1], 所给定理 3 的证明作直接修改而得证，我们将其推迟到第 3 节再证明，并在该节讨论实现细节和测试结果。在此，我们先给出几个可由定理 5 得出的结论示例。

▶ **例6.** 顶点按顺序为 0, . . . , 4 的五圈 C_5 的图多项式为

$$\begin{aligned} & x_0x_1x_2x_3(x_0 - x_1 + x_2 - x_3) \\ +& x_0x_1x_2x_4(-x_0 + x_1 - x_2 + x_4) \\ +& x_0x_1x_3x_4(x_0 - x_1 + x_3 - x_4) \\ +& x_0x_2x_3x_4(-x_0 + x_2 - x_3 + x_4) \\ +& x_1x_2x_3x_4(x_1 - x_2 + x_3 - x_4) \\ +& \ldots \end{aligned}$$

将定理 5 应用于 $f(1,1,1,1,0), (1,1,1,0,1), (1,1,0,1,1), (1,0,1,1,1)$, 和 $(0,1,1,1,1)$，我们得出结论：每种颜色的特征向量 χ 都满足以下线性方程组：

$$\begin{aligned}
\chi(0)-\chi(1)+\chi(2)-\chi(3)&=0\\
-\chi(0)+\chi(1)-\chi(2)+\chi(4)&=0\\
\chi(0)-\chi(1)+\chi(3)-\chi(4)&=0\\
-\chi(0)+\chi(2)-\chi(3)+\chi(4)&=0\\
\chi(1)-\chi(2)+\chi(3)-\chi(4)&=0
\end{aligned}$$

该系统在$\{0,1\}^{V(C_5)}$中只有两个解，即 $\chi=(0,0,0,0,0)$ 或 $\chi=(1,1,1,1,1)$。也就是说，如果 C_5 不是 L-可着色的，并且对每个 $v\in V(C_5)$都有 $|L_v|=2$，那么 L 中使用的每种颜色都必须出现在所有列表中，因此 $L_0=L_1=L_2=L_3=L_4$（并且我们可以验证，按照这样的列表赋值，G 确实不是 L-可着色的）。

▶ **例 7.** 设 G 由路径 1234 以及一个与该路径所有顶点都相邻的顶点 0 构成，并考虑一个列表赋值 L，使得 $|L_0|=4$ 且 $|L_1|=|L_2|=|L_3|=|L_4|=2$。$G$ 的图多项式为

$$\begin{aligned}
&x_0^2x_1x_2x_3x_4(x_1-x_2+x_3-x_4)\\
+&x_0^3x_2x_3x_4(-x_0+x_2+x_4)\\
+&x_0^3x_1x_3x_4(x_0-x_1-x_4)\\
+&x_0^3x_1x_2x_4(-x_0+x_1+x_4)\\
+&x_0^3x_1x_2x_3(x_0-x_1-x_3)\\
+&\dots
\end{aligned}$$

因此，定理 5 给出了特征向量χ对应的以下线性方程组。

$$\begin{aligned}
\chi(1)-\chi(2)+\chi(3)-\chi(4)&=0\\
-\chi(0)+\chi(2)+\chi(4)&=0\\
\chi(0)-\chi(1)-\chi(4)&=0\\
-\chi(0)+\chi(1)+\chi(4)&=0\\
\chi(0)-\chi(1)-\chi(3)&=0.
\end{aligned}$$

这意味着 $\chi(1)=\chi(2)$，$\chi(4)=\chi(3)$，且 $\chi(0)=\chi(2)+\chi(3)$。由于 $\chi(0)\le 1$，这意味着 $\chi(2)$ 和 $\chi(3)$ 不可能同时为 1，因此，如果 G 不是 L-可着色的，那么 $L_2\cap L_3=\emptyset$。因此，G 是 L-可着色的，当且仅当从 G 中删除边 23 后得到的图 G' 是L-可着色的。

方程组在 $\{0,1\}^5$ 中的非零解为 $(1,1,1,0,0)$ 和 $(1,0,0,1,1)$。只有一种方法可以组合这些向量以得到指定的列表大小 $(4,2,2,2,2)$，这意味着如果 G 不是 L-可着色的，那么（在颜色置换的意义下）$L_1=L_2=\{a,b\}$，$L_3=L_4=\{c,d\}$，且 $L_0=\{a,b,c,d\}$。我们可以轻松验证，G 确实不是 L-可着色的。

▶ **例 8.** 设 G 为中心为 0、辐条为 $1,\dots,5$ 的轮图，并设 $s(0)=5$,$s(1)=s(2)=s(3)=2$ 和 $s(4)=s(5)=3$。P_G 中度小于 s 给出的度数的所有单项式均为零，因此定理 3 对.....没有给出任何结论

s-可选择性。利用定理 5 确定的线性方程组所满足的非零特征向量只有 (1, 1, 1, 1, 0, 0) 和 (1, 0, 0, 0, 0, 1, 1)。这只给出一种可能的 s-列表赋值 L，使得 G 可能无法 L-着色（在颜色置换意义下），即 $L_1 = L_2 = L_3 = \{a, b\}$，$L_4 = L_5 = \{c, d, e\}$，以及$L_0 = \{a, b, c, d, e\}$。然而，我们很容易验证 G 是 L-可着色的，因此 G 是s-可选择的。

或者，注意到颜色的特征向量表明，任何满足 $L_3 \cap L_4 = \emptyset$ 和 $L_5 \cap L_1 = \emptyset$ 的 s-列表赋值 L，若 G 不可 L-着色，则图 G 可 L-着色，当且仅当删除边 34 和 51 后得到的图 G' 可 L-着色。然而，定理 3 表明 G' 可 s-选择着色，因此 G 也可 s-选择着色。

▶ **例 9.** 令 G 为一个轮图，其中心为 0，辐条为 1, . . . , 5，并增加顶点 6（与 1 和 2 相邻）及顶点 7（与 2 和 3 相邻）；赋予其列表 L，使得 $|L_1| = |L_2| = |L_3| = 3$、$|L_4| = |L_5| = |L_6| = |L_7| = 2$ 和 $|L_0| = 5$。若 G 不可 L-着色，则根据定理 5，颜色的可行特征向量为 (0，0，0，0，0，0，0) 和 (1，1，1，1，1，0，0)。显然，无法用具有这些特征向量的颜色组成 L，因此 G 可 L-着色。

因此，在定理 3 不成立的情况下，定理 5 的结果可以通过多种方式为图 G 的 L-可着色性提供有趣的信息（可行特征向量是指利用定理 5 得到的线性方程组中，属于$\{0,1\}^{V(G)}$ 的解）：

- 可能不存在由可行特征向量构成、且系数为非负整数的线性组合，能够等于表示列表大小的向量 s，这表明G 是 s-可选择的（例 9）。也可能只有少数几种方式能够组合可行特征向量，从而得到大小符合要求的列表；通过逐一考察相应的s-列表指派，我们可以证明 G 是 s-可选择的（例 8），或找出所有使得 G 不可 L-着色的 s-列表指派 L（例 6 和例 7）。还可以利用特征向量得出结论：如果 G 无法通过某个 s-列表指派 L实现 L-着色，那么相邻顶点 u 和 v 所分配的列表必定不相交（例 8 和例 7）。因此，G 是 L-可着色的，当且仅当 $G - uv$ 是 L-可着色的。这可能正是我们所需要的（例如，这足以执行上述可约构形方法描述中概述的论证，只需不删除整个可约构形，而只删除这条边 uv）。或者，我们可以迭代处理 $G - e$，并得出它是 s-可选择的结论（例 7），从而 G 也同样是s-可选择的。

与定理 3 不同，利用定理 5，我们可以根据特定的 s-列表分配 L，得出图 G 是 L-可着色的结论，即使 G 不是 s-可选择的。或许可以利用这一点，加快寻找 L-着色的回溯算法：每当我们进行分支时，就用定理 5 的条件检查其中某个分支是否能保证图的其余部分可以用它们仍可用的颜色列表着色。如果可以，我们就只继续这个分支，并剪除其余分支。不过需要注意的是，这项额外测试的开销可能超过缩减搜索树所带来的收益。

当然，也存在一些图（甚至相当简单的图），定理 5 对它们无法提供任何信息。通过考察图多项式的更多系数，可以进一步拓展这一方法；我们将在第 4 节中对此进行更详细的讨论。

2 Alon-Tarsi 方法的高效实现

在定理 3 的理论应用中，人们通常会猜测图多项式中哪些单项式（或哪些单项式）是相关的，然后证明其中至少有一个的系数非零；这可能会用到 Alon 和 Tarsi [1] 已经给出的系数的以下标准组合解释。对于函数 $f\colon V(G) \to \mathbb{N}$，$f$-定向 G 是指对 G 的边进行定向，使得每个顶点 $v \in V(G)$ 的出度恰好为 $f(v)$。给定 G的一个固定参考定向，与参考定向不同、且在 m 条边上方向相反的另一种定向 G，其符号 为 $(-1)^m$。

▶ **引理 10**。设 G 是一个图。对于任意 $f\colon V(G) \to \mathbb{N}$，$G$ 的图多项式中 x^f 处的系数等于 G的 f-定向的符号之和。

定向的符号之和乍看之下与行列式的定义相似，因此人们可能希望能够在多项式时间内计算它。然而，情况似乎并非如此。注意，对于二部图，所有 f-定向的符号都相同（因为它们之间的差异仅在于翻转欧拉子图的边）。考虑一个 4-正则图 G，并令 G' 为将 G 的每条边细分一次后得到的图。设 f 为 $V(G)$ 的每个顶点赋值 2，并为 $V(G') \setminus V(G)$ 的每个顶点赋值 1。则 $|[x^f]\, P_{G'}|$ 是 G'的 f-定向数量，这等于 G的欧拉定向数量。然而，计算一个 4-正则图的欧拉定向数量是 #P-困难的 [11],，即使对于平面图也如此 [10]。因此，对于最大度数为四的二部平面图，计算图多项式的系数是 #P-困难的。

要应用定理 3，实际上只需判断相关系数中是否有一个非零，而不必精确计算它们；我们也没有排除这一判断可以在多项式时间内完成的可能性。然而，对于一般图而言，这似乎不太可能[2]。此外，要应用定理 5，我们实际上需要确定这些系数的值。

因此，我们将注意力转向这样一个问题：设计一种虽需指数时间、但在实践中有用的算法，用于计算图多项式的相关系数。我们先讨论一些方案，再逐步引出所选的方法。

2.1 直接枚举

要确定某个特定系数 $[x^f]\, P_G$，一种可能考虑的首要方法是直接枚举图 G的所有 f-定向。虽然这种方法的效率还不够高，但在继续介绍一种更实用的算法之前，我们先深入讨论一下。

可以在多项式时间内，针对每个 f-定向枚举图 G 的所有 f-定向。为此，只需注意到：通过简单地归约为辅助二分图中的最大匹配问题（该辅助二分图由细分尚未定向的边，并将每个原始顶点 v 扩展为 $f(v)$ 减去当前出度的 v 个顶点而得到），即可在多项式时间内判断一个部分定向能否扩展为 f-定向。有了这个子程序，我们可以通过回溯，尝试以两种方式定向 G 的每条边，并立即剪去无法导向至少一个 f-定向的分支。

[2] 不过值得一提的是，对于二部图，由于所有 f-定向的符号都相同，这个问题便简化为判断是否存在一个图 f的 f 函数定向，使得对每个顶点 v，均满足$f(v) < s(v)$；通过将问题标准地归约为二部图中的最大匹配，即可在多项式时间内解决。

实际上，分支很少会被剪去，因此运行可扩展性测试所带来的开销并不值得。事实证明，更有效的做法是只剪去显然无用的分支（即某个顶点 v 的出度超过 $f(v)$ 的分支），并结合适当的选边规则来决定要定向哪条边（例如，选择与当前出度最接近 $f(v)$ 的顶点v 相连的边；尤其优先选择定向已被确定的边，因为这些边与出度等于 $f(v)$ 的顶点 v 相连）。

这种方法的一个问题是，通常，满足列表条件的 f定向数量会随着边数快速增长。此外，我们可能不得不遍历所有受列表大小限制的 f 可能取值。

▶ **例 11.** 考虑一个团 K_n，其列表大小为 $n-1$。要验证定理 3 不适用于这种情况（它确实不适用，因为 K_n 不可（$n-1$）-着色），我们必须计算所有满足以下条件的函数 f对应的系数 $[x^f]\,K_n$：对于每个 $v \in V(K_n)$，$f(v) \leq n-2$。为此，我们要列出 K_n 的所有定向，但排除那些含有出度为 $n-1$的顶点的定向；此类定向的数量为

$$2^{|E(K_n)|} - n2^{|E(K_n)|-n+1} = 2^{|E(K_n)|}\big(1 - \frac{n}{2^{n-1}}\big).$$

即使实现得非常高效，当边数超过约 40 时，这种方法也会变得慢到难以实际使用。

例 11 基本上是直接枚举算法的最坏情况。尽管如此，我们的实验表明，即使输入没有那么人为构造，对于这个规模的图，该算法也会变得相当缓慢（处理一个有 40 条边的图需要几分钟）。值得一提的是，逐个计算系数有一个小小的优点：一旦找到一个非零系数，我们就可以立即停止；如果答案为肯定，这会提高该方法的效率。

2.2 截断乘法

一种更高效的方法源于一个简单的观察：如果我们一次性计算所有相关系数，就可以逐条处理边。对于两个函数f，s：$I \to \mathbb{N}$，如果对每个 $i \in I$都有 $f \prec s$ $f(i) < s(i)$，则记作 $f \prec s$。对于变量$\{x_i : i \in I\}$ 上的多项式 q以及函数 s：$I \to \mathbb{N}$，令

$$\mathrm{trunc}_s(q) = \sum_{s' \prec s} [x^{s'}]\, q \cdot x^{s'},$$

也就是说，多项式截断$_s(q)$是从 q 中截取得到的，只保留每个变量 x_i 的度小于 $s(i)$的单项式。

▶ **观察 12。** 对于变量$\{x_i : i \in I\}$上的任意多项式 q 和 m 以及函数s：$I \to \mathbb{N}$，

$$\mathrm{trunc}_s(q \cdot m) = \mathrm{trunc}_s(\mathrm{trunc}_s(q) \cdot m).$$

特别地，定理 3 表明，如果截断$_s(P_G)$ $\neq 0$，那么 G 是 s-可选的；而观察12表明，我们可以将截断$_s(P_G)$通过逐个乘以 $uv \in E(G)$的项（$x_v - x_u$）并在每次乘法后执行截断来计算。在实现中，当当前截断多项式为 q 时，我们按如下方式处理边 uv：对于 q 中满足 $a \neq 0$的每个单项式 $a \cdot x^f$，

(i) 如果 $f(v) + 1 < s(v)$，则将 $a \cdot x^{f+1_v}$ 添加到输出中；
(ii) 如果 $f(u) + 1 < s(u)$，则将 $-a \cdot x^{f+1_u}$ 添加到输出中。

当然，我们将多项式 q 表示为一张表，其中每个度向量 f 映射到系数 $[x^f]\,q$。因此，这只需遍历表示 q的表，并将各个值加到表示输出的表中。请注意，系数可能（而且经常会）相互抵消——如果 $[x^{f-1_u}]\,q = [x^{f-1_v}]\,q$，那么输出中 x^f 处的系数贡献之和为零，因此需要从输出中删除该单项式。

▶**例 13.** 为了与直接枚举进行快速的定性比较，我们考虑团 K_n，其列表大小为 $n-1$。满足以下条件的函数数量f： $V(K_n) \to \mathbb{N}$，使得 $f(v) \le n-2$对每个 $v \in V(K_n)$ 都成立，至多为 $(n-1)^n = 2^{n\log^2 n}$。假设可用内存达到 GB 量级，只要 $n\log_2 n$ 不超过约 30，我们就能存储相应的系数；也就是说， n 最多约为 10，此时 K_n 有 45 条边。

这当然是一个粗略的高估，没有考虑到我们并非一次存储所有系数这一事实（处理完 b 条边后，每个单项式的度函数 f 满足 $\sum_{v\in V(G)} f(v) = b$），也没有考虑到有些单项式的系数为零，因此可以将其丢弃。在我们的实验中，采用截断乘法方法的直接实现处理边数约为 50–60 的图时，开始出现内存不足。此时程序运行时间还不到一分钟，因此时间并非限制因素。我们将在第 2.4 节介绍一种降低内存需求的方法。在此之前，先作几点说明。

与直接枚举方法类似，除了删除系数为零的单项式外，我们还可以删除这样的单项式：它与图多项式其余各项相乘后，不可能得到各变量度数均不超过 s 的单项式。具有这一性质的单项式可以通过同样的归约方法识别，即归约为二分图中的最大匹配问题。然而，我们的测试表明，增加这项额外开销并不值得，除非是在某些极其罕见的情况下；详情见第 2.6 节。

还需指出，与直接枚举方法不同，我们无法提前终止，因为只有在处理完所有边之后，才能一次性计算出非零系数。因此，在“易于着色的图”中，截断乘法方法可能效率较低，因为这类图的图多项式中有许多适用的非零系数。第 2.4 节提出的改进主要旨在降低内存需求，同时也缓解了这一问题。

在介绍这项改进之前，我们先说明截断乘法算法的高效实现中的一个重要细节。

2.3 多项式的表示

为了执行截断乘法算法，我们需要能够将特定单项式的系数加到输出多项式中。因此，将多项式表示为一种关联数组似乎很自然：该关联数组将单项式的度向量映射到其系数。然而，实现关联数组的各种标准方式（哈希表、搜索树、字典树等）都会带来显著的时间和空间开销。幸运的是，还存在一种更加简单、高效的替代方案。

令 $<$ 为从 $V(G)$ 到 $\mathbb{N}$的函数的字典序；也就是说，我们任意固定 G 的顶点顺序$v_1, \dots, v_n$，并且当存在 $a \in \{1, \dots, n\}$ 使得对于 $i \in \{1, \dots, a-1\}$，$()()$，且 $()()$ 时，记 。我们将多项式存储为所有满足 的有序对 $(, c)$ 的列表，并按照 f 以递增字典序排序。关键观察如下：若 $f_1 < f_2$，那么对于任意顶点

$v \in V(G)$，$f_1 + 1_v < f_2 + 1_v$。也就是说，如果我们按照输入多项式中单项式的度向量递增字典序处理它们，那么截断乘法算法的第 (i) 部分加入输出多项式的单项式也会按递增字典序生成，第 (ii) 部分生成的单项式也同样如此。因此，我们可以像归并排序那样合并 (i) 和 (ii) 中生成的系数。

也就是说，采用有序列表表示时，截断乘法算法可按如下方式实现：设 (f_u, c_u) 和 (f_v, c_v) 是输入列表中满足 $f_u(u) + 1 < s(u)$ 且 $f_v(v) + 1 < s(v)$ 的最早元素。重复执行以下操作，直到这两个元素都到达输入列表末尾：

- 如果 $f_u + 1_u = f_v + 1_v$，则：如果 $c_u \neq c_v$，将 $(f_v + 1_v, c_v - c_u)$ 添加到输出末尾。将 (f_u, c_u) 和 (f_v, c_v) 前进到满足 $f_u(u) + 1 < s(u)$ 且 $f_v(v) + 1 < s(v)$ 的下一个输入元素。否则，如果 $f_u + 1_u > f_v + 1_v$，则将 $(f_v + 1_v, c_v)$ 添加到输出末尾，并将 (f_v, c_v) 前进到满足 $f_v(v) + 1 < s(v)$ 的下一个输入元素。否则，将 $(f_u + 1_u, -c_u)$ 添加到输出末尾，
- 并将 (f_u, c_u) 前进到满足 $f_u(u) + 1 < s(u)$ 的下一个输入元素。

由于这些列表只需按顺序访问，且我们只需在末尾添加元素，因此我们将它们简单地存储为数组；这不仅简洁，还带来了缓存友好的内存访问模式。为尽量减少内存需求，我们以紧凑方式存储度向量，仅占用 $\lceil \log_2 \max\{s(v) : v \in V(G)\} \rceil \cdot |V(G)|$ 位[3]。除了减少内存占用，这种方式还降低了将（修改后的）度向量从输入复制到输出的时间复杂度。

举例来说，与我们最初使用哈希表（`unordered_map` 来自标准 C++ 库）的实现相比，这种类似归并排序的方法使性能提升了约 10 倍。

2.4 顺序化（及并行化）

截断乘法算法的主要局限在于内存消耗——对于约有 50 条边的图，它通常就会消耗数 GB 内存。关键问题是，我们一次性计算截断图多项式的所有系数（以及中间多项式）。为避免这一问题并将工作拆分为更小的块，我们采用以下观察。若两个多项式 p 和 q 中不存在同时以非零系数出现的单项式，则称它们不相交。对于图 G的顶点集合 S，若对于每个 $v \in S$，两个向量 $f, f' : V(G) \to \mathbb{N}$满足 $f(v) = f'(v)$，则称它们是 S-等价的。若变量为 $\{x_v : v \in V(G)\}$的多项式 q 中，任意两个满足 $[x^f]\, q \neq 0$和 $[x^{f'}]\, q \neq 0$的向量 f和 f' 都是 S-等价的，则称该多项式为 S-齐次。变量为 $\{x_v : v \in V(G)\}$的多项式 p 的 S-划分，是满足 $p = p_1 + \ldots + p_m$的、由两两不相交的 S-齐次多项式 $p_1, \ldots, p_m$ 构成的最小系统。

▶ **观察 14**。设 H 是图 G 的生成子图，设 $S \subseteq V(G)$ 由满足 G 的所有关联边均属于 $E(H)$ 的顶点组成，并设 H' 是生成子图

[3] 实际上会稍多一些，因为我们会添加填充，以确保表示每个顶点度的位都存储在同一个内存字中，而不会在 $\lceil \log_2 \max\{s(v) : v \in V(G)\} \rceil$ 不能整除字宽时，可能被拆分到两个连续的内存字中。这样可以简化将 1_v 加到某个顶点 $v \in V(G)$ 的度向量等操作。

其边集为 $E(G) \,||\, E(H)$ 的 G。设 $s: V(G) \to \mathbb{N}$ 是任意函数，并设 $p_1, \ldots, p_m$ 是 $\mathrm{trunc}_s(P_H)$ 的 S-划分。对于 $i = 1, \ldots, m$，设 $q_i = \mathrm{trunc}_s(p_i \cdot P_{H'})$。则 $q_1, \ldots, q_m$ 是 $\mathrm{trunc}_s(P_G)$ 的 S-划分。

特别是，$\mathrm{trunc}_s(p_1 \cdot P_{H'})$、$\ldots$、$\mathrm{trunc}_s(p_m \cdot P_{H'})$ 两两不相交，因此这些截断乘法可以完全独立地执行。

这启发了以下算法。固定图G的顶点 $v_1, v_2, \ldots, v_n$的一个排序。从多项式 $p = 1$开始，依次对 $i = 1, 2, \ldots$，针对与 v_i关联且尚未处理的边应用截断乘法算法。处理完每个顶点 v_i 后，如果 p 中非零系数单项式的数量超过某个界 N，则停止此过程，

- 将 p划分为其$\{v_1, \ldots, v_i\}$-划分 $p_1, \ldots, p_m$，并对 $j = 1, \ldots, m$，从 p_j开始递归运行相同
- 的过程，处理由$\{v_{i+1}, \ldots, v_n\}$诱导的子图的边。

请注意，在上一节所述的 p 表示中，$\{v_1, \ldots, v_i\}$-等价单项式在列表中连续出现，因此我们可以轻松地将 p 划分为若干个 $\{v_1, \ldots, v_i\}$。对于标准 Alon-Tarsi 方法，如果某个分支中出现非零多项式，我们就可以立即停止整个过程，从而获得直接枚举算法的一个优点。对于扩展 Alon-Tarsi 方法（定理 5），我们需要收集所有分支的信息；我们将在第 3 节中对此作更详细的讨论。

可以从以下评估结果看出界限 N 的选择所产生的影响。该评估针对一个有 25 个顶点和 60 条边的非可选择图（由六个 K_5 副本沿不同顶点以类路径方式粘合而成，且列表大小等于顶点度）。

N	10^3	10^4	10^5	10^6	10^7
time	30 s	25 s	23 s	24.5 秒	25.5 秒

在测试机器上，N 的最优值似乎约为 10^5：当N较小时，我们需要更频繁地承担分支例程的开销，也无法充分利用高效的类似归并排序的顺序处理。当 N较大时，则无法将数据全部放入 L2 缓存。

需要指出的是，由于各分支中的计算彼此独立，因此可以并行执行，甚至可以采用分布式方式；我们没有实现这些改进（原因之一是我们主要关注扩展 Alon-Tarsi 方法，而在这种情况下，以非顺序方式合并不同分支所得的信息似乎并非易事）。

2.5 排序的选择

图的边应按什么顺序处理？显然，如果截断能尽早消去尽可能多的系数，就会更有利，因此我们应从最难定向的子图中的边开始，也就是说，从满足 H、 $\left(\sum_{v \in V(H)} s(v)\right) - |E(H)|$ 最小的子图中的边开始。因此，理想情况下，我们希望确定一个顶点排序 $v_1, \ldots, v_m$，使这类难以定向的子图尽早出现在排序中，然后按顺序处理 $a = 2, \ldots, m$，依次处理从 v_a 到 v_1、$\ldots$、v_{a-1} 的边。

遗憾的是，这与处理少数几个顶点所关联的所有边的需求不相容，而这正是上一节所述序列化所需要的。作为一个

VSEP 数据集		MD	MD+PROC	OVER	LIST	LIST+DEG	MDR
crit12	59%	61%	59%	64%	94%	85%	88%
平面	55%	61%	51%	49%	45%	45%	50%

表1 排序启发式方法比较。

极端例子，对于完全图 K_n，如果我们采用上一段所述的边的顺序，那么在考虑边 v_nv_1 时，处理完另外 $\binom{n-1}{2}$ 条边后，我们才会首次完成处理与某个顶点关联的所有边。

我们将讨论范围限制在上一节所述类型的边排序，即固定顶点 $v_1, \ldots, v_n$ 的排序，先处理所有与 v_1 相 incident 的边，再处理所有与 v_2 相 incident 的剩余边，依此类推。选择顶点排序时，我们现在力求平衡两个有些矛盾的目标：我们希望每个初始段 $v_1, \ldots, v_a$ 所诱导的子图尽可能难以定向，同时也要尽量减少尚未处理且与该初始段中的顶点相邻的顶点数，以限制与 $v_{a+1}, \ldots, v_n$ 对应的变量的度数选择数。

后一个目标表明，应按照顶点分离数最小的排序进行处理。然而，图的顶点分离数等于其路径宽度，因此精确求解该问题或将其近似到一个常数加性项以内都是 NP 困难的 [4]; 尽管路径宽度是固定参数可解的 [5], ，相应的算法在实践中并无实用价值。此外，采用这种排序可能与我们希望已处理部分的图尽可能稠密（从而尽可能难以定向）的目标相冲突。

因此，我们采用启发式方法选择排序；表1比较了我们测试的几种启发式方法。前六种启发式方法依次贪心地选择 $v_1, v_2, \ldots$，每次都按照以下某条规则选择顶点 v_i 。

- VSEP：使处理完 v_i后，至少有一个已处理邻点的顶点数量最少。在这些顶点中，选择度最小的顶点。MD：在 $G - \{v_1, \ldots, v_{i-1}\}$ 中选择度最小的顶点。MD+PROC：与 MD 相同，但在度最小的顶点中，优先选择在 $\{v_1, \ldots, v_{i-1}\}$ 中邻点最多的顶点。
- OVER：使离开初始段的边数最少，即使 $G - \{v_1, \ldots, v_{i-1}\}$ 中的度与$\{v_1, \ldots, v_{i-1}\}$中的邻点数之差最小。LIST：列表大小最小；若相同，则选择 $G - \{v_1, \ldots, v_{i-1}\}$中度最小的顶点。LIST+DEG：选择 $G - \{v_1, \ldots, v_{i-1}\}$中列表大小与度之和最小的顶点。
-
-

最后，MDR 取 MD 所得排序的逆序。

对于这些数据集，crit12 是一个包含 844030 个图的集合，每个图的列表大小至多为 23 条边，且不存在与定理 3 相关的非零系数（这是一个旨在生成环面上绘制的图的 5-可选择性障碍的项目的一部分）；planar 是一个有 16 个顶点和 37 条边的可选择平面图（带对角线的平面 3×4 网格，且有四个

[4] 图的顶点排序的顶点分离数 是满足以下条件的最小整数 k：对于每个顶点 v，排序中位于 v 之后且与 v 相邻，或位于 v 之前的顶点至多有 k 个。图的顶点分离数是其顶点的所有排序的顶点分离数中的最小值。

个额外顶点分别与第一行、最后一行、第一列和最后一列相邻；与外面关联的顶点的列表大小为三，其余顶点的列表大小为五）。对于第二个图，我们禁用了序列化，以消除在某个分支中找到第一个非零系数时提前停止所带来的（半随机）影响。在表中，我们列出了整个计算过程中得到的非零系数单项式数量，并将其与仅保留输入中顶点排序时得到的结果进行比较。

在其他几个测试图上，启发式方法的选择并未发挥任何作用（例如需要注意的是，在所有顶点的列表大小都相同的正则图上，除 VSEP 和 MDR 之外的所有启发式方法都会产生相同的排序）。基于这些实验，我们选择使用 MD+PROC 启发式方法来确定待处理顶点的顺序——除了在上述两个数据集上表现都相当好之外，它还具有改善顺序化粒度的优势。也就是说，请注意，在该算法中，只有在处理完离开某个顶点的所有边之后，我们才能检查系数数量是否超过界限 N。如果输入图是 d-退化的，则 MD+PROC（以及 MD）启发式方法保证在连续两次检查之间至多处理 d 条边。

需要指出的是，存在一些图，在这些图上 VSEP 启发式方法的表现显著优于 MD+PROC；以下小节末尾的示例对此有所说明。因此，在具体应用中，尝试不同排序的选择可能是值得的。

排序的选择似乎还有很大的改进空间。例如，对于给定的子图 H，或许可以找到一种足够高效的方法，计算或估算满足 H 具有f-定向的不同向量 $f \prec s$ 的数量。这样一来，自然可以选择排序中的第 i 个顶点 v_i，使得在由与 v_1，.....，v_i关联的边组成的子图中，这一数量最小。除了逐个选择顶点之外，在选择 v_i时，也可以考虑处理多个顶点所产生的影响。最后值得指出的是，我们可以自适应地选择排序，也就是说，在序列化的每个分支中都可能选择不同的排序。

2.6 测试结果

我们在多个测试用例上测试了所述算法的性能。测试是在一台配备 Intel Core i5-7200U 2.50GHz CPU 和 8GB 内存的机器上进行的。

- 上一节所述的 844030 个 crit12 集中的图均不存在与定理 3 相关的非零系数，处理这些图耗时 4 秒。由 a 个 K_b 以类路径方式通过不同顶点粘合而成的不可列表着色图，其
- 列表大小等于顶点度。我们针对 $b \in \{3, 4, 5, 6\}$ 以及 a 的多个取值进行了测试；对于给定的 b，运行时间随边数变化的情况（边数四舍五入至最接近的 5 的倍数）如表 2 所示。这反映了该方法在更稠密的图上的性能如何下降。我们还测试了若干类可列表着色图。由于找到合适的系数后会提前终止，这种情况下的耗时可能更受运气影响（即
- 顶点排序的选择及顺序化中相应分支是否能迅速导向一个非零系数）。

- 由 a 个 K_b ($b \geq 4$)副本沿不同顶点以类路径方式粘合而成的图，其中一个副本删除了一条边，且列表大小等于顶点度。这里列出一些我们得到的计时结果

b \ 边	45	50	55	60	65	70
3	<0.01 秒	0.01 秒	0.08 秒	0.12 秒	0.3 秒	1 s
4		0.15 秒	0.7 秒	4 s	26 s	186 秒
5		1 s		23 s		703 秒
6	0.5 秒			50 s		

表 2 标准 Alon-Tarsi 方法在图 aK_b 上的运行时间。

这些结果体现了性能（在所有情况下，a 均取满足图中顶点数不超过 2^7 的最大值）：

b	a	边	time
4	42	251	0.2 秒
5	31	309	0.4 秒
6	25	374	0.5 秒

- 平面 $a \times a$ 网格，带有对角线，并在第一行、最后一行、第一列和最后一列旁各增加四个顶点；与外面相接的顶点的列表大小为三，其余顶点的列表大小为五。众所周知，可以用定理 3 证明这些图的可选择性，Zhu 已给出证明。对于 $a = 11$（364 条边），程序运行约 0.6 秒。
- 顶点为 0、1、.....、$3n-1$ 的环，并添加所有形如 $\{i,(i+n) \bmod 3n\}$ 的边，且所有列表大小均为 3。注意，该图是一个长度为 $3n$ 的环与 n 个三角形的边不交并；因此，根据著名的环加三角形定理 [9]，由定理 3 可知它是 3-可选择的。在这种情况下，即使是 $n = 15$（90 条边），程序也需要约 9 秒。更重要的是，此时程序需要约 7GB 内存。

最后一种情况下性能较差，值得探讨一番，尤其是内存消耗表明序列化改进未能奏效。值得注意的是，对于定理 3，x^f 在 f 将所有顶点赋值为 2 时，只有一个有用的单项式。具体而言，进行顺序化时，只有一个分支包含相关单项式。幸运的是，程序中选取的单项式排序方式使得我们在进行序列化时，会先考虑那些在已处理顶点上的限制具有字典序最大次数的单项式，也就是说，所有对应于已处理顶点的变量的度均为二的单项式。因此，我们会优先沿着这个唯一的相关分支进行，从而避免任何回溯。然而，问题在于，即使施加了这一限制，非零系数单项式的数量仍然过多——程序跟踪的单项式数量在处理完 15 个顶点后达到峰值，超过 4700 万[5]。

[5] 我们测试了这样一种可能性：移除那些无法对 x^f 的系数产生贡献的单项式。这些单项式是通过第 2.1 节开头所述的、将问题化归为二分图中的最大匹配的方法检测出来的。这样一来，单项式峰值数量降至 2600 万以下，内存消耗大约减半。然而，与这种剪枝相关的额外开销使运行时间增加到 36 秒，即变为原来的四倍。因此，即使是在可以说对剪枝极为有利的情况下，这种方法似乎也不值得采用。不过要指出的是，使用剪枝可以消除序列化时恰好先选择正确分支这一偶然因素。

增长如此迅速的一个原因是，所选的顶点排序最终导致顶点分离数非常大。因此，对于这张特定的图，VSEP 排序启发式明显优于其他方法，因为它使我们能够按照“三角形逐个处理”的顺序处理该图，顶点分离数为六。采用 VSEP 排序时，即使对于 $n = 40$（240 条边），运行时间也不到 0.01 秒。如果将三角形随机添加到环上，VSEP 仍优于 MD+PROC，但优势较小：MD+PROC 在 $n = 26$ 时开始耗尽内存（此时 VSEP 排序需要处理的单项式数量约少 9 倍），而 VSEP 则在$n = 34$时开始耗尽内存。

3 扩展 Alon-Tarsi 方法

我们首先证明支撑 Alon-Tarsi 方法强化结果的理论结论。

定理 5 的证明。 由于 L-可着色性和定理陈述中的约束都只取决于颜色是否相等，不失一般性，我们可以假设 $\bigcup_{v\in V(G)} L_v$ 的元素是代数独立的。对于每个 $v \in V(G)$，令 $\ell_v = |L_v|$ 并

$$p_v = x_v^{\ell_v} - \prod_{c\in L_v} (x_v - c) = \left(\sum L_v\right) \cdot x^{\ell_v - 1} + \dots . \tag{1}$$

注意，对于每个 $c \in L_v$，都有 $p_v(c) = c^{\ell_v}$。

令 q 为如下得到的多项式：从 $q = P_G$ 开始，只要存在一个顶点 $v \in V(G)$ 使得 $\deg_{x_v} q \geq \ell_v$，就将 q 替换为

$$\sum_{f\in\mathbb{Z}^{V(G)}, f(v)<\ell_v} [x^f]\, q \cdot x^f + \sum_{f\in\mathbb{Z}^{V(G)}, f(v)\geq\ell_v} [x^f]\, q \cdot x^{f-\ell_v 1_v} \cdot p_v.$$

注意，每次这样的替换都会使 $\deg_{x_v} q$ 至少减少 1；此外，对每个 $\varphi \in \bigtimes_{v\in V(G)} L_v$，$q(\varphi) = P_G(\varphi)$。最终得到一个多项式 q，使得对每个 $v \in V(G)$，$\deg_{x_v} q < \ell_v$。

注意，P_G 是齐次的，且是 $|E(G)|$ 项的乘积，因此，对每个满足 $[x^{f'}]\, P_G \neq 0$ 的 $f' \in \mathbb{Z}^{V(G)}$，$\sum_{v\in V(G)} f'(v) = |E(G)|$。由于 $\sum_{v\in V(G)} f(v) = |E(G)| - 1$，$q$ 中 x^f 处的系数来自于将顶点 $v \in V(G)$ 的单项式 x^{f+1_v}（属于 P_G）中的 x^{ℓ_v} 替换为 p_v，其中这些顶点满足 $f(v) = \ell_v - 1$。根据 (1)，每个这样的单项式对 $[x^f]\, q$ 的贡献为$[x^{f+1_v}]\, P_G \cdot \sum L_v$。此外，根据定理 3，由于 G 不是 L-可着色的，对每个满足 $f(v) < \ell_v - 1$ 的 $v \in V(G)$，我们有$[x^{f+1_v}]\, P_G = 0$。因此，

$$[x^f]\, q = \sum_{v\in V(G)} [x^{f+1_v}]\, P_G \cdot \sum L_v.$$

我们有 $q = 0$，否则定理 2 将表明存在 $\varphi \in \bigtimes_{v\in V(G)} L_v$，使得 $q(\varphi) = P_G(\varphi) \neq 0$，因此根据观察 1，$G$ 将是 L-可着色的。因此，令 $C = \bigcup_{z\in V(G)} L_z$，我们有

$$\begin{aligned} 0 = [x^f]\, q &= \sum_{v\in V(G)} [x^{f+1_v}]\, P_G \cdot \sum L_v = \sum_{v\in V(G)} [x^{f+1_v}]\, P_G \cdot \sum_{c\in C} c \cdot \chi_{L,c}(v) \\ &= \sum_{c\in C} c \cdot \sum_{v\in V(G)} [x^{f+1_v}]\, P_G \cdot \chi_{L,c}(v). \end{aligned}$$

由于这些颜色代数独立，系数

$$\sum_{v\in V(G)} [x^{f+1_v}]\, P_G \cdot \chi_{L,c}(v)$$

对于每种颜色 c，都必须为零。 ◀

对于函数 $s\colon V(G) \to \mathbb{N}$，称函数 $f\colon V(G) \to \mathbb{N}$ 为 s-紧的，如果对于每个 $v \in V(G)$，都有 $f(v) \le s(v)$，并且对于恰好一个顶点 $v \in V(G)$，有 $f(v) = s(v)$。对于一个 s-紧函数 f，其 s-基是函数 $f'\colon V(G) \to \mathbb{N}$，满足对于每个顶点 $v \in V(G)$，只要 $f(v) < s(v)$ 且 $f'(v) = f(v)$，就有 $f'(v) = f(v)$；而对于满足 $f(v) = s(v)$ 的唯一顶点 $v \in V(G)$，则有 -1。注意，对于每个系数出现在定理 5 等式左侧的单项式 x^f，$f \prec s$ 或 f 必为 s- 紧的。如果对于某个 $f \prec s$，有 $[x^f]\, P_G \neq 0$，那么根据定理 3，该图是 s-可选染色的。因此，为了应用定理 5，我们需要关注单项式 x^f 的系数，其中 f 是 s-紧的。然后，我们根据它们的 s-基将这些系数分组，每组都会给出一个关于特征向量的线性约束。

为了枚举系数，我们使用第 2 节中描述的算法，并作出修改，使截断中也包含具有 s-紧度向量的单项式。下面就实现方面谈几点：

- 将多项式存储为三元组 (f', v, c) 的列表较为方便，其中 v 要么是 G 的一个顶点，要么是 $\varnothing$，$f' \prec s$ 而 c 是在 $x^{f'}$ 处的系数（若 $v = \varnothing$），以及在 $x^{f'+1_v}$ 处的系数（若 $v \neq \varnothing$）；在此情况下，$f'(v) = s(v) - 1$。

  列表按递增字典序排序，首先依据 f，其次依据 v。这样一来，具有相同 s-基的单项式会在表示结果的列表中聚集在一起，我们可以直接据此构造约束，无需进一步后处理。

- 在顺序化过程中，我们仅根据 f' 对单项式进行划分，不考虑 v。这样可以确保所有具有相同 s-基的单项式都归入同一分支，因此我们仍可独立处理各个分支。

- 然而，这种表示方式会带来一个问题：如何在输出中保持排序。在标准 Alon-Tarsi 方法中，我们利用了这样一个事实：如果 $f_1 < f_2$，那么对于任意顶点 $v \in V(G)$，$f_1 + 1_v < f_2 + 1_v$，详情请参见第 2.3 节。然而，对于上述表示方式，我们需要的类似性质并不成立。事实上，考虑元素 $(f_1, \varnothing, c_1)$ 和 $(f_2, \varnothing, c_2)$，使得 $f_1 < f_2$, $f_1(v) = s(v) - 2$ 且 $f_2(v) = s(v) - 1$。处理与 v 相 incident 的边时，这些元素将分别产生 $(f_1 + 1_v, \varnothing, c_1)$ 和 (f_2, v, c_2)，但 $f_1 + 1_v \le f_2$ 不一定成立。为避免这个问题，在处理边 uv 时，我们通过合并以下四个流来生成输出；这些流都保证是递增的：

  - $(f + 1_v, w, c)$，对于输入中满足 $f(v) \le s(v) - 2$ 的 (f, w, c)；(f, v, c)，对于输入中满足 $f(v) = s(v) - 1$ 的 $(f, \varnothing, c)$；$(f + 1_u, w, -c)$，对于输入中满足 $f(u) \le s(u) - 2$ 的 (f, w, c)；以及 $(f, u, -c)$，对于输入中满足 $f(u) = s(u) - 1$ 的 $(f, \varnothing, c)$。
- 由于定理 5 给出的约束都是线性的，因此可以对它们进行紧凑表示：我们可以忽略那些与其他约束线性相关的约束，所以只需存储至多 $|V(G)|$ 个线性无关约束组成的列表。此外，随着新约束的生成，这个列表自然可以逐步更新；因此，我们不必

先生成完整的约束列表（这对序列化尤其重要，因为若必须存储结果的所有系数，其在降低内存消耗方面的用处将会大受限制）。

顺便一提，在我们的实现中，为避免在 $\mathbb{Q}$ 上进行高斯消元时系数增长带来的问题，我们会针对素数 $p = 2^{32} - 1$ 在有限域 $\mathbb{F}_p$ 上检验线性相关性。这可能会（极少见地）导致某些约束被不必要地舍弃，因为它们在 $\mathbb{F}_p$ 上线性相关，但在 $\mathbb{Q}$ 上并非如此。

收集完约束后，我们需要找出满足这些约束的特征向量，即线性方程组的 {0,1}-解。这可以使用任意整数线性规划求解器来实现。由于变量数量通常相当少，即使采用简单的方法，先进行高斯消元，再遍历自由变量的所有 {0,1}-取值，并检查由方程组确定的变量值是否也属于 {0,1} ，通常也是可行的。

如果可能的特征向量数量适中，我们就可以构造另一个整数线性规划，要求这些特征向量的线性组合（系数为非负整数）等于指定的列表大小向量。该程序的每个解都对应一种列表赋值，而从该列表赋值出发，图未必可着色。我们的实现仍然只采用简单的方法：执行高斯消元，并检查自由变量的所有可能取值；不过，对于这一部分，更适合使用更高级的整数线性规划求解器。

最后，如果可能的列表赋值数量不太多，可以尝试从每种列表赋值出发为图着色（由于顶点数量通常相当少，任何 CSP 或 SAT 求解器，甚至相当简单的穷举着色算法，都足以胜任这项任务）。在我们的实现中，我们将根据列表进行可着色性判断的问题表示为合取范式下的可满足性问题，并在这一部分使用 MiniSat [7] SAT 求解器。

3.1 测试结果

扩展 Alon-Tarsi 方法当然需要计算图多项式中更多的系数，相比之下，标准 Alon-Tarsi 方法所需计算的系数较少。此外，除非找到的单项式表明该图是 s-可选着色的（根据定理 5），否则我们不能在找到第一个非零系数的单项式后立即终止；而在这种情况下，实际上就不需要扩展 Alon-Tarsi 方法了[6]。因此，我们应当预期，其时间复杂度（以及在顺序化不起作用的情况下，可能还有内存消耗）会比标准 Alon-Tarsi 方法更差。

为说明这一点，考虑由 a 个 K_b 以类路径方式通过不同顶点粘合而成的不可列表着色图，其列表大小等于顶点度；这些图的标准 Alon-Tarsi 方法的评估结果见第 2.6 节开头。表 3 列出了给定 b 时，计算时间与边数的关系（边数四舍五入到最接近的 5 的倍数）。计算时间包括收集约束并消去线性相关的约束，但不包括进一步处理（列出可行特征向量，……）。将这些结果与表 2 中标准

[6] 虽然我们确实可以在任意时刻终止枚举，但这可能意味着我们会遗漏之后才能发现的一些约束（不过，如果搜索耗时过长，这仍可能是一个有效的替代方案；当解空间长时间没有变化时，停止搜索也可能是合理的）。我们报告的时间测量结果对应于扩展 Alon-Tarsi 方法的完整运行过程，即枚举所有紧单项式。

b \ 边	30	35	40	45	50
3	0.03 秒	0.2 秒	0.5 秒	1 s	4 s
4	0.1 秒	0.5 秒	3.5 秒		29 s
5	0.1 秒		3 s		128 秒
6	0.1 秒			19 s	

表 3 扩展 Alon-Tarsi 在图 aK_b 上的运行时间。

Alon-Tarsi 方法的结果进行比较，我们得出结论：对于这种特定类型的图，扩展 Alon-Tarsi 所需的时间，大致相当于标准 Alon-Tarsi 方法处理边数多约 15 条的图所需的时间。

为了进行更贴近现实的测试，我们在 crit12 数据集中的 844030 个图上评估了性能，这些图具有指定的列表大小 s （另见第2.5节）。该数据集中的图是在一个项目中生成的，该项目旨在寻找环面上绘制的图的 5-可选择性障碍；这些图经过多种启发式方法筛选（包括检查它们是否没有可用于定理 3 的非零系数），作为临界图的候选对象；也就是说，它们不具备 s-可选择性，但所有真子图都具备 s-可选择性。因此，这些图通常与具备 s-可选择性相当接近，故应能预期定理 5 会为其可着色性提供有趣的信息。对于其中每个图，我们

- 运行标准 Alon-Tarsi 方法（并检查该方法是否不适用），收集扩展 Alon-Tarsi 方法的
- 约束，列出可行特征向量，并利用它们找出“可删除”的边 uv；这些边具有这样的性质：
- 对于所有使图不可 L-着色的 s-列表赋值 L， u 和 v 的列表都不相交。如果存在可删除的边，则在删除这些边后得到的图上重新运行标准 Alon-Tarsi 方法，以检验其 s-可选择性；列出最多 100 种特征向量可行的可能 s-列表赋值；如果此类 s-列表赋值少于 100 种，则使用 MiniSat 求解器确定该图是否能由其中任何一种赋值着色。
-
-

总运行时间为 62 s，其中 5% 用于标准 Alon-Tarsi 方法（包括少数几次重新运行），21% 用于扩展 Alon-Tarsi，4% 用于寻找可行向量和可删除边，29% 用于列出 s-列表赋值，35% 用于使用 MiniSat 求解器进行着色（剩余的 6% 用于读取输入及其他杂项开销）。结果如下：

- 对于 548502 个（65%）图，程序找到了一个 s-列表赋值，使它们无法着色。对于 19322 个（2%）图，程序找到了可删除边，并在删除这些边后重新运行标准
- Alon-Tarsi 方法，从而判定这些图是 s-可选择的。对于 732 个图，程序直接证明了它
- 们是 s-可选择的（其中 78 个没有可行向量，215 个的向量无法组合成任何 s-列表赋值，另有 439 个经 MiniSat 求解器证明，对所有可能的 s-列表赋值均可着色）。对于 11554 个（1.4%）图，程序仅成功找到了可删除边（从而表明这些图不是临界的）。
-

- 对于 263526 个（31%）图，程序未能得出结论，因为可供探索的 s-列表赋值过多。对于 394 个图，程序也未能得出结论，因为与定理 5 相关的所有系数均为零。

值得注意的是，对于该数据集中的图，扩展 Alon-Tarsi 方法完全无法提供任何信息的情况只占极小比例（不到 0.1%），这表明该方法在类似情形下（枚举临界图、确认绘制在固定曲面上的图中的构型可约性）很有用。

对于该数据集，列出潜在坏分配并据此检查可着色性的步骤，其耗时与其用途相比并不成比例，因此除非需要确认图不是s-可选择的，否则运行这些步骤或许并不值得。另一方面，值得指出的是，我们主要致力于优化标准 Alon-Tarsi 方法和扩展 Alon-Tarsi 方法的实现，因此其他部分的运行时间肯定还有改进空间。

4 局限性与扩展

我们指出扩展 Alon-Tarsi 方法有一项重要限制：设 s 是将图 G 的顶点映射到列表大小的函数。假设单项式 x^f 的系数非零，其中 f 是 s-紧的。设 v 是满足 $f(v) = s(v)$ 的顶点。令$s' = s + 1_v$，定理 3 表明 G 是 s'-可选择的。因此，定理 5 只能为那些“接近 s-可选择”的图 G 提供有意义的信息，也就是说，至少存在一个顶点 v，使得 G 是 $(s + 1_v)$-可选择的。

此外，尽管上一节讨论的例子和实验表明，定理 5 适用于图与列表大小的许多有趣组合，但也存在一些相当简单（且接近 s-可选择）的图，使得定理 5 无法提供任何信息，因为所有相关系数最终都为零；例如，见下文的例 16。

还需注意，利用定理 5 得到的颜色特征向量上的约束必然是线性的，因此，如果可行特征向量满足更复杂的条件，该方法至多只能给出这些条件的线性松弛。

利用图多项式的更多系数，自然可以缓解这些顾虑。事实上，用于证明定理 5 的论证显然可以进一步推进；例如，下一步可得到以下结果。

▶ 定理 *15.* 设 G 为一个图，$L = \{L_v \subset \mathbb{R} : v \in V(G)\}$ 为赋给 G 的顶点的列表分配，并假设 $f : V(G) \to \mathbb{Z}$ 满足 $f(v) < |L_v|$ 对每个 $v \in V(G)$ 和 $\sum_{v \in V(G)} f(v) = |E(G)| - 2$。若 G 不可 L-着色，则

$$\sum_{\{u,v\} \subseteq V(G)} [x^{f+1_u+1_v}]\, P_G \cdot \chi_{L,c}(u)\chi_{L,c}(v) + \sum_{v \in V(G)} [x^{f+2\cdot 1_v}]\, P_G \cdot \chi_{L,c}(v) = 0 \tag{2}$$

对于每种颜色 c 和

$$\sum_{\{u,v\} \subseteq V(G)} [x^{f+1_u+1_v}]\, P_G \cdot (\chi_{L,c_1}(u)\chi_{L,c_2}(v) + \chi_{L,c_2}(u)\chi_{L,c_1}(v)) + \sum_{v \in V(G)} [x^{f+2\cdot 1_v}]\, P_G \cdot \chi_{L,c_1}(v)\chi_{L,c_2}(v) = 0 \tag{3}$$

对于任意两个不同的颜色 c_1 和 c_2。

证明。设 ℓ_v 和 p_v 为 $v \in V(G)$ 和 q，如定理 5 的证明中所述。设 $S = \{v \in V(G) : f(v) = \ell_v - 1\}$ 和 $D = \{v \in V(G) : f(v) = \ell_v - 2\}$。注意到

$$[x^f]\, q = \sum_{\{u,v\}\subseteq S} [x^{f+1_u+1_v}]\, P_G \cdot \sum L_u \sum L_v + \sum_{v\in S} [x^{f+2\cdot 1_v}]\, P_G \cdot \Big(\sum L_v\Big)^2 - \sum_{v\in S\cup D} [x^{f+2\cdot 1_v}]\, P_G \cdot \sum_{\{c_1,c_2\}\subseteq L_v} c_1c_2.$$

由于 G 不可 L-着色，因此对于互不相同的 $u, v \notin S$，根据定理 3 有 $[x^{f+1_u+1_v}]\, P_G = 0$；根据定理 5，对于 $u \notin S$，有 $\sum_{v\in V(G)} [x^{f+1_u+1_v}]\, P_G \cdot \sum L_v = 0$。因此，

$$\begin{aligned}
&\sum_{\{u,v\}\subseteq V(G)} [x^{f+1_u+1_v}]\, P_G \cdot \sum L_u \sum L_v \\
&\quad = \sum_{\{u,v\}\subseteq S} [x^{f+1_u+1_v}]\, P_G \cdot \sum L_u \sum L_v + \sum_{u\in V(G)\setminus S} \sum L_u \cdot \sum_{v\in V(G)} [x^{f+1_u+1_v}]\, P_G \cdot \sum L_v \\
&\qquad - \sum_{\{u,v\}\subseteq V(G)\setminus S} [x^{f+1_u+1_v}]\, P_G \cdot \sum L_u \sum L_v - \sum_{v\in V(G)\setminus S} [x^{f+2\cdot 1_v}]\, P_G \cdot \Big(\sum L_v\Big)^2 \\
&\quad = \sum_{\{u,v\}\subseteq S} [x^{f+1_u+1_v}]\, P_G \cdot \sum L_u \sum L_v - \sum_{v\in V(G)\setminus S} [x^{f+2\cdot 1_v}]\, P_G \cdot \Big(\sum L_v\Big)^2.
\end{aligned}$$

此外，对于 $v \notin S \cup D$，有 $[x^{f+2\cdot 1_v}]\, P_G = 0$。因此，我们得到

$$\begin{aligned}
0 = [x^f]\, q = &\sum_{\{u,v\}\subseteq V(G)} [x^{f+1_u+1_v}]\, P_G \cdot \sum L_u \sum L_v \\
&+ \sum_{v\in V(G)} [x^{f+2\cdot 1_v}]\, P_G \cdot \Big(\Big(\sum L_v\Big)^2 - \sum_{\{c_1,c_2\}\subseteq L_v} c_1c_2\Big).
\end{aligned}$$

由于这些颜色代数独立，对于每个颜色 c，系数

$$\sum_{\{u,v\}\subseteq V(G)} [x^{f+1_u+1_v}]\, P_G \cdot \chi_{L,c}(u)\chi_{L,c}(v) + \sum_{v\in V(G)} [x^{f+2\cdot 1_v}]\, P_G \cdot \chi_{L,c}(v)$$

在 c^2 处必须为零；对于任意两个互不相同的颜色 c_1 和 c_2，系数

$$\begin{aligned}
&\sum_{\{u,v\}\subseteq V(G)} [x^{f+1_u+1_v}]\, P_G \cdot (\chi_{L,c_1}(u)\chi_{L,c_2}(v) + \chi_{L,c_2}(u)\chi_{L,c_1}(v)) \\
&+ \sum_{v\in V(G)} [x^{f+2\cdot 1_v}]\, P_G \cdot \chi_{L,c_1}(v)\chi_{L,c_2}(v)
\end{aligned}$$

在 c_1c_2 处必须为零。 ◀

下面给出一个简单的图示例，在该示例中，定理 15 比定理 5 提供了更多信息。

▶ **例 16. </style}令 由中心为 0、辐条为 1, . . . , 4 的轮图构成，并具有列表赋值，使得 且 。通过直接分类讨论可知，在颜色置换意义下，满足 不可 - 着色的此类规模列表赋值只有少数几种：{**a, b, c**} </style}并令** $L_i = L_{i-4}$，其中 $i \geq 5$,

- for some $i \in \{1, 2\}$, $L_i = L_{i+1} = \{a, b\}$ and $L_{i+2} = L_{i+3} = \{a, c\}$, or

- for some $i \in \{1,2,3,4\}$, $L_i = L_{i+1} = \{a,b\}$ and $L_{i+2} = L_{i+3} = \{c,d\}$, or
- for some $i \in \{1,2,3,4\}$, $L_i = \{a,d\}$, $L_{i+1} = \{a,b\}$, $L_{i+2} = \{b,c\}$ and $L_{i+3} = \{c,d\}$.

然而， G 的图多项式中没有任何与定理 5 相关且系数非零的单项式（事实上，要应用定理 5，我们必须有 $f(0) < 3, f(1) \ldots, f(4) < 2$ 且 $f(0) + \ldots + f(4) = |E(G)| - 1 = 7$，这是不可能的）。我们来说明

$$\begin{aligned} P_G = x_0^2 x_1 x_2 x_3 x_4 (x_1^2 + x_2^2 + x_3^2 + x_4^2 \\ - 2x_1x_2 - 2x_2x_3 - 2x_3x_4 - 2x_4x_1 \\ + x_1x_3 + x_2x_4 \\ + x_0x_1 + x_0x_2 + x_0x_3 + x_0x_4 \\ - 2x_0^2) + \ldots, \end{aligned}$$

因此，定理 15 意味着任意颜色的特征向量 χ 必须满足

$$\begin{aligned} \chi(1) + \chi(2) + \chi(3) + \chi(4) \\ -2\chi(1)\chi(2) - 2\chi(2)\chi(3) - 2\chi(3)\chi(4) - 2\chi(4)\chi(1) \\ +\chi(1)\chi(3) + \chi(2)\chi(4) \\ +\chi(0)\chi(1) + \chi(0)\chi(2) + \chi(0)\chi(3) + \chi(0)\chi(4) \\ -2\chi(0) = 0, \end{aligned}$$

并且任意两种不同颜色的特征向量 χ 和 χ' 必须满足

$$\begin{aligned} \chi(1)\chi'(1) + \chi(2)\chi'(2) + \chi(3)\chi'(3) + \chi(4)\chi'(4) \\ -2 \sum_{(i,j) \in \{(1,2),(2,3),(3,4),(4,1)\}} \chi(i)\chi'(j) + \chi'(i)\chi(j) \\ +\chi(1)\chi'(3) + \chi'(1)\chi(3) + \chi(2)\chi'(4) + \chi'(2)\chi(4) \\ + \sum_{j \in \{1,2,3,4\}} \chi(0)\chi'(j) + \chi'(0)\chi(j) \\ -2\chi(0)\chi'(0) = 0. \end{aligned}$$

这足以将大小为 3，2， 2， 2 的列表的可能列表赋值限制为 22 种选项之一，其中十种是上文所述的赋值 L ，对于这些赋值， G 不可 L-着色。

定理 15 对颜色的可行特征向量施加了更复杂的二次约束；此外，它还对列表赋值中可能出现的颜色对的可行特征向量施加约束，因此可能揭示比定理 5 中针对单个颜色的线性约束多得多的信息。另一方面，应用定理 15 的计算成本会高得多——必须计算图多项式的更多系数，而且目前尚不清楚如何紧凑地表示所得约束，也不清楚如何从这些约束中获得有关可行特征向量的信息。

或许最实用的做法是，先按照第 3 节中的讨论，枚举定理 5 所允许的可行向量，并构造一个完全图 K（带自环），其顶点为可行特征向量；对于大约有 10–20 个顶点的图，预计 K 的规模会相当小。随着定理 15 的约束

逐步生成，我们删除 K 中违反 (2) 的顶点以及 K 中违反 (3) 的边。最终，对于任意具有给定列表大小的列表赋值 L，若 G 不可 L着色，则 L 中各颜色的特征向量必须在 K中构成一个团；如果两个不同的颜色具有相同的特征向量，则它必须在 K中构成一个自环。

当然，我们还可以进一步推进这一论证，得到三次、四次.....约束。不过，尚不清楚实现这一强化是否值得付出相应努力，尤其是考虑到第 3 节的方法在实践中似乎表现相当不错。

参考文献

1 Noga Alon 和 Michael Tarsi。图的着色与定向。《组合数学》，12(2):125–134，1992。2 K. Appel 和 W. Haken。每个平面图都是四可着色的，第一部分：放电法。《伊利诺伊数学杂志》，21:429–490，1977。3 K. Appel、W. Haken 和 J. Koch。每个平面图都是四可着色的，第二部分：可约性。《伊利诺伊数学杂志》，21:491–567，1977。4 Hans L. Bodlaender、John R. Gilbert、Hjálmtýr Hafsteinsson 和 Ton Kloks。树宽、路径宽度、前沿大小和最短消去树的近似。《算法杂志》，18(2):238–255，1995。网址：`http://dx.doi.org/10.1006/jagm.1995.1009`，`doi:10.1006/jagm.1995.1009`。5 Hans L. Bodlaender 和 Ton Kloks。图的路径宽度和树宽的高效构造性算法。《算法杂志》，21:358–402，1996。6 Daniel W Cranston 和 Douglas B West。通过图着色介绍放电法。《离散数学》，340:766–793，2017。7 Niklas Een。MiniSat：一种带有冲突子句最小化功能的 SAT 求解器。载于 SAT-05：第 8 届可满足性测试理论与应用国际会议论文集，第 502–518 页，2005。8 P. Erdős、A. L. Rubin 和 H. Taylor。图的可选择性。《数论大会论文集》，26:125–157，1980。9 H. Fleischner 和 M. Stiebitz。P. Erdős 的一个着色问题的解。《离散数学》，101:39–48，1992。10 Heng Guo 和 Tyson Williams。带复权重的平面布尔 #CSP 的复杂性。载于自动机、语言与编程国际研讨会论文集，第 516–527 页，2013。11 Sangxia Huang 和 Pinyan Lu。实数加权 Holant 问题的二分定理。载于第 27 届 IEEE 计算复杂性会议论文集，第 96–106 页，2012。12 C. Thomassen。每个平面图都是 5-可选着色。《组合理论杂志》B辑，62:180–181，1994。13 V. G. Vizing。给定颜色的顶点着色（俄文）。《离散分析方法》，新西伯利亚，29:3–10，1976。14 M. Voigt。平面图的列表着色。《离散数学》，120:215–219，1993。15 Xuding Zhu。平面图的 Alon-Tarsi 数。《组合理论杂志》B辑，134:354–358，2019。