基于 Beck 方法的超图着色局部计算算法

AndrzejDorobisz ！ 理论计算机科学系，数学与计算机科学学院，雅盖隆大学，波兰克拉科夫

Jakub Kozik
雅盖隆大学数学与计算机科学学院理论计算机科学系，波兰克拉科夫

arXiv:2305.02831v1 [cs.DS] 4 May 2023

摘要

我们研究均匀超图的二着色局部计算算法（LCA）。我们关注满足 Lovász局部引理强化假设的超图实例，其形式为 $2^{1-\alpha k}(\Delta+1)e < 1$，其中 Δ 是最大边度的上界。由此产生的主要问题是，对于多大的 α，存在一种 LCA，能够在每次查询的多对数时间内对这类超图进行正确着色。我们简要介绍如何通过 Moser 和 Tardos 的重采样算法升级 Beck 于 1991 年提出的经典顺序过程，从而得到一种适用于 α 最大达到 $1/4$的多对数时间 LCA。随后，我们提出一种改进的过程，通过允许 α 最大达到 $1/3$，从而解决更广泛范围的实例。

2012年 ACM 学科分类计算数学 → 超图；计算数学 → 概率算法；计算理论 → 流式、次线性和近线性时间算法

关键词和短语B性质、超图着色、局部计算算法

资助本研究部分获得波兰国家科学中心（2016/21/B/ST6/02165）的资助

1 引言

超图着色问题常被用作各种概率技术的基准。任务是判断是否存在（或明确找出）合法着色，即为超图的顶点分配颜色，使得没有任何一条边包含的顶点全部同色。事实上，在线性超图上进行二着色[1] 是 Erdős 和 Lovász 在其开创性论文 [8] 中提出局部引理的主要动机之一。众所周知，判断给定超图是否存在合法二着色是 NP完全问题 [14]。即使对于所有边的大小均为 3 的超图，这一结论也成立。本文将在局部计算算法的框架下，讨论均匀超图二着色的次线性算法。

我们将研究 k均匀超图[2]。在本文余下部分中，n 表示所考虑均匀超图的顶点数，m 表示其边数，k 表示边的大小。我们假设 k 是固定的（但足够大，以避免技术细节），且 n 趋于无穷大。对于一个固定的超图，我们用 Δ 表示其最大边度。在我们将研究的实例中，Δ 由 k的一个函数所界定，因此就 n 而言，它是$\mathcal{O}(1)$。这意味着边数 m 至多与 n呈线性关系。我们还假设所考虑的超图没有孤立顶点。因此，我们还有 $m = \Theta(n)$。

[1] 在二染色问题中，我们可以为每个顶点分配两种可用颜色中的一种。[2] 在 k-均匀超图中，每条边恰好包含 k 个顶点。

1.1 局部计算算法

Rubinfeld、Tamir、Vardi 和 Xie 在 [20] 提出了一种称为局部计算算法（LCA）的一般性次线性顺序算法模型。该模型旨在描述这样一种情形：需要在一个大型实例上执行某些计算，但在任何特定时刻，只需要获得答案的一部分。与局部计算算法的交互通过一系列关于全局解片段的查询来组织。算法应在次线性时间内（相对于实例规模）回答每个连续查询，并以系统的方式逐步生成与某个全局解一致的部分答案。该模型允许使用随机性，算法也可能偶尔失败。

例如，对于超图二着色问题，LCA 过程的目标是为给定的超图找到一种合法着色。算法可以接受关于任意顶点的查询，并为被查询的顶点分配两种可用颜色之一。对于任意查询序列，以高概率，返回的部分着色都应能够扩展为合法着色。

形式化地，对于一个固定问题，如果某个过程对于任意大小为 n 的实例和任意查询序列，都能在时间 $t(n)$ 内一致地回答每个查询，并且计算所用的空间至多为 $s(n)$，那么它就是一个 $(t,\ s,\ \delta)$-局部计算算法。时间 $t(n)$ 必须关于 n 为次线性，但最好对其具有多对数依赖。$\delta(n)$ 应当界定整个查询序列发生失败的概率，通常要求该值较小。计算内存、输入和随机比特源都表示为可随机访问的磁带（后两者不计入 $s(n)$ 限制）。计算内存可以在各次查询之间保留；具体而言，它可以存储先前调用中确定的部分答案。有关该模型的精确定义，请参见 [20]。

如果返回的解不依赖查询顺序（即只取决于输入和随机比特），则称该过程为 查询无关 。这通常意味着算法只使用计算内存来回答当前查询，无需在不同查询之间保留信息。这是一种理想的性质，因为它允许并行执行算法查询。在一篇后续论文中， [3], Alon、Rubinfeld、Vardi 和 Xie 提出了通用方法，用于消除对查询顺序的依赖，并减少 LCA 过程所需的随机比特数。在同一篇论文中，他们将这些技术应用于 [20] 中的示例过程（包括超图着色），将其转化为查询无关的 LCA。这些改进后的过程不仅能在多对数时间内运行，也能在多对数空间内运行。Mansour、Rubinstein、Vardi 和 Xie 在 [15] 中改进了对这一方法的分析。

1.2 构造性局部引理与 LCA

Lovász局部引理（局部引理）是局部算法领域最重要的工具之一。在其基本形式中，它可以非构造性地证明某些组合对象的存在，这些对象不具有一组不理想的性质，即所谓的坏事件。Faragó [10] 最近发表的综述简要介绍了这一主题，并总结了局部引理的各种版本。

对于一个固定的 k-均匀超图，令 $p = 2^{-k}$ 表示在均匀随机着色中，某条固定的边为特定颜色的单色边的概率。对称版本的局部引理（例如参见 [10]）的直接应用可以证明，对于最大边度为 Δ 的超图，条件$2p\ (\Delta + 1)\ \mathrm{e}\ < 1,$ 足以保证其可二着色。

多年来，人们一直试图将局部引理转化为高效的算法，但始终未能成功。

第一次突破出现在 1991 年，当时 Beck [5],以超图二着色为例，展示了一种将局部引理的一些存在性证明转化为多项式时间算法过程的方法。然而，要实现这一点，必须加强局部引理的假设，并将其表述为

$$2\,p^{\alpha}\,(\Delta+1)\,\mathrm{e}<1. \tag{1}$$

对于 $\alpha=1$，该不等式简化为标准假设。上述不等式对Δ施加了约束，并且随着 α 变小，约束会更加严格。Beck 的原始证明适用于 $\alpha<1/48$。从那时起，人们投入了大量精力研究其在具体问题中的应用，并推动 α 向前发展，使其尽可能接近标准的局部引理判据 [2, 16, 7, 21, 18]。

下一项突破由 Moser 于 2009 年取得。随后，Moser 与 Tardos 合作，将 Moser 的思想重新表述为 [19] 中该引理的一般构造性形式。他们证明，假设满足 LLL 的所谓变量设定，一种称为 Resample[3] 的自然随机化过程能够快速找到相关随机变量的一组取值，使得所有坏事件均不发生。他们还证明，在典型情况下，该过程的期望运行时间与实例规模呈线性关系。对于 k-均匀超图的二着色问题，重新采样的总期望次数有上界 m/Δ （参见 [10] 中的定理 7 ）。

将构造性局部引理调整为 LCA 模型，仍是该领域最具挑战性的问题之一。不过事实证明，先前关于局部引理算法化的结果可以自然地加以改造。事实上，针对 [20], 中超图着色的首个 LCA 算法，建立在 Beck 算法的一个变体之上；该变体见于 Alon 和 Spencer 合著的书籍 [4]。该版本适用于 $\alpha<1/11$，且每次查询的运行时间为多对数时间。之后的改进主要致力于优化空间和时间需求（[3], [15]），但对于多对数时间的 LCA， α 的上界一直未能得到改善。在一项近期研究中，Achlioptas、Gouleakis 和 Iliopoulos [1] 展示了如何将重新采样调整为 LCA 模型。不过，他们未能实现多对数时间。他们的版本以 $t(n)=n^{\beta(\alpha)}$ 的时间回答查询。他们证明， α 的上界与回答查询所需的时间之间存在某种权衡。特别是，当 α 趋近于 $1/2$ 时， $\beta(\alpha)$ 趋近于 1，因此每次查询的运行时间上界非常宽松。

1.3 主要结果

我们的研究聚焦于 Lovász局部引理的局部构造版本领域中的以下一般性问题：对于满足条件 $2(\Delta+1)\mathrm{e}<2^{\alpha k}$ 的 k-均匀超图的二着色问题， α 的取值最多能达到多少，才能存在一个多对数时间的 LCA。我们证明了以下定理：

▶ **定理 1**（主要结果）。 对于每个 $\alpha<1/3$ 以及所有足够大的 k，存在一种局部计算算法，能够以每次查询多对数时间的复杂度，以概率 $1-O(1/n)$ 解决最大边度为 Δ的 k-均匀超图的二着色问题，且满足 $2\mathrm{e}(\Delta+1)<2^{\alpha k}$.

在 [20] 的记号下，我们提出了一种 (polylog(n),$\mathcal{O}(n)$, $\mathcal{O}(1/n)$)-LOCAL 计算算法，能够为满足上述假设的超图正确着色。我们的算法是

[3] 只要有一些坏事件被违反，该过程就会选取任意一个此类事件，并对该事件所依赖的所有变量进行重新采样。

而非查询无关。此外，消除对查询顶点顺序的依赖的典型方法似乎无法在不牺牲常数 α 的情况下应用。我们的主要结果更具技术性和精确性的表述见附录 C 中的定理 9。

作为比较，Alon 等人 [3] 在 Rubinfeld 等人 [20] 之后提出了一种查询无关的（polylog(n)，polylog(n)，$\mathcal{O}(1/n)$)-局部计算算法，适用于满足条件的超图 时时彩

$$\begin{aligned} 16\,\Delta(\Delta-1)^3(\Delta+1) &< 2^{k_1},\\ 16\,\Delta(\Delta-1)^3(\Delta+1) &< 2^{k_2},\\ 2e(\Delta+1) &< 2^{k_3}, \end{aligned} \tag{2}$$

其中 k_1，k_2 和 k_3 是正整数，满足 $k = k_1 + k_2 + k_3$。这些假设对应于 $\alpha < 1/11$。

[3] 对 LCA 过程的分析仅保证运行时间为$\mathcal{O}\left(\log^{\Delta}(n)\right)$量级。Mansour 等人在 [15] 中致力于在多对数级范围内改进时间和空间界，消除指数中对最大边度的依赖。他们得到了一个 LCA，其时间和空间复杂度为$\mathcal{O}\left(\log^4(n)\right)$，前提是$k \geq 16\log(\Delta) + 19$，因此它要求对 α有更强的界。

1.4 LOCAL 分布式算法

局部计算算法模型与 Linial 提出的经典局部分布式计算模型（称为 LOCAL）有关。关于这两种模型的比较，参见 Even、Medina 和 Ron [9]的研究。Chang 和 Pettie 最近在 [6] 中指出，在 LOCAL 模型中，求解依赖图具有有界度的局部引理实例这一一般问题，在某种意义上对于一大类问题是完备的（这些问题均可在次对数轮数内求解）。他们还猜想，对于经过充分强化的局部引理条件（例如，在式（1）中取足够小的 α），存在一种分布式 LOCAL 算法，能够在$\mathcal{O}(\log\log\, n)$轮内解决该问题。在 LCA 框架内对这种算法进行直接模拟，将得到一个过程：至少对于固定的最大度，该过程能够在多对数时间内回答查询。

最近，Fischer 和 Ghaffari [11], 在这一猜想方面取得了进展，他们证明存在一种用于局部引理实例的算法，能够在 $2^{\mathcal{O}\left(\sqrt{\log_{\log n}}\right)}$轮内运行。此后，Ghaffari、Harris 和 Kuhn 在 [12] 中进一步改进了底层依赖图的度对运行时间的影响。特别是，对于约束足够严格的超图二着色问题，该结果允许构造一种 LCA 过程，使其能够在次线性时间内回答查询。然而，运行时间仍会是超多对数级的。此外，所需的局部引理假设强化程度，似乎远高于应用 Rubinfeld 等人结果所需的程度 [20]。

在 LCA 模型中模拟 LOCAL 算法的可能性意味着，如果 Chang 和 Pettie 的猜想成立，那么任何满足充分强化的 LLL 条件的问题，都可以在 LCA 模型中以每次查询多对数时间的复杂度求解。因此，我们可以提出一个较弱的猜想：对于某个 α，每个这样的 α-强化问题都可以在 LCA 中以每次查询多对数时间的复杂度求解。对于超图着色这一具体问题，已知该性质成立。然而，我们可以进一步询问：对于一个固定的问题，这样的 α 的最大值是多少。这正是第 1.3 节开头提出的一般问题。值得注意的是，我们的算法必不可少地利用了 LCA 的顺序性

。正因为如此，它们无法转换为 $\mathcal{O}(\log \log n)$ LOCAL 算法。这也说明了这些模型之间的一个重要差异。

2 证明的主要技术与思想

Beck [5] 的算法过程分为两个阶段。在第一个阶段，即我们称之为碎裂阶段的阶段，它构建一个随机的部分着色，以确保所有边中有一定比例已经被正确着色。此外，对于尚未处理的边，它们包含足够多的未着色顶点，以确保部分着色能够完成为一个正确的着色。这些边还形成了规模为对数级的连通分量，可以独立地进行着色。随后，在第二个阶段，即我们称之为最终着色阶段的阶段，使用穷举搜索来完成每个分量的着色。这将得到一个具有多项式运行时间的顺序过程。为了将运行时间降低到近乎线性，可以将碎裂阶段应用两次。这样，最终连通分量 w.h.p. 的大小为 $\mathcal{O}(\log \log(n))$。[20] 中的超图着色多对数级 LCA 过程遵循了这一方法，在回答单个查询时，在局部模拟两个碎裂阶段和一次穷举搜索。这三个阶段的划分直接体现在该过程所要求的条件 (2) 中。

虽然尚不清楚能否仅基于重新采样设计出 LCA 算法，但将其与先前的局部算法结合，能带来显著改进。事实证明，在多对数时间内，仅经过一个碎裂阶段，就可以借助重新采样完成着色。只需进行这一简单修改并稍微改进分析，就足以为 $\alpha \leq 1/4$ 推导出定理 1。这是我们的首项贡献。该过程为阐释我们后续改进背后的直觉和动机提供了参照。具体而言，我们定义了分量超图 这一概念，以便对残余超图的各个分量进行更细致的分析。因此，我们在第 3 节详细介绍基础算法。

为改进基础算法，我们做的第一项修改是在碎裂阶段为所有顶点采样颜色。随后，部分顶点的颜色就此确定，而其他顶点则可以在最终着色阶段更改其已分配的颜色。在第一阶段为所有顶点着色，在某种程度上模糊了碎裂阶段与最终着色阶段之间的界限。这样做的主要目的是能够将残余超图更精细地划分为互不相交的片段。此外，这也有助于确定残余超图中无需重新着色的某些分量。这对应于重新采样中首次采样的颜色恰好构成合法着色的情形。总的来说，我们成功显著降低了在最终着色阶段进行着色的互不相交片段的保守估计规模，从而进一步放宽了关于 α 的必要条件，使其适用于 $\alpha < 1/3$。改进后的过程见第 4 节。

为了分析这些过程，我们采用一种常见技巧：为需要重新着色的分量关联一些类似树的见证结构。每个此类结构都描述一组与超图中某些边相关的事件。所有这些事件都由碎裂阶段分配的颜色决定。对于基础算法而言，这些结构相当典型。然而，为了在 α 上获得更好的界，我们设计了更复杂的结构，能够追踪不同类型的事件；这些事件还可能取决于允许重新着色的颜色。不同类型的事件具有不同的概率界。分析中的一个重要方面，是在单个结构内对不同类型的事件进行摊还。这些结构的构造是我们主要的技术贡献。我们在附录 D 中详细描述了这一构造。

最后需要指出，尽管我们的方法还不够通用，无法适用于所有满足局部引理强化假设的实例，但它们可以应用于许多与超图着色类似的问题，例如 k-SAT。

3 建立基本结果

本节将说明如何把 Beck 算法与重新采样相结合，构造一种局部计算算法，使其每次查询的运行时间为多对数时间，适用于 α至 $1/4$。换言之，我们将在更强的假设 $\alpha \le 1/4$ 下证明定理 1。为使论述简明，我们先介绍一种全局随机化算法，然后说明如何将这一过程调整为 LCA 模型。该过程的分析见附录 B。

设 $H = (V, E)$ 是一个满足定理 1 假设的超图，其中$\alpha \le 1/4$固定。为方便技术处理，我们假设 αk 是一个整数[4]。赋予随机颜色是指从两种可用颜色中均匀随机地选取一种。对于边集 S，$V(S)$ 表示由 S 中的边覆盖的所有顶点。对于一条边 f， $N(f)$ 表示与 f 相交的边集。我们采用一种与该领域其他研究类似的命名约定；具体而言，我们对 Beck 算法的理解受到 Alon 和 Spencer [4] 、Molloy 和 Reed [17]，的描述，以及 [20] 中给出的局部计算算法实现的影响。

3.1 全局着色过程

算法首先为顶点任意选定一个顺序。随后分两个阶段进行：碎裂阶段 和最终着色阶段。在碎裂阶段，为输入超图中的部分顶点着色，然后将尚未正确着色的超图的边拆分为最终连通分量 ——这些子超图可以独立着色。最终着色阶段则逐一处理各个最终分量，完成着色。

3.1.1 碎裂阶段

该过程按照固定顺序依次处理顶点。对于每个顶点，要么为其指定一个随机颜色，要么在它属于坏 边的情况下不为其着色。若一条边包含 $(1-\alpha)k$ 个已着色顶点，且仍未正确着色（也就是说，这些顶点的颜色都相同），则称该边为坏 边。一条边一旦变坏，就不再为该边中的其他顶点着色——这些顶点称为受困顶点。已指定颜色的顶点称为已接受顶点。

碎裂阶段完成后，边分为三种：安全边 – 由已接受顶点正确着色，坏边 – 恰好包含 ($1-\alpha)k$ 个已接受顶点，且颜色相同，不安全边 – 包含少于 $(1-\alpha)k$ 个已接受顶点，且颜色相同

请注意，在所得的（部分）着色中，每条未正确着色的边都至少有 αk 个问题顶点，它们将在下一阶段着色。还要注意，某些不安全边可能一个已着色顶点都没有。

已接受顶点的颜色不会改变，因此安全边已经处理妥当。于是，我们关注坏边和不安全边。令 E_{bad} 表示所有

[4] 事实上，对于给定的 k，唯一合理的做法是将 α 取为 t/k 的形式，其中 t 是一个整数 $2 \le t \le k$。

坏边的集合。考虑超图（$V(E_{bad})$ E_{bad}）。它自然分解为连通分量。

▶ **定义 2**。 超图 $(V(E_{bad})$ $E_{bad})$ 的每个分量都称为一个坏分量。

注意，每个受扰顶点都属于某个坏分量。在此基础上，我们构建一种抽象结构，用于表示坏分量之间通过不安全边形成的依赖关系。

▶**定义 3.** 按如下方式构造一个分量超图：其顶点是 H 的坏分量，并且对于每条与多个坏分量相交的不安全边 f，向其中添加一条包含 f 所相交的所有坏分量的边。

对于分量超图中的每个连通分量（即分量超图中相互连通的极大坏分量集合），我们通过取这些坏分量的并集来构造一个最终分量 （因此，最终分量是 H的一个子超图）。如果每个最终分量包含至多 $2(\Delta+1)\log(m)$ 条坏边，则碎裂阶段成功 。如果不满足这一条件，该过程将宣告失败。事实证明，这种情况发生的可能性极低。

3.1.2 最终着色阶段

对于碎裂阶段确定的每个最终分量 $\mathcal{C}$，我们将所有与其相交的不安全边添加到 $\mathcal{C}$ 中，然后将 $\mathcal{C}$ 限制为问题顶点[5]。由此得到一个超图$\mathcal{C}'$，其中至多包含 $2(\Delta+1)^2\log(m)$ 条边，且每条边至少包含 αk 个顶点。$\mathcal{C}'$ 中的最大边度不会超过 Δ，即 H中的最大边度。由于 $2e(\Delta+1) < 2^{\alpha k}$ （根据定理 1 的假设），Lovász局部引理保证$\mathcal{C}'$ 是二着色的。因此，根据 Moser 和 Tardos 的定理，重新采样算法平均只需进行 $|E(\mathcal{C}')|/\Delta$ 次重新采样，即可找到其合法着色（参见 [10]中的定理 7）。

最终着色阶段结束后，所有最终连通分量都已被正确着色。由于每条坏边或不安全边都在某个最终连通分量中得到处理，而每条安全边在碎裂阶段已经被正确着色，因此现在可以保证，所构造的着色对于整个 H 都是正确的。

3.2 局部计算算法实现

我们采用相当标准的技术来实现所述算法的 LCA 实现。下面对其进行阐述，以便为介绍我们的主要算法提供背景。所述过程的一个重要性质是，无需预先固定顶点的顺序。事实上，甚至可以由对手在线选择该顺序。按照 [20], 我们将利用顺序选择的自由。该算法的 LCA 版本将模拟在特定顺序下运行的全局版本。该顺序在计算过程中动态构造，并由查询驱动。当算法被查询顶点 v时，除了一些因适应 LCA 模型而产生的细微调整外，它会执行标准算法为 v分配最终颜色所需的全部工作。LCA 版本见列表 1、2、3 和 4。算法运行期间分配的所有颜色都存储在计算内存中（该内存在查询之间保留）。为方便起见，我们还在那里存储每个顶点的状态——未着色、已接受 或 受困。初始时所有顶点均未着色。

[5] $H=(V,E)$ 的限制定义为 $H'=(V'\{e\cap V'|\ e\in E, e\cap V'\neq\emptyset\})$。

算法 1 一致超图着色的 LCA——主函数

```
过程 QUERY(v - 顶点):
    如果 v 是 uncolored 则
        如果 包含 v 的所有边都不是坏边则
            为 v 分配一个随机颜色并将其标记为 accepted        // shattering
        否则 将 v 标记为 troubled
    如果 v 是 troubled 则
        C_v ← BUILD_FINAL_COMPONENT(v)                      // shattering
        COLOR_FINAL_COMPONENT(C_v)                          // final coloring
    返回 分配给 v 的颜色
```

3.2.1 查询

当顶点 v 已经被标记为已接受时，会立即返回其颜色。如果此前尚未处理过该顶点，算法会检查 v 是否属于某条坏边（这需要检查包含 v的所有边的当前状态）。如果不属于，则为 v分配一个随机颜色，将该顶点标记为已接受，并返回所分配的颜色。另一方面，当 v 属于某条坏边时，会将其标记为受困。随后，算法在过程 BUILD_FINAL_COMPONENT中确定包含 v 的最终分量。这些步骤可视为碎裂阶段。之后，在过程 COLOR_FINAL_COMPONENT 中对该最终分量执行最终着色阶段。

算法 2 为属于某条坏边的 v 构建最终分量

```
过程 BUILD_FINAL_COMPONENT(v - 受困顶点):
    B ← ∅                  // initialize set of bad edges of the component
    U ← ∅                  // initialize set of unsafe edges to process
    e ← 包含 v的任意坏边
    将 e 标记为已探索 并 运行 EXPAND_BAD_COMPONENT(e, B, U)
    // process surrounding unsafe edges
    当 U 不为空 时，执行
        f ← 来自 U 的下一条边（将其从 U中移除）
        EXPAND_VIA_UNSAFE(f, B, U)
    // return hypergraph built on set of bad edges
    返回 C = (V(B) B)
```

3.2.2 构建_最终_分量

该过程通过探索 H[6] 的线图，构建 v 的最终分量中的坏边集合 B。它使用临时标记 已探索 来标记已访问的边（此标记不会在查询之间保留）。构造从一条包含受扰顶点 v 的坏边开始，并将其扩展为一个坏分量。然后，只要还有可能，就通过与 B 相邻的不安全边访问相邻坏分量中的边，并将这些边加入集合 B。如果

[6] 线图 $L(H)$ 是建立在 $E(H)$ 上的图，其中两个不同的顶点（代表 H 的边）相邻，当且仅当相应的边相交。

某个时刻 B 中的坏边数量超过规定上限 $2(\Delta+1)\log(m)$，该过程便宣告失败（请注意，由于 LCA 模型不允许更改先前查询返回的颜色，因此无法重新启动）。当没有更多坏边可添加时，最终分量的构造即告完成。随后，返回基于收集到的坏边构建的超图 $\mathcal{C} = (V(B), B)$。

坏分量的扩展在子程序 `EXPAND_BAD_COMPONENT` 中完成。它从给定的坏边开始，通过检查相邻的边来探索线图。对于每条相邻边，使用 `DETERMINE_EDGE_STATUS` 确定其类型（安全、不安全或坏）。确定一条边的状态可能需要处理该边上一些未着色的顶点。对于其中每个顶点，该过程都会检查它是否受困。如果没有，则为该顶点分配一个随机颜色，并将其标记为已接受。

算法 3 最终分量构造的子程序

```
过程 EXPAND_BAD_COMPONENT(e - 坏边, B - 坏边, U - 不安全边):
    Q ← {e}                                  // initialize set of bad edges to process
    当 Q 非空时执行
        f ← 从 Q 中取下一条边（将其从 Q中移除）
        将 f 添加到 B，如果 |B| > 2(Δ + 1) 记录(m)，则失败
        对于 g ∈ N(f) 中尚未探索的项，执行
            将 g 标记为已探索并 DETERMINE_EDGE_STATUS(g)
            如果 g 为 bad，则将 g 添加到 Q
            如果 g 是 unsafe 则 将 g 添加到 U

过程 EXPAND_VIA_UNSAFE(f - 不安全边, B - 坏边, U - 不安全边):
    对于 g ∈ N(f) 中尚未已探索的项 执行
        DETERMINE_EDGE_STATUS(g)
        如果 g 是 bad 则
            将 g 标记为已探索并 运行 EXPAND_BAD_COMPONENT(g, B, U)

过程 DETERMINE_EDGE_STATUS(g - 边):
    对于每个 w 在 g 中，只要 uncolored 尚未变为 g，就执行 safe 操作
        如果 包含 w （包括 g）的某条边是坏的 那么 将 w 标记为 troubled
        否则 为 w 分配一个随机颜色并将其标记为 accepted
    统计已接受顶点，并检查其颜色，以确定 g的状态
```

在通过不安全边进行扩展时，我们维护集合 U，其中包含尚未处理且与 B 的任意边相交的不安全边。只要 U 非空，我们就从 U 中选取任意一条不安全 f，并按照 `EXPAND_VIA_UNSAFE` 对其进行处理。在此过程中，我们确定与 f 相邻的所有边的状态；如果遇到一条不在 B 中的坏边，就将其加入其中，并扩展包含该边的坏分量。为方便技术处理，在坏分量扩展期间，我们收集尚未探索的相邻不安全边，并将它们加入 U。

3.2.3 color_final_分量

最终分量$\mathcal{C}$ 会扩展为与其相交的不安全边，然后将其限制为自身的问题顶点集合。所得超图记作$\mathcal{C}'$。算法尝试使用重新采样过程为$\mathcal{C}'$ 找到合法着色。为确保运行时间为多对数时间

10 遵循 Beck 方法的超图着色局部计算算法

算法 4 在最终分量内寻找着色

```
过程 COLOR_FINAL_COMPONENT(C - 超图):
    将所有与 C 相交的 unsafe 边添加到 C中
    C′ ← C 对问题顶点的限制
    t_e ← |E(C′)|/Δ                    // expected time of one RESAMPLE trial
    对于 trial = 1 到 2 log(m) 执行
        // RESAMPLE with limited number of steps
        为 V(C′) 分配随机颜色
        对于 step = 1 到 2t_e 执行
            如果 存在单色的 f ∈ E(C′) 则
                为 f的所有顶点分配新的随机颜色
            else
                //C′ is properly colored
                将C′ 的所有顶点标记为 accepted 并返回
    FAIL
```

，该过程只运行有限次重采样步骤。为降低失败概率，可以重新启动该过程几次。找到合法着色后，$\mathcal{C}'$ 中的每个顶点都会标记为已接受。从现在起，$\mathcal{C}$ 的所有边都视为安全边。然而，如果所有尝试均未成功，该过程就会宣告失败。

4 主要结果——算法

我们将展示如何改进上一节中描述的基本过程，从而得到一个可用于证明定理 1 的算法，也就是说，该算法对于满足强化的 LLL 条件 (1) 的输入超图，在每次查询时都能在多对数时间内运行。实际上，对于任何满足该条件且 $\alpha \in (0, 1)$ 的实例，我们的过程也能找到合法着色，但无法保证其运行时间适用于 $\alpha \geq 1/3$。我们首先介绍改进算法背后的主要思路，并描述其全局版本。接着，我们讨论如何将其调整为局部计算算法模型，最后给出 LCA 过程的描述。算法的分析见附录 C。

4.1 一般思路

随机着色算法的一种常见做法是从初始随机着色开始，然后进行一些修正，将其转换为合法着色（例如 Resample[19] 或 Alon 的并行算法 [2]）。Beck 的过程并非如此：它逐步构造合法着色，但会将某些顶点（即标记为受困的顶点）的着色推迟到后续阶段。我们的方法介于两者之间。总体而言，我们倾向于采用后一种做法，但会在碎裂阶段就为受困顶点采样颜色。这些颜色被视为提议的颜色，我们保留在最终着色阶段更改它们的可能性。我们利用提议颜色的信息来缩小最终着色阶段的处理范围。具体而言，查看为受困顶点提议的颜色后，只有包含单色边的最终分量才需要重新着色。此外，如果我们仔细跟踪坏分量之间的依赖关系（见定义 2），还可以缩小这些

最终分量的规模。我们将在以下小节中更详细地阐述这一想法。

4.1.1 坏分量的激活

想象所有顶点都已在碎裂阶段着色，而我们现在要确定最终连通分量。我们查看分量超图（见定义 3），并必须决定哪些坏分量需要重新着色。我们从被单色边相交的坏分量开始——将它们标记为初始激活，并将它们视为最终连通分量的种子。其余坏分量当前未激活。我们的意图是在最终着色阶段仅对活跃分量重新着色。请注意，仅通过改变着色使初始激活的分量获得正确着色，可能并不足够，因为在它们重新着色之后，某些在碎裂阶段包含两种颜色的不安全边可能会变成单色边。因此，必须传播激活。我们使用以下规则

- 令 A_t 为由活跃坏分量覆盖的问题顶点集合，f 为与 A_t 相交的不安全边。如果 $f \setminus A_t$ 是单色的，则所有与 f 相交的未激活坏分量都会被激活，而所有与 f 相交的坏分量都会合并为一个（最终的）分量。

上述传播规则会尽可能地持续应用。规则停止时，可以保证所有单色边都位于活跃分量内，而活跃分量外的所有不安全边和坏边都已由位于活跃坏连通分量之外的顶点正确着色。特别地，我们可以接受为非活跃顶点提出的所有颜色。

4.1.2 边修剪

我们采用一种额外的技术，可以进一步缩小最终分量的范围。注意，为了保证最终分量具有二着色性，只需确保每条边在某个最终分量内至少有 αk 个顶点可以重新着色即可。这意味着，如果某个活跃分量已经包含某条边的 αk 个问题顶点，那么就没有必要通过该边传播激活。因此，我们可以按如下方式改进传播规则。考虑一条不安全边 f，其中 $f \setminus A_t$ 是单色的（回顾一下，A_t 表示当前活跃的问题顶点集合）。如果某个活跃分量包含 f的至少 αk 个问题顶点，则将 f 修剪至该活跃分量。否则，激活所有与 f 相交的坏分量，并将它们合并为一个分量（如上一节所述）。

我们指出，这项技术的直接灵感来自 Czumaj 和 Scheideler 的工作 [7]，他们在构造待重新着色区域时积极使用了边修剪。使用这一方法的后果之一是，最终分量的形状取决于激活传播的具体顺序。

4.2 全局着色过程

与第 3.1 节中的基础算法类似，改进后的过程分为碎裂阶段和最终着色阶段。前者根据上一小节所述的思路进行了修改。特别是，我们使用提议的和已接受的颜色这两个概念。整个过程的伪代码见附录 A 中的清单 5。

4.2.1 碎裂阶段

碎裂阶段的第一部分几乎相同，不同之处在于现在会为每个顶点着色。该过程按照固定顺序处理顶点，并对每个顶点将其标记为已接受或受困，同时为其选择一个随机颜色。若处理顶点 v 时，v 不属于任何坏边，则该顶点已接受；否则，该顶点受困。当一条边中已接受顶点的数量达到 $(1-\alpha)k$ 且该边仍为单色时，这条边就会变成坏边。处理完所有顶点后，按照与基础算法相同的方式确定安全边和不安全边。此外，我们将所有顶点（已接受和受困）颜色都相同的边称为单色的边。已接受顶点的颜色称为已接受颜色。受困顶点的颜色称为提议颜色。接受分配给某个顶点的颜色，指的是将其状态改为已接受。

下一步是确定最终分量。我们使用分量超图。我们将把一些坏分量和不安全边标记为激活。活跃分量指的是分量超图中通过活跃不安全边彼此连通的极大活跃坏分量集合。我们首先将所有单色的不安全边，以及所有与任意（坏边或不安全边）单色边相交的坏分量标记为激活。令 A_t 表示当前被活跃坏分量覆盖的问题顶点集合。接下来，只要存在满足以下条件的非活跃不安全边 f:

- f 在活跃受困区域之外是单色的（即，$f \setminus A_t$ 是单色的），并且每个活跃分量包含的 f
- 问题顶点少于 αk ,

我们就激活 f ，并激活所有与 f相交的坏分量。当无法再应用此传播规则时，我们接受所有非活跃坏分量中问题顶点的颜色。此时，每个活跃分量都确定一个最终分量，即其坏分量的并集。与基础算法一样，如果每个最终分量包含至多 $2(\Delta+1)\log(m)$ 条坏边，则碎裂阶段成功 。否则，该过程宣告失败。

4.2.2 最终着色阶段

我们在最终着色阶段开始时进行一项修改。对于每个最终分量 $\mathcal{C}$，我们加入与其相交的不安全边，但不加入所有这类边，而只加入那些在 $V(\mathcal{C})$ 中至少有 αk 个受困顶点的边。然后，我们完全按照基础算法进行：将 $\mathcal{C}$ 限制在受困顶点上，并应用重新采样。

4.3 LCA 实现背后的思路

在基本情况下，将全局算法转换为 LCA 是直截了当的。事实上，LCA 版本确定的重新着色区域相同（假设两个版本按相同顺序处理顶点）。对于前一小节描述的改进算法，转换为 LCA 则更复杂，而且会改变算法的行为。主要难点在于，对于一个单独且初始时未激活的坏分量，要迅速判断它是否会被激活并不容易。可能存在一条很长的激活链，最终导致所考虑的坏分量被激活，而我们不知道应朝哪个方向搜索，以找到最终触发这次激活的源头。此外，即使我们发现它会被激活，也无法轻易确定包含它的最终分量会是什么形状，因为这还需要执行激活传播，并确定相邻坏分量的激活状态。为了解决这些问题，当查询某个坏分量中的一个受扰顶点时，我们着重寻找一个包含该顶点的区域。

该顶点可以独立于输入超图的其余部分进行重新着色。这意味着，从过程开始时起，该顶点所属的分量就被视为激活状态，并允许对该分量中的不安全边进行裁剪。此外，我们还采用下文所述的其他技术，以限制单次查询中已处理区域的扩展。

4.3.1 修剪至坏分量

我们将边修剪扩展到这样的情况：不安全边 f 在某个坏分量 S中至少有 αk 个受困顶点，并且这些顶点与f 的已接受顶点组成的集合不是单色的。在这种情况下，可以通过从 f 中移除不属于 S的受困顶点来修剪 f。注意，我们在此处不检查 S 是否处于激活状态。这一步背后的想法是，从现在起， S 负责 f的合法着色。如果在某个时刻， S 的顶点颜色无需任何重新采样便被接受，那么 f显然会被正确着色。否则，如果 S 变为激活状态，那么 f 无论如何都会被修剪，并且 S 拥有足够多的 f 受困顶点，不会破坏 S的二着色性。

4.3.2 激活排除

要使一个未激活的坏分量 S 被激活，必要条件是存在一条不安全边 f，其在 $f \cap V(S)$中的已接受顶点和受困顶点具有相同的颜色。当不存在这样的边，或所有这样的边都已被修剪至其他分量时，S 就无法被激活。因此，如果它最初不是激活的，就会保持未激活状态。在这种情况下，我们可以接受 S各顶点的所有提议颜色。这样一来，一些不安全边会变为合法着色的边，我们可以将其视为安全边。反过来，这可能有助于证明相邻的坏分量也不会被激活。同样的推理可以应用于一组 C 坏分量。如果 C 中的坏分量均未处于初始激活状态，并且不存在与 C 之外某个可能激活 C的坏分量相交的不安全边，那么我们可以得出结论：C 中的所有坏分量都将保持未激活状态。

4.3.3 条件扩展

上一小节所述的思路可用于坏分量，以某种方式搜索可能导致激活的原因。如果 S_1 最初并未激活，我们就检查可能导致 S_1激活的不安全边。我们可以任选一个这样的 f，并询问由 f 相交的其他坏分量 S_2 是否可能变为激活状态。只要仍存在激活迄今访问过的坏分量组中任一 S_i 的风险，我们就可以继续这一过程。最终，我们要么找到某个最初已激活的分量，要么证明所有已考察的坏分量都无法被激活。事实证明，如果我们不沿着那些可通过修剪至坏连通分量技术修剪的边进行搜索，那么此类搜索过程中处理的区域不太可能很大。

可以利用找到初始活跃坏分量的可能性，在扩展分量时将其延伸至一个邻接区域。对于一个与当前构造中的最终分量相邻的选定坏分量，我们启动搜索：要么找到某条单色边（初始活跃分量），并将整个搜索区域并入该分量；要么确认该区域无法被激活。后一种情况下，我们可以直接接受该区域中的提议颜色。前一种情况下，我们可以进行扩展，因为单色边作为一种不太可能发生的事件，在某种意义上抵消了扩展分量的代价。实际上，停止搜索过程的条件不仅可以是找到单色边，也可以是找到一种限制较少的情况，即找到俺去也

一条不安全边与搜索区域外至少两条不相交的坏边相交。这种可能性可由分析的技术细节推导得出。

4.4 LCA 过程

我们参照第 3.2 节介绍的基础算法来描述改进的 LCA 过程。与之前一样，顶点的顺序是动态构建的，并由查询和算法的执行过程决定。对于边集 S，我们用 by $V_t(S)$ 表示 $V(S)$ 中的所有问题顶点。对于边 f，我们用 $f|_t$ 表示 f 的问题顶点集，用 $f|_a$ 表示 f 的已接受顶点集。

4.4.1 查询

主过程与基础算法（列表 1）中的对应过程几乎完全相同。唯一的区别是，在处理坏边上的顶点 v 时，不仅会将其标记为问题顶点，还会为 v 分配一个随机颜色。

4.4.2 build_final_分量

此过程是算法的核心，与基础版本中的对应过程相比复杂得多。其内容见附录 A 中的清单 6 和 7。该过程还使用了前面定义的子程序（参见清单 3），但有一处修改：一旦某个顶点 w 被标记为受困，也会为 w 分配一个随机颜色。与之前一样，该过程在线图 H 上运行，并扩展一个由坏边组成的集合 B，在过程结束时，这些坏边将被转换为最终分量。该过程总是从包含被查询顶点 v 的坏分量开始，并通过不安全边扩展相邻坏分量。主要变化在于，基础算法中每条不安全边都会导致分量扩展，而在此处会更谨慎地处理不安全边。在整个过程中，我们确保 B 的大小不超过边数上界 $2(\Delta+1)\log(m)$；如果超过，该过程就会停止并宣布失败。

令 U 为与 V（B）相交的尚未处理的不安全边集合。如果某条边可以裁剪为 V（B），则可以将其从 U中安全地移除。因此，我们可以假设U 中的每条 f 在 V（B）中包含少于 αk 个受困顶点。由于每条不安全边都包含多于 αk个受困顶点，因此 U 中的每条 f 都必须与 V（B）之外至少一个坏分量相交。只要能够应用，过程就会使用以下扩展规则：

(r1) 如果存在 f 在 U 中，其在 B之外与至少两条不相交的坏边相交，或 (r2) 如果存在 f 在 U 中，使得 f 在 $V_t(B)$之外的所有顶点都是单色的，那么 B 将扩展为包含与 f相交的坏分量中的所有坏边；(r3) 如果 U 中没有满足条件（r1）或（r2）的边，但存在 f 在 U 其外部的问题顶点少于 αk 个 V (B), 然后针对 f 调用 `EXPAND_OR_ACCEPT` 过程（将在下一小节中介绍），该过程实现条件扩展技术，并将返回的坏边集合（可能为空）加入 B。

请注意，当不存在满足条件（r1）或（r2）的边时，对于来自 U 的任意剩余 f，可以保证 f 恰好与外部的一个坏分量相交，且 $f \setminus V_t(B)$ 不是单色的。如果这样的 f 不满足条件（r3），那么它在该外部坏分量中至少有 αk 个问题顶点，因此可以依据修剪至坏连通分量技术将其修剪至该分量。由此，f 可以从 U 中移除。

每次应用扩展规则后，都会将已处理的边从 U 中移除。另一方面，当扩展 B 时，可能会向 U 中添加新的不安全边，但我们会移除那些现在可以

修剪至 $V(B)$ 的边。由于不符合任何扩展规则的边也会从 U 中移除，最终 U 会变为空集，过程随之停止。此时， B 是一个坏边集合，其中的边周围只有安全边和修剪后的不安全边。

4.4.3 扩展_或接受

该过程是条件扩展技术的一种实现，针对给定的不安全边 e进行操作。与 BUILD_FINAL_COMPONENT类似，它会生成一组坏边 A ，我们称之为 搜索区域，并确保其大小不超过 $2(\Delta+1)\log(m)$ 的界限（如果超过该界限，整个算法就会停止并宣告失败）。初始时， A 为空。随后，它会通过坏分量进行扩展，这些坏分量可能会导向初始激活的坏分量；扩展从与 e相交的未探索坏分量开始。当没有更多候选坏分量时，扩展自然停止。然而，如果发现某条单色边，或某条与两条互不相交的未探索坏边相交的不安全边，该过程也可能提前停止。

设 Q 为待处理的不安全边集合（初始为空）。设 C 为当前正在扩展的坏分量中的坏边集合。令 U_C 表示与 $V(C)$相交、但不与 B 和 A 中的边相邻的不安全边集合（这些边正是与 C 中的边相邻、但在扩展 C之前尚未探索的不安全边）。该过程将 A 中的所有边加入 C，然后查找以下 摊销配置：

- (e1) 如果 C 包含单色边 f，则过程停止并返回集合 A；(e2) 如果 U_C 包含单色边 f，或者 (e3) 如果 U_C 包含一条边 f，该边与 C 外至少两条不相交的坏边相交，则首先将 A 与被 f 相交的坏分量中的所有坏边合并，然后过程停止并返回 A。

当找不到此类配置时， U_C 中的所有不安全边都不是单色的，而且每条不安全边在 A 外至多与一个坏分量相交。我们关注 U_C 中可能导致 C 激活的边——这些边在 V（C）中的问题顶点与已接受顶点一起构成单色边。每条此类边 f 必须与恰好一个外部坏分量相交，并且该分量的问题顶点与 $f|_a$ 一起确保对 f 的合法着色。如果该外部坏分量中至少有 αk 个 f 的问题顶点，则可以将 f 修剪至该分量（根据修剪至坏连通分量的技术）。因此，我们只将 U_C 中可能导致 C 激活且在 V（C）之外的问题顶点少于 αk 个的边添加到 Q 。

处理完 C 后，我们从 Q （待处理的不安全边集合）中任取一条边，并针对该选定边所相交的外部坏连通分量重复上述步骤。该分量可能已经被加入 A；在这种情况下，过程会继续从 Q 中选取边。如果过程结束时未遇到摊销配置，则 A 中没有单色边，并且所有与 V（A）相交的不安全边，要么由已接受顶点和来自 V_t（A）的顶点的颜色正确着色，要么被裁剪至其外部的坏分量。因此，整个 A 的激活可以排除。接着，我们将 V_t（A）中的所有顶点标记为已接受，并将由其颜色正确着色的边视为安全边。在这种情况下，过程返回空集。

请注意，在此过程中，如果 $V(A)$ 覆盖了某条不安全边上的至少 αk 个问题顶点，我们就不会对其进行边修剪，因为这可能导致误激活（若该边在 $V(A)$ 内为单色）。我们还会忽略所有与 $V(B)$ 相交的不安全边（它们在调用 EXPAND_OR_ACCEPT 之前已被探索），因为它们不满足 (r1)

和 (r2)，所以不能用于摊销配置，也不会导致激活（可以保证它们在 $V_i(B)$ 之外不是单色的）。

4.4.4 颜色_最终_分量

最后一个过程与基础算法（列表 4）中的对应过程几乎完全相同。回想一下，唯一的变化在于过程的开头。我们不再将与C相交的所有不安全边都加入其中，而只加入在 $V(C)$中至少有 αk 个问题顶点的不安全边。随后，我们按照基础算法继续进行。

参考文献

1 Dimitris Achlioptas、Themis Gouleakis 和 Fotis Iliopoulos。《一般Lovász局部引理的简单局部计算算法》。载于第32届ACM算法与体系结构并行性研讨会论文集，SPAA '20，第1–10页，美国纽约州纽约，2020年。计算机协会。 doi:10.1145/3350755.3400250。 2 Noga Alon。《局部引理的并行算法版本》。随机结构与算法，2(4):367–378，1991年。 doi:10.1002/rsa.3240020403。 3 Noga Alon 、 Ronitt Rubinfeld 、 Shai Vardi 和 Ning Xie。《空间高效的局部计算算法》。载于第二十三届ACM-SIAM离散算法研讨会论文集，第1132–1139页。ACM，纽约，2012年。 4 Noga Alon 和 Joel H. Spencer。《概率方法》（第二版）。John Wiley，2000年。 doi:10.1002/0471722154。 5 József Beck。《Lovász局部引理的算法方法》。第一部分。随机结构与算法，2(4):343–365，1991年。 doi:10.1002/rsa.3240020402。 6 Yi-Jun Chang 和 Seth Pettie。《LOCAL模型的时间层级定理》。SIAM计算杂志，48(1):33–69，2019年。 doi:10.1137/17M1157957。 7 Artur Czumaj 和 Christian Scheideler。《非均匀超图着色：一般Lovász局部引理的一种新算法方法》。载于第十一届ACM-SIAM离散算法研讨会论文集（美国加利福尼亚州旧金山，2000年），第30–39页。ACM，纽约，2000年。 8 Paul Erdős 和 László Lovász。《关于三色超图的问题与结果及若干相关问题》。载于《无限集与有限集》（研讨会，凯斯特海伊，1973年；献给Paul Erdős六十岁生日），第二卷，János Bolyai数学学会研讨会论文集第10卷，第609–627页。North-Holland，阿姆斯特丹，1975年。 9 Guy Even、Moti Medina 和 Dana Ron。《两种局部模型的优势：集中式局部算法与分布式局部算法》。信息与计算，262:69–89，2018年。 doi:10.1016/j.ic.2018.07.001。 10 András Faragó。《概率、图与算法的交汇点：Lovász局部引理及相关结果综述》。算法，14(12)，2021年。网址： https://www.mdpi.com/1999-4893/14/12/355, doi:10.3390/a14120355。 11 Manuela Fischer 和 Mohsen Ghaffari。《Lovász局部引理的次对数分布式算法及复杂度层级》。载于第31届国际分布式计算研讨会，LIPIcs第91卷。Leibniz国际信息学会议论文集，第18号文章，共16页。达格施图尔城堡。Leibniz信息学中心，瓦德恩，2017年。 12 Mohsen Ghaffari、David G. Harris 和 Fabian Kuhn。《关于局部分布式算法的去随机化》。载于第59届IEEE计算机科学基础年度研讨会——FOCS 2018，第662–673页。IEEE计算机协会，洛斯阿拉米托斯，美国加利福尼亚州，2018年。 doi:10.1109/FOCS.2018.00069。 13 Nathan Linial。《分布式图算法：基于局部数据的全局解》。载于第28届计算机科学基础年度研讨会（SFCS 1987），第331–335页，1987年。 doi:10.1109/SFCS.1987.20。

14 László Lovász。超图的覆盖与着色。载于 第四届东南部组合学、图论与计算会议论文集（佛罗里达大西洋大学，佛罗里达州博卡拉顿，*1973*年），第3–12页，1973年。**15** Yishay Mansour、Aviad Rubinstein、Shai Vardi和Ning Xie。将在线算法转换为局部计算算法。载于Artur Czumaj、Kurt Mehlhorn、Andrew Pitts和Roger Wattenhofer编， 自动机、语言与编程，第653–664页，柏林、海德堡，2012年。Springer Berlin Heidelberg。**16** Michael Molloy和Bruce Reed。局部引理的更多算法方面。载于 *STOC '98*（德克萨斯州达拉斯），第524–529页。ACM，纽约，1999年。**17** Michael Molloy和Bruce Reed。 图着色与概率方法。Springer，2002年。doi:10.1007/978-3-642-04016-0。**18** Robin A. Moser。更有效地对Lovász局部引理去随机化。*CoRR*，abs/0807.2120，2008年。网址： http://arxiv.org/abs/0807.2120， arXiv:0807.2120。**19** Robin A. Moser和Gábor Tardos。一般Lovász局部引理的构造性证明。*J. ACM*，57(2)：第11条，15页，2010年。**20** Ronitt Rubinfeld、Gil Tamir、Shai Vardi和Ning Xie。N.：快速局部计算算法。载于 *ICS 2011*，第223–238页，2011年。**21** Aravind Srinivasan。Lovász局部引理的改进算法版本。载于 第十九届*ACM-SIAM*离散算法年会，第611–620页。ACM，纽约，2008年。

18 超图着色的局部计算算法——遵循 Beck 方法

A 改进过程列表

A.1 全局算法列表

算法5 一致超图着色改进算法

过程 HYPERGRAPH_COLORING(H - 超图):
    // I. SHATTERING PHASE
    令 $(v_1, v_2, \ldots v_n)$ 为 $V(H)$ 的一个排列
    对于 $i = 1$ 到 n，执行
        为 v_i 分配一种随机颜色
        如果 包含 v_i 的所有边都不是坏的那么 将 v_i 标记为 accepted
        否则 将 v_i 标记为 troubled
    确定每个 $e \in E$（H）的状态 // e is bad、 safe、 or unsafe
    探索线图并构建分量超图 $H_C = (V_C, E_C)$
    // activation of bad-components
    // - let U_C be the set of unsafe edges corresponding to E_C
    // - let $U(B)$ denote unsafe edges intersecting component B
    // - let $U_C(B) = U(B) \cap U_C$
    // - let V_t ($\mathcal{C}$) denote set of troubled vertices in component $\mathcal{C}$
    $\mathcal{A} \leftarrow \emptyset$ // initialize set of active components
    $Q \leftarrow \emptyset$ // unsafe edges to process
    // - initial activation
    对于每个 $B \in V_C$ 执行
        如果某个 $e \in E(B)$ 或 $f \in U(B)$ 是单色的，则
            将 B 标记为激活
            将 B 添加到 $\mathcal{A}$ and，并将来自 $U_C(B)$ 的所有边添加到 Q
        else 标记 B 为未激活
    对每个 $f \in U_C$ 执行
        如果 f 是单色的 那么 将 $\mathcal{A}$ 中所有被 f 相交的 $\mathcal{C} \in \mathcal{A}$ 合并。
    // - activation propagation
    当 Q 不为空 时执行
        $f \leftarrow$ 下一个边（从 Q 中将其移除）来自 Q
        如果 $\forall_{\mathcal{C} \in \mathcal{A}} |f \cap V_t(\mathcal{C})| < \alpha k$ 且 $f \setminus V_t(\bigcup \mathcal{A})$ 是单色的那么
            // - activate new bad-components through f
            对于每个 $B \in V_C$，使得 B 为未激活且 f 与 B 相交时执行
                将 B 标记为激活
                将 B 添加到 $\mathcal{A}$ 和 将所有边添加到 Q，这些边来自 $U_C(B)$
            // - merge active components through f
            合并所有与 $\mathcal{A}$ $\mathcal{C} \in \mathcal{A}$ 相交的 f
    // II. FINAL COLORING PHASE - color each final component
    对每个 $\mathcal{C} \in \mathcal{A}$ 执行
        对每个 $f \in U(\mathcal{C})$ 满足 $|f \cap V_t(\mathcal{C})| \geq \alpha k$，将 添加 f 到 $\mathcal{C}$
        $\mathcal{C}' \leftarrow$ 对受困顶点的 $\mathcal{C}$ 限制
        重新采样($\mathcal{C}'$)

A.2 build_final_分量(LCA) 的列表

算法6 用于最终分量构造的改进LCA过程

```
过程 BUILD_FINAL_COMPONENT(v - 受困顶点):
    B ← ∅                    // initialize set of bad edges of the component
    U ← ∅                    // initialize set of unsafe edges to process
    U_s ← ∅                  // unprocessed unsafe edges able to launch search
    包含 v 的任意坏边中的 e ←
    将 e 标记为已探索并运行 EXPAND_BAD_COMPONENT(e, B, U)
    // process surrounding unsafe edges according to extension rules
    当 U ≠ ∅ 或 U_s ≠ ∅ 时 do
        当 U 不为空时执行
            f ← 从 U 取出下一条边（并将其从 U中移除）
            如果 f 在 V（B）中有 < αk 个问题顶点则
                如果 f 满足规则（r1）或（r2）则
                    EXPAND_VIA_UNSAFE(f, B, U)
                否则，如果 f 可以满足规则（r3）则
                    将 f 添加到 U_s          // f \ V(B) has < αk troubled vertices
        如果 U_s 不为空则
            f ← U_s 中的下一条边（将其从 U_s中移除）
            如果 f 在 V(B) 中有 < αk 个受困顶点则
                // f satisfies rule (r3)
                (A, U_A) ← EXPAND_OR_ACCEPT(f, B, U)
                B = B ∪ A 并且如果 |B| > 2(Δ + 1)log(m) 则失败
                U = U ∪ U_A
    // return hypergraph built on set of bad edges
    返回 C = (V(B) B)
```

20 遵循 Beck 方法的超图着色本地计算算法

A.3 expand_or_accept(LCA) 的列表

算法 7 通过不安全边 e 进行条件扩展（探索搜索区域）

过程 EXPAND_OR_ACCEPT(e - 不安全边)**:**
  $A \leftarrow \emptyset$ // initialize set of bad edges of the search area
  $U_A \leftarrow \emptyset$ // initialize set of unsafe edges around search area
  $Q \leftarrow \{e\}$ // unprocessed unsafe edges allowing expansion
  // process selected surrounding unsafe edges
  当 Q不为空**时**
    $f \leftarrow$ 来自 Q 的下一条边（将其从 Q 中移除）
    // expand with the external component to which leads f
    $(C,\ U_C) \leftarrow (\emptyset,\ \emptyset)$
    EXPAND_VIA_UNSAFE(f, C, U_C)
    $A = A \cup C$ **并且如果** $|A| > 2(\Delta+1)\log(m)$ **则失败**
    $U_A = U_A \cup U_C$
    // inspect new edges - look for amortizing configuration
    如果 ***(e1)*** 得到满足（ C 中存在单色边）**则**
      返回 $(A,\ U_A)$
    否则，如果存在一条不安全边 f，位于 U_C 中且满足 ***(e2)*** 或 ***(e3)*****则**
      EXPAND_VIA_UNSAFE(f, A, U_A)
      返回 $(A,\ U_A)$
    // select edges that may cause an activation
    else
      对于 g，U_C **执行**
        if $g|_a \cup (g|_t \cap V(C))$ 是单色的，**则**
          如果 $g \mid V(C)$ 有 $< \alpha k$ 个受困顶点，**则** 将 g 添加到 Q
  // activation exclusion
  将 $V(A)$中的所有受困顶点标记为 accepted
  返回 $(\emptyset,\ \emptyset)$

B 基础算法的分析

我们证明，第 3.2 节中描述的算法是一个用于均匀超图二着色问题的 $(\mathcal{O}(\log^2(n)),\mathcal{O}(n),\mathcal{O}(1/n))$-LCA。下面我们简要讨论其正确性和运行时间。空间需求 $\mathcal{O}(n)$ 是显而易见的。接下来，我们重点分析碎裂阶段的失败概率，这是整个分析中最重要的部分。我们证明，对于任意查询序列，该概率至多为 $1/m$。此外，在保持相同时间界的情况下，可以将其任意降低。最后，我们证明，对于最终着色阶段，同样的结论也成立。综合起来，这意味着算法的失败概率为 $\mathcal{O}(1/n)$。

B.1 正确性

我们证明，只要该过程没有失败，由已接受顶点的颜色所定义的当前部分着色就可以扩展为 H 的合法着色。注意，在每次查询得到回答后，我们都有以下性质：每条边要么已经被合法着色（即为安全边），要么至少包含 αk 个未着色或受困顶点。因此，从 H 中移除安全边，并将其限制在未着色顶点和受困顶点上，我们便得到一个最大边度至多为 Δ、且每条边的大小至少为 αk 的超图。由于假设 $2e(\Delta+1) < 2^{\alpha k}$，根据局部引理（LLL），我们知道剩余超图是二着色的，并且该超图的任意合法着色都可以用于扩展 H 的当前部分着色。

为了在 LCA 的背景下完成正确性的证明，我们还应提及，算法构造的部分着色只会不断扩展；也就是说，一旦某个顶点被标记为已接受，其颜色就不会再改变。特别是，多次查询的顶点会得到一致的答案。

B.2 运行时间

回顾一下，我们将 k 和 Δ 视为固定值（即相对于超图的规模为$\mathcal{O}(1)$），因此，检查任意指定边中每个顶点的当前状态、遍历包含某个指定顶点的所有边，或确定边 f 的集合 $N(f)$，都能在常数时间内完成。特别地，`DETERMINE_EDGE_STATUS` 的运行时间为$\mathcal{O}(1)$。

考虑一次调用 `QUERY`(v)。如果 v 已被接受，或不属于任何坏边，则计算可在常数时间内完成。否则，算法必须构造并着色包含 v 的最终分量。正如下文所论证的，这两个步骤都能在$\mathcal{O}(\log^2(n))$ 时间内完成。

过程 `BUILD_FINAL_COMPONENT` 会遍历集合 B 以及与 B相邻的所有边，因此单次调用最多处理 $(\Delta+1)|B|$ 条边。注意，在该过程及其子程序中，直接处理单条边所花费的时间是常数。因此，由于 B 的大小有明确上界，这一步只需 $\mathcal{O}(\log(n))$ 时间。

过程 `COLOR_FINAL_COMPONENT` 从超图 $\mathcal{C}$ 开始，该超图至多包含 $2(\Delta+1)\log(m)$ 条边，因此其大小 $|\mathcal{C}|$ 为 $\mathcal{O}(\log(n))$。在相对于 $|\mathcal{C}|$ 的线性时间内，它得到一个大小同阶的 $\mathcal{C}'$。于是，重新采样过程至多执行 $\mathcal{O}(\log(n))$ 次。单次重新采样需要 $\mathcal{O}(|\mathcal{C}'|)$ 时间来初始化颜色并找出所有单色边；此后，每一步重新采样都可以在常数时间内完成（重新采样 f 后，我们只需检查 f 及其相邻边的状态）。由于我们将重新采样次数限制为 $t_e = |E(\mathcal{C}')|/\Delta$，即$\mathcal{O}(\log(n))$，每次重新采样过程需要$\mathcal{O}(\log(n))$ 时间。总之，整个过程的运行时间为 $\mathcal{O}(\log^2(n))$。

B.3 碎裂阶段发生失败的概率

我们关注这样的事件：在某次查询期间，过程 BUILD_FINAL_COMPONENT构造出的坏边集合 B 大小超过 $2(\Delta+1)\log(m)$。为了进行分析，假设在这种情况下，过程不会失败，而是继续执行，直到构造出一个完整的最终分量。然后，我们测量发现大型最终分量的概率，即边数超过 $2(\Delta+1)\log(m)$ 的分量。我们按照 Molloy 和 Reed 的证明分析这种情况（参见 [17] 中的引理 25.2），但对估计稍作改进。

采用一种常见的技巧，我们为算法发现的每个最终分量关联一个见证结构，其大小与该分量的大小密切相关。我们为每个见证结构定义一个事件，使得每当构造出某个最终分量时，与之关联的见证结构所定义的事件就会成立。最后，我们证明，由所有足够大的见证结构定义的任何事件成立的可能性都极低。这也限制了发现大型最终分量的概率。

回顾一下，$L(H)$ 是 H 的线图。若边集 $T \subseteq E$ 在这样一个图中连通，则称其为 $L(H)$ 中的 (1，2)- 树：当且仅当 T 中的元素在 $L(H)$ 中的距离为 1 或 2 时，它们在该图中相邻。若边集 $T \subseteq E$ 在这样一个图中连通，则称其为 $L(H)$ 中的 (2，3)-树：当且仅当 T 中的元素在 $L(H)$ 中的距离为 2 或 3 时，它们在该图中相邻；此外，T 中任意两个元素在 $L(H)$ 中都不相邻。采用上述记号，我们将 $L(H)$ 中的任意 (2，3)-树定义为见证结构。我们为每个见证结构 τ 指定一个事件$\mathcal{X}(\tau)$，表示 τ 中的每条边在碎裂阶段之后都是坏的。当 $\mathcal{X}(\tau)$ 成立时，我们称 τ 成立。

▷ 命题 4。如果 $\mathcal{C}$ 是一个最终分量，那么 $E(\mathcal{C})$ 是 $L(H)$ 中的一个 (1,2)-树。

证明。 这很容易由以下事实得出： $E(\mathcal{C})$ 包含了在分量超图中连通的坏分量的边。根据定义，每个坏分量的边在 $L(H)$中是连通的；当两个坏分量在分量超图中相邻时，必然存在两条边，分别属于这两个坏分量，且它们在$L(H)$中的距离为 2。 ◁

对于一个最终分量 $\mathcal{C}$ ，令 $\tau(\mathcal{C})$ 为与 $\mathcal{C}$ 相关联的见证结构，其定义为 $E(\mathcal{C})$ 的一个极大子集，该子集构成一个 (2,3)-树。

▷ 命题 5. 若$\mathcal{C}$ 是最终分量，则满足 $\tau(\mathcal{C})$，且 $|\tau(\mathcal{C})| \geq |E(\mathcal{C})|/(\Delta+1)$。

证明。 根据定义 $\tau(\mathcal{C}) \subseteq E(\mathcal{C})$，并且 $E(\mathcal{C})$ 仅包含坏边，因此 $\tau(\mathcal{C})$ 满足条件。其大小的下界由命题 4 得出，事实上，所有(1,2)-树中的元素都必须与其最大的 $(2,3)$-树相邻（参见 [17] 中的事实 25.5）。 ◁

命题 5 使我们能够将分析聚焦于见证结构。由此直接可知，如果要证明发现一个含有超过 $2(\Delta+1)\log(m)$ 条边的最终分量是不太可能的，只需证明找到一个大小超过 $2\log(m)$ 的满足条件的见证结构是不太可能的。

▷ 命题 6. $L(H)$ 中大小为 u 的见证结构数量少于 $m(4\Delta^3)^u$。

证明。 这可由对大小为 u 的树的可能形状进行简单计数得出4^u （其形状至多有这些），以及将其嵌入到具有 m 个顶点且最大度至多为 Δ^3 的图中的方式数量 $m(\Delta^3)^{u-1}$ （至多有这些种方式）得出。另请参见命题 25.6 的证明 [17]。 ◁

▷ 命题 7。设 $p_B = 2^{1-(1-\alpha)k}$。对于大小为 u 的见证结构 τ，我们有

$$\Pr[\mathcal{X}(\tau)] \leq p_B^u.$$

证明。对于单条边 f，f 变坏的概率至多为 p_B，因为在碎裂阶段，算法必须为 $(1-\alpha)k$ 的顶点分配至少 f 个随机颜色（这可能不会发生），并且前 $(1-\alpha)k$ 次分配必须得到相同的结果。对于一组互不相交的边 τ，满足事件 $\mathcal{X}(\tau)$ 要求在 u 个互不相交的顶点集上出现这种行为。◁

▶**命题 8**。设

$$\frac{8\Delta^3}{2^{(1-\alpha)k}} < \frac{1}{2} \tag{3}$$

且 $u = \lceil (c+1)\log(m) \rceil$，对于某个 $c > 0$，则

$$\Pr[\ \textit{there is a satisfied witness structure of size } u\ ] < \frac{1}{m^c}.$$

证明。我们采用一阶矩方法，通过满足条件的、大小为 u的见证结构的期望数量，来界定找到此类见证结构的概率。我们使用断言 6 和 7 得到界 $m(4\Delta^3 p_B)^u$。根据假设，有 $4\Delta^3 p_B < \frac{1}{2}$，结合 $u \geq (c+1)\log(m)$ 即可完成证明。◀

最后请注意，对于所有 $\alpha \leq 1/4$ 和所有 k，只要强化的 LLL 条件 (1) 成立，(3) 也同样成立。在命题 8 中取 $u = \lceil 2\log(m) \rceil$ 后，我们得到找到大小恰好为 u 且满足条件的见证的概率小于 $1/m$。这也限制了找到更大且满足条件的见证的可能性，因为每个大小大于 u 的 (2,3)-树都包含一个大小恰好为 u 的 (2,3)-树。因此，根据命题 5，发现大型最终分量的可能性很低，并且在碎裂阶段（涵盖所有查询）发生失败的概率小于 $1/m$。此外，通过线性增加最终分量的可接受大小，我们可以使这一界限呈指数下降。

B.4 最终着色阶段的失败概率

考虑过程 `COLOR_FINAL_COMPONENT` 的一次执行，以及从输入最终分量 $\mathcal{C}$ 中获得的超图 $\mathcal{C}'$。由于它满足局部引理的假设，Moser 和 Tardos 的定理可以将重新采样的预期步骤数界定为 $|E(\mathcal{C}')|/\Delta$（见 [10] 中的定理 7）。运行该过程两倍于预期步骤数的次数，可以确保单次试验未找到解的概率至多为 $1/2$。重复该过程 $2\log(m)$ 次后，失败概率将降至 m^{-2}。

现在考虑 `COLOR_FINAL_COMPONENT` 在所有查询中的执行情况。由于最终分量的数量不会多于边的数量，因此调用次数不会超过 m 次。因此，在最终着色阶段（涵盖所有查询）发生失败的概率小于 $1/m$。同样需要指出的是，通过线性增加重新采样过程的试验次数，我们可以使这个界限呈指数下降。

C 主要结果的证明

本节将证明定理 1 中表述的主要结果。事实上，我们将证明以下略强的结论。

▶ **定理 9**（主要结果——技术性表述）。 存在一种局部计算算法，可在每次查询的多对数时间内，以概率 $1 - O(1/n)$ 解决 k 均匀超图的二着色问题，其中最大边度为 Δ，且满足

(i) $2\mathrm{e}(\Delta + 1) < 2^{\alpha k}$，其中 $\alpha \leq 1/3$，以及 *(ii)* $\Delta \leq \frac{1}{192}\, 2^{k/3}$。

上述定理中的第一个条件保证了我们算法的正确性。第二个条件来自分析，并且是确保以高概率每个最终分量的大小至多为对数级所必需的。

注意，对于 $k \geq 8$，第二个条件在 $\alpha = 1/3$ 时蕴含第一个条件。另一方面，取更小的 k 并不合理，因为这样一来，第一个条件就意味着超图没有边（$\Delta < 1$）。因此，原则上我们可以将定理的假设缩减为仅包含第二个条件。相较于定理 1，技术性表述允许取 $\alpha = 1/3$，并阐明了何谓足够大的 k（注意，对于 $\alpha < 1/3$，加强的局部引理判据在 k 足够大时蕴含第二个条件）。

固定 $\alpha \leq 1/3$，并令 H 为一个 k-均匀超图，满足定理 9 针对该 α 的假设。与基础算法的分析（见 B 节）类似，我们证明，4.4 节所述的改进算法满足 $(\log^2(n), \mathcal{O}(n), \mathcal{O}(1/n))$- LCA 的条件，适用于 k-均匀超图的二着色问题。为证明正确性，我们可以直接使用与基础算法相同的论证（见 B.1 节）。空间需求也保持不变。因此，下面我们只讨论运行时间和失败概率。由于改进算法与基础版本的显著差异仅在于 BUILD_FINAL_COMPONENT 的实现，我们将分析重点放在该过程上。

C.1 碎裂阶段的运行时间

考虑一次调用 BUILD_FINAL_COMPONENT。其运行时间与执行过程中处理的边数呈线性关系。这些边是在扩展规则和条件扩展过程中探索的坏边和不安全边，而这些过程最终会接受其搜索区域。由于所有已处理的不安全边都与已探索的坏边相邻，且 Δ 为 $\mathcal{O}(1)$，我们只需估算后者的数量。我们证明其为 $\mathcal{O}(\log^2(n))$，因此最终分量构造的运行时间满足 $\mathcal{O}(\log^2(n))$ 的界。

聚焦于 EXPAND_OR_ACCEPT 的一次单独执行。当选定的坏分量被扩展时，会探索坏边，随后将这些坏边添加到搜索区域。由于在分量扩展过程中会显式检查大小（在 EXPAND_BAD_COMPONENT 中），并且还会检查搜索区域的大小，因此单次条件扩展最多可以探索 $\mathcal{O}(\log(n))$ 条坏边。

回到 BUILD_FINAL_COMPONENT，现在我们可以注意到，在每条扩展规则之前、期间和之后，集合 B 的大小均为 $\mathcal{O}(\log(n))$。因此，只需界定那些最终接受其搜索区域的条件展开过程中已探索的坏边数量。注意，所有条件展开的数量均受与 B 相邻的不安全边数量的限制，因此它们至多有 $\mathcal{O}(\log(n))$ 个。这意味着，已探索边的总数确实为 $\mathcal{O}(\log^2(n))$。

C.2 扰乱阶段中的失败概率

与基础算法中对碎裂阶段的分析类似（见 B.3 节），我们假设过程 BUILD_FINAL_COMPONENT 和 EXPAND_OR_ACCEPT 不会失败，而是持续运行，直到构造出一个完整分量（最终分量或搜索区域）。因此，我们衡量发现一个包含超过 $2(\Delta+1)\log(m)$ 条边的分量的概率。我们在下面的命题中界定这一概率。

▶**命题 10.** 在定理 *9* 的假设下，对于任意查询序列，算法发现包含超过 $2(\Delta+1)\log(m)$ 条边的最终分量或搜索区域的概率小于 $\frac{2}{m}$。 —

这个命题的证明采用了与基础算法中类似性质的证明相似的构造，不过相应步骤比之前更复杂。首先，我们定义与 H 的线图相关联的见证结构，用于跟踪最终分量和搜索区域的扩展。这一次，见证结构更加复杂，由其定义的事件可以描述其中边上的多种行为。接着，我们界定特定大小的可能结构数量，并证明找到足够大的、满足条件的见证结构的可能性很低。最后，我们说明如何按照 BUILD_FINAL_COMPONENT 和 EXPAND_OR_ACCEPT 过程的工作构造满足条件的见证结构。整个证明相当技术性，因此我们将其推迟到下一节。

D 命题 10 的证明

D.1 见证结构

本节定义一种见证结构，用于描述与 4.4节所述算法的运行相关的事件。该算法作用于一个 k 一一致超图 H，该超图满足定理 9 对某个 $\alpha \le 1/3$ 的假设。

D.1.1 基本性质和定义

见证结构的主要部分是一棵树，它通过同态（即保持邻接关系的映射）嵌入线图 $L(H)$ 中。此外，它还配备了一些额外信息——用于表示每个顶点类型的标签，以及边的方向。

▶ **定义 11**。大小为 u 的见证结构是一个元组 $\tau = (T、f、\sigma、\phi)$ 其中 *1.* T 是一个大小为 u 的树（含 u 个顶点的无标号树） *2.* $f : T \to L(H)$是从 T 到 H的线图的同态， *3.* σ将标签$\{(M)、(B)、(B_{in})、(B_{out})、(U)、(E)\}$赋予 T 的顶点， *4.* ϕ 是一个为 T的边指定方向的函数。

我们使用“节点”一词来表示见证结构的顶点，以便明确区分 T 的顶点和 H 的顶点。出于类似的原因， T 的边称为弧。对于每个节点 x ，令 f_x 表示通过同态映射 f 分配给它的边（两个节点可能映射到同一条边）。标记为 (E) 的节点为空节点，所有其他节点均为活跃节点。令 $E(\tau) \subset E(H)$ 表示分配给 τ 中各节点的所有边组成的集合。令 $A(\tau) \subset E(\tau)$ 表示分配给 τ 中活跃节点的所有边组成的集合。没有出边的节点称为根。对于每个节点 x ，我们将其深度 $d(x)$ 定义为从 x 到某个可达根的最长有向路径（由弧组成）的长度。根的深度为 0。对于边集 S，令 $N^+(S)$ 表示被 S 中的边覆盖的边集，即属于 S 或与 S 相邻的边所组成的集合（形式化地， $N^+(S) = \{S \cup \bigcup_{f \in S} N(f)\}$）。对于一个正数 d 和一个见证结构 τ:

- 令 $A_{<d}(\tau) \subset A(\tau)$ 为分配给 τ 中深度严格小于 d 的活跃节点的边集；令 $R(\tau) = V(N^+(A(\tau)))$ 为属于由 $A(\tau)$ 覆盖的边的顶点集，令 $R_{<d}(\tau) = V(N^+(A_{<d}(\tau)))$ 为属于由 $A_{<d}(\tau)$ 覆盖的边的顶点集。

见证结构的定义概括了我们将要使用的结构的基本形态。这足以为给定大小的结构数量获得一个合理的上界。

▷ 断言 12. 大小为 u 的见证结构数量小于 $m(48\Delta)^u$。

证明。至多有 4^u 棵无标号树，其上有 u 个顶点。对于每棵这样的树 T，至多有 $m\Delta^{u-1}$ 个从 T 到 $L(H)$的同态映射，因为第一个节点可以映射到 $L(H)$中的任意顶点，而每个其他节点都必须以保持邻接关系的方式映射。最后，标记函数 6^u 有 σ种候选选择，且 2^{u-1} 有 ϕ种选择。

◁

D.1.2 由见证结构描述的事件

对于每个活跃节点 x，分配给 x 的标签决定了某个基本事件。它仅取决于碎裂阶段中分配给 f_x 顶点的颜色。回想一下，对于一条边 f，$f|_a$表示 f中已接受顶点的集合，而 $f|_t$表示 f中的问题顶点集合。

▶ **定义 13**。 设 x 是见证结构 τ中的一个活跃节点。基本事件 $X(x)$ 定义如下，具体取决于分配给 x的标签 $\sigma(x)$ ：

- (M) – f_x 是单色的，(B) – f_x 是坏的，(B_{in}) – f_x 是不安全的，且集合 $f_x|_a \cup ($
- $f_x|_t \cap R_{<d(x)}(\tau))$ 是单色的，大小至少为 $(1-\alpha)k$，(B_{out}) – f_x 是不安全的，且集合
- $f_x|_a \cup (f_x|_t \setminus R_{<d(x)}(\tau))$ 是单色的，大小至少为 $(1-\alpha)k$，(U) – f_x 是不安全的，且受困顶点少于 $2\alpha k$ 个，也就是说， $f_x|_a$ 是单色的，大小至少为 $(1-2\alpha)k$。

请注意，标记为 (B_{in}) 或 (B_{out}) 的节点的基本事件定义涉及其外层见证结构的形状。

▶ **定义 14.** 对于见证结构 τ，事件 $X(\tau)$ 定义为与 τ 的活跃节点相关联的基本事件的合取。当 $X(\tau)$ 成立时，我们称见证结构 τ 为满足。

注意， τ的空节点不会影响事件$\mathcal{X}(\tau)$（或者，我们可以假设分配给空节点的事件总是成立）。下面，我们为基本事件的概率定义上界 p_M、 p_B、 p_U 。

▷ 断言 15. 对于标记为（M）的 x， $\mathcal{X}(x)$的概率为 $p_M = 2/2^k$.

▷ 断言 16. 对于标记为（B）、（B_{in}）或（B_{out}）的 x ， $\mathcal{X}(x)$的概率至多为 $p_B = 2/2^{(1-\alpha)k}$。

证明。在每种情况下， f_x 的所有已接受顶点都必须是单色的；当 x 标记为（B_{in}）或（B_{out}）时，部分问题顶点也必须获得相同的颜色。注意，对于每个顶点 $v \in f_x$ ，在为 v抽取颜色时，我们已经知道它会被标记为已接受还是问题顶点。我们也知道 v 是否属于 $R_{<d(x)}$（τ），因为该集合完全由 τ确定。因此，为 v 选择颜色时，我们可以知道它是否属于 f_x 中应该是单色的那一部分。为了满足 X（x），这类顶点至少应有（$1-\alpha$）k 个（这种情况可能不会发生）；而且由于它们必须得到相同的结果，该事件的概率至多为 p_B。◁

▷ 断言 17。对于 x 标记的 (U)， $\mathcal{X}(x)$ 的概率至多为 $p_U = 2/2^{(1-2\alpha)k}$。

证明。 这可由与以下事件相同的论证得到： f_x <style id='6}是坏的，只是这里已接受顶点的集合可能更小，但至少应有 $(1-2\alpha)k$ 个。

◁

某些 τ 的活跃节点的基本事件并不独立，即使分配的边两两不相交也是如此。不过，在这种情况下，基本事件合取的概率确实有一个上界，即相应上界的乘积。这是因为此时每个基本事件都规定了不同顶点集上的期望行为，而这些上界只涉及特定采样颜色取相同值的概率。

D.1.3 合法见证结构

我们所关注的事件的性质允许对该结构施加额外的性质。这些性质用于推导此类见证结构成立的概率上界。首先，空节点的数量不能过多。所需的比例由以下定义刻画。

▶ **定义 18**。 对于见证结构 τ，我们通过以下公式定义 τ 的平衡 。

$$2\#_{\{(M)\}} + \#_{\{(B),(B_{in}),(B_{out})\}} - \#_{\{(E)\}},$$

其中$\#_X$ 是 τ 中标签属于集合 X的节点数。见证结构具有 a 非负平衡称为平衡的。

其次，我们要求每个基本事件都指向一个互不相交的顶点集合。这两个条件刻画了我们所关注的见证结构。

▶**定义 19.** 如果一个见证结构是平衡的，并且分配给活跃节点的边两两不相交（特别是，同一条边不能分配给两个不同的活跃节点），则称其为适当的 。

对于适当的见证结构 τ，$\mathcal{X}(\tau)$ 的概率至多为与τ的活跃节点相对应的基本事件概率上界的乘积。见证结构是平衡的这一事实，可用于通过 τ 中活跃节点的存在，对空节点的出现进行摊销。

▷ 命题 20. 设 $q = 2^{1-k/3}$。对于 $\alpha \le 1/3$，有 $\max(p_M^{1/3}, p_B^{1/2}, p_U) \le q$。

上述命题意味着，从某种意义上说，每个标记为 (M) 的节点都可以摊销两个空节点，而每个标记为 (B)、(B_{in}) 或 (B_{out}) 的节点都可以摊销一个空节点。利用这一观察，对于适当的见证结构 τ，我们能够推导出满足 τ 的概率上界，该上界仅涉及此见证结构的大小。

▷ 命题 21. 设 $q = 2^{1-k/3}$。对于大小为 u 的适当见证结构 τ，我们有

$$\Pr[\mathcal{X}(\tau)] \le q^u.$$

证明。 将基本事件的上界相乘，并利用命题 20 以及 τ 是平衡的这一事实，我们得到

$$\Pr[\mathcal{X}(\tau)] \le p_M^{\#_{\{(M)\}}} p_B^{\#_{\{(B),(B_{in}),(B_{out})\}}} p_U^{\#_{\{(U)\}}} \le q^{3\#_{\{(M)\}} + 2\#_{\{(B),(B_{in}),(B_{out})\}} + \#_{\{(U)\}}} \le q^u.$$

◁

现在，我们可以对在碎裂阶段变为满足状态的、大小足够大的某个适当见证结构的概率给出上界。

▶ **命题 22.** 设 $q = 2^{1-k/3}$。

$$\Pr[\text{存在一个大小至少为 } 2\log(m) \text{ 的满足的适当见证结构}] < \frac{2}{m}$$

证明。 由命题 12 和命题 21，我们得到：

$$\mathbb{E}\left[\begin{array}{c}\text{number of satisfied proper witness structures}\\ \text{of size at least } 2\log(m)\end{array}\right] < \sum_{u \ge 2\log(m)} m(48\Delta)^u \cdot q^u.$$

回想一下，$\Delta \le \frac{1}{192}\, 2^{k/3}$（根据定理 9 的假设）。因此 $48\Delta q \le \frac{1}{2}$，并且上述总和很容易上界为 $\frac{2}{m}$。◀

D.2 满足的适当见证结构的构造

我们通过证明算法所完成的工作与满足的适当见证结构的出现之间的关系，来完成命题 10 的证明。命题 10 是命题 22 和命题 23（如下所述）的直接推论。后者的证明一直延续到本节末尾。

▶**命题 23.** 对于算法执行过程中构造的每个分量（无论是最终分量还是搜索区域），都存在一个大小为$|\tau| \geq \frac{|C|}{(\Delta+1)}$的满足的适当见证结构 τ，其中 C表示该分量的坏边集合。

为了证明命题 23，我们将说明，随着算法的推进，我们可以同时构造出一个大小相应的满足的适当见证结构。构造过程分为一系列扩展，每次都在迄今构造出的见证结构上进行。这些扩展对应于将分量扩展到相邻的坏分量（通过不安全边）。首先，我们明确构造过程中拟保持的见证结构性质。然后，我们介绍构造和扩展见证结构的基本技巧。接着，我们讨论如何为在 EXPAND_OR_ACCEPT中已探索的搜索区域构造满足的适当见证结构。最后，我们说明如何为在 BUILD_FINAL_COMPONENT中得到的最终分量构造满足的适当见证结构。

D.2.1 构造的性质

对于见证结构 τ，边 g 与 τ 之间的距离，是 L（H）中 g 与 E（τ）中任意元素之间最短路径的长度。我们称一条边与 τ **相邻**，如果它到 τ 的距离恰好为 1。如果一条边与 E（τ）中的某条边相交，即它属于 E（τ）或与其相邻，则称该边 **与 τ 相交**。我们将属于 A（τ）的边称为 **活跃边**。对于边集 S，令 $S|_b \subset S$ 表示 S 中所有坏边构成的集合。当 S 被 A（τ）覆盖时，我们称 τ **覆盖** S。

▶ **定义 24** (适当构造). 设 τ 为一个见证结构，C 为一个边集。当 τ 满足以下所有性质时，我们称其为 C 的适当构造：
(a) 的活跃边互不相交，且没有边被分配给两个不同的 nt 激活节点 s，
(b) τ 具有正平衡，**(c)** 由 τ 中的激活节点定义的每个基本事件均成立，
(d) τ覆盖了 C中包含的所有坏边，也就是说，$C|_b \subset N^+(A(\tau))$，**(e)** $A(\tau) \subset C$。

前两个条件意味着 τ 是适当的。第三个条件确保它成立。条件 (d) 意味着 $C|_b$ 的大小至多是 τ大小的 $(\Delta + 1)$ 倍（如命题 23 中所要求）。它还与条件 (e) 一起用于证明其他性质。请注意，适当构造的正平衡确保活跃边集合非空。

在以下部分中，我们将展示如何为单个坏分量的边集获得适当构造。随后，我们将展示如何扩展适当构造 τ，使其成为 $C \cup D$ 的适当构造，其中 C 和 D 是以某种方式相关的边集。我们将这一过程称为在 D 中扩展 τ。我们将分析 C 与 D 之间关系的几种典型情况。目标构造通过一系列添加新激活节点来实现，这些节点包含从 D 到 τ 的边。在大多数情况下，会将一个新激活节点与某个空节点一同添加。

为了确保最终的见证结构是 $C \cup D$ 的适当构造，我们遵循以下规则。每次添加时，只能将属于 D 且未被当前活跃边集覆盖的边作为新活跃节点添加。这确保了条件 (a) 和 (e) 得到满足。扩展会持续进行，直到 $D|_b$ 未被覆盖，最终满足条件 (d)。每次添加一组新节点后，我们都会检查见证结构的平衡是否降低，从而保持条件 (b)。最后，为确保满足条件 (c)，对于每个新活跃节点，我们都会检查相应的基本事件在该节点添加时是否成立。每次添加时，我们还必须确保之前插入的活跃节点所对应的基本事件定义不被改变。标记为 (M)、(B) 和 (U) 的节点所对应的事件具有自描述性，不会发生改变。对于标记为 (B_{in}) 和 (B_{out}) 的节点，则必须特别谨慎。为避免改变其定义，我们遵循以下规则：

(f) 当一个新的活跃节点 x 被添加到 τ时， x 在 τ 中的深度必须大于标记为(B_{in}) 或 (B_{out}) 且对应于距离 2 以内边的节点的深度 f_x。

该规则确保，一旦将标签为（B_{in}）或（B_{out}）的节点 x 添加到 τ 中，顶点集 $f_x \cap R_{<d(x)}$（τ）在之后对 τ 的扩展中不会发生变化。

在以下证明中，我们遵循上述规则。因此，我们只明确说明平衡不会降低，且添加新活跃节点的方式符合规则 (f)。我们还会检查分配给新添加的活跃节点的基本事件是否得到满足；不过，对于坏边和单色边，无需赘述，它们分别可以标记为 (B) 和 (M)。

D.2.2 基本技术

▶**定义 25**（添加一条边）。对于见证结构 τ 和一条与 E（τ） 中某条边相邻的边 g，将 g 添加到 τ，使其成为 （ X ） 指的是以下步骤。设 y 是 τ 中的一个节点，使得 f_y 与 g 相邻。我们添加一个新节点 x，令 $f_x = g$，将其标记为 （X），并在 τ 的树中添加一条连接 x 和 y 的弧。默认情况下，我们将该弧指向 y。

▷ 引理 26（坏分量的适当构造）。令 B 为某个坏分量的边集。则存在一种适用于 B的适当构造。

证明。我们从标记为 (B) 的单个节点 τ 开始，并为其分配一条边 $e \in B$。然后，只要 B 未被 $A(\tau)$ 覆盖，我们就从 B 中任取一条与 $A(\tau)$ 的距离为 2 的边 f。这样的边必定存在，因为 B 中未被 $A(\tau)$ 覆盖的边与该集合的距离至少为 2， $e \in B$ 到 $A(\tau)$ 的距离为 0，且 B 是连通的。边 f 与 τ 的距离至多为 2。以下构造确保分配给空节点的边与活跃边相邻，因此 f 不属于 $E(\tau)$。于是，我们可以考虑两种情况。如果 f 与 τ 相邻，我们就将其作为 (B) 添加到 τ 中。否则， f 与 τ 的距离为 2。在这种情况下，设 g 是从 f 到 τ 的长度为 2 的路径上的一条边。我们将 g 作为 (E) 添加到 τ 中；随后 f 与 τ 相邻，我们就将其作为 (B) 添加到 τ 中。

设置第一个节点（用于 e）后， τ 的初始平衡为 1。扩展 τ 的方式不会降低该平衡，因此满足条件 (b)。所有活跃节点都已标记（B），且只将属于 B 的坏边分配给这些节点，因此满足条件 (c) 和 (e)。将一条边作为新的活跃节点添加时，它与 A 的距离为 2（τ），因此满足条件 (a)。构造最终停止时，也满足条件 (d)。◁

▶**定义 27** (2-可达性)。对于见证结构 τ 和边 g，当 g 与 $A(\tau)$不相交，且 τ 中存在某个节点 x，使得两者之间的距离

为 g 和 f_x 之间的距离为2时，我们称 g从 是 *2*-可达的。在这种情况下，我们所说的“从 g可2-到达 τ 的最深节点”，是指 τ 中深度最大的节点 x，且 f_x 与 g之间的距离为2 。如果同一深度上有多个这样的节点，我们可以任选其中一个。

注意，当某个 g 未被 $A(\tau) \neq \emptyset$覆盖，但它到 τ 的距离至多为 2 时， τ 从 g是 2-可达的。这是因为， $E(\tau)$ 中有一条边到 g的距离至多为 2， $A(\tau) \subset E(\tau)$ 中有一条边到 g的距离至少为 2，并且$E(\tau)$是连通的。

▶ **定义 28**（附加一条边）。 对于一条边 g 和一个从 g2可达的见证结构 τ，我们称将 g 附加到 τ 作为 （X）进行如下操作。设 $x \in \tau$ 是 τ 中从 g2可达的最深节点， h 是一条位于从 g 到 f_x 的最短路径上的边。我们将 h 作为空节点添加到 τ 中（即使 h 已分配给此前添加的某个节点），并使其朝向 x连接。然后，我们将 g 添加到 τ 中，并用（X）为其标记。

假设 (X) 既不是 (E)，也不是 (U)，则 τ 的平衡不会降低。此操作遵循规则 (f)，因此不会改变在 g 之前添加到 τ 的活跃节点的基本事件。它还保留了在将 g 添加到 τ 时，关于 g 的哪些顶点位于 $R(\tau)$ 中的信息。我们在以下断言中表述这一性质。

▷ 断言 29（附加保留覆盖性）。对于边 g 和一个可从 g 出发 2-可达的见证结构 τ，令 τ' 表示将 g 附加到 τ 后得到的见证结构。那么，$g \cap R(\tau) = g \cap R_{<d(z)}(\tau')$，其中 z 表示 τ' 中对应于 g 的新添加节点。

证明。 根据 2-可达的定义， g 到 $A(\tau)$ 的距离至少为 2。尽管集合$R(\tau)$由 τ的所有活跃边定义，但当我们考察它在 g上的限制时，只有距离 2 以内的那些边会产生影响，这些边距 g。根据 g 附加到τ的方式定义，对应于这些活跃边的所有节点，其深度都小于新添加的 z，因此这些边属于 $A_{<d(z)}(\tau')$。 ◁

接下来，我们将说明如何扩展适当构造 τ，使其覆盖与 τ距离较近的坏分量的边。为简单起见，在此处我们将坏分量等同于其边集。

▷ 命题30（坏分量内的扩展）。设 τ 是边集 C 的一个适当构造。设 B 是某个坏分量的边集，且该坏分量与 τ 的距离至多为 2。则 τ 可以扩展为 $C \cup B$ 的一个适当构造，同时不降低平衡。

证明。当 B 未被 $A(\tau)$覆盖时，可以保证存在 $f \in B$，使得τ 是 2-可达的。只要这一条件成立，我们就任取一个这样的 f 并将其作为 τ 的一部分附加上去，即 (B)。此操作不会降低平衡，且符合规则 (f)。存在这样一条边，是因为 B 是连通的，并且某个 $g \in B$ 与 τ之间的初始距离至多为2。◁

设 C 为一组边， B 为 C 之外某个坏分量的一组边（即 $C \cap B = \emptyset$），并且存在一条不安全边 f，同时与 $C|_b$ 和 B相交，且$f|_t \subset V(C|_b) \cup V(B)$。当$f|_a \cup (f|_t \cap V(C|_b))$为单色的且大小至少为$(1-\alpha)k$时，我们称 f 可以从 B 激活 C。当$f|_a \cup (f|_t \setminus V(C|_b))$为单色的且大小至少为$(1-\alpha)k$时，我们称 f 可以从 C 激活 B 。这些定义对应于搜索区域的扩展方式，以及构造最终分量时的扩展规则（r2）。

32 超图着色的局部计算算法——遵循 Beck 方法

▷ 断言31（遵循激活的潜在原因）。设 τ 为边集 C的一个适当构造， B 为一个坏分量的边集，并且存在一条不安全边 f ，它可能使 C 从 B激活。那么，在不降低平衡的情况下，τ 可以扩展为$C \cup \{f\} \cup B$ 的一个适当构造。

证明。若 B 与 τ 的距离至多为 2，则可根据命题30扩展 τ ，从而为 $C \cup B$ 得到适当构造。注意，由于 f 是一条不安全边，为 $C \cup B$ 得到适当构造就足以为 $C \cup \{f\} \cup B$ 得到适当构造。否则，B 的所有边与 τ 的距离至少为 3。特别地， $R(\tau) \cap V(B) = \emptyset$。另一方面，由于条件 (d)，$V(C|_b) \subset R(\tau)$。由于 f 与 $V(C|_b)$ 和 $V(B)$ 相交，因此它与 τ 的距离恰为 2。根据假设， f 可以从 B 激活 C，所以 $f|_a \cup (f|_t \cap V(C|_b))$ 是单色的。由于 f 中每个不属于 $V(C|_b)$ 的受扰顶点都必须位于 $V(B)$ 中，而后者与 $R(\tau)$ 不相交，因此 $f|_t \cap V(C|_b) = f|_t \cap R(\tau)$。这使得可以按照 (B_{in}) 将 f 附加到 τ。根据命题29，将 f 添加到 τ 后，相应事件便得到满足，因此我们得到 $C \cup \{f\}$ 的适当构造。接着，可以使用命题30在 B 内对其进行扩展，从而得到所需的 $C \cup \{f\} \cup B$ 的适当构造。根据命题30附加边和进行扩展都不会降低 τ 的平衡。◁

▷ 断言32（沿着一次激活）。设 τ 是边集 C的一个适当构造，且 B 是一个坏分量的边集，并且存在一条不安全边f ，它可能从 C激活 B 。那么， τ 可以扩展为$C \cup \{f\} \cup B$ 的一个适当构造，且不降低平衡。

证明。 我们仿照断言31的证明进行。唯一的变化是，当 B与3 的距离至少为 τ时，我们将 f 附加到 τ 上，采用(B_{out}) 而不是(B_{in})。◁

▷ 断言33（摊销配置的应用）。设 τ 是边集 C的一个适当构造。设 f 是一条与 τ的距离至多为2的边，且它要么是单色的，要么与至少两条未被 $A(\tau)$覆盖的不相交坏边相交。令 S 表示那些未包含在 C中、但与 f相交的坏分量的边集之并。那么， τ可以扩展为 $C \cup \{f\} \cup S$ 的一个适当构造，且不降低平衡。

证明。 当 f 与 τ相交时， τ 可以在包含于S 中的每个坏分量内，通过命题30进行扩展（因为这些分量随后与 2 的距离至多为 τ）。因此，我们可以假设 f与 τ的距离为 2 τ，所以 τ从 f是2-可达的。

在第一种情况下，当 f 是单色的时，我们将其作为 (M) 附加到 τ ，从而为 $C \cup \{f\}$ 获得一个适当构造。然后，根据命题30在 S 内扩展它。在第二种情况下， f 与至少两条不相交的坏边相交。将它们记为 g_1 和 g_2。根据假设，它们不被 A（τ）覆盖，并且属于 S。如果任何 g_i 与 τ 的距离至多为 2，那么 τ 可从其 2- 可达。在这种情况下，我们将 g_i 作为 (B) 附加到 τ ，从而为 $C \cup \{g_i\}$ 获得一个适当构造。结果， f 与 A（τ）相交，因此包含在 S 中的所有坏分量现在与 τ 的距离至多为 2。因此，对每个坏分量应用命题30，我们在 S 内扩展 τ。如果 g_1 和 g_2 与 τ 的距离都至少为 3，我们将 f 作为 (E) 附加到 τ。然后，我们将 g_1 和 g_2 作为 (B) 添加到 τ ，将它们朝向对应于边 f 的新添加节点连接。这样，平衡得以保持，并且我们遵守规则 (f)。随后， τ 是 $C \cup \{f, g_1, g_2\}$ 的一个适当构造。再次，我们充分多次应用命题30，在 S 内扩展 τ。在每种情况下，我们都为 $C \cup \{f\} \cup S$ 获得一个适当构造。

◁

D.2.3 搜索区域的构造

考虑由某条边 e 的 EXPAND_OR_ACCEPT(e) 完整执行过程所构造的搜索区域（回想一下，为了分析方便，我们不声明失败，而是继续执行该过程）。我们可以将其表示为 $A = C_1 \cup C_2 \cup ... \cup C_t \cup F$，其中

- C_i 是使搜索区域得以扩展的第 i个坏分量的边集，t 是此类扩展的次数，
- F 表示在找到摊销配置 (e2) 或 (e3) 时添加到 A 中的坏边。更具体地说，集合 C_i 对应于通过 EXPAND_VIA_UNSAFE获得的连续集合 C，跳过返回空集的调用。当未找到摊销配置或出现 (e1) 时，集合 F 为空。请注意，(e2) 情况下也可能如此。回想一下，当 F非空时，它由某些坏分量的边组成。设 $U = \{f_1, f_2, ..., f_t\}$为满足以下条件的不安全边集： f_i 触发了搜索区域向 C_i 的扩展（特别是 $f_1 = e$）。对于找到摊销配置的情况，令 f_F 表示使摊销配置成为可能的边。它与 C_t 相交，并且满足 (e1)、(e2) 或 (e3) 之一。在这种情况下，令 $C_F = C_t \cup \{f_F\} \cup F$。当未找到摊销配置时， f_F 未定义。令 υ_F 表示当 F 为空时的 $\emptyset$，而当其不为空时，令 $\upsilon_F = \{f_F\}$。然后，我们将 $U_A = U \setminus f_1\} \cup \upsilon_F$ 定义为将搜索区域从 C_1 扩展到 A的不安全边集。

接下来，我们将通过跟踪 EXPAND_OR_ACCEPT 的评估，说明如何为搜索区域构造适当构造。形式上，我们为 $A \cup U_A$ 构造适当构造。然后我们证明，实际上可以从任意 C_i 而非 C_1 开始，得到适当构造。最后我们论证，如果找到了摊销配置，就能得到平衡至少为 2 的适当构造。

▷ 命题 34（跟踪搜索区域的扩张）。设 $A_i = C_1 \cup C_2 \cup ... \cup C_i \cup \{f_2, ..., f_i\}$为第 i 次扩张后的搜索区域及其不安全边。对于每个 $i \leq t$，存在一个适当构造 τ_i，用于 A_i。

证明。 回顾一下，当调用 EXPAND_OR_ACCEPT(e) 时，边 e 仅与一个尚未探索的坏分量相交，而该分量由 C_1捕获。因此，对于 $i = 1$，令 τ_1 为依据引理 26 得到的、用于 $A_1 = C_1$ 的适当构造。对于归纳步骤，设 τ_i 为用于 A_i的适当构造。注意， f_{i+1} 可能会激活A_i，使其脱离 C_{i+1}。因此，我们可以应用断言31，将 τ 扩展为一个适当构造 τ_{i+1}，用于$A_{i+1} = A_i \cup \{f_{i+1}\} \cup C_{i+1}$。 ◁

▷ 断言35（搜索区域的适当构造）。存在一个针对$A \cup U_A$的适当构造。

证明。令 A_i 如断言35中所定义。于是， $A \cup U_A = A_t \cup \upsilon_F \cup F$。令 $\tau = \tau_t$ 为通过命题 34 得到的、针对 A_t 的适当构造。当 $F = \emptyset$ 时， τ 已经是针对 A 的适当构造。现在假设 F 非空，因此该过程因遇到摊销配置而停止。在这种情况下， f_F 在该处定义为一条不安全边，且满足 (e2) 或 (e3)（若满足 (e1)，则 F 为空）。 F 由所有与 f_F 相交且位于 A_t 之外的坏分量中的坏边组成。注意， f_F 与由 τ 覆盖的 C_t 相交，因此它到 τ 的距离至多为 2。注意， F 和 A_t 不相邻，并且由于 $A(\tau) \subset A_t$， F 未被 $A(\tau)$ 覆盖。这种情况满足断言33的假设，因此我们可以将 τ 扩展为针对 $A_t \cup \{f_F\} \cup F = A \cup U_A$ 的适当构造。◁

34 超图着色的局部计算算法——遵循 Beck 方法

断言35证明了在搜索区域这一情形下的命题 23。本节其余部分将介绍搜索区域的构造的一些其他性质。我们重点讨论这样的情形：搜索区域旨在并入最终分量，也就是说，找到了摊销配置。我们将说明如何利用这种配置。

来看分量超图，并关注对应于坏分量 $V_A = \{C_1, C_2, ..., C_{t-1}, C_t\}$ 的顶点，以及对应于不安全边 $\{f_2, ..., f_t\}$ 的边 $E_A = \{F_2, ..., F_t\}$。回想一下，每条 f_i 只与 $\{C_1, C_2, ..., C_{t-1}, C_t\}$ 中的两个坏分量相交。其中一个是这样的 C_j：将 f_i 添加到 Q 后得到它——我们用 $C_{Q(i)}$ 表示。另一个是 C_i。因此，$T_A = (V_A, E_A)$ 实际上是一棵树。此外，我们将每条 F_i 从 $C_{Q(i)}$ 指向 C_i。

前一个论断中得到的 A 适当构造，其构造顺序由算法访问分量 C_i 的顺序决定。这个顺序可以看作沿着 T_A 遍历，方向与边 F_i 的朝向一致，同时保持已访问顶点集合连通。然而，事实证明，我们可以在搜索区域内扩展适当构造：从某个 C_i 出发，以任何能保持已访问坏分量集合连通的顺序遍历 T_A。特别是，我们可以允许按与 T_A 中边的方向相反的顺序遍历这些边。

▷ 断言36（搜索区域内的扩展）。令 τ 为边集 C 的适当构造，使得对每个 C_i，满足 $C_i \subset C$ 或 $C_i \cap C = \emptyset$，但对至少一个 $s \le t$，满足 $C_s \subset C$。此外，我们要求满足 $F \subset C$，或者 F 中没有任何边被 $A(\tau)$ 覆盖。于是，τ 可以扩展为 $C \cup A \cup U_A$ 的适当构造，且平衡不会降低。

证明。我们从 C_s 开始，并按任意顺序遍历 T_A，同时保持已访问的坏分量集合连通（忽略边的方向）。遍历T_A 的同时，我们以类似命题 34 的方式，用已访问的坏分量扩展适当构造 τ。

令 C' 为初始集合 C 与截至目前已处理的集合 C_i 的并集。当我们通过边 F_l 从 C_i 移动到 C_j时，首先检查 $C_j \subset C$是否成立。若成立，则无需进行任何操作。否则，C_j 位于 C'之外。如果从 C_i 到 C_j 的方向与 F_l 的定向一致（即，当 $i < j$ 和 $f_l = f_j$时），则 f_l 可以从 C_j 激活 C'，并应用断言31来扩展 τ。当我们沿与 F_l 相反的方向移动时（即，当 $i > j$和 $f_l = f_i$时），则 f_l 可以从 C' 激活 C_j，并应用断言32来扩展 τ。

当遍历 T_A 时，τ 是 $C \cup A_t$ 的适当构造。如果初始时 $F = \emptyset$ 或 $F \subset C$，那么我们就完成了。否则，我们可以按照断言35中的相同方式（即，通过应用断言33）在 $\{f_F\} \cup F$ 内扩展 τ，从而为整个 $C \cup A \cup U_A$ 获得一个适当构造。◁

▷ 断言37（摊销配置的适当构造）。存在一个适当构造 τ_F，其 C_F 平衡至少为 2，并且 f_F 被 $A(\tau_F)$覆盖。此外，τ_F 中的所有活跃节点均标记为 (B) 或 (M)。

证明。当搜索区域包含摊销配置时，f_F 有定义。设 F可能为空，不过我们的证明也涵盖这种情况。回想一下，$C_F = C_t \cup \{f_F\} \cup F$。注意，$f_F$ 与 C_t 和 F（当它非空时）都相交，因此 C_F 是连通的。

首先考虑 f_F 满足 (e1) 或 (e2) 的情况，即它是一条单色边。我们从一个标记为 (M) 的单节点开始构造 τ，并为其分配边 f_F。其初始平衡为 2。然后，我们可以按照引理 26 中的相同方式，将 τ 扩展为针对 C_F 的适当构造。所得的 τ 的平衡至少与初始平衡相同，因此满足要求，至少为 2。显然，f_F 被 $A(\tau_F)$ 覆盖。

现在考虑 f_F 满足 (e3) 的情况。它必须与 F 中至少 2 条不相交的坏边相交。此外，它还与 C_t 相交，因此与 f_F 相邻的不相交坏边至少有 3 条。将它们记为 e_1、e_2 和 e_3。我们可以从一个标记为 (*E*) 的单个节点开始构造 τ，并将 f_F 分配给它。然后，我们将 e_1、e_2 和 e_3 添加到 τ 中，并将它们都标记为 (*B*)。完成这些初始步骤后，τ 的平衡为 2，而 f_F 被 A（τ_F）覆盖。现在，与第一种情况一样，我们以与引理 26 相同的方式，将 τ 扩展为针对 C_F 的适当构造，且不降低平衡。◁

▶**推论38**。 对于包含摊销配置的搜索区域，存在一种平衡至少为 2 的适当构造。

证明。 这是断言37和断言36的推论。前者允许获得 τ，这是对 $C_F = C_t \cup F \cup \{f_F\}$ 的适当构造，并且其平衡至少为 2。然后，根据断言36，可以将其扩展到整个搜索区域，而不会降低初始平衡。◀

▷ 断言39（纳入摊销配置）。设 τ 是边集 C的适当构造，使得 C_F位于距 τ至多为 2 的距离内，并且 $C_t \cup F$中的任何边都未被 $A(\tau)$覆盖。则，τ可以扩展为 $C \cup C_F$的适当构造，且不会降低平衡。

证明。当 f_F 与 τ 相交时，C_t 以及包含在 F 中的所有坏分量到 τ 的距离至多为 2。然后，对每个这样的坏分量应用命题30，可以将 τ 扩展为 $C \cup C_F$ 的一个适当构造。现在假设 f_F 与 τ 不相交。令 τ_F 为通过断言37获得的 C_F 的适当构造。注意，根据假设，来自 $A(\tau)$ 的边与来自 $A(\tau_F)$ 的边不相交。由于 C_F 到 τ 的距离至多为 2，且 $A(\tau_F)$ 覆盖了 C_F（包括 f_F），因此 τ 与 $A(\tau_F)$ 之间的距离至多为 3。

令 x 为 τ 中最深的节点，使得对应的边 f_x 与 A（τ_F）的距离至少为 1 且至多为 3。令 p 为从 f_x 到 A（τ_F）的最短路径，并令 z 为 τ_F 中的一个节点，使得 f_z 位于该路径的末端。从 f_x 沿着 p 移动到 f_z 时，对于每条内部边，我们将其作为 (E) 加入 τ，并朝向 p 中与其前驱对应的节点。之后，我们重新定向 τ_F 中的所有弧，使得 z 成为唯一的根。重新定向边不会改变基本事件的定义，因为 τ_F 中的所有激活节点都标记为 (B) 或 (M)。最后，我们通过添加一条从 z 指向 τ 中某个节点 y 的弧，将修改后的 τ_F 并入 τ；该节点满足 f_y 是 p 中 f_z 的前驱。以这种方式将 τ_F 连接到 τ 符合规则 (f)，因此所得的 τ 满足所有基本事件。由于 p 的长度至多为 3，最多会向 τ 中添加 2 个空节点。它们的加入由 τ_F 的平衡所抵消，因此 τ 的平衡不会降低。◁

D.2.4 最后一个分量的构造

这是命题 23 证明的最后一步，同时完成命题 10 的证明。我们展示如何为通过对 `BUILD_FINAL_COMPONENT`(v) 进行完整求值而获得的最终分量构建一个适当构造。形式上，这是针对 $B \cup U_e$ 的一个适当构造，其中 B 是最终分量的坏边集合，而 U_e 是导致最终分量发生扩展的不安全边集合（它既包含触发扩展规则的边，也包含用于扩展添加到最终分量中的搜索区域的边）。

我们从包含 v 的初始坏分量的适当构造 τ 出发。通过应用引理 26 得到该构造。然后我们遵循扩展规则，并据此扩展 τ。

36 超图着色的 LOCAL 计算算法——遵循 Beck 方法

设 B' 是扩展最终分量时其中的坏边集合。设 U'_e 是迄今为止导致扩张的不安全边集合。假设 τ 是 $B' \cup U'_e$ 的适当构造。设 f 是一条满足某条扩展规则的不安全边。注意，它与 τ 的距离至多为 2，因为它与 B' 相交。

▷ 断言40（遵循规则（r1））。设 f 满足扩展规则（r1），且 S 为属于被 f相交的外部坏分量的边集。那么， τ 可以扩展为针对 $B' \cup U'_e \cup \{f\} \cup S$的适当构造。

证明。 对于（r1）， f <style id='6}至少与两条彼此不相交且不与$B' \cup U'_e$相交的坏边相交 $A(\tau)$，因此它们未被覆盖。这种情况与在 EXPAND_OR_ACCEPT中找到摊销配置（e3）相对应。因此，我们可以应用断言33，在 τ 内将其扩展为$\{f\} \cup S$。◁

▷ 断言41（遵循规则（r2））。设 f 满足扩展规则（r2），且 C 为被 f相交的、位于 B' 之外的唯一坏分量中的坏边集。那么， τ 可以扩展为针对 $B' \cup U'_e \cup \{f\} \cup C$的适当构造。

证明。 在 (r2) 的情况下，注意到 f 可能会激活 C，该激活来自 $B' \cup U'_e$。这种情况类似于断言36中所述的搜索区域扩展（当我们沿着与 C_i 到C_j 相反的方向，从 F_i移动时）。我们可以应用断言32，在 τ内扩展 $\{f\} \cup C$。◁

在进行条件扩展时，令 A 为由 EXPAND_OR_ACCEPT(f) 返回的集合。当 A 为空时，无需采取任何操作，因此我们着重讨论它非空的情形，也就是说，找到了一个摊销配置。此时，扩展适当构造的过程会更加复杂。与上一节一样，我们使用形式 $A = C_1 \cup C_2 \cup ... \cup C_t \cup F$ 表示搜索区域，并用 $U_A = \{f_2, ..., f_t\} \cup \upsilon_F$表示触发连续扩展的不安全边集合。每个 f_i 都从 $C_{Q(i)}$ 指向 C_i，而 f_F 表示使摊销配置得以成立的那条边。我们还使用 $C_F = C_t \cup \{f_F\} \cup F$。

▷ 命题 42（遵循规则（r3））。设 f 满足扩展规则（r3），且 A 为由 EXPAND_OR_ACCEPT(f) 返回的搜索区域，其中包含一个摊销配置。那么， τ可以扩展为 $B' \cup U'_e \cup \{f\} \cup A \cup U_A$的适当构造。

证明。 根据 A 与 τ 之间的距离，我们区分两种情况。假设A 或 U_A 与 2 的距离至多为 τ。然后，我们在至少一个 τ 内按以下方式扩展 C_i：

- 如果 C_F 与 τ 的距离至多为 2，则我们利用断言39将 τ 扩展为$B' \cup U'_e \cup C_F$ 的适当构造；否则，如果某个 C_i 与 τ 的距离至多为 2，则我们利用命题30将 τ 扩展为 $B' \cup U'_e \cup C_i$ 的适当构造；否则，存在某个 f_i 与 τ 的距离至多为 2，且$C_{Q(i)}$ 和 C_i 中的任何一条边都未被 $A(\tau)$覆盖；在这种情况下， f_i 与至少两条不相交的坏边相交

（一个在 $C_{Q(i)}$，另一个在 C_i），因此我们使用断言33，将 τ扩展到$\{f_i\} \cup C_{Q(i)} \cup C_i$内。完成这一步后， τ 满足断言36的假设，因此我们将其扩展至覆盖整个搜索区域。

现在假设 A 和 U_A 中的所有边与 τ 的距离都至少为 3。在这种情况下，f 与 τ 的距离必须为 2。令 p 为一条从 B' 到 A 的路径 $e_1 \in B'$，f，$e_2 \in A$。注意， $e_1 \notin E(\tau)$。令 τ_A 为 $A \cup U_A$ 的一个适当构造，其平衡至少为 2，由推论 38 得到。由于 e_2 被 τ_A 覆盖， f 与 τ_A 的距离至多为 2。

如果 f 与 A（τ_A）相交，那么令 x 为 τ_A 中一个激活节点，使得 f 与 f_x 相邻。我们将 e_1 作为空节点加入 τ。随后， f 与 τ 相邻，因此我们将其作为空节点 y 加入 τ。最后，我们通过添加一条从 y 指向 x 的弧，将 τ_A 纳入 τ。

在剩余的情形中，当 f 与两个构造中的所有活跃边都不相交时，注意到 τ 和 τ_A 都从 f 出发是2-可达的。现在观察到， f 在 B 中的问题顶点少于 αk ，而 A也是如此。由于 f 没有其他问题顶点，因此可以将其标记为（U）。我们为 f创建一个节点 x，并将其作为（U）连接到 τ。然后，通过将 f （使用同一个节点 x）连接到 τ_A，我们将τ_A 并入 τ。注意，每次连接都会插入一个空节点。

在这两种情形中， τ_A 中的超额平衡抵消了连接 τ_A和 τ所需的两个额外空节点，因此 τ 的正平衡得以保留。将 τ_A并入 τ后，激活节点的深度保持不变。由于$\tau_A \subset A \cup U_A$，且与 τ 的距离至少为3，因此并入 τ_A后，基本事件的定义不会发生改变。此外，在最后一种情形中，将 f 作为（U）添加进去，也不会改变基本事件的定义。这意味着最终， τ是 $B' \cup U'_e \cup \{f\} \cup A \cup U_A$ 的适当构造。$\lhd$