Halin图的快速S-装箱着色算法

XinZhang Dezhi Zou
中国西安西安电子科技大学数学与统计学院，710071
xzhang@xidian.edu.cn dezhizou@stu.xidian.edu.cn

2025年12月30日

arXiv:2512.22809v1 [math.CO] 28 Dec 2025

摘要

受无线广播网络中的频率分配问题启发，Goddard、Hedetniemi、Hedetniemi、Harris 和 Rall 于 2008年 引入了S-装箱着色的概念。给定一个正整数非递减序列$S = (s_1, s_2, \ldots, s_k)$，图 G 的 S- 装箱着色是将其顶点集划分为 k 个子集 $\{V_1, V_2, \ldots, V_k\}$，使得对于每个 $1 \le i \le k$，V_i 中的任意两个不同顶点u和 v 之间的距离至少为$s_i + 1$。本文研究最大度为 $\Delta \le 5$ 的 Halin图 的S-装箱着色问题。具体而言，我们给出一种线性时间算法，可为任意满足 $\Delta \le 5$ 的 Halin 图构造一个 (1, 1, 2, 2, 2)-装箱着色。值得注意的是，存在一些 Halin 图无法进行 (1, 2, 2, 2)-可装箱着色。

关键词：顶点着色；装箱着色；Halin 图。

1 引言

设 $S = (s_1, s_2, \ldots, s_k)$ 是一个非递减的正整数序列（即 $s_1 \le s_2 \le \cdots \le s_k$）。简单图 $G = (V(G), E(G))$ 的 S-装箱着色是将 V(G) 划分为 k 个互不相交的子集 $V_1, V_2, \ldots, V_k$，使得对于每个 $i \in [k] := \{1, 2, \ldots, k\}$，任意两个不同的顶点 $u, v \in V_i$ 均满足 $d_G(u, v) \ge s_i + 1$，其中 $d_G(u, v)$ 表示 G 中 u 与 v 之间的距离。G 的装箱色数记作 $\chi_p(G)$，是使 G 存在 (1, 2, . . . , k)-装箱着色（也称为 k-装箱着色）的最小整数 k。装箱色数这一概念由 Goddard、Hedetniemi、Hedetniemi、Harris 和 Rall [15] 于 2008年提出，最初称为广播色数，其研究动机源于无线广播网络中的频率分配问题。

G 的正常 k-着色是一个映射 $\varphi : V(G) \to [k]$，满足对于任意相邻顶点 u、$v \in V(G)$，$\varphi(u) \ne \varphi(v)$；若存在这样的映射，则称 G 是 k-可着色的。与之相对，G 的 2-距离着色是一个顶点着色 $\varphi : V(G) \to [k]$，满足对于任意两个不同的顶点 u、$v \in V(G)$，若 $d_G(u, v) \le 2$，则 $\varphi(u) \ne \varphi(v)$；使这一条件成立的最小 k 称为 2-距离色数$\chi_2(G)$。值得注意的是，这些概念与装箱着色有关：(1, 1, . . . , 1)-装箱着色对应普通正常着色，而 (2, 2, . . . , 2)-装箱着色则对应 2-距离着色。此外，(2, 2, . . . , 2)-装箱着色与精确2-距离着色 [26, 第 11 节9]密切相关，但条件更严格：精确 2-距离着色只要求距离

恰好为 2 的顶点着不同颜色（允许相邻顶点同色），而 (2，2，. . . ，2)-装箱着色则要求每个颜色类中任意两点之间的距离至少为 3，因此必然满足精确2-距离着色的要求，但反之不成立。如果图是无三角形图，那么精确2-距离着色就等价于单射着色——即任何两个拥有公共邻点的顶点都着不同颜色的着色 [9, 10]。由此可知，即使对于无三角形图，(2， 2，. . . ，2)-装箱着色仍严格强于单射着色。值得注意的是，Bu、Qi 和 Zhu [11] 证明了最大度不超过五的 Halin 图具有单射 7 着色。关于这些相关主题的更多阅读资料，读者可参阅 [13, 27, 28, 31]。

我们将$S(G)$定义为对G的每条边进行细分后得到的图，即将G中每条边$uv \in E(G)$替换为一条经过新顶点 w 的路径u-w-v。多项研究 [7, 14]探讨了次三次图（其中每个顶点的度至多为 3）的$\chi_p(G)$和$\chi_p(S(G))$的上界问题。Gastineau 和 Togni [14] 率先提出，对于所有次三次图G，$\chi_p(S(G)) \leq 5$ 是否成立。Brešar 等人 [7], 随后将这一问题正式表述为一个猜想，并提出如下猜想：

猜想 1.1. 如果 G 是次三次图，那么 $\chi_p(S(G)) \leq 5$。

对于图 G 及其细分图 S (G)，Gastineau 和 Togni [14] 确立了它们的包着色性之间的基本关系：如果 G 是 $(s_1, s_2, \ldots, s_k)$-可装箱着色的，那么 S (G) 就是 $(1, 2s_1 + 1, 2s_2 + 1, \ldots, 2s_k + 1)$-可装箱着色的。他们还指出，当 G 是 $(1^2, 2^2)$-可装箱着色的（序列中的 p^r 表示 p 重复 r 次）时，便可直接推出 $\chi_p(S(G)) \leq 5$。因此，要验证猜想 1.1 对次三次图的任意子类成立，只需证明该子类具有 $(1^2, 2^2)$-包着色性。

为实现这一目标，Liu、Zhang 和 Zhang [20] 证明，每个次三次图G都可进行 $(1^2, 2^2, 3)$-装箱着色，从而得到 $\chi_p S(G)$ 的装箱色数 ≤ 6。这一结果距离猜想界限仅差一个参数，并显著改进了 Balogh、Kostochka 和 Liu [2], 此前的结果；他们证明，次三次图的 1-细分的装箱色数不超过 8。

已有多项研究探讨了次三次图的各种装箱 S-着色 [1, 4–6、8、17–19、21，24]。特别是，Gastineau 和 Togni [14] 证明，每个次三次图都存在 $(1^2, 2^3)$-装箱着色和 $(1, 2^6)$-装箱着色。Balogh、Kostochka 和 Liu [2] 进一步证明，任意次三次图都有一种 $(1^2, 2^2, 3^2, k)$-装箱着色，其中颜色 $k \geq 4$ 至多使用一次；此外，每个 2-退化次三次图都可进行 $(1^2, 2^2, 3^2)$-装箱着色。

具体而言，对于三次图，Cranston 和 Kim [12] 证明，除彼得森图外，所有三次图都可以进行 (2^8)-装箱着色。Thomassen [30] 补充了这一结果；Hartke、Jahanbekam 和 Thomas [16], 则独立证明了三次平面图是 (2^7)-可装箱着色的。

为了系统地完善这些结果，研究人员将注意力转向i-饱和次三次图，这类图由对高次顶点的结构约束定义。k-度图 是满足$\Delta(G) \leq k$的图G，而i-饱和次三次图 是指每个3度顶点至多与i 个其他3度顶点相邻的图。这种分类带来了显著进展：Yang 和 Wu [33]证明，所有0-饱和次三次图都存在$(1^2, 3)$-装箱着色；Bazzal [3] 将这一结果推广到具有$(1^2, 2)$-着色的1-饱和图；Mortada 和 Togni [23] 进一步推广到具有$(1^2, 2, 3)$-着色的2-饱和图；他们后续的工作 [22] 则证明了(3, 0)-饱和变体具有$(1^2, 2^2)$-可着色性（这是一个更精细的子类，其中(3, 0)-饱和指重顶点不相邻的3-饱和图；重顶点定义为其所有邻居均为3度顶点的3度顶点）。

尽管对最大度数较大的图中的S-装箱着色研究有限，Mortada 和 Togni [25] 最近通过分析具有不同饱和约束的k-度图，推动了这一领域的发展；若每个度为k的顶点至多有t个度为k的邻居，则称k-度图G为t-饱和。其结果如下：对于0-饱和k-度图，G存在$(1^{k-1}, 3)$-距离染色；当饱和度提高到 $1 \leq t \leq k-2$时，则转为具有$(1^{k-1}, 2)$-装箱着色性。此外，$(k-1)$-饱和k-度图（$k \geq 4$）需要$(1^{k-1}, 2^{k-1})$-装箱着色，而所有k-度图（$k \geq 3$）都具有$(1^{k-1}, 2^k)$-装箱着色性。这些发现不仅推广了关于次三次图的既有研究，也为分析度约束图可着色性提供了统一框架。

Halin 图是由一棵没有度为二的顶点的树T（称为特征树）构造而成的平面图，方法是通过一个圈C将所有叶子连接起来（该圈称为伴随圈）。Halin图表现出独特的S-装箱着色模式。Tarhini 和 Togni [29] 证明了三次Halin图是（$1^2, 2, 3$）-可装箱着色的，而 Wang、Hu 和 Wang [32] 表明，由于其平方图的色数为 6，5-度 Halin图需要（2^6）-距离染色。本文加强了这一界限，证明 5-度 Halin图实际上允许（$1^2, 2^3$）-距离染色，从而改进了该图类的现有最佳结果。值得注意的是，轮图W_5（最大度为 5）是一个不允许（$1, 2^3$）-距离染色的 Halin 图。

定理1.2。 最大度至多为 5 的 $Halin$ 图是（$1^2, 2^3$）-可装箱着色的。

定理 1.2 的证明是算法性的：它给出了一个明确的过程，用于构造所需的 $1^2, 2^3$-装箱着色，所需时间为 $O(n)$，其中 n 表示 Halin 图中的顶点数。算法在第 2 节介绍，并在第 3节验证。第 4节汇总了有关 Halin 图的 S-装箱着色的未解决问题。

2 一种快速算法

设 G 是一个最大度不超过 5 的 Halin 图，其中 T 是特征树，$C := a_1a_2 \cdots a_na_1$ 是 G 的伴随圈。我们将使用的颜色集合为 $\{1, 1', 2_a, 2_b, 2_c\}$。

首先，使用两种颜色 1 和 $1'$ 对树 T 进行正常着色，并将此着色记为 φ。接着，移除 C 上的所有颜色，并按照 RECOLORING 算法对顶点 $a_1, a_2, \ldots, a_n$ 重新着色。在该算法中，我们引入布尔变量 all_same（初始化为 True），以区分两种情况。如果 $\varphi(a_1) \neq \varphi(a_n)$ 初始成立，或者存在索引 $2 \leq i \leq n$，使得 $\varphi(a_i) \neq \varphi(a_{i-1})$，则对顶点 $a_1, a_2, \ldots, a_n$ 进行循环重标记，以确保 $\varphi(a_1) \neq \varphi(a_n)$，将 all_same 重置为 False，并应用 CASE-1。否则（即，对于所有 $i \in [n]$，$\varphi(a_i) = 1$），all_same 的值保持为 True，并直接应用 CASE-2。注意，每个 a_i 都是 T 中的叶，且设 b_i 为其在 T 中的唯一邻点。对于任意 i，令 1_i表示 $\{1, 1'\} \setminus \varphi(b_i)$中的颜色，并令 $\mathcal{N}[b_i]$ 表示与 b_i相邻的顶点 a j 的集合。

引理 2.1。 对于每个满足 $i \in [n]$ 的 b_i，$\mathcal{N}[b_i]$ 至多包含三个顶点。

证明。反设 $\{a_i、a\,j、a_k、a_\ell\} \subseteq \mathcal{N}[b_i]$，其中 $i < j < k < l$。由于$\{\varphi(a_i)、\varphi(a_j)、\varphi(a_k)、\varphi(a_\ell)\} \subseteq \{2_a, 2_b, 2_c\}$，根据 RECOLORING 算法，差值 $j-i$、$k-j$ 和 $l-k$ 中至多有一个等于 1。不失一般性，设 $j-i \neq 1$ 且 $k-j \neq 1$。这意味着存在顶点 $a_{i'}$ 和 $a_{j'}$，使得 $i < i' < j < j' < k$。由于 $a_{i'}$ 和 $a_{j'}$ 是树 T 的叶子，因此在 T 中必然存在从 b_i到 $a_{i'}$以及从 b_i到 $a_{j'}$的路径。因此，b_i必然至少关联六条边，这与条件 $\Delta(G) \leq 5$矛盾。□

算法 1：重新着色($\varphi(a_1)$, $\varphi(a_2)$, $\cdots$, $\varphi(a_n)$)

输入：顶点的颜色 $\varphi(a_1)$、$\varphi(a_2)$、$\cdots$、$\varphi(a_n)$，其中顶点为 a_1, a_2、$\cdots$、a_n

输出：a_1, a_2、$\cdots$、a_n 重新着色后的颜色

```
全部_相同 ← 真
如果 φ(a_1) = φ(a_n)则
    对于i ← 2到n 执行
        如果 φ(a_i) ≠ φ(a_{i−1})则
            对于 j ← 0 到 n − 1 执行
                a_{j+1} ← a_{(i+j−1 模 n)+1}
            all_相同 ← 假
            中断
否则
    all_相同 ←假
如果全部_相同 = 假则
    CASE-1(a_1, a_2, ⋯, a_n, φ(a_1), φ(a_2), ⋯, φ(a_n))
否则
    CASE-2(a_1, a_2, ⋯, a_n, φ(a_1), φ(a_2), ⋯, φ(a_n))
```

算法2：CASE-1($a_1, a2, \cdots, a_n, \varphi(a_1), \varphi(a_2), \cdots, \varphi(a_n)$) 输入：顶点 $a_1, a2, \cdots, a_n$ 的颜色 $\varphi(a_1)$、$\varphi(a_2)$、$\cdots, \varphi(a_n)$ 输出：重新着色后 $a_1, a2, \cdots, a_n$ 的颜色 1 m $\leftarrow \lfloor n/4 \rfloor * 4$ // 确保 m 是小于或等于 n 的最大 4 的倍数 2 for k $\leftarrow$ 1to m do 3 if k $\equiv 1 \pmod 2$ then4 $\varphi(a_k) \leftarrow 1_k$ // 为 a_k 着色，避开 b_k上的颜色 5 else if k $\equiv 2 \pmod 4$ then6 $\varphi(a_k) \leftarrow 2_a$7 else8 $\varphi(a_k) \leftarrow 2_b$ // 覆盖 n $\equiv 0 \pmod 4$ 的情形 9 switch (n –m) do 10 case 1 do

```
11          φ(a_n) ← 1_n // 覆盖 n ≡ 1 (mod 4) 的情形 12 case 2 do13          φ(a_{n−1}) ← 2_a; φ(a_n) ← 1_n // 覆盖 n ≡ 2 (mod 4) 的情形 14 case 3 do15          φ(a_{n−2}) ← 1_{n−2}; φ(a_{n−1}) ← 2_a; φ(a_n) ← 1_n // 覆盖 n ≡ 3 (mod 4) 的情形
```

算法 3：CASE-2($\mathbf{a}_1, \mathbf{a}2, \cdots, \mathbf{a}_n, \varphi(\mathbf{a}_1), \varphi(\mathbf{a}_2), \cdots, \varphi(\mathbf{a}_n)$) 输入：顶点 $\mathbf{a}_1, \mathbf{a}2, \cdots, \mathbf{a}_n$的颜色 $\varphi(\mathbf{a}_1), \varphi(\mathbf{a}_2), \cdots, \varphi(\mathbf{a}_n)$ 输出：重新着色后 $\mathbf{a}_1, \mathbf{a}2, \cdots, \mathbf{a}_n$ 的颜色 1 $\mathbf{m} \leftarrow \lfloor \mathbf{n}/4 \rfloor * 4$ // 确保 m 是小于或等于 n 的最大 4 的倍数 2 对于 $\mathbf{k} \leftarrow 1$到 m 执行 3 若 $\mathbf{k} \equiv 1 \pmod 2$ 则 4 $\varphi(\mathbf{a}_k) \leftarrow 1_k$ // 对 $\mathbf{a}_k$ 着色，避开 $\mathbf{b}_k$上的颜色 5 否则若 $\mathbf{k} \equiv 2 \pmod 4$ 则 6 $\varphi(\mathbf{a}_k) \leftarrow 2_a$ 7 否则 8 $\varphi(\mathbf{a}_k) \leftarrow 2_b$ // 覆盖 $\mathbf{n} \equiv 0 \pmod 4$ 的情况 9 根据 $(\mathbf{n} - \mathbf{m})$ 执行 10 情形 1 执行 11 $\varphi(\mathbf{a}_n) \leftarrow 2_c$ // 覆盖 $\mathbf{n} \equiv 1 \pmod 4$ 的情况 12 情形 2 执行 13 $\varphi(\mathbf{a}_{n-1}) \leftarrow 1_{n-1}$; $\varphi(\mathbf{a}_n) \leftarrow 2_c$ // 覆盖 $\mathbf{n} \equiv 2 \pmod 4$ 的情况 14 情形 3 执行 15 $\varphi(\mathbf{a}_{n-2}) \leftarrow 2_a$; $\varphi(\mathbf{a}_{n-1}) \leftarrow 1_{n-1}$; $\varphi(\mathbf{a}_n) \leftarrow 2_b$ // 覆盖 $\mathbf{n} \equiv 3 \pmod 4$ 的情况 // 检查特定条件并相应修改颜色 16 对于 $\mathbf{i} \leftarrow 1$ 到 $\mathbf{n} - 2$ 执行 17 若 $\varphi(\mathbf{a}_n) = 2_c$ 且 $\varphi(\mathbf{a}_{n-1}) = \varphi(\mathbf{a}_i) = 2_b$ 且 $\mathbf{b}_n = \mathbf{b}_{n-1} = \mathbf{b}_i$ 则 18 $\varphi(\mathbf{a}_2) \leftarrow 2_c$; $\varphi(\mathbf{a}_{n-1}) \leftarrow 2_c$; $\varphi(\mathbf{a}_n) \leftarrow 2_a$ 19 跳出 20 not_分离的 $\leftarrow$True 21 若 $\varphi(\mathbf{a}_n) = 2_c$ 则 22 对于 $\mathbf{i} \leftarrow 1$ 到 $\mathbf{n} - 1$ 执行 23 若 $\varphi(\mathbf{a}_i) \in \{2_a, 2_b\}$ 且 $\mathbf{b}_n = \mathbf{b}_i$ 则 24 not_分离的 $\leftarrow$ False 25 跳出 26 若 not_分离的则 27 对于 $\mathbf{i} \leftarrow 4$ 到 $\mathbf{n} - 1$ 执行 28 若 $\varphi(\mathbf{a}_i) = 2_a$ 且 $\mathbf{b}_i = \mathbf{b}_2$ 则 29 $\varphi(\mathbf{a}_2) \leftarrow 2_c$; $\varphi(\mathbf{a}_n) \leftarrow 2_a$ 30 跳出

对于一列n个顶点$a_1, a_2, \ldots, a_n$，其颜色序列 是一个字符串 $\mathrm{l}_1\mathrm{l}_2 \cdots \mathrm{l}_n \in \Sigma^n$，其中 $\Sigma \subseteq \mathbb{N}$是一个有限颜色集，$\mathrm{l}_j$表示顶点$a_j$的颜色。在本文中，如果索引$i$以 l_j（即 $\mathrm{l}_j = i$）的形式出现在颜色序列中，其含义由顶点b_j的颜色决定：如果b_j 的颜色为 $1'$，则i表示颜色 1；否则表示颜色 $1'$。

以下引理是算法CASE-1 和CASE-2的直接推论，此处不加证明。值得注意的是，CASE-2 首先为顶点 $a_1, a_2, \ldots, a_n$生成一个颜色序列，随后在不同条件下对某些顶点重新着色。

- 如果存在唯一的顶点 $b_i \in V(T)$（由引理 2.1 保证），满足 $N[b_i] = \{a_i, a_{n-1}, a_n\}$ and

aa_i,a_{n-1},a_n 的颜色序列为 $2_b2_b2_c$，我们在 CASE-2 的第 16–19 行CASE-2中执行以下重新着色 2_c：将其重新赋给 a_i 和 a_{n-1}，并将 2_a 赋给 a_n（示意见图 1(c)）。

- 如果 $\mathcal{N}$[bn]中没有任何顶点被染成 2a 或 2b，且 b2 与某个被染成 2a 的 a j ≠a2相邻，那么我们将 a2 重新着色为 2c，并将 an 重新着色为 2a（示例见图 1(d) 或 1(g)）。在算法中，我们引入一个布尔变量 not_separated（初始化为 True），以确定 $\mathcal{N}$[bn]中是否没有被染成 2a 或 2b 的顶点：如果 not_separated 变为 False，算法将不执行任何操作并终止；否则，它会检查 b2 是否与某个被染成 2a 的 a j ≠a2 相邻。如果满足该条件，则执行重新着色操作（实现细节见 CASE-2 的第 20–30 行）。

引理2.2。如果 t *he CASE-1* 算法应用后，分配给 $a_1, a_2, \ldots,$ a_n 如下：

当 $n \equiv 0$（模 4）时：颜色序列为 $i2_ai2_b \cdots i2_ai2_b$。

当 $n \equiv 1$（模 4）时：颜色序列为 $i2_ai2_b \cdots i2_ai2_bi$。

当 $n \equiv 2$（模 4）时：颜色序列为 $i2_ai2_b \cdots i2_ai2_b2_ai$。

当 $n \equiv 3$（模 4）时：颜色序列为 $i2_ai2_b \cdots i2_ai2_bi2_ai$。

如果使用*CASE-2* 算法，颜色将由更复杂的条件决定：

当 $n \equiv 0$（模 4）时：颜色序列为 $i2_ai2_b \cdots i2_ai2_b$。

当 $n \equiv 1$ 模 4 时，颜色序列取决于图结构，且为以下之一：

- $i2ai2b \cdots i2ai2b2c$，当且仅当以下条件之一成立时出现：– $\mathcal{N}$[bn]有一个顶点 aj 被染成 $2a$ 或 $2b$，且当 $j = n - 1$时，$\mathcal{N}$[bn]中没有其他顶点被染成 $2b$（见图*1(a)*）；– 或 $b2$ 不与一个被染成 $2a$ 的 $aj \neq a2$ 相邻（见图*1(b)*）；

- $i2ci2bi2ai2b \cdots i2ai2bi2ai2c2a$，当且仅当 bn 与 an_{-1} 和 an 均相邻，且还与另一个被染成 $2b$ 的顶点 aj 相邻时出现（见图*1(c)*）；

- $i2ci2bi2ai2b \cdots i2ai2b2a$，当且仅当 $\mathcal{N}$[bn]中没有顶点被染成 $2a$ 或 $2b$，且 $b2$ 与一个被染成 $2a$ 的 $aj \neq a2$ 相邻时出现（见图*1(d)*）；

当 $n \equiv 2$（模 4）时：颜色序列为以下之一：

- $i2_ai2_b \cdots i2_ai2_bi2_c$，其条件与 $n \equiv 1$（模 4）的第一种情况相同（见图*1*（*e*）和（*f*））；
- $i2_ci2_bi2_ai2_b \cdots i2_ai2_bi2_a$，其条件与 $n \equiv 1$（模 4）的第三种情况相同（见图 *1*（*g*））。

当 $n \equiv 3$ (模 4)时：颜色序列为 $i2_ai2_b \cdots i2_ai2_b2_ai2_b$。 □

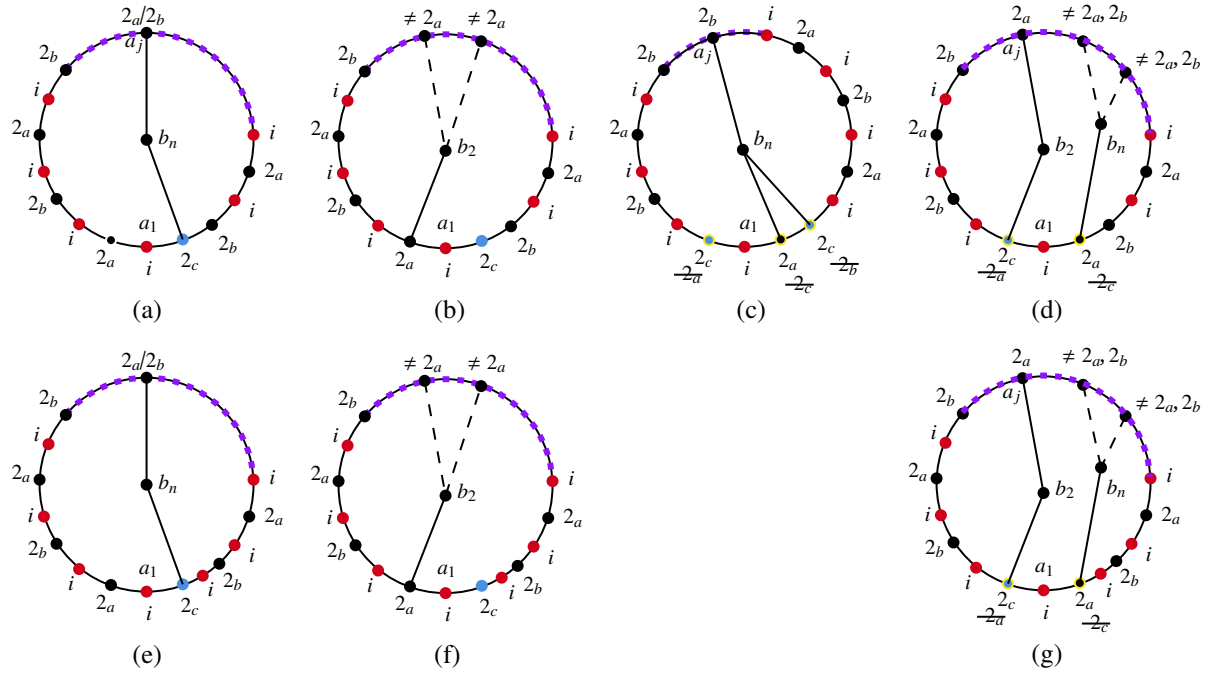

图1：算法中涉及的（重新）着色过程CASE-2

引理 2.3. 对于每个 b_i，若 $i \in [n]$, *if* $\mathcal{N}[b_i]$ 恰好包含三个顶点，则其中至少有两个在 C 上彼此相邻，且它们不属于同一种颜色。

证明。设 $\mathcal{N}[b_i] = \{a_i, a\,j, a_k\}$。不失一般性，假设 $i < j < k$。现在，$j - i = 1$或 $k - j = 1$ 必有其一成立。否则，根据与引理 2.1 中类似的论证，我们将推得 b_i 的度至少为六，这与给定条件 $\Delta(G) \leq 5$矛盾。鉴于$\{\varphi(a_i), \varphi(a\,j), \varphi(a_k)\} \subseteq \{2_a, 2_b, 2_c\}$，根据引理2.2，RECOLORING 算法表明，分配给 a_i, a_j, a_k的颜色序列为以下之一：

当 $n \equiv 1$ (模 4)，$j = n - 1$，且 $k = n$ 时：颜色序列为 $2_a2_b2_c$ 或 $2_b2_c2_a$；当 $n \equiv 2$ (模 4)，$j = n - 1$，且 $k = n$ 时：颜色序列为 $2_a2_b2_a$ 或 $2_b2_b2_a$；当 $n \equiv 3$ (模 4)，$i = n - 3$，$j = n - 2$，且 $k = n$ 时：颜色序列为 $2_b2_a2_b$；当 $n \equiv 3$ (模 4)，$j = n - 3$，且 $k = n - 2$ 时：颜色序列为 $2_a2_a2_b$ 或 $2_b2_a2_b$。

无论哪种情况，$\mathcal{N}[b_i]$ 中的元素颜色都不相同。 □

接下来，我们应用冲突消解算法，通过将其他顶点重新着色为颜色 2_c，以解决由颜色 2_a 或 2_b 引发的潜在冲突对。开始执行冲突消解算法时，我们通过构建字典 $\mathcal{N}$来确定每个 $1 \leq i \leq n$ $\mathcal{N}[b_i]$。给定一个字典 $\mathcal{N}$：$K \to Y$（其中K 是键的集合，Y 是值的集合），其定义域定义如下：

$$\mathrm{dom}(\mathcal{N}) = \{k \in K \mid \exists y \in Y, \mathcal{N}(k) = y\}.$$

这意味着，$\mathrm{dom}(\mathcal{N})$是所有满足以下条件的键的集合：$k$ 对应有值 $\mathcal{N}[k]$ 。很容易看出，此处字典 $\mathcal{N}$的键为$b_1, b_2, \ldots\ldots, b_n$。之后，我们检查每个偶数$i$，判断$\mathcal{N}[b_i]$是否包含一对顶点$a_i$和$a_j$，使它们都被着成 2_a或 2_b。注意，如果存在这样一对

若存在这样的配对，则 j 根据引理唯一确定为 i，依据引理 2.3；随后，我们将 a_i 重新着色为 2_c，如果$\varphi(a_{i-2}) \neq 2_c$；否则，将 aj 重新着色为 2_c。

算法 4：冲突消解($\varphi(a_1)$，$\varphi(a_2)$，$\cdots$，$\varphi(a_n)$)

输入： 顶点的颜色 $\varphi(a_1)$、$\varphi(a_2)$、$\cdots$、$\varphi(a_n)$，其中顶点为 a_1, a_2、$\cdots$、a_n

输出： a_1, a_2、$\cdots$、a_n 重新着色后的颜色

```
L ← {(a1, b1)、(a2, b2)、.....、(an, bn)}
N ← ∅                                              // 初始化一个空字典
对每个 (a, b) ∈ L和 φ(a) ∉ {1, 1′} 执行
    如果 b ∈ dom(N) 则
                                                   // 检查b是否是 N 中的键
        N[乙] ← N[乙] ∪ {甲}                        // 将a 添加到b的邻居集合中
    else
        N[乙] ← {甲}                               // 用 {b 的邻居集合初始化 {a}
返回 N                                             // 返回构造的字典
a0 ←an
对于i ← 1 到n 执行
    如果i ≡ 0 (模 2)则
        J ← {j |a j ∈ N[bi]} \ {i}
        对 jin J 中的每个 j 执行
            如果 φ(ai) = φ(aj) 且 φ(ai) ≠ 2c 则
                如果 φ(ai−2) ≠ 2c 则
                    φ(—i) ← 2c
                如果 φ(ai−2) = 2c 则
                    φ(aj) ← 2c
                                                   // 处理完毕后退出内层循环
                跳出
```

最后，我们整合重新着色算法和冲突消解算法，以构建装箱着色算法。最终得到所需的S-装箱着色 Halin 图G，满足序列S 所规定的所有距离约束：$= (1^2, 2^3)$。然而，这一结论并非显而易见。下一节将对此进行验证。

Alg法 5：装箱着色 (*G*

）输入：一个 Halin 图 G = T ∪C，其中 Δ(G) ≤ 5，T 是特征树，C := $a_1a_2\cdots a_na_1$ 是 G 的伴随圈。输出：G 的一个 $(1^2, 2^3)$-装箱着色 // 使用两种颜色 1 和 1′1 对 T 进行 2-着色 φ 初始化空着色 φ，针对 T 中的所有顶点 2 选择 T 中任意一个根顶点 r 3 从 r 开始执行 BFS，以遍历树 T 4 对于按遍历顺序排列的 T 中每个顶点 v，执行 5 如果 v 是根 r，则6 φ (v) ← 17 否则 8 如果 v 的父顶点颜色为 1，则9 φ(v) ← 1′10 否则11 φ (v) ← 112 重新着色($\varphi(a_1)$，$\varphi(a_2)$，⋯，$\varphi(a_n)$) 13 冲突消解($\varphi(a_1)$，$\varphi(a_2)$，⋯，$\varphi(a_n)$)



3 算法验证与复杂度

该装箱着色算法输出了一个使用以下颜色为G着色的方案：$\{1, 1', 2_a, 2_b, 2_c\}$。为验证这确实是一个 $(1^2, 2^3)$-装箱着色，我们给出以下一系列引理。

引理3.1。由1（或 1′）着色的顶点构成一个独立集。

证明。装箱着色算法首先使用两种颜色 1 和 1′对特征树 T 进行正常着色。到达第 12 行时，算法根据两种不同的情况，对伴随圈上的所有顶点重新着色。在每种情况下，当集合 $\{1, 1'\}$ 中的顶点 a_k 通过 CASE-1 或 CASE-2 算法重新着色时，所选颜色都不同于当前赋予 b_k 的颜色。在 CASE-1、CASE-2 和 RECOLORING 算法的整个执行过程中，伴随圈 C 中任意两个相邻顶点都不会被赋予相同的颜色。值得注意的是，在 CASE-1 算法中，顶点 a_1 会被重新赋予颜色 1_1, ，而顶点 a_n 始终会被重新赋予颜色 1_n。鉴于根据装箱着色算法第 1–11 行和 RECOLORING 算法第 3–6 行确定的 a_1 和 a_n的原始颜色不同，因此 $\varphi(b_1) \neq \varphi(b_n)$，从而 $1_1 \neq 1_n$。因此，由颜色 1（或 1′）着色的顶点构成一个独立集。

鉴于冲突消解算法不会引入任何新的颜色为 1 或 1′的顶点，因此，颜色为 1（或 1′）的顶点所诱导的独立集，在执行第13行后仍然成立。□

A*2*-独立集定义为一个顶点子集，其中任意一对顶点之间的距离至少为 3。

Le引理 3.2。冲突消解算法执行完毕后，

1. 用 2_a （或 2_b）着色的顶点构成一个2-独立集；

2. 对于每个 $i \in [n]$, , 集合 $\mathcal{N}[b_i]$ 中不包含两个使用颜色 2_c着色的顶点 .

证明。 完成 12 行的装箱着色 算法后，颜色 2_a 或 2_b 仅出现在伴随圈C中。

现在，很容易观察到，如果两个顶点 a_i 和 a_j 都使用 2_a（或 2_b）着色，且它们之间的距离至多为 2，那么$b_i = b_j$。在这种情况下，我们简单地说，顶点a_i 存在一个由b_i引起的AB-冲突 。

下面我们证明，冲突消解 算法能够解决所有 AB-冲突。假设a_i 存在一个由 b_i引起的 AB-冲突。由此可得 $|\mathcal{N}[b_i]| \geq 2$。

如果 $|\mathcal{N}[b_i]| = 2$，那么存在另一个顶点 a_j与 b_i相邻，且其颜色与 a_i相同。冲突消解 算法将使用颜色 2_c重新为 或 a_j 中的一个顶点着色，从而解决由b_i引起的 AB-冲突。

若 $|\mathcal{N}[\mathrm{b}_i]| \neq 2$，则根据引理 2.1， $|\mathcal{N}[\mathrm{b}_i]| = 3$。设 a_i、$\mathrm{a\,j}$ 和 a_k 为 C 上与顶点 b_i 相邻的三个顶点。引理 2.3 的证明指出了分配给 a_i、a_j 和 a_k 的初始颜色。执行冲突消解算法后，这些顶点（a_i、a_j 或 a_k）中恰有一个会被重新着色。因此，算法完成后，a_i、a_j 和 a_k 的颜色将分别为 2_a、 2_b 和 2_c。这就消除了由 b_i 引起的 AB-冲突。

另一方面，冲突消解算法不会引入新的 AB-冲突，因为 $\mathcal{N}[b_i] \cap \mathcal{N}[b_j] = \emptyset$对每个$b_i \neq b_j$。因此，应用装箱着色算法时，所有颜色为 2_a 或 2_b 的顶点对之间的距离至少为 3。这证明了第一个结论。

接下来，我们可以用类似的方法证明第二个结论。假设在当前阶段，存在某个 b_i ，使得 $\mathcal{N}[b_i]$ 中有两个顶点被染成颜色 2_c。那么，必然曾发生过由 b_i 和 $|\mathcal{N}[b_i]| = 3$引起的 AB-冲突。然而，前面的证明表明，应用 冲突消解 算法后， $\mathcal{N}[b_i]$ 中的三个顶点分别被染成 2_a、 2_b和 2_c。这与我们的假设矛盾，因此排除了这种情况。□

如果顶点 a_i 在应用 2_c 冲突消解 算法之前的颜色是 ，那么我们称 a_i 为一个 原始 C-顶点。如果顶点 a_i 是一个原始 C-顶点，且 $\mathcal{N}[b_i]$ 中包含颜色为 2_a 或 2_b的顶点，那么我们称 a_i 是分离的。

如果顶点 a_i 在应用冲突消解算法之前的颜色不是 2_c ，但在应用算法后被重新染成 2_c ，那么我们称 a_i 为新引入的 C-顶点。对于新引入的 C-顶点 a_i，在应用冲突消解算法之前，它必然存在由 b_i 引起的 AB-冲突。因此，我们令 $\mathrm{a\,j} \in \mathcal{N}[\mathrm{b}_i]$为一个顶点，使得 a_i和 $\mathrm{a\,j}$ 的颜色相同。注意，根据引理 2.1 和 2.3，j 由 i 唯一确定。为方便起见，我们令 $\mathrm{i}^* := \mathrm{j}$。

引理 3.3。 假设$_i$ 和$_j$ 在执行 2_a 或 2_b 之前均已着色，冲突消解算法，并且 $b_i = b_j$其中 $i < j$。如果 a_j是新引入的 C-顶点，那么在应用冲突消解算法之前，以下性质至少有一项成立：

性质 ***I*** 或者 a_{i-2} 是原始 C-顶点（见图 2（a）），或者是新引入的 C-顶点，其 $(i-2)^* > j$（见图2（b））；或者

性质***II*** 性质 I 不成立，并且存在一个原始 C-顶点 a_{k_0} 以及一列新引入的 C-顶点 $\{a_{k_\ell}:\ 1 \le l \le r\}$，它们沿着 C 上从 a_n 到 a_i 的顺时针路径排列（$k_0 = n$ 是可能的），使得：

- $k_r = i - 2$,
- $k^*_l = k_{l-1} + 2$ 对所有 $1 \le l \le r$，且
- 顶点位于 $\{a_{k_l+1}\}$ 被染成 1 或 1′ 对于每个 $1 \le l \le r$（见图 2（c））；或

性质 ***III*** 性质 I 和性质 II 均不成立，并且存在一列新引入的 C-顶点 $\{a_{k_\ell}:\ 0 \le l \le r\}$沿着从 a_n 到 a_i 的顺时针路径位于 C 上，使得：

- $k^*_0 > j$,
- $k_r = i - 2$,
- $k^*_l = k_{l-1} + 2$ 对于所有 $1 \le l \le r$，且
- 图中的顶点 $\{a_{k_{l-1}+1}\}$ 被着色为 1或 1′ 对于每个 $1 \le l \le r$（见图2 （d））。

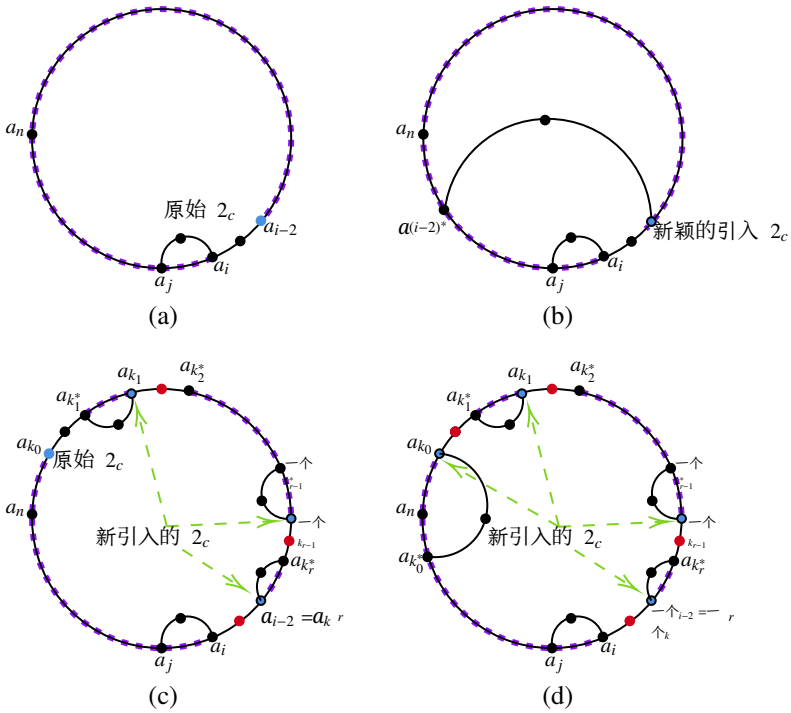

图 2：性质 I、II 和 III 的示意图

证明。由于 a_j 是新引入的 C-顶点，因此我们有 $i = j^*$。根据冲突消解算法第 17 行和第 18 行，当第 10 行的循环进行到这个特定的 i 时，a_{i-2} 的颜色必须是 2_c。如果 a_{i-2} 是原始 C-顶点，或者是满足 $(i-2)^* > j$ 的新引入的 C-顶点，那么该过程终止，结论成立。否则，a_{i-2} 是满足 $(i-2)^* < i$ 的新引入的 C-顶点。根据引理 2，a_{i-1} 的颜色必须是 1 或 1′。

在前面的分析中，将 a_{i-2}（即 a_{j^*-2}）替换为 a_j，我们可以推导出还存在另一个新引入的C-顶点，它沿着 C 更早被遇到（利用 Halin 图没有边交叉这一性质）。同样，根据第 17 行和第 18 行，当循环到达 $(i-2)^*$时，$a_{(i-2)^*-2}$ 的颜色必为 2_c。

– 如果 $a^{(i-2)^*-2}$ 是原始 C-顶点，或是一个新引入的 C-顶点，且 $((i-2)^*-2)^* > j$，则该过程以 $r = 1$ 终止。根据引理 2.2，$a^{(i-2)^*-1}$ 的颜色为 1 或 1′。

– 否则，我们将新引入的 C -顶点 $a^{(i-2)^*-2}$ 替换为 a_{i-2} ，并将 $a_{((i-2)^*-2)^*-2}$ 合并。该顶点可能是：

- 一个原有的 C -顶点，或一个新引入的 C -顶点，其 $(((i-2)^*-2)^*-2)^* > j$（以 $r = 2$ 结束该过程），或者
- 一个新引入的 C -顶点，其 $(((i-2)^*-2)^*-2)^* < j$（需要进一步迭代）。

然而，由于 n 是有限的，该迭代不可能无限进行，因此必然终止。 □

引理3.4。 冲突消解 算法 执行完毕后，使用 2_c 着色的顶点构成一个2-独立集。

证明。 我们通过反证法证明这一结果。假设 a_s 和 a_t $(1 < s < t)$ 由 2_c 装箱着色 算法 着色，且它们之间的距离至多为 2。根据引理 3.3，沿着 a_s 和 a_t 之间的 C 的距离必须至多为 2。

如果 a_s 和 a_t 都不是新引入的 C-顶点，那么引理2.2 告诉我们，这是唯一一种情况：分配给 $a_1, a_2, \ldots, a_n$ 的颜色序列为 $i2_c i2_b i2_a i2_b \cdots i2_a i2_b i2_a i2_c 2_a$。因此，沿 C 从 a_s 到 a_t 的距离为 3，矛盾。

接下来，我们考虑三个不同的情形进行进一步分析。

情形 1。a_s 不是新引入的 C-顶点，而 a_t 是新引入的 C-顶点。

在此情形下，我们有 $s \neq n$。因此，根据引理2.2，分配给 $a_1, a_2, \ldots, a_n$ 的颜色属于以下三个子情形之一：

(i) $i2_c i2_b i2_a i2_b \cdots i2_a i2_b i2_a i2_c 2_a$;

(ii) $i2_c i2_b i2_a i2_b \cdots i2_a i2_b 2_a$;

(iii) $i2_c i2_b i2_a i2_b \cdots i2_a i2_b i2_a$.

For 子情形 (i) 有两种可能：要么是 $s = 2$ 和 $t = 4$，要么是 $s = n-1$ 和 $t = n$.

在 $s = 2$ 和 $t = 4$ 的情况下，参考冲突消解算法的最后六行，a_4 不可能是新引入的 C-顶点。这是因为 $\varphi(a_2) = 2_c$。

在 $s = n-1$ 和 $t = n$ 的情况下，a_n 不可能是新引入的 C-顶点。原因是 b_n 与 a_{n-1} 和 a_n 均相邻，而且还与另一个着色为 2_b 的顶点 a_j 相邻。因此，不存在由 b_n 导致的 AB-冲突。

对于子情形 (ii) 或 (iii)，我们有 $s = 2$ 和 $t = 4$。类似地，a_4 不可能是新引入的 C-顶点，因为 $\varphi(a_2) = 2_c$。

Case 2。a_s 是新引入的 C-顶点，而 a_t 不是新引入的 C-顶点 x

在这种情况下，我们有 $t \neq 2$ (因为 $\varphi(a_1) \notin \{2_a, 2_b, 2_c\}$ 因此它不可能是新引入的 C-顶点）。因此，根据引理 2.2，分配给 $a_1, a_2, \ldots\ldots, a_n$ 的颜色属于以下三种子情形之一：

(i) $i2_a i2_b \cdots i2_a i2_b 2_c$，其中 $\mathcal{N}[b_n]$ 有一个被 2_a 或 2_b 着色的顶点 a_j，且 $\mathcal{N}[b_n]$ 除被 2_b 着色的顶点 a_j 外没有其他顶点（当 $j = n-1$ 时），或者 b_2 不与被 2_a 着色的 $a_j \neq a_2$ 相邻；

(ii) $i2_c i2_b i2_a i2_b \cdots i2_a i2_b i2_a i2_c 2_a$，其中存在一个顶点$b_j$，它同时与$a_{n-1}$和$a_n$相邻，并且还与另一个被 2_b着色的顶点a_j 相邻。

(iii) $i2_a i2_b \cdots i2_a i2_b i2_c$，其中 $\mathcal{N}[b_n]$没有任何顶点被染成 2_a或 2_b，且 b_2与一个被染成 2_a 的 $a_j \neq a_2$相邻。

对于子情形 (i)，我们有 $s = n-1$ 和 $t = n$。由于 a_n 被染成 2_c，因此对于所有 i，都有 $i^* \not> n-1$。此外，a_{n-1} 是一个新引入的 C-顶点，而 a_{n-3} 未被染成 2_c。因此，我们不满足引理 3.3 的性质 I 或性质 III（其中 $j := n-1$ 且 $i := (n-1)^*$）。相反，我们属于该引理的性质 II，该性质要求存在一个原始 C-顶点 a_{k_0}，以及沿 C 上从 a_n 到 a_i 的顺时针路径上的一系列新引入的 C-顶点 $\{a_{k_\ell} : 1 \le l \le r\}$，使得对于所有 $1 \le l \le r$，都有 $k_r = i-2$ 且 $k^* l = k_{\ell-1}+2$。

由于 a_n 是唯一的原始 C-顶点，因此必有 $k_0 = n$。于是，该序列满足 $k_1^* = 2$，并且 b_2 与某个顶点相邻，该顶点不是 a_2 且被标记为 2_a。根据子情形 (i) 的条件，这意味着 $\mathcal{N}[b_n] \cap \{2_a, 2_b\} \neq \emptyset$。假设 b_n 与被标记为 2_a 或 2_b 的 a_x $(x \neq n)$ 相邻。由于 G 是一个 Halin 图，因此它必须满足 $k_1^* > x$。鉴于 $k_1^* = 2$，唯一可能的 x 是 $x = 1$。然而，a_1 被标记为 1 或1′，这与 a_x 被标记为 2_a 或 2_b 的假设矛盾。

对于子情形 (ii)，由于 a_1 不可能是新引入的 C-顶点，我们有 $s = n-3$ 且 $t = n-1$。在此子情形中，b_n 不会造成任何 AB-冲突，而 a_{n-2} 被标记为 1 或 1′。因此，对所有 i，都有 $i^* \not\geq n-3$。于是，按照前面的类似推理，我们得到引理 3.3 的性质 II（其中 $j := n-3$ 且 $i := (n-3)^*$）。该性质要求存在一个原始 C-顶点 a_{k_0}，以及沿着从 a_n 到 a_i 的 C 上顺时针路径排列的一系列新引入的 C-顶点 $\{a_{k_\ell} : 1 \le l \le r\}$。该序列必须满足 $k_r = i-2$，并且对于所有 $1 \le l \le r$，都有 $k^* l = k_{\ell-1} + 2$。

只有两个原始 C-顶点。如果 $k_0 = n$，那么 $k_1^* = 2$，这不可能，因为 a_2 是一个原始 C-顶点。如果 $k_0 = 2$，那么 $k_1^* = 4$。由于 a_j 的颜色为 2_b，且 $\mathcal{N}[b_j] = \{a_j、a_{n-1}、a_n\}$，由 G 是 Halin 图这一事实可推知 $k_1^* > j$。因此 $j \le 3$。然而，a_1 和 a_3 的颜色都是 1 或 1′，而 a_2 的颜色为 2_c。这与假设 a_j 的颜色为 2_b相矛盾。

对于子情形 (iii)，我们可以按照处理子情形 (i) 的类似方式处理。

情形 3。a_s 和 a_t 都是新引入的*C*-顶点。

首先，我们有 $t^* < s$。否则，根据冲突消解算法第 17–18 行，a_t 不会被重新着色为 2_c。根据 t^*的定义，我们知道 b_t 引发了 AB-冲突，且 a_{t^*} 的颜色为 2_a 或 2_b。由引理2.2可知，无论哪种情况，a_1 的颜色都是 1 或 1′，并且原始 C-顶点至多只能出现在 a_2,、a_{n-1}和 a_n中。因此，$t^* \ge 2$，且 a_{t^*} 不是原始 C-顶点。如果存在一个原始 C-顶点 a_j，满足 $t^* < j < s$，那么 $j = 2$。这意味着 $t^* = 1$，矛盾。

现在，由于每个 a_j with $t^* < j < s$ 都不是原始的 *C*-顶点，因此 a_{s-2} 也不是原始的*C*-顶点。鉴于*G* 是一个 Halin 图，对于每个$t^* < x < s$，我们有 $t^* < s^* < s$ 以及 $x^* \le s$。

应用引理3.3with $j = s$ and $i = s^*$，我们观察到其性质 I 不成立。

若性质 II 成立，则存在一个原始 C-顶点 a_{k_0}，以及一列新引入的 C-顶点 $\{a_{k_\ell} : 1 \le l \le r\}$，它们沿着 C 上从 a_n 到 a_i 的顺时针路径排列，使得对所有 $1 \le l \le r$，都有 $k_r = i-2$ 且 $k^* l = k_{\ell-1} + 2$。由于 G 是 Halin 图，$k_0 + 2 \pmod n = k_1^* > t^* \ge 2$。根据引理2.2，$k_0$必为 $\{2, n-1$或 $n\}$。因此，$k_0 = 2$，进而 $k_1^* = 4$。这迫使 $t^* = 3$。然而，根据引理2.2，a_3 始终着色为 1 或 1′，这与 a_{t^*} 着色为 2_a 或 2_b这一事实矛盾。

现在考虑最后一种情形，即引理 3.3 的性质 III 成立。也就是说，存在一列新引入的 C-顶点 $\{a_{k_\ell} : 0 \le l \le r\}$，它们沿着 C 上从 a_n 到 a_i 的顺时针路径排列，并满足：$k_0^* > s$、$k_r = i - 2$，且对所有 $1 \le l \le r$，都有 $k^* l = k_{\ell-1} + 2$。由于 G 是 Halin 图，其结构性质意味着 $t^* < k_0 < s$。然而，这与已确立的条件直接矛盾：对于每个 $t^* < x < s$，都有 $x^* \le s$。□

定理 3.5。 该 *PACKING-COLORING* 算法返回一个 $(1^2, 2^3)$-装箱着色。

证明。 这可直接由引理 3.1、3.2 和 3.4得出。 □

定理 3.6。 该 *PACKING-COLORING* 算法在线性时间内运行，即其时间复杂度为 $O(|G|)$。

证明。首先，使用深度优先搜索（BFS）遍历对特征树 T 执行二着色过程，运行时间为 O(t)，其中 t 表示 T 中顶点的数量。重新着色算法包含 CASE-1 和 CASE-2 子程序，每种情况的处理时间均为 O(n)，其中 n 为伴随圈 C 中顶点的数量。用于消解颜色冲突的冲突消解算法也在 O(n) 时间内运行。这是因为构造集合 $\mathcal{L}$ 和字典 $\mathcal{N}$ 各需 O(n) 时间。冲突消解循环遍历从 1 到 n 的每个偶数索引 i。对于每个偶数 i，算法都会计算集合 J。由于 J 的大小大致不超过 4（注意，$\mathcal{N}[b\ j]$最多包含 5 个顶点，因为 G 的最大度不超过 5），遍历 J 的内层循环以常数时间 O(1) 运行，因为其最多执行 4 次迭代。因此，整个冲突消解循环的时间复杂度仍为线性，即 O(n)，因为外层循环为 O(n)，内层循环为 O(1)。因此，PACKING-COLORING 算法的总体时间复杂度为 $O(t + n) = O(|G|)$。□

4 开放问题

当应用 装箱着色 算法为 Halin 图G 着色时，$\Delta(G) \le 5$，我们观察到颜色 2_c 只被分配给相对较少的顶点。这一观察促使我们研究能否修改该算法，以确保任意两个使用 2_c着色的顶点之间的最小距离超过当前约束。我们将这一问题形式化如下：

问题 4.1。 最大度至多为 *5* 的 *Halin* 图 $(1^2, 2^2, 3)$-可装箱着色吗？

在装箱着色 算法中，Halin 图的特征树的所有非叶顶点G 均被着色为 1 或 $1'$。这实际上给出了 $1^2, 2^3$-装箱着色 G。或许下面的问题很有趣：

问题 4.2。 设 G 是最大度为 Δ 的 *Halin* 图，其中 T 是特征树，C 是伴随圈。满足什么最小 $l := l(\Delta)$ 使得 G 存在一个 $(1_2, 2^\ell)$-装箱着色，并且 $V(T) \setminus V(C)$ 中的所有顶点（即 T 中不在 C 上的内部顶点）都使用序列中标记为 1的前两种颜色着色？

对于最大度为 $\Delta \ge 6$ 的 Halin图，我们提出装箱着色理论中的以下基本问题：

问题 4.3。 确定最优参数(k, l)，使得每个最大度为 Δ 的 *Halin* 图 G 都存在$(1^k, 2^\ell)$-装箱着色。具体而言，随着Δ 增大，(k, l) 的可行区域如何扩展？

致谢

作者感谢 Ping Chen、Mengyao Dai 和 Yali Wu 参与了关于本文主题的初步讨论。特别是，第一作者感谢 Weichan Liu 为本文所含图示的绘制提供指导。

参考文献

[1] J. Balogh、A. Kostochka 和 X. Liu. 三次图细分的装箱色数。*Graphs and Combinatorics*, 35(2):513–537, 2019年。[2] J. Balogh、A. Kostochka 和 X. Liu. 三次图细分的装箱着色。*GraphsandCombinatorics*, 35(2):513–537, 2019年。[3] H. A. Bazzal. 关于次三次图的 S-装箱着色。arXiv:2409.01769。[4] O. V. Borodin 和 A. O. Ivanova. 平面次三次图的二距离四着色。*Diskretnyĭ Analizi Issledovanie Operatsiĭ*, 18(2):18–28, 2011年。[5] B. Brešar、J. Ferme 和 K. Kamenická. 距离图 S-装箱着色 $G(\mathbb{Z}, \{2, t\})$。离散应用数学, 298:143–154, 2021年。[6] B. Brešar、J. Ferme、S. Klavžar 和 D. F. Rall. 装箱着色综述。*Discussiones Mathematicae. Graph Theory*, 40(4):923–970, 2020年。[7] B. Brešar、S. Klavžar、D. F. Rall 和 K. Wash. 装箱色数、(1, 1, 2, 2)-着色及彼得森图的刻画。*Aequationes Mathematicae*, 91(1):169–184, 2017年。[8] B. Brešar、S. Klavžar、D. F. Rall 和 K. Wash. 图中局部变化下的装箱色数。*Discrete Mathematics*, 340(5):1110–1115, 2017年。[9] B. Brešar、B. Samadi 和 I. G. Yero. 重新审视图的单射着色。*DiscreteMath.*, 346(5):12, 2023年。编号/文号 113348。[10] Y. Bu、D. Chen、A. Raspaud 和 W. Wang. 平面图的单射着色。*Discrete Appl. Math.*, 157(4):663–672, 2009年。[11] Y. Bu、C. Qi 和 J. Zhu. Halin 图的单射着色。载于信息与管理中的算法方面。第*14*届国际会议，*AAIM 2020*，中国金华，*2020*年*8*月*10*日至*12*日。论文集，第492–500页。Cham：Springer，2020年。[12] D. Cranston 和 S. Kim. 次三次图平方图的列表着色。*Journal of GraphTheory*, 57:65–87, 2008年。[13] F. Foucaud、H. Hocquard、S. Mishra、N. Narayanan、R. Naserasr、Éric Sopena 和 P. Valicov. 平面次三次图的精确平方染色。离散应用数学, 293:74–89, 2021年。

[14] N. Gastineau 和 O. Togni. 三次图的 S-装箱着色。Discrete Mathematics, 339(10):2461–2470, 2016。[15] W. Goddard、S. T. Hedetniemi、S. M. Hedetniemi、J. M. Harris 和 D. F. Rall. 图的广播色数。组合学, 86:33–49, 2008年。[16] S. Hartke、S. Jahanbekam 和 B. Thomas. 平面次三次图平方图的色数。arXiv:1604.06504。[17] P. Holub 和 J. Hofman. 距离图 D(1, t) 和 D(1, 2, t) 的 S- 装箱着色。应用数学与计算 , 447:11, 2023 年。编号 / 文号 127855。[18] A. Kostochka 和 X. Liu. 次三次外平面图的装箱 (1, 1, 2, 4)-着色。离散应用数学, 302:8–15, 2021年。[19] R. Liu、X. Liu、M. Rolek 和 G. Yu. 某些次三次图的装箱 (1, 1, 2, 2)-着色。离散应用数学, 283:626–630, 2020年。[20] X. Liu、X. Zhang 和 Y. Zhang. 每个次三次图都可进行装箱 (1, 1, 2, 2, 3)-着色。Discrete Mathematics, 348(11):7, 2025年。编号/文号 114610。[21] M. Mortada. 关于3-非正则次三次图的 S-装箱着色。Discrete Applied Mathemat- ics, 359:16–18, 2024年。[22] M. Mortada 和 O. Togni. 关于次三次图的 S-装箱着色。Discrete Mathematics, 347(5):6, 2024年。编号/文号 113917。[23] M. Mortada 和 O. Togni. 关于2-饱和次三次图的 S-装箱着色。澳大利亚组合学杂志, 90:155–167, 2024年。[24] M. Mortada 和 O. Togni. 次三次图 S-装箱着色的更多结果与问题。Discrete Mathematics, 348(4):9, 2025年。编号/文号 114376。[25] M. Mortada 和 O. Togni. 关于有界度图的 S-装箱着色。arXiv:2503.18793[math.CO] (2025), 2025年。[26] J. Nešetˇril 和 P. Ossona de Mendez. 《稀疏性——图、结构与算法》。Springer-Verlag，柏林、海德堡，2012年。[27] Priyamvada 和 B. Panda. 某些图类的精确平方染色：复杂性与算法。理论计算机科学, 932:84–101, 2022年。[28]D. A. Quiroz. 弦图的精确距离图着色。Discrete Mathematics, 343(5):111769, 2020年。[29] B. Tarhini 和 O. Togni. 三次Halin图的S-装箱着色。离散应用数学, 349:53–58, 2024年。[30] C. Thomassen. 平面三次图的平方图可用7种颜色着色。组合论杂志B辑, 128:192–218, 2018年。

[31] J. van den Heuvel、H. Kierstead 和 D. A. Quiroz。精确距离图的色数。组合理论杂志*B*辑，134：143–163，2019年。[32] Y. Wang、X. Hu 和 W. Wang。最大度为五的 Halin 图的平方图的色数为六。组合学，133：217–231，2017年。[33] W. Yang 和 B. Wu。关于次三次图的 *S*-着色打包。离散应用数学，334：1–14，2023年。