arXiv:2302.13644v1 [cs.DS] 27 Feb 2023

3-COLORING 在时间 $\mathcal{O}^*(1.3217^n)$内

Lucas Meijer[1]

l.meijer2@uu.nl[1]乌得勒支大学，信息与计算科学系

2023年2月28日

摘要

我们提出了一种新的 3-COLORING 算法，其运行时间为 $\mathcal{O}^*(1.3217^n)$。对于该算法，我们使用 Beigel 和 Eppstein 提出的时间 $\mathcal{O}^*(1.3289^n)$算法来解决 3-COLORING 。他们描述了一种适用于所有图的结构，其中的顶点可以相对容易地着色。本文改进了这一结构，并提出了新的方法，以确定相关顶点如何降低算法的运行时间。

1 引言

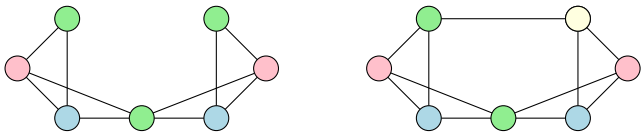

图 1：左：可3着色图。右：不可3着色图。

3-COLORING 问题是图论中最基本的问题之一。 3-COLORING问题提出了这样一个问题：给定一个有 n 个顶点的图 G，我们能否为每个顶点分配三种颜色（即红色、绿色或蓝色）中的一种，使得任意两个相邻顶点的颜色都不同?

值得注意的是， 3-COLORING 是 GRAPH COLORING 问题的一个特例。在 GRAPH COLORING中，我们的目标是最小化为图着色所需的颜色数，同时确保任意两个相邻顶点的颜色不同。 GRAPH COLORING 是复杂性理论中的经典问题；它是 Karp 最初提出的21 NP-完全问题之一 [5]。Karp 提出该问题一年后，Lov´asz 证明了 3-COLORING 是 NP-完全问题 [7]。具体来说，这是使判断一个图能否用该数量的颜色着色成为 NP-完全问题的最小颜色数： 0-COLORING 和 1-COLORING 都可以通过多项式时间算法轻松判定，而如果存在 2-COLORING 图，则可以通过贪心地为顶点着色找到它。

1.1 3-着色的历史

显然， 3-COLORING 可以通过尝试每一种颜色赋值，并在多项式时间内验证其是否能得到正确解，在时间 $\mathcal{O}^*(3^n)$ 内求解。第一个非平凡算法于 1976 年提出

Year	作者	Time
1976	Lawler	$\mathcal{O}^*(1.4425^n)$
1994	Schiermeyer	$\mathcal{O}^*(1.415^n)$
2000	Beigel 和 Eppstein	$\mathcal{O}^*(1.3289^n)$

表 1： 3-COLORING 以往算法概览。

，由 Lawler [6] 提出。他发现， 3-COLORING 可以通过以多项式延迟遍历所有极大独立集来求解。对于图 G 中的每个极大独立集 I，我们为该极大独立集中的所有顶点赋予一种颜色。然后，我们验证 $G\backslash I$ 是否可 2-着色。如果不存在独立集 I，使得 $G\backslash I$ 可 2-着色，则 G 不可 3-着色。Moon 和 Moser 表明，任何图 [8], 中至多存在 $\mathcal{O}(3^{n/3}) = \mathcal{O}(1.44225^n)$ 个极大独立集，我们最多可在时间 $\mathcal{O}(3^{n/3}) = \mathcal{O}(1.44225^n)$ [4] 内遍历它们。验证 $G\backslash I$ 是否可 2- 着色需要多项式时间，因此 Lawler 算法的运行时间为 $\mathcal{O}^*(3^{n/3})$。

1994 年，Schiermeyer 找到了一种改进算法，可在时间 $\mathcal{O}^*(1.415^n)$ [9] 内解决 3-COLORING 。关键在于，Schiermeyer 的算法基于这样一个想法：如果图可以 3-着色，那么任意顶点的邻居集合都必须可 2-着色。

在本文提出时间复杂度为 $\mathcal{O}^*(1.3217^n)$ 的算法之前，已知解决 3-COLORING 的最佳算法是 Beigel 和 Eppstein 提出的时间复杂度为 $\mathcal{O}^*(1.3289^n)$ 的算法 [1]。Beigel 和 Eppstein 于 2000 年发表了这篇论文。他们设计了一种算法，可在时间$\mathcal{O}^*(1.36444^n)$ 内求解 (3,2)-CONSTRAINT SATISFACTION PROBLEM ((3,2)-CSP)。3-COLORING 问题可以归约为 (3,2)-CSP ，且不会增加实例规模，这意味着 3-COLORING 也可以在时间 $\mathcal{O}^*(1.36444^n)$ 内求解。此外，Beigel 和 Eppstein 发现，有些顶点比使用 (3,2)-CSP 算法时更容易着色。通过先为一部分颜色着色，他们得到了时间复杂度为 $\mathcal{O}^*(1.3289^n)$ 的算法。我们的算法将在 Beigel 和 Eppstein 的算法基础上进一步拓展和改进。

1.2 图着色的现状

给定一个图 G，我们可以确定 G 是否可以用至多 k 种颜色着色。值得注意的是，在 GRAPHCOLORING 中，我们希望最小化 k 的值，使得可以用 k 种颜色为 G 着色。利用容斥技术 [2]， GRAPH COL-ORING 问题可以在 $\mathcal{O}^*(2^n)$ 时间内解决。令人惊讶的是，对于任意 $k \geq 7$，时间复杂度为 $\mathcal{O}^*(2^n)$ 的算法都是目前已知最快的。

目前已知最快的 4-COLORING 算法的运行时间为 $\mathcal{O}^*(1.7272^n)$，由 Fomin 等人于 2007 [3] 发表。Fomin 等人证明，图要么极大独立集的数量较少，要么路径宽度较小。他们创建了一种遍历所有极大独立集的算法，以及一种关于图的路径宽度固定参数可解的算法；根据图的具体情况，他们选择运行其中一种。遍历所有独立集的算法将目前已知的最佳 3-COLORING 算法用作子程序。因此，我们对 3-COLORING 的改进也改进了 4-COLORING。现在，我们可以在 $\mathcal{O}^*(1.7247^n)$ 时间内解决 4-COLORING 。

最后，对于 5-COLORING 和 6-COLORING，Zamir 最近的研究进展表明， 5-COLORING和 6-COLORING 都可以在时间 $\mathcal{O}^*((2-\epsilon)^n)$内求解，其中 ϵ取较小值（两者的 ϵ不同） [10]。对于任意$k \geq 7$，是否存在一个时间为 $\mathcal{O}^*((2-\epsilon)^n)$的算法来求解 k-COLORING，仍然是一个悬而未决的问题。

1.3 我们的贡献

在本文中，我们证明了以下定理：

定理 1.1。 存在一个用于 3-COLORING 的算法，其运行时间为 $\mathcal{O}^*(1.3217^n)$ ，适用于具有 n个顶点的图 s.

我们改进了 Beigel 和 Eppstein 的时间为 $\mathcal{O}^*(1.3289^n)$的算法。在二十多年来，他们的算法一直是已知的最佳结果。因此，我们认为运行时间方面的改进对于 3-COLORING 问题而言是一项重要成果。

1.4 组织结构

我们进一步发展了 Beigel 和 Eppstein 的许多想法；并在第 2 节讨论他们的算法。我们在第 3 节介绍一种新的图结构——极大低幅度繁茂森林，它有助于我们找到相对容易着色的顶点。我们在第 4 节阐明极大低幅度繁茂森林如何帮助我们更快地为顶点着色。我们在第 5 节结合各项发现，通过构造一个线性规划来分析算法的运行时间。我们在第 6 节总结算法。

2 Beigel 和 Eppstein 算法概要

在本节中，我们讨论 Beigel 和 Eppstein 提出的用于求解 3-coloring 的时间复杂度为 $\mathcal{O}^*(1.3289^n)$ 的算法[1]。在本文提出改进之前，Beigel 和 Eppstein 的算法是已知的求解 3-coloring 速度最快的算法。值得注意的是，我们通过改进他们的算法构建了自己的算法。我们将介绍 Beigel 和 Eppstein 提出的、对我们的改进至关重要的概念。我们对 Beigel 和 Eppstein 算法中的一些引理略作修改，以便使其适用于我们的算法。

Beigel 和 Eppstein 表明，可以将 (3,2)-Constraint Satisfaction Problem ((3,2)-CSP) 当作黑盒来高效求解 3-coloring 。(3,2)-Constraint Satisfaction Problem由一组变量组成，每个变量都必须被赋予至多三种颜色中的一种。变量及其某种颜色的组合称为变量-颜色对。(3,2)-Constraint Satisfaction Problem 包含两个变量-颜色对之间的约束：两个顶点不能同时被赋予各自变量-颜色对中的颜色。我们将 3-coloring 转换为 (3,2)-Constraint SatisfactionProblem ，具体如下：

1. 每个变量代表一个顶点。
2. 所有变量的颜色域都与其所代表顶点的颜色域相同。
3. 对于代表相邻顶点的任意两个变量，我们添加一项约束，规定它们不能同时取相同的颜色。

定理 2.1 ([1, 定理 1])。存在一种求解 (3,2)-Constraint Satisfaction Problem 的算法，其运行时间为 $\mathcal{O}^*(1.36443^n)$，其中 n 是变量的数量。

Beigel 和 Eppstein 证明，(3,2)-Constraint Satisfaction Problem 可以在时间$\mathcal{O}^*(1.36443^n)$内求解。该归约不会增加实例规模，因此 3-coloring 也可以在时间$\mathcal{O}^*(1.36443^n)$内求解。

值得注意的是，给定一个部分着色图，我们也可以使用 (3,2)-CSP 的算法来判断是否存在一种对未着色顶点的颜色赋值，使其成为有效的 3-着色。在 Beigel 和 Eppstein 的 (3,2)-Constraint Satisfaction Problem 算法中，存在一条归约规则，可以在多项式时间内从实例中移除任何可选颜色不超过两种的变量。已着色顶点的邻居至多有两种可能的颜色。因此，可以将这些顶点从实例中移除。例如，考虑一个较小的顶点集 S ，它有一个较大的邻居集 T。对 S中的顶点，至多存在 $3^{|S|}$ 种有效着色赋值。通过遍历这些颜色赋值，我们可以在时间 $\mathcal{O}^*(1.36443^{|V|-|S|-|T|} \cdot 3^{|S|})$内求解 3-coloring 。

显然，当 $|T|$ 相对于 $|S|$足够大时，这将改善算法的运行时间。本节将说明 Beigel 和 Eppstein 如何找到一个较小的顶点集进行颜色标记，以改进用于 3-coloring 的算法，以及我们如何调整他们的引理，使其在新算法的背景下得到改进。

2.1 工作因子

Beigel 和 Eppstein 的算法使用许多分支规则来从图中排除某些情况。分支规则是一种通过递归求解多个实例来求解一个实例的策略。为了分析这些分支，他们使用工作因子：

定义 2.1 （工作因子）。工作因子，$\lambda(r_1, r_2, \dots)$，表示分支规则的复杂度。分支规则会探索若干个分支。每个分支都会探索一个规模缩减后的实例。在工作因子中，每个数值 $r_1, r_2, \dots$ 表示某个分支中的实例规模缩减量。每个工作因子 $\lambda(r_1, r_2, \dots) = c$，其中 c是函数 $f(x) = 1 - \sum x^{-r_i}$的最大零点。工作因子有助于我们计算算法的运行时间。每一步都会应用某个分支规则。每条分支规则都有一个工作因子 λ：算法的运行时间为 $\mathcal{O}^*((\max \lambda)^n)$。因此，如果所有工作因子均为 $\lambda \leq 1.3217$，算法的运行时间为 $\mathcal{O}^*(1.3217^n)$。

2.2 移除低度顶点

根据定义，低度顶点的邻居很少。因此，我们希望避免尝试它们的所有颜色分配。事实上，为低度顶点着色，只会减少其他少数顶点可用的颜色。此外，它们也相对不太可能拥有已着色的邻居。幸运的是，我们可以移除许多低度顶点。

拥有两个或更多邻居的顶点可以直接从实例中移除：它们的邻居集合不可能包含全部三种可能的颜色。因此，这些顶点总能获得一种可用的颜色。

此外，我们还可以限制度为三的顶点数量：我们使用一条分支规则，确保不存在包含九个或更多顶点或任何环的三度顶点连通子图。

引理 2.2 ([1, 引理 20])。 设 G 是一个 3-COLORING 实例，其中某个环仅由度为三的顶点组成。那么，我们可以将 G 替换为更小的实例，其工作因子至多为 1.2433。

证明。 参见 Beigel 和 Eppstein[1,引理 20]。 □

下一个引理改编自 Beigel 和 Eppstein [1, 引理 21]。原引理移除了包含八个或更多度为三的顶点的连通子图，但我们将其修改为移除包含九个或更多度为三的顶点的连通子图。

引理 2.3。 设 G 是一个包含至少九个度为三顶点的连通子图的 3-COLORING 实例。那么，我们可以将 G 替换为更小的实例，且工作因子至多为1。3022。

证明。 注意，任意顶点的邻居都必须是可 2-着色的。因此，如果某个顶点有三个邻居，那么其中至少两个邻居必须被赋予相同的颜色。可供选择的邻居对有三种，我们将在每个分支中分别探索一种。

当我们选择两个顶点使其颜色相同时，可以将这两个顶点合并：新顶点将与这两个原始顶点中任意一个曾经相邻的所有顶点相邻。由于合并后的顶点都与同一个三度顶点相邻，该顶点在合并后将变为度为二。此时，可以从实例中移除该顶点。此外，移除该顶点后，我们还会移除它的所有三度邻居。

假设一个由三度顶点构成的连通子图有 $n \geq 9$ 个顶点。现在，我们选择一个顶点 v，并在三个分支中分别合并它的邻居。如果子图中存在一个在该子图内有三个邻居的顶点，就令任意一个这样的顶点为 v。否则，该子图必定是一条路径。此时，选择一个顶点作为 v，使得从子图中移除该顶点后，子图被分为大小分别为$\lceil \frac{n}{2} \rceil$和 $\lfloor \frac{n}{2} \rfloor$的连通分量。

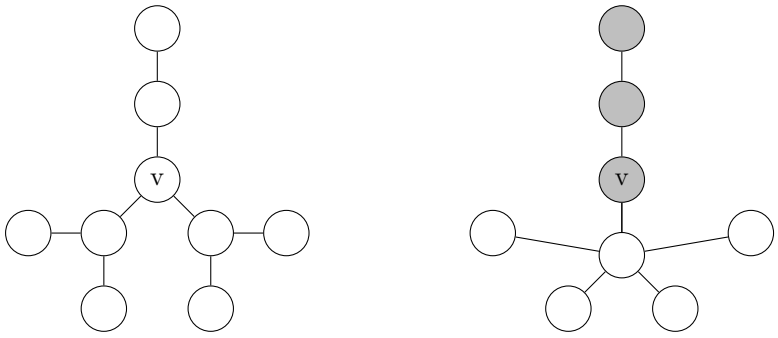

图2：由九个三度顶点构成的连通子图（左），以及三个分支之一，它合并了两个邻居（右）。 v 底部的邻居被合并。灰色：可从实例中移除的顶点。

在每个分支中， v 会从实例中移除，并将它的两个邻居合并为一个：每个分支都会使实例规模至少减少一个。然后， v 的未被合并的邻居会失去一个邻居，并且可能只剩下两个顶点。在这种情况下，它也会被移除。我们继续这一过程，直到子图中不再存在度为二的顶点。从子图中的每个

顶点到 v 都恰好存在一条路径，因此除 v 外的每个顶点都能在恰好一个子图中从实例中移除。因此，我们得到一个工作因子 $\lambda(2+\alpha_1,\ 2+\alpha_2,\ 2+\alpha_3)$，其中 $\alpha_1+\alpha_2+\alpha_3=n-1$。此外，如果 α_1、α_2 或 α_3 中任意一个为零，那么另外两个必定分别为 $\lceil \frac{n}{2} \rceil$ 和 $\lfloor \frac{n}{2} \rfloor$。

α_1、α_2 和 α_3 的最坏情况值分别为 0、4 和 4。我们得到的工作因子为 $\lambda(2,\ 6,\ 6)\ =1.3022$。 □

2.3 更快地为顶点着色

在限制图中低度顶点的数量后，Beigel 和 Eppstein 使用一种称为茂密森林的图结构，来寻找一个较小的顶点集，使其拥有一个较大的邻接顶点集。

定义 2.2（茂密森林）。茂密森林 是一种森林，其中每棵树都至少有一个内部顶点，并且每个内部顶点都必须与树中至少四个其他顶点相邻。当不存在任何位于茂密森林之外且有四个邻居也位于茂密森林之外的顶点、不存在任何茂密森林中有三个邻居位于茂密森林之外的叶节点，并且不存在任何与茂密森林的内部顶点相邻的茂密森林外顶点时，该茂密森林称为极大的。

我们找到一个极大繁茂森林，并为其中的内部顶点着色。所有叶子都与某个内部顶点相邻，因此每个叶子都有一个已着色的邻居。每个内部顶点在繁茂森林中至少有四个邻居，因此我们预计森林中会有许多叶子。为分析可能的颜色分配数量，我们根据图的顶点与极大繁茂森林之间的关系，将这些顶点划分如下：

- R：繁茂森林中各棵树的根顶点，其中每棵树都任意选取一个内部顶点作为根顶点。
- I：繁茂森林中除根顶点以外的所有内部顶点。
- L：繁茂森林的叶子。
- N：繁茂森林外与繁茂森林相邻的顶点。
- U：繁茂森林外与繁茂森林不相邻的顶点。

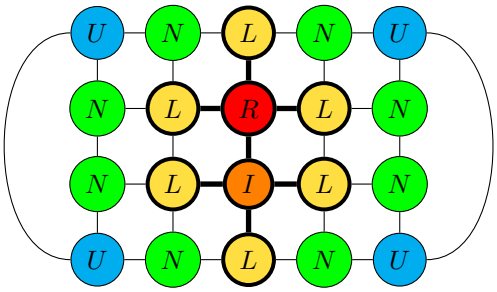

图 3：图中由一棵树构成的极大繁茂森林（粗体）。红色：根顶点（R）。橙色：内部顶点（I）。黄色：叶子（L）。绿色：茂密森林的邻居（N）。青色：其他顶点（U）。

给定一个具有任意极大繁茂森林的图，我们使用这些划分来分析算法的运行时间。对于 R 和 I，即 F 的内部顶点，我们尝试所有可能的颜色分配。每个根顶点都有三种可能的颜色，因此 $|R|$ 中顶点的可能颜色分配共有 $3^{|R|}$ 种。之后，每棵树的根顶点都已着色。与 R 相邻的 I 中的顶点有两种可用颜色。在一个邻居着色后，我们为 I 中的每个顶点着色：这些顶点的可能颜色分配只有 $2^{|I|}$ 种。然后，L 中的每个顶点至少有一个已着色的邻居，这意味着可以在多项式时间内将这些顶点从实例中移除。这样一来，N 和 U 中的顶点就是需要通过

(3,2)-Constraint Satisfaction Problem 来解决的顶点：这需要 $\mathcal{O}^*(1.36443^{|N|+|U|})$ 时间。总的来说，给定一个茂密森林 F，求解 3-coloring 需要 $\mathcal{O}^*(3^{|R|} \cdot 2^{|I|} \cdot 1.36443^{|N|+|U|})$ 时间。

幸运的是，我们还可以对 N 和 U 中的一些顶点着色，从而缩短算法的运行时间。为此，我们在未被茂密森林 F覆盖的顶点 V 上寻找另一片森林。我们将这个图记作 $G[V-F]$：即 G中 $V-F$的导出子图。

定义 2.3（色彩森林）。由有根树构成的森林，其中每棵树的根恰有三个子节点，且至多有五个孙节点。根的每个子节点自身至多有两个子节点。

Beigel 和 Eppstein 证明，U中必定存在一片覆盖所有顶点的色彩森林，即一片极大色森林。Beigel 和 Eppstein 证明，以下算法总能找到一个有效的极大色森林。这里，$K_{1,3}$ 树是指具有一个根顶点和三个子节点的树。

1. 创建一片由$K_{1,3}$ 树构成的极大森林。
2. 只要可行，就移除一棵$K_{1,3}$ 树，并添加两棵新的树。
3. 将 U 中所有剩余顶点分配为某个 $K_{1,3}$ 树的孙节点，使得每棵树都没有六个孙节点。

Beigel 和 Eppstein 对色彩森林中树的分析与我们的分析略有不同。他们用三度顶点的数量表示颜色分配的数量。相反，我们用 $G[V-F]$ 中顶点的数量表示颜色分配的数量。这样做是为了通过证明某些邻居超过三个的顶点必须包含在极大色森林中，进一步改进运行时间。

关于下一个引理，类似的证明见 Beigel 和 Eppstein 的 [1, 引理 24]。他们用树中三度顶点的数量表示对色彩森林中的树进行着色的运行时间。此外，他们还假设，由三度顶点构成的连通子图中最多有八个顶点。

引理 2.4。令 T' 是色彩森林中的一棵树。那么 T' 的工作因子为 1。34004 .

证明。假设 T 至多有四个孙节点。在这种情况下，我们为根指定三种可能颜色中的一种。这样一来，它的三个子节点便少了一种可选颜色，因此可以通过 (3,2)-CSP 算法将它们从实例中移除。孙节点仍然存在，必须由 time$\mathcal{O}^*(1.36443^n)$ 算法处理。由于每个孙节点都由 $\mathcal{O}^*(1.36443^n)$ 算法着色，因此着色速度相对较慢。最坏情况是孙节点数量达到上限，即四个。我们遍历根顶点的三种可能颜色分配，而有四个顶点需要由 (3,2)-CSP 算法着色。平均而言，为树中的每个顶点着色、从而为树中的所有顶点着色所需的运行时间为 $(3 \cdot 1.36443^4)^{\frac{1}{8}} < 1.34004$。

现在，设 T 是一棵恰有五个孙节点的树。色彩森林中的顶点最多有三个邻居，因此每个子节点最多有两个孙节点。共有五个孙节点，因此两个子节点各有两个孙节点，另一个子节点有一个孙节点。我们选出有两个孙节点的那两个子节点，并遍历这些顶点的所有颜色分配。如果它们获得相同的颜色，那么根以及它们合计的四个孙节点都会有一个已着色的邻居。这意味着，只有剩下的另一个子节点和孙节点通过 (3,2)-Constraint Satisfaction Problem着色。如果它们获得不同的颜色，那么根的颜色必须是第三种颜色。此时，只有最后一个孙节点通过 (3,2)-CSP 算法着色。这样，每个顶点的运行时间为 $(3 \cdot 1.36443^2 + 6 \cdot 1.36443)^{\frac{1}{9}} < 1.338302 < 1.34004$。

因此，最坏情况是一棵有四个孙节点的树，由此可得引理中的结论：在最坏情况下，我们可以以工作因子 1.34004 为一棵树 T' 着色。 □

我们总结了 Beigel 和 Eppstein 提出的对我们算法至关重要的所有概念。我们从图中移除邻居少于三个的顶点，以及包含一个环或九个及以上顶点的三度顶点连通子图。之后，我们寻找一个极大繁茂森林和一个极大色森林，以找出拥有许多邻居的顶点集。我们在这两个森林中为特定顶点着色，从而可以将它们的邻居从实例中移除。剩余顶点将使用 (3,2)-CSP 算法着色。

3 限制难着色顶点

现在我们将讨论我们的主要改进：对繁茂森林进行修改，进一步限制繁茂森林之外可能存在的顶点数量。我们将限制集合 N 中高幅值顶点的存在。这个顶点会使繁茂森林之外存在许多顶点，而这是我们希望避免的。

定义 3.1（高幅度顶点）。 与极大茂密森林相邻，且在极大茂密森林之外有三个邻居的顶点，称为高幅度顶点。

定义 3.2（极大低幅度繁茂森林）。 如果一个极大茂密森林中，所有相邻的高幅度顶点都与一棵仅有一个内部顶点和四个叶子的树相邻，且与同一棵树相邻的任意两个高幅度顶点都共享一个公共邻居，那么该森林称为极大低幅度繁茂森林。这个邻居必须是这棵树的叶子，或是繁茂森林之外的一个顶点。

我们可以在多项式时间内将任意给定的极大茂密森林转换为极大低幅度繁茂森林。如果存在一个高幅度顶点，导致茂密森林不再是低幅度繁茂森林，我们可以修改与该高幅度顶点相邻的树。每次这样的修改都会增加内部顶点，或向茂密森林中添加一棵新树。此外，任何修改都不会减少茂密森林中的树的数量。茂密森林中的树或内部顶点的数量永远不会超过图中顶点的数量，因此我们最终会找到一个极大低幅度繁茂森林。如果某次修改导致茂密森林不再是极大的，我们就向茂密森林中添加一个新的内部顶点。这也会增加内部顶点的数量。

引理 3.1。 设 G 是一个图，其中所有顶点的度都至少为三，并且不存在由三度顶点构成的环，也不存在包含九个或更多三度顶点的连通子图。那么存在一个极大低幅度繁茂森林，且我们可以在多项式时间内找到它。

证明。 首先，我们在多项式时间内找到一个极大繁茂森林：贪心地向繁茂森林中添加新树，并向树中添加新的内部顶点，直到无法继续为止。现在，我们将修改这个极大繁茂森林，以找到一个极大低幅度繁茂森林。具体而言，我们将列举几种情形：存在一个高幅值顶点，它与一棵具有多个内部顶点或四个以上叶子的树相邻；或者存在一棵树，其中有一对高幅值顶点不共享公共邻居。

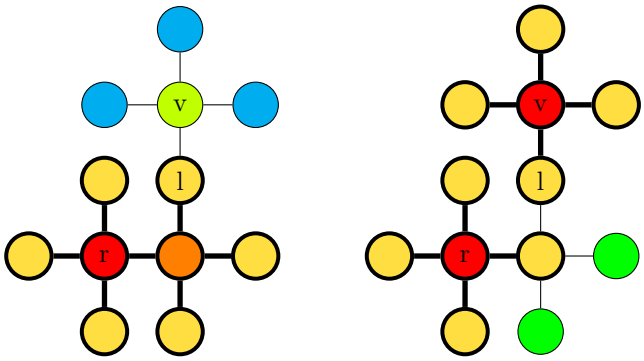

图4：一个包含繁茂森林（粗体）的图，展示了移除与大树相邻的高幅值顶点之前（左）和之后（右）的情况。红色：根顶点（R）。橙色：内部顶点（I）。黄色：叶子（L）。黄绿色：高幅值顶点。绿色：繁茂森林的其他邻居（N）。青色：其他顶点（U）。

考虑一棵由多个内部顶点组成的树，该树与一个高幅度顶点 v 相邻，如图4所示。设与 v 相邻的叶节点为 l，并设该树的根顶点 r 为任意一个在茂密森林中不与 l 相邻的内部顶点。然后，我们可以移除原来的树，并添加两棵树，分别以 r 和 v 为根。以 r 为根的树将包含原树中的四个顶点，而以 v 为根的树将包含茂密森林之外与其相邻的三个顶点，以及与其相邻的叶节点 l。

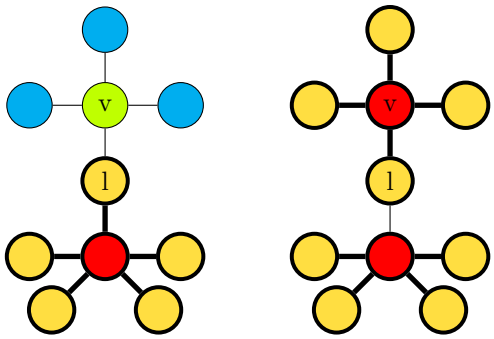

图5：一个带有茂密森林（以粗体标出）的图，展示了移除与叶子很多的树相邻的高幅值顶点前（左）和后（右）的情况。红色：根顶点（R）。黄色：叶子（L）。黄绿色：高幅值顶点。青色：其他顶点（U）。

其次，考虑一个与一棵树相邻的高幅度顶点 v，该树只有一个内部顶点且至少有五个叶子，如图5所示。设与 v 相邻的叶节点为 l。从现有的树中移除 l，剩下的树仍然是一棵有四个叶子的有效树。然后，添加一棵以 v 为根且有四个叶子的树：原本位于茂密森林之外的三个邻居，以及 l。

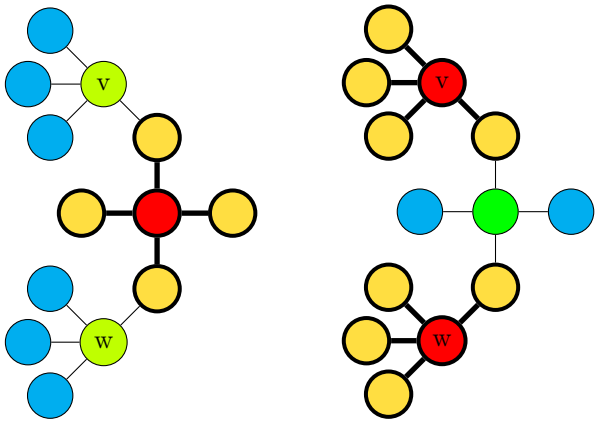

图6：一个带有茂密森林（以粗体标出）的图，展示了移除与同一棵树相邻的高幅值顶点前（左）和后（右）的情况。红色：根顶点（R）。黄色：叶子（L）。黄绿色：高幅值顶点。绿色：茂密森林的其他邻居（N）。青色：其他顶点（U）。

接下来，考虑一棵树：它与两个分别邻接于高幅值顶点 v 和 w 的不同叶子相邻，其中 v 和 w 既不相邻，也不在茂密森林之外共享邻居，如图6所示。此时，我们可以移除这棵树，转而以 v 和 w 为根创建新的树。这两棵新树都将有四个叶子：原本位于茂密森林之外的三个邻居，以及与它们相邻的叶子。由于 v 和 w 分别与不同的叶子相邻，且不在茂密森林之外共享邻居，因此二者都将形成有效树。

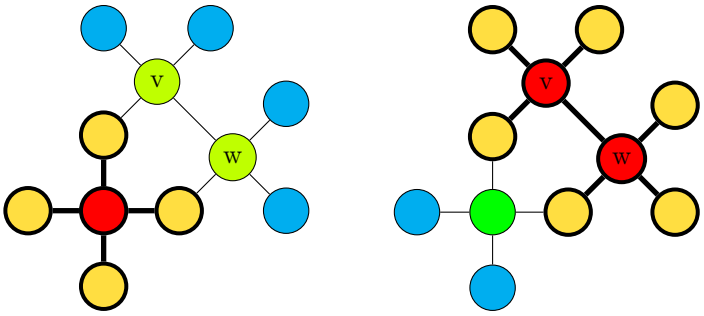

图7：一个带有茂密森林（粗体显示）的图，展示了移除与大树相邻的高幅度顶点前（左）和后（右）的情况。红色：根顶点（R）。黄色：叶子（L）。石灰色：高幅值顶点。绿色：茂密森林的其他邻居（N）。青色：其他顶点（U）。

最后，考虑一棵树，它有两个不同的叶子分别与高幅值顶点 v 和 w 相邻，其中 v 和 w 不在茂密森林之外共享公共邻居，但二者相邻。此时，我们可以移除这棵树（它只有一个内部顶点），转而创建一棵以 v 和 w 这两个高幅值顶点为内部顶点的新树，如图7所示。

因此，高幅度顶点只能与极大低幅度繁茂森林中的一棵树相邻，而该树恰好有一个内部顶点和四个叶子。如果有多个高幅度顶点与同一棵树相邻，那么它们要么与该树的同一个叶子相邻，要么在茂密森林之外共享一个公共邻居。

□

现在，任何剩余的高幅度顶点都必须与低幅度繁茂森林相邻，这限制了它们的出现方式。此外，这些高幅度顶点必须与任何其他和同一棵树相邻的高幅度顶点共享一个公共邻居。由于它们必须共享一个公共邻居，当一棵树与许多高幅度顶点相邻时，这会在图中形成可预测的结构。下一节将说明，这些结构还会限制茂密森林之外的顶点数量。

4 难着色顶点的界

高幅值顶点只存在于与特定树相邻的位置，如上一节所述。如果高幅值顶点与同一棵树相邻，则它们必须共享一个公共邻居。利用这些约束，我们可以列出一组方程，表示不同类型顶点之间的关系。我们将利用这些约束构建一个线性规划，以找出对于我们的算法而言的最坏情况图。

我们进一步对茂密森林之外的顶点进行划分，以便分析高幅值顶点的影响：

- N_1： N 中度为三的顶点，其中一个邻居位于 L中。
- N_2： N 中在 L中有多个邻居的顶点。
- $N_{3,\ i}$：高幅值顶点（ N 中在极大低幅度繁茂森林之外有三个邻居的顶点），它们与一棵树相邻，而总共有 i 个高幅值顶点与该树相邻。如果一个高幅值顶点与茂密森林中的多棵树相邻，则我们将其归入其相邻树中i 值最高的那一类。我们将这些顶点统称为 N_3。

其次 ，我们将把 U 中的顶点划分出来，以表示它们与 N的新划分之间的关系 ：

- U'： U 中的顶点，其三个邻居都必须位于 N_3中。
- U_j： U 中的顶点（不在 U'中），这些顶点出现在一个包含j N_1中顶点的三度顶点连通子图中。

U 中的所有顶点的度都必须为三，因此它们必须全部恰好出现在一个三度顶点子图中。此外，三度顶点的连通子图最多包含八个顶点。因此， U 中的所有顶点要么位于 U' 中，要么属于某个集合 U_j （对于 $0 \geq j < 8$）。

最后，我们注意到， N_2 和 N_3 中的顶点数量决定了茂密森林之外三度顶点的最大数量（包括N_1 以及 U中的所有顶点）。我们统计 N_2 或 N_3 中的顶点与茂密森林之外的三度顶点之间可能存在的边数，并将其分为两类：

1. U'中的每个顶点有三条边，用于将 N_3与 U'连接起来。

2. 这组边 S，连接 N_2 或 N_3 中的顶点与 N_1 中的顶点，或与满足 $0 \leq j \leq$的集合 U_j 中的顶点之间 7.

现在，我们将根据顶点划分之间的关系，说明各种约束。这些关系将根据另一个划分中的顶点数量，限制一个划分中可以存在的顶点数量：

引理 4.1. 设 G 是一个图，其中所有顶点的度至少为三，并且不存在由三度顶点构成的环，也不存在包含九个或更多三度顶点的连通子图。设 F 是 G中的一个极大低幅度繁茂森林。那么，

$$4 \cdot |R| + 2 \cdot |I| \leq |L| \tag{1}$$

$$|N_1| + 2 \cdot |N_2| + |N_3| \leq 2 \cdot |L| \tag{2}$$

$$|U'| \geq \frac{1}{5} \cdot |N_{3,5}| + \frac{2}{6} \cdot |N_{3,6}| + \frac{5}{7} \cdot |N_{3,7}| + |N_{3,8}| \tag{3}$$

$$\sum_{j=0}^{7} \left(\frac{10-j}{8-j} \cdot |U_j| \right) \leq 2 \cdot |N_2| + 3 \cdot \sum_{i=1}^{8} |N_{3,i}| - 3 \cdot |U'| \tag{4}$$

证明。 这些方程描述了顶点划分之间的关系。我们将逐一证明这些方程：

$$4 \cdot |R| + 2 \cdot |I| \leq |L|$$

这源于茂密森林的定义：如果一棵树只有一个内部顶点，那么它至少必须有四个叶子。每增加一个内部顶点，该树至少会增加两个叶子：我们将一个叶子变为内部顶点，这会移除一个叶子。然而，由于该叶子原本只与内部顶点相邻，因此现在这个内部顶点必须与三个叶子相邻。由此，每增加一个内部顶点，茂密森林中至少会出现两个叶子。

$$|N_1| + 2 \cdot |N_2| + |N_3| \leq 2 \cdot |L|$$

这源于定义 L、 N_1、 N_2和 N_3： L中的顶点在茂密森林之外至多有两个邻居。否则，茂密森林就不是极大的。 N_1 和 N_3 中的顶点在 L中至少有一个邻居，而 N_2 中的顶点在 L中必须至少有两个邻居。

$$|U'| \geq \frac{1}{5} \cdot |N_{3,5}| + \frac{2}{6} \cdot |N_{3,6}| + \frac{5}{7} \cdot |N_{3,7}| + |N_{3,8}|$$

回顾 U'的定义： U 中的顶点的所有邻居都必须是高幅值顶点（N_3）。当有五个或更多高幅值顶点与茂密森林中的一棵树相邻时，顶点会出现在 U' 中。所有高幅值顶点都必须共享一个公共邻居。此时，如果它们都必须共享一个公共邻居，那么 U 中的某些顶点必须与其中三个高幅值顶点相邻。对于茂密森林中分别与五个、六个、七个或八个高幅值顶点相邻的每棵树，我们将分析 U' 中必须存在多少个顶点。

考虑一棵树，其中有五个相邻的高幅值顶点（在 $N_{3,\ 5}$ 中的五个顶点）。至多有两对高幅值顶点通过树的叶节点拥有公共邻居。因此，至少有一个顶点通过叶节点与其余四个顶点中的任何一个都没有公共邻居。这个顶点必须与其余四个高幅值顶点都在茂密森林之外共享一个公共邻居。然而，它在茂密森林之外只能有

三个邻居。因此，它的邻居中至少有一个必须与其余四个高幅值顶点中的两个相邻：它在 N_3 中有三个邻居，且必须位于 U'中。对于 $N_{3,\,5}$ 中的任意五个顶点， U' 中必定至少存在一个顶点。

N_1 中的顶点和 N_2 中的顶点不可能在 N_3中拥有三个邻居，因此它们永远无法充当 U'中的顶点。 N_3 中与另一棵树相邻的顶点可能在 N_3中拥有三个邻居。然而，这比 U 中的顶点担任相同角色具有严格更优的运行时间：它会从实例中移除一个可能属于 U 的顶点。 U 中的顶点通过色彩森林进行着色，这比整个算法耗时更多。因此，我们可以假设始终是 U 中的顶点必须与 N_3中的三个顶点相邻。

利用这一技术，我们确定：每六个N_3中至少有两个 $U'_{,\,6}$；每七个N_3中至少有五个 $U'_{,\,7}$；每八个N_3中至少有八个 $U'_{,\,8}$。

$$\sum_{j=0}^{7}\left(\frac{10-j}{8-j}\cdot|U_j|\right)\leq 2\cdot|N_2|+3\cdot|N_3|-3\cdot|U'|$$

回顾 S： N_2 与 N_3之间以及 N_1 与 U 之间的边（但不包括 U'）。我们将计算 N_2 和 N_3中的每个顶点可能存在的 S 边数，以及 S 中每条边可能存在的 U 顶点数（但不包括 U'），以证明该等式。

根据定义， N_2 在茂密森林外部至多有两个邻居，而 N_3 恰好有三个邻居。不过，我们要减去所有连接到 U'中顶点的边。 U' 中的每个顶点在 N_3中都有三个邻居，因此，对于 U'中的每个顶点，我们要减去三条不可能从 N_3 连接到 U 中另一个顶点的边。于是得到$|S|\leq 2\cdot|N_2|+3\cdot|N_3|-3\cdot|U'|$。

接下来，我们计算 S中的每条边对应的、可能存在于 $U-U'$中的顶点数（位于 U中，但不在 U'中）。首先，考虑 $G[(U-U')\cup N_1$中的连通分量：这些顶点的度均为三，并且根据引理 2.2不能包含环。因此，它们构成一棵树，这意味着一个包含 n 个顶点的连通分量恰好包含 $n-1$ 条边。每条边在该连通分量内都有两个端点。这些顶点的度数之和为 $3\cdot n$，因此，一个端点位于该连通分量内、另一个端点位于其外部的边数必须为 $3\cdot n-2\cdot(n-1)=n+2$。顶点数与出边数之比为 $n/(n+2)$，当 n 最大时该比值最大： $n=8$。根据定义，所有与 $U-U'$ 中的顶点相连的出边，其另一个端点都必须位于 N_2 或 N_3 中。然而， N_1 中的每个顶点在 L中恰好有一个邻居。因此，对于连通分量中的 N_1 个顶点中的每个顶点，我们都要减去一个必定位于N_2 或 N_3中的连通分量邻居。此外，我们已经计算出 U_j 中可以存在的顶点数：还必须从中减去 N_1 中的顶点数。于是得到$\sum_{j=0}^{7}\left(\binom{10-j}{8-j}\cdot|U_j|\right)\leq|S|$。结合这两个结果，可得$\sum_{j=0}^{7}\left(\binom{10-j}{8-j}\cdot|U_j|\right)\leq 2\cdot|N_2|+3\cdot|N_3|-3\cdot|U'|$。 □

接下来，我们证明可以覆盖极大色森林中与 U' 中顶点相邻的所有高幅值顶点。由于对色彩森林中的顶点进行颜色染色，比仅使用 (3,2)-CSP 算法更快，因此这将缩短算法的运行时间。

定义 4.1（极大高幅值色森林）。 极大高幅值色森林是一个极大色森林，它覆盖 U 中的所有顶点，以及 U' 中顶点的所有相邻 N_3 顶点。

引理 4.2。 设 G 是一个图，其中所有顶点的度数均为三或更高，且不存在由三度顶点构成的环，也不存在包含九个或更多三度顶点的连通子图。设 G 中的 F 是一个极大低幅度繁茂森林。那么，我们可以在 $G[V-F]$ 中找到一个极大高幅值色森林。

证明。 Beigel 和 Eppstein 证明了我们可以在 $G[V-F]$ 中找到一个色彩森林。为了找到色彩森林，他们首先在 $G[V-F]$ 中找到了一个由 $K_{1,\,3}$ 棵树组成的极大森林：每棵树都有一个根顶点和三个子节点。在可能的情况下，他们通过移除一棵 $K_{1,\,3}$ 树并至少添加两棵，来修改这个由 $K_{1,\,3}$ 棵树组成的森林。借助这个由 $K_{1,\,3}$ 棵树组成的森林，他们证明了 U 中的每个顶点都可以分配给其中一棵 $K_{1,\,3}$树，并且每棵树分配到的孙节点不超过五个 [1, 引理 25]。

现在我们将证明，也可以将所有与U' 中的顶点相邻的高幅值顶点纳入色彩森林。这样一来，它就会从极大色彩森林变成极大高幅值色彩森林。

所有高幅值顶点或 U 中的顶点，在茂密森林之外都有三个邻居。因此，如果它们没有被纳入由 $K_{1,\ 3}$ 棵树组成的森林中，它们就至少与其中一棵、至多与三棵树相邻。它们不可能与任何树都不相邻，否则我们就可以以该顶点为根添加一棵新的 $K_{1,\ 3}$ 树。假设顶点 v 与 i $K_{1,\ 3}$ 棵树相邻，我们赋予顶点 v 权重 $1/i$。Beigel 和 Eppstein 证明，对于任意一棵树，只要所有潜在孙节点的权重之和不超过五，我们就能找到一个极大色彩森林。

考虑一棵任意的$K_{1,\ 3}$ 树，其中潜在孙节点的权重总和大于五。对于三个子节点中的每一个，至多有两个孙节点：总共至多六个。由于每个孙节点的权重至多为一，因此必定有六个潜在孙节点。此外，至少有五个孙节点的权重为一，因此它们各自只与一棵$K_{1,\ 3}$ 树相邻。

假设存在一个可能的孙节点 v ，它与一个并非T的可能孙节点的顶点相邻。根据 v的权重，我们总能将 T 替换为两棵$K_{1,\ 3}$ 树：

- v 的权重为一。我们移除 T ，并创建一棵以 v为根的新$K_{1,\ 3}$ 树。由于 v 至多与 T的另一个孙节点相邻，因此 T 必定有一个子顶点，其孙节点不与 v相邻。我们还创建一棵以该子顶点为根的新树。

- v 的权重大于一。设 v' 为 T 的孙节点，且它与v和 T 的同一个子节点相邻。我们移除 T ，并创建一棵以 v的父节点为根的新$K_{1,\ 3}$ 树。 v 至多与 T的另一个孙节点相邻，而 v' 至多与两个孙节点相邻。 T 有六个潜在孙节点，因此必定有一个不与 v 或 v'相邻：我们可以以该顶点为根创建一棵新的$K_{1,\ 3}$ 树。

设 v 的这个邻居不属于一个 $K_{1,\ 3}$ 树。这样，我们可以移除 T ，并创建两棵新树：一棵以 v为根，另一棵以不与 v相邻的 T 的子顶点为根。设 v 改为邻接两棵 $K_{1,\ 3}$ 树。在这种情况下，我们同样可以创建两棵新树。其他孙节点不能与另一棵 $K_{1,\ 3}$树相邻。因此，我们移除原来的树，并分别以 v的父节点和不与 v的父节点相邻的某个孙节点为根建立新树。顶点唯一可能的剩余配置如图 8 所示

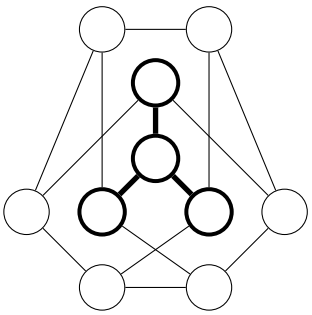

图 8：在极大高幅值色森林中，一棵（加粗的）$K_{1,\ 3}$ 树的孙节点权重大于五时，唯一可能的配置。

为了找到一棵潜在孙节点权重超过六的 $K_{1,\ 3}$ 树，六个潜在孙节点都不能有位于 $K_{1,\ 3}$ 树之外的邻居。所有孙节点都必须有三个邻居，其中一个是 T的子顶点。 T的每个孙节点都必须邻接另外两个 T的孙节点。六个孙节点之间要么存在两个长度为三的环，要么存在一个长度为六的环。如果存在两个环，我们就可以建立两棵新树，分别以其中一个孙节点为根。因此，六个孙节点之间必然存在一个长度为六的环。

如果所有潜在孙节点都是 U 中的顶点，就会形成一个由三度顶点构成的环。因此，在这种情况下，必须存在一个高幅度顶点作为可能的孙节点。我们只将与 U' 中顶点相邻的高幅度顶点视为潜在孙节点，因此也必须至少存在一个 U' 中的顶点。由于 U' 中的一个顶点与三个高幅度顶点相邻，而 $K_{1,3}$ 树中的顶点和潜在孙节点都不可能有该结构之外的邻居，因此 $K_{1,3}$ 树或其潜在孙节点中至少有三个高幅度顶点。

假设不再包含其他高幅度顶点或 U' 中的顶点。这样，我们就可以移除当前的 $K_{1,3}$ 树，并改为以 U' 中的顶点为根创建一棵新树，将三个高幅度顶点作为其子节点。这样，这棵树的潜在孙节点将少于六个，因为只有 U 中的顶点以及与 U' 中顶点相邻的高幅度顶点才是潜在孙节点。否则，所有孙节点都将是 U 中的顶点。在这种情况下， U 中的顶点会形成一个环，这是不可能的。

考虑这样一种情况：树及其潜在孙节点中，存在更多与 U' 中某个顶点相邻的高幅值顶点。那么， U' 中必定存在另一个顶点。与 U' 中该顶点相邻的高幅值顶点，不可能与茂密森林中的另一棵树相邻：在该树及其潜在孙节点中，不可能存在两棵互不相交的 $K_{1,\ 3}$ 树。否则，我们就可以用这两棵树替换这棵 $K_{1,\ 3}$ 树。因此，$K_{1,\ 3}$ 中的两个顶点必定共享一个公共邻居。

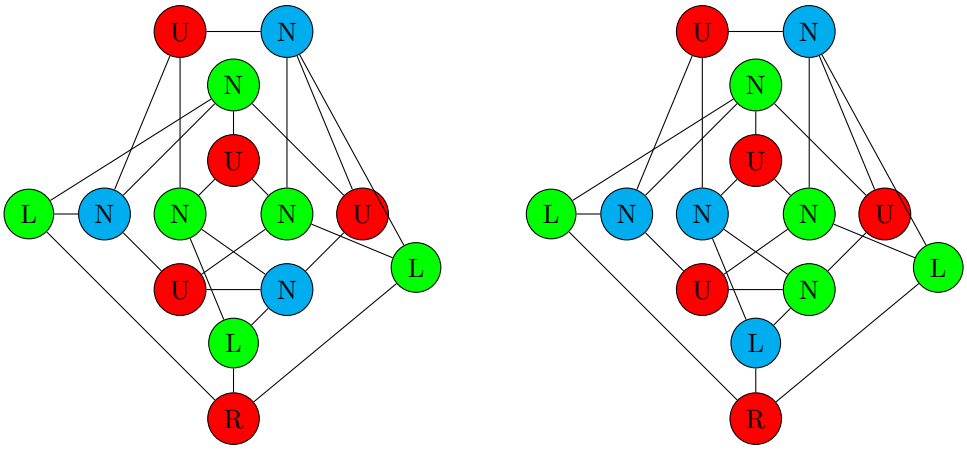

图 9：当极大高幅值色森林中的一棵树的权重大于五时，唯一可能配置所对应的一个有效 3-着色。

现在，$K_{1,\ 3}$ 树及其潜在孙节点中必定包含 U' 中的多个顶点。于是，它们相邻的茂密森林中的那棵树，必须至少与六个高幅值顶点相邻。唯一可能的配置如图 9 所示。

在这种配置中，茂密森林里的树中，必定有三对高幅值顶点与同一个叶节点相邻。接着，$K_{1,\ 3}$ 树及其六个孙节点，只通过这三个叶节点与图的其余部分相连。这三个叶节点共享同一个根顶点作为邻居，因此它们必定可 2-着色。无论它们如何着色，都总能为这组顶点配置着色。这三个叶节点要么颜色相同，要么其中两个颜色相同，第三个颜色不同。图 9 展示了这两种情况下的示例着色。

我们现在知道，如果$K_{1,\ 3}$ 树的孙节点权重大于五，那么我们可以替换这棵树并降低孙节点的权重，否则这棵树就很容易着色。因此，我们可以得出结论：能够找到一个极大高幅值色森林。对于 U'中的每个顶点，极大高幅值色森林都能让我们更轻松地为 N_3 中的顶点着色。具体来说，对于$N_{3,\ 5}$中的每五个顶点，U' 中必定有一个顶点与$N_{3,\ 5}$中的三个顶点相邻。对于$N_{3,\ j}$ 中的顶点，针对 $6 \leq j \leq 8$，它们都会与 U'中的一个顶点相邻。 □

我们发现了顶点划分之间的约束：根据它们的性质以及 L 中顶点的数量，这些约束限制了 N and$U$ 中顶点的数量。此外，一些高幅值顶点总是可以利用色彩森林着色。现在，我们创建一个线性规划，以确定我们算法的最坏情况图。

5 分析我们的算法

我们通过寻找极大低幅度繁茂森林和极大高幅值色森林，改进了 Beigel 和 Eppstein 的算法。我们枚举这些森林中所选顶点的所有可能颜色分配。

删除度数为 1 和度数为 2 的顶点。删除度数为 –3 的环和至少包含↪9 个度数为 –3 的顶点的连通子图。找一个极大的低–幅值树丛。找一个极大的高–幅值色彩树丛。对于树丛内部顶点以及↪色彩树丛中选定顶点的每种着色方案：按照该着色方案为所有顶点着色。对–(3, 2)-CSP 算法应用于剩余顶点。

现在我们将证明定理 1.1。我们已知第 4 节中关于与茂密森林存在不同关系的划分的约束。我们将这些约束与为图中的每个顶点着色所需的时间结合起来，构造一个线性规划。该线性规划将找出我们算法的最坏情况图，并确定其运行时间为 $\mathcal{O}^*(1.3217^n)$。

5.1 线性规划

我们找到了顶点划分之间的一组约束（约束 1、2、3 和 4）。我们利用这些约束创建一个线性规划，使算法的运行时间最大化。

首先，我们将讨论线性规划的最大化函数，以及它如何帮助我们找出算法的最坏情况运行时间。然后，我们将进一步讨论线性规划正常运行所需的其他约束。最后，我们将讨论该线性规划如何导出最终结果：在时间 $\mathcal{O}^*(1.3217^n)$ 内求解3-COLORING 。

5.2 创建线性规划

$$\max \quad \log(3) \cdot |R| + \log(2) \cdot |I| + \log(1.36443) \cdot |N^*| + \log(1.34004) \cdot |U^*| \tag{5}$$

$$|N| - \frac{3}{5} \cdot |N_{3,5}| - |N_{3,6}| - |N_{3,7}| - |N_{3,8}| = |N^*| \tag{6}$$

$$|U| + \frac{3}{5} \cdot |N_{3,5}| + |N_{3,6}| = |U^*| \tag{7}$$

回顾一下，我们遍历极大低幅度繁茂森林中内部顶点的所有 $3^{|R|} \cdot 2^{|I|}$ 颜色分配。这使我们能够在多项式时间内移除叶子（$|L|$），因此不会增加指数运行时间。对于剩余顶点，U 中的所有顶点都将由极大高幅值色森林覆盖，此外还包括每个与 U'中的某个顶点相邻的高幅值顶点。我们将集合 U^* 定义为所有将由极大高幅值色森林覆盖、从而在时间 $\mathcal{O}^*$内着色的顶点（$1.34004^{|U^*|}$：U中的所有顶点，以及 N_3 中与U'中的顶点相邻的所有顶点（引理 4.2）。剩余顶点将由 (3,2)-CSP 算法进行着色。令 N^*为既未被极大低幅度繁茂森林覆盖，也未被极大高幅值色森林覆盖、因而将直接使用 (3,2)-CSP 算法着色的顶点集。事实上，N^* 等于 N中的所有顶点，但不包括与 U 中的顶点相邻的高幅值顶点（引理 4.2）。

请注意，$3^{|R|} \cdot 2^{|I|} \cdot 1.36443^{|N^*|} \cdot 1.34004^{|U^*|}$根据图的划分计算 3-COLORING 算法的运行时间。不过，这并不是线性函数。幸运的是，对该函数取对数后会得到一个线性函数：$\log(3^{|R|} \cdot 2^{|I|} \cdot 1.36443^{|N^*|} \cdot 1.34004^{|U^*|})$。因此，该函数确定参数 $|R|$、$|I|$、$|N^*|$和 $|U^*|$，使运行时间达到最大。

$$|R| + |I| + |L| + |N| + |U| = n \tag{8}$$

$$\sum_{k=1}^{3} |N_k| = |N| \tag{9}$$

$$\sum_{i=1}^{8} |N_{3,i}| = |N_3| \tag{10}$$

$$\sum_{j=0}^{7} |U_j| + |U'| = |U| \tag{11}$$

$$\sum_{i=1}^{8} (\frac{8}{i} \cdot |N_{3,i}|) \leq 8 \cdot |R| \tag{12}$$

约束 1、2、3 和 4 在线性规划中充当约束。约束 8 确保顶点集的各个划分之和等于任意数量的顶点 n。我们还添加了其他约束（约束 9、10、11），以确保其他划分之和等于其父集。最后，我们添加约束 12：根据定义，$N_{3,\ i}$ 中的每个顶点都与极大繁茂森林中的一棵树相邻，而该树总共与 i 个高幅值顶点相邻。

5.3 线性规划的结果

表2展示了线性规划的结果。在这些结果中，顶点集以其占顶点总数的比例计算。请注意，大多数顶点集的值为零：它们不会出现在最坏情况中。 N （极大低幅度繁茂森林的邻居）中的顶点若确实存在，则位于N_3、6 或 N_2中。 N 中的顶点没有一个位于 N_1中，因此 U 中的所有顶点要么位于 U_0 中，要么位于 U'中。 U_0 中的顶点存在于 U_0中由八个三度顶点构成的连通子图中：这是能够存在的最大的三度顶点连通子图。

$\lvert\mathbf{R}\rvert$	0.0396825	$\lvert\mathbf{E}\rvert$	0.5555556	$\lvert\mathbf{N_{3,4}}\rvert$	0	$\lvert\mathbf{U_2}\rvert$	0
$\lvert\mathbf{I}\rvert$	0	$\lvert\mathbf{N_1}\rvert$	0	$\lvert\mathbf{北_{3,5}}\rvert$	0	$\lvert\mathbf{U_3}\rvert$	0
$\lvert\mathbf{L}\rvert$	0.1587302	$\lvert\mathbf{N_2}\rvert$	0.0396825	$\lvert\mathbf{N_{3,6}}\rvert$	0.2380952	$\lvert\mathbf{U_4}\rvert$	0
$\lvert\mathbf{N}\rvert$	0.2777778	$\lvert\mathbf{N_3}\rvert$	0.2380952	$\lvert\mathbf{N_{3,7}}\rvert$	0	$\lvert\mathbf{U_5}\rvert$	0
$\lvert\mathbf{U}\rvert$	0.5238095	$\lvert\mathbf{N_{3,1}}\rvert$	0	$\lvert\mathbf{N_{3,8}}\rvert$	0	$\lvert\mathbf{U_6}\rvert$	0
$\lvert\mathbf{N}^*\rvert$	0.0396825	$\lvert\mathbf{N_{3,2}}\rvert$	0	$\lvert\mathbf{U_0}\rvert$	0.4444444	$\lvert\mathbf{U_7}\rvert$	0
$\lvert\mathbf{U}^*\rvert$	0.7619048	$\lvert\mathbf{N_{3,3}}\rvert$	0	$\lvert\mathbf{U_1}\rvert$	0	$\lvert\mathbf{U'}\rvert$	0.0793651

表2：线性规划的结果。

最坏情况图由许多相同的树组成。每棵树周围都有六个高幅值顶点。树的其余邻居都是 N_2中的顶点。每棵树都会导致 U' 中出现两个顶点，而所有其他不与树相邻的顶点都属于 U_0。图10展示了其中一棵树。

Sp具体而言，对于 R中的每个顶点，其他集合中的顶点数量如下 ：

1. L：4
2. N_2：1
3. $N_{3,\ 6}$:

6 4. U_0:

11.2 5. U':

2

通过利用这些顶点集合的频率，我们确定算法的最坏情况运行时间为 $\mathcal{O}^*((3^1 \cdot 1.36443^1 \cdot 1.34004^{19.2})^{n/25.2})$ $= \mathcal{O}^*(1.3217^n)$。

6 结论

本文介绍了一种针对 3-COLORING 的改进算法。该算法执行以下步骤：

值得注意的是，我们引入了高幅度顶点这一概念。在 Beigel 和 Eppstein 的算法中，高幅度顶点的存在使得许多顶点可以位于极大繁茂森林之外。相反，我们构造了极大低幅度繁茂森林：一种高幅度顶点较少的极大繁茂森林。此外，我们还构造了极大高幅值色森林。借助这些森林，我们确定了一个顶点集，并遍历该集合所有可能的颜色分配。不属于该集合且与该集合中的顶点均不相邻的所有顶点，将由 (3,2)-CSP 算法解决。

我们对图中各顶点的不同性质进行了深入分析，这些性质由顶点与茂密森林之间的关系所决定。这项分析以线性规划的形式表述，使我们能够生成算法运行时间最大的图：时间 $\mathcal{O}^*(1.3217^n)$。这项分析帮助我们找到了以下结构，它是最坏情况图族中的一棵树：

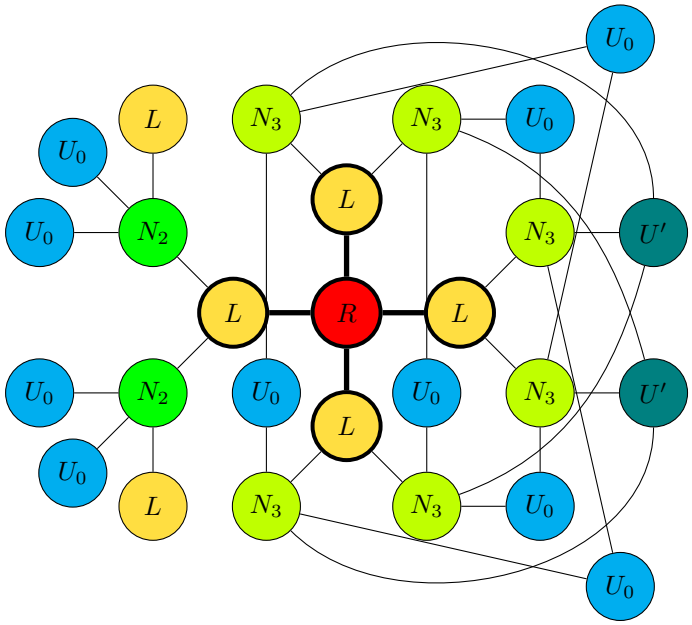

图10： 3-COLORING （单棵树）。红色：根顶点（R）。橙色：内部顶点（I）。黄色：叶子（L）。黄绿色：高幅值顶点。绿色：茂密森林的其他邻居（N）。蓝绿色：茂密森林之外、邻居均为高幅值顶点的顶点（U'）。青色：其他顶点（U）。

我们将 (3,2)-CONSTRAINT SATISFACTION PROBLEM 算法作为黑盒算法使用，因此，对 (3,2)-CONSTRAINT SATISFACTION PROBLEM 的任何改进都会自动改进我们用于 3-COLORING 的算法。我们的算法会通过茂密森林和色彩森林为许多顶点着色，因此， (3,2)-CONSTRAINT SATISFACTION PROBLEM 的输入将相对容易预测。通过研究 (3,2)-CONSTRAINTSATISFACTION PROBLEM 的输入中是否包含易于约简的结构，或许可以改进用于 3-COLORING 的算法。

目前尚不清楚，本文提出的概念能否用于改进 k-COLORING for $k \geq 4$ 的运行时间。特别是对于 4-COLORING，目前已知的 (4,2)-CONSTRAINTSATISFACTION PROBLEM 最佳算法与 (3,2)-CONSTRAINT SATISFACTION PROBLEM 完全相同。不过， (4,2)-CONSTRAINT SATISFACTION PROBLEM 增添了一个复杂的新情况：三种和四种可能颜色的变量都需要指数时间，但四种颜色的变量比三种颜色的变量更难着色。如果一个顶点有一个已着色的邻居，它仍有三种可能的颜色。因此，除非一个顶点有两个已着色且颜色不同的邻居，否则我们无法在多项式时间内对其进行约化。因此，

使用部分着色图和 (4,2)-Constraint SatisfactionProblem 来求解 4-coloring 要复杂得多。

7 致谢

我们感谢 Carla Groenland 和 Jesper Nederlof 对本文早期草稿提出的详尽意见，以及就图着色展开的有益讨论。我们感谢 Till Miltzow 和 Ivan Bliznets 为本文的写作风格提供建议。

参考文献

[1] Richard Beigel 和 David Eppstein。《3-着色的时间复杂度》。载于：《算法杂志》54.2（2005年2月），第168–204页。 DOI: 10.1016/j.jalgor.2004.06.008。 URL: https://doi.org/10.1016/j.jalgor.2004.06.008。[2] Andreas Bj¨orklund、Thore Husfeldt 和 Mikko Koivisto。《通过容斥原理进行集合划分》。载于：《SIAM计算杂志》39.2（2009年1月），第546–563页。 DOI: 10 . 1137 / 070683933。 URL: https://doi.org/10.1137/070683933。[3] Fedor V. Fomin 等人。《分支与树宽结合的两种技术》。载于：《算法》54.2（2007年12月），第181–207页。 DOI: 10.1007/s00453-007-9133-3。 URL: https://doi.org/10.1007/ s00453-007-9133-3。[4] David S. Johnson、Mihalis Yannakakis 和 Christos H. Papadimitriou。《生成所有极大独立集》。载于：《信息处理快报》27.3（1988年3月），第119–123页。 DOI: 10.1016/ 0020-0190(88)90065-8。 URL: https://doi.org/10.1016/0020-0190(88)90065-8。[5] Richard M. Karp。《组合问题之间的可归约性》。载于：《计算复杂性》。Springer US，1972年，第85–103页。 DOI: 10 . 1007 / 978 - 1 - 4684 - 2001 - 2 _ 9。 URL: https : // doi.org/10.1007/978-1-4684-2001-2_9。[6] E.L. Lawler。《关于色数问题复杂度的注记》。载于：《信息处理快报》5.3（1976年8月），第66–67页。 DOI: 10.1016/0020-0190(76)90065-x。 URL: https://doi.org/ 10.1016/0020-0190(76)90065-x。[7] L. Lov´asz。《超图的覆盖与着色》。载于：第四届东南组合学、图论与计算会议论文集（1973年），第3–12页。 URL: https://cir.nii.ac.jp/crid/1572261549354589440。[8] J. W. Moon 和 L. Moser。《图中的团》。载于：《以色列数学杂志》3.1（1965年3月），第23–28页。 DOI: 10.1007/bf02760024。 URL: https://doi.org/10.1007/bf02760024。[9] Ingo Schiermeyer。《以少于 O(1.415n) 步判定 3-可着色性》。载于：《计算机科学中的图论概念》。Springer Berlin Heidelberg，1994年，第177–188页。 DOI: 10 . 1007 /3 - 540 - 57899-4_51。 URL: https://doi.org/10.1007/3-540-57899-4_51。[10] Or Zamir。打破 5-着色和 6-着色的 2^n 障碍。2020年。 DOI: 10.48550/ARXIV.2007. 10790。 URL: https://arxiv.org/abs/2007.10790。