arXiv:2302.02290v1 [cs.DS] 5 Feb 2023

通过局部随机收缩求解带权图中的最大 k-边连通子图

Chaitanya Nalam* Thatchaphol Saranurak†

2023年2月7日

摘要

“最大 k-边连通子图 问题”是一个经典的图聚类问题，自 20 世纪 70 年代以来便受到研究。令人惊讶的是，目前尚无针对带权图中这一问题的非平凡技术：一种非常直接的递归最小割算法，运行时间为 $\Omega(mn)$，直到现在仍是最快的算法。此前所有进展都只能在图为无权图且 k 足够小时实现加速（例如 Henzinger 等人（ICALP’15）、Chechik 等人（SODA’17）和 Forster 等人（SODA’20）的工作）。

我们提出了首个突破这一长期 $O(m\cdot n)$ 时间障碍，在带权无向图中。更具体地说，我们展示了一种极大 k-边连通子图算法，其运行时间仅为 $\tilde{O}(m\cdot\min\{m^{3/4},n^{4/5}\})$。作为直接应用，我们可以在相同的运行时间内，对无向图中所有边的强度进行 $(1+\epsilon)$ 近似。

我们的关键技术是首个具有精确割值保证的局部割算法，其运行时间仅取决于输出规模。此前所有局部割算法的运行时间要么取决于输出的割值——在带权图中，这可能导致运行时间任意长；要么只能提供近似割保证。

*密歇根大学，安娜堡。电子邮箱：nalamsai@umich.edu
†密歇根大学，安娜堡。电子邮箱：thsa@umich.edu

目录

1 引言 3 2 技术概述 6 3 预备知识 9 3.1 递归最小割算法.9

4 小体积割的局部算法 10
4.1 局部 Prim 算法.10 4.2 优美 秩函数.12 4.3 用于（ν， σ， k）-集合的局部子程序.13

5 $\tilde{O}$（$m^{1.75}$）最大 k-连通划分算法 15
5.1 使用 LocalKCut 的全局算法.15 5.2 运行时间分析.17 **6 局部算法**

小基数割的算法 19
6.1 排序邻接表.20 6.2 加速 LocalPrim .20

7 $\tilde{O}(mn^{0.8})$ 最大 k-连通划分算法 22
7.1 对 n .22 7.2 运行时间分析.22

8 条件下界 23
8.1 LocalKCut 的下界.23 8.2 LocalMST 的下界.25 8.3 SmallLocalSeparation 的下界.26 **9 边强度估计 28 A 秩函数及其保证 29 B 极小极端集的局部算法 31 C 最大 k-边连通划分的唯一性 33 D 边强度估计的反例 33**

1 引言

我们重新考察一个自然的图聚类问题：高效计算极大的 k-边连通子图。自 70 年代以来，这一问题便受到研究 [Tar72]。给定一个（有向）图 $G = (V, E)$，如果所有边割的值都至少为 k，我们称 G 是 k-边连通的；类似地，如果所有点割的值都至少为 k，我们称 G 是 k-顶点连通的。G 的极大 k-边连通子图包含所有满足下述条件的子集 $\{V_1, \ldots, V_z\}$：对于每个 i，诱导子图 $G[V_i]$ 是 k-边连通的，并且不存在严格超集 $V_i' \supset V_i$ 使得 $G[V_i']$ 是 k-边连通的。G 的极大 k-顶点连通子图定义类似。

事实证明，对于这些问题的所有变体（无论图是带权还是无权、有向还是无向），都存在一种非常简单的递归最小割算法：对于 k-边连通性版本，通过调用全局最小割子程序计算一个值小于 k 的边割（A, B）；如果不存在这样的割，则返回 $\{V\}$。否则，分别对 $G[A]$ 和 $G[B]$ 递归，并返回两次递归所得答案的并集。对于 k-顶点连通性版本，类似地计算一个值小于 k 的顶点割（A, S, B）（其中 S 将 A 与 B 分隔开，即 $E(A, B) = \emptyset$）。如果存在这样的割，则分别对 $G[A \cup S]$ 和 $G[B \cup S]$ 递归。该算法在一个有 n 个顶点和 m 条边的图上的最坏情况运行时间为 $\Theta(n \cdot T_{\text{mincut}}) = \Omega(mn)$，其中 $T_{\text{mincut}} = \Omega(m)$ 是计算最小割所需的时间。[1][2]

令人惊讶的是，上述直接的算法至今仍是最快的算法！尽管这个问题自 20 世纪 70 年代以来就一直受到研究，但所有已知算法只有在图为无权图且 k 足够小时才能实现加速。经典的深度优先搜索算法 [Tar72]可在 $O(m)$ 时间内，求解无向无权图中的极大 2-边连通子图和 2-顶点连通子图问题。研究者曾尝试将这一结果推广至有向图[ES80、Mak88、Jab16], ，但这些算法的运行时间仍为 $\Omega(mn)$。随后，Henzinger 等人 [HKL15] 在 $k = 2$ 时，提出了适用于有向图的 $\tilde{O}(n^2)$ 时间算法。接着，Chechik 等人 [CHI+17]给出了首个突破 $O(mn)$ 界的算法，其中 k 可为任意常数。对于问题的所有变体，他们的算法运行时间均为 $\tilde{O}(m\sqrt{mk^{O(k)}})$。他们还将无向图中极大 k-边连通子图的计算时间改进至 $\tilde{O}(m\sqrt{nk^{O(k)}})$。随后，Forster 等人 [FNY+20] 将这两个界分别改进为 $\tilde{O}(m\sqrt{mk^{3/2}})$ 和 $\tilde{O}(mk + n^{3/2}k^3)$。尽管如此，对于 k 可能任意大的加权图，甚至对于 $k \geq m^{1/3}/n^{1/6}$的无权图，简单的递归最小割算法至今仍是最快的算法。

我们的结果。我们给出了首个突破递归最小割算法长期以来 $O(mn)$-时间瓶颈的算法，适用于加权图中的极大 k-边连通子图，研究对象为无向图。这一问题变体在实践中备受关注，也已有多种启发式方法被提出 [ZLY+12, AIY13]。形式化地说，我们的主要结果如下：

定理 1.1。存在一种随机化算法，给定带权无向图 $G = (V, E, w)$，其中 $w \in \mathbb{Z}_{\geq 0}^E$ 有 n 个顶点和 m 条边，以及任意参数 $k > 0$，该算法可在 G 中以高概率于 $O(m \cdot \min\{m^{3/4} \log^{3.75} n, n^{4/5} \log^{3.6} n\})$ 时间内计算出极大的 k-边连通子图。

[1]Karger证明了 [Kar00] 在无向图中计算最小边割需要 $T_{\text{mincut}} = \tilde{O}(m)$，但其他变体的最先进结果所需时间更长。对于有向图中的最小边割，$T_{\text{mincut}} = \widehat{O}(m \cdot \min\{n/m^{1/3}, \sqrt{n}\})$ [CLN+22]，而当图为无权图时则为 $T_{\text{mincut}} = \tilde{O}(mk)$ [Gab95]。对于最小点割，$T_{\text{mincut}} = O(mn)$ [HRG00] 适用于无向图和有向图；若图为无权图，则无向图的结果为 $T_{\text{mincut}} = \widehat{O}(m)$ [LNP+21]，有向图的结果为 $T_{\text{mincut}} = \min\{\widehat{O}(n^2)\ \tilde{O}(mk^2)\}$ [LNP+21, FNY+20]。[2]全文中，我们用 $\tilde{O}(\cdot)$隐藏一个polylog(n)因子，并用 $\widehat{O}(\cdot)$隐藏 $n^{o(1)}$ 因子。

众所周知，解 $\{V_1, \ldots, V_z\}$（G的极大 k-边连通子图）是唯一的，并构成 V的一个划分。此外，它可以精确判定一条边的边强度是否至少为 k：边 $(u, v) \in E$ 的边强度至少为 k，当且仅当 u 和 v 都位于同一个子集$V_i \in \{V_1, \ldots, V_z\}$内。Benczur 和 Karger [BK02] 提出的边强度是图稀疏化的核心概念，其推广形式见 [FHHP19, CKN20, CCPS21]。 [BK02] 给出了一种近线性时间算法，可对边强度进行低估。尽管这些估计足以用于图稀疏化 [BK02, CX18, FHHP19], ，但它们无法近似每条边的边强度；某些边的估计值与其边强度之间甚至可能相差 $\Omega(n^{1/3})$ 倍。相关讨论见附录 D。

目前计算精确边强度最快的已知算法仍是递归 Mincut˜算法，其运行时间为 $O(mn$。即使允许较大的近似比，也没有已知的更快算法。相比之下，定理 1.1 通过给出一个（$1+\varepsilon$）近似算法，改进了边强度近似方面的当前最佳结果。

推论 1.2。 存在一种随机化算法，给定一个带权无向图，可以以高概率在$(1+\epsilon)$时间内近似计算图中所有边的强度，近似比为 $\tilde{O}(m \cdot \min\{m^{3/4}, n^{4/5}\}/\epsilon)$。

主要技术贡献：带权图中的局部精确算法。 定理 1.1 背后的关键工具是首个局部割算法，它在带权图中既能提供精确的割值保证，运行速度也很快。

我们通常将局部割算法称为这样一类算法：给定一个种子顶点 x，找到一个包含 x 的割S，使其具有理想性质，但运行时间仅与 S 的大小相近，且与整个图的大小无关。例如，一系列颇具影响力的研究致力于设计用于寻找稀疏割的局部割算法（例如 [ST13、ACL06、OZ14、HRW20]），并在多个领域得到广泛应用，包括求解线性系统 [ST14] 和动态图算法 [NSWN17、SW19]。遗憾的是，所有这些算法都只能提供近似保证。[3] 另一类算法源自性质测试领域，其用于检验是否存在一个包含种子 x 的小割 S，且其割大小至多为 k。这些算法的运行时间为 exp(k) [YI10、OR11、YI12]，并可改进至 poly(k) [GR97、PR02、FNY+20、NSY19]。尽管这些算法在性质测试之外也有一些有趣的应用 [FNY+20、CDK+21、JHSZ22], ，但它们不适用于带权图，因为 k 可能任意大。

总而言之，现有文献中的所有局部割算法要么是近似算法，要么运行时间与割值 k 成正比。我们的关键技术是一种能够避开这两项缺陷的算法。

为介绍我们的关键工具，我们需要先说明一些记号。从现在起，我们考虑的所有图 $G = (V, E, w)$ 都是无向图。对于任意边集 $F \subseteq E$，$w(F)$ 表示 F 的总权重。对于任意集合 $X \subset V$，我们记作 $X = V \setminus X$。

令 $\delta(X) = E(X, \overline{X})$ 表示 X 的割集，而 $w(\delta(X))$ 表示 X 的割值。X 的（无权）体积记为 vol(X) $= |E(X, V)|$，表示与 X 相连的所有边的数量。若对于每个真子集 $Y \subset X$，$w(\delta(Y)) > w(\delta(X))$，则称 X 为极端集。等价地，在将 X 收缩为单个顶点后得到的图 G/X 中，X 是唯一的最小割。

定义 1.3. 对于任意顶点 $x \in V$，我们称 X 为一个(x, ν, σ, k)-集，如果 $x \in X$，$\mathrm{vol}(X) < \nu$，$|X| < \sigma$，且 $w(\delta(X)) < k$。如果 X 还是一个极端集，那么 X 称为一个(x, ν, σ, k)-极端集。

直观而言，(x, ν, σ, k)-集合是一个包含我们想要找到的种子顶点 x 的“小型局部割”。从现在起，我们用 ν 表示小型局部割的体积，用 σ 表示

[3]如果存在一个 割 S，其导通率为 ϕ，则这些算法会找到一个割 S'，其导通率为 ϕ'，其中 $\phi' > \phi$。

局部割较小一侧的顶点数量。我们要解决的关键问题如下：

问题 1.4 (LocalKCut)。给定一个图 $G=(V,E,w)$、一个顶点 x以及参数 ν、σ和k，要么找到一个 (x,ν,σ,k)-集，要么返回 $\perp$，表示不存在(x,ν,σ,k)-极端集。

注意，当不存在 (x,ν,σ,k)-极端集，但存在一个 (x,ν,σ,k)-集时，我们允许返回 $\perp$或任意一个 (x,ν,σ,k)-集。现在，我们的关键技术贡献如下：

定理 1.5。存在满足以下保证的随机化算法，且这些保证以高概率成立：

1. 给定对邻接表 G 的访问权限，问题 *1.4* 可在 $O(\nu\sigma^2\log^2 n)$时间内解决。
2. 在 $O(m\sigma^2\log^2 n)$ 预处理时间后，针对 G，问题 *1.4* 可在 $O(\sigma^4\log n)$时间内解决。

定理 1.5 最重要的保证是，其运行时间与 k 无关。定理 1.5 提供了精确的割大小保证，即 k 中不存在间隙：它要么返回一个 (x,ν,σ,k)-集，要么报告不存在 (x,ν,σ,k)-极端集。可以将定理 1.5 的保证加强为“要么返回一个 (x,ν,σ,k)-极端集，要么报告不存在 (x,ν,σ,k)-极端集”。我们在附录 B 中给出了这样一种算法。不过，该算法速度较慢，因此我们不会用它来证明定理 1.1。故将其推迟到附录中介绍。

我们用条件下界来补充这一算法：

定理 1.6。给定参数 x、ν、σ、k，判断是否存在一个 $(x$、ν、σ、$k)$-集是 *W[1]*-难的，即使以 σ 为参数。进一步地，假设 *OMv* 猜想 *[HKNS15]*，成立，那么不存在任何算法能够在获得 G 的邻接表和矩阵（不进行任何额外预处理）的情况下，在 问题 *1.4* 上实现 $O(\sigma^{2-\Omega(1)})$ 时间复杂度。

定理 1.6的第一项表述指出，问题 1.4中极端集这一概念对于高效算法至关重要；如果我们只想判断是否存在一个 $(x,\ \nu,\ \sigma,\ k)$-集，那么很可能不存在运行时间为 $f(\sigma)n^{O(1)}$ 的算法，其中 f 是任意函数（参见[1]CFK [第 13.3 节中 W^+15]-困难性的定义）。

定理 1.6的第二项表述表明，定理 1.5的运行时间无法进一步改进，除非 $O(\sigma^{2-o(1)})$，前提是我们只能访问邻接表和矩阵。若将定理 1.5的运行时间改进至 $\tilde{O}(\sigma^2)$，甚至在进行非平凡的预处理后达到 $\tilde{O}(\sigma)$，那么定理 1.1的运行时间将分别改进至 $\tilde{O}(mn^{2/3})$ 和 $\tilde{O}(mn^{1/2})$。我们将此留作一个非常有趣的开放问题。

关键技术：局部随机收缩。定理 1.5背后的核心思想如下：著名的随机收缩技术由 [Kar93]提出，用于计算全局最小割；它可以局部化为局部割算法，用于检测带权图中的不平衡极端集合。

Goldreich 和 Ron 的性质测试算法中，早先已出现将随机收缩技术应用于局部割算法的做法 [GR97]，此外 [PR02]中也有类似做法。然而，事先并不清楚他们的结果能否推广到带权图。这是因为在 [GR97、PR02]的设定中，(1) 图是无权且度有界的；(2) 他们要寻找的目标割满足 X $|\delta(X)|<k$ 且 $|X|=O(k)$；(3) 他们给出的运行时间为 poly(k)。因此，时间界本质上可能取决于 k。GR97] 中的部分论证关键地利用了图无权这一条件，因而无法延续到我们的带权设定中。我们的观察是，“局部随机收缩”技术在带权图中仍然有效，且运行时间界不依赖于 k。

这展现了随机收缩技术一个令人振奋的特点：据我们所知，目前文献中其他所有具有精确割值保证的局部割算法，其运行时间都不可避免地依赖于 k。这些技术包括采用确定性分支策略的局部搜索算法 [YI10、OR11、CHI+17]，采用随机停止点的局部搜索算法 [FNY+20]，甚至还有 Goldberg-Rao 流算法的局部化版本 [OZ14、NSY19]。

我们将定理 1.5视为迈向更强的带权图局部割算法的第一步，这些算法能够提供精确割值保证。鉴于这些算法的强大保证，我们期待它们会有更多应用。例如，若能加强定理 1.5 使其适用于有向图，就能立即将我们的主要结果定理 1.1推广到有向图。

相关问题。计算图$G=(V,E)$的 k-边连通连通分量这一相关问题，旨在找到 V的唯一划分 $\{V'_1,\dots,V'_{z'}\}$，使得对于同一部分 V'_i中的任意顶点对 (s,t)，G 中（而非 $G[V'_i]$中）的 (s,t)-最小割至少为 k。最大k-边连通子图 $\{V_1,\dots,V_z\}$ 可能不同于 k-边连通连通分量$\{V'_1,\dots,V'_z\}$。例如，考虑一个无权无向图 G'，其中顶点 s 和 t 通过三条并行路径 (s,u_1,t)、(s,u_2,t) 和 (s,u_3,t) 相连。尽管 s 和 t 属于 G'的同一个 3-边连通连通分量，但 G'的最大 3-边连通子图中没有任何两个顶点属于同一个子图。

我们注意到，计算 k无向图中的边连通分量 G 可以归约为计算 G中的全点对最大流，这可以使用 $\tilde{O}(n^2$时间，通过 [AKL+21]最近提出的算法完成。然而，该算法既不能解决，也不能对我们计算极大 k边连通子图的问题提供任何结论。

章节安排。我们先在第 2 节给出高层次的技术概述，随后在第 3 节正式介绍用于寻找极大 k-边连通子图的递归最小割算法，为类似的证明做好铺垫。在第 4 节中，我们给出一个用于定理 1.5(1) 的局部割算法。接着，在第 5 节中，我们利用该局部割算法，在 $\tilde{O}(mm^{3/4})$ 时间内求出极大 k-边连通子图。这证明了定理 1.1 中所述的第一项运行时间。在第 6 节中，我们给出定理 1.5(2) 中的另一种局部割算法；当割的体积较大时，该算法比第 4 节中的算法更快。与第 5 节类似，我们在第 7 节中应用这一局部算法，在 $\tilde{O}(mn^{4/5})$ 时间内求出极大 k-边连通子图，从而完成定理 1.1 的证明。

在第8节中，我们证明了求解问题 1.4的困难性，从而证明了定理 1.6。在第9节中，我们展示了主要结果在近似边强度方面的一个应用，即推论 1.2。

2 技术概述

如果 X是一个 $(x,\ \nu,\ \sigma,\ k)$-集合，只是不要求$|X|<\sigma$，则称 $X\subset V$为一个 $(x,\ \nu,\ k)$-集合。同样地，如果不要求 $\mathrm{vol}(X)<\nu$，则称 X为一个 $(x,\ \sigma,\ k)$-集合。

框架。 我们的算法基于 [CHI+17]的框架，因此我们先解释他们的思路。他们的第一个关键观察是，递归最小割算法只有在递归深度为 $\Omega(mn)$ 时，才会花费 $\Omega(n)$ 时间。而当许多已找到的、值小于 k 的割高度不平衡（例如 $|S|=O(1)$）时，就会出现这种情况。[CHI+17] 的主要技术结论是，若有一个用于寻找这类不平衡割的亚线性时间算法 S，就能改进 $O(mn)$运行时间。

更具体地说，他们的归约表明，给定一个顶点 x，如果我们能在 $O(\nu)$ 时间内检查是否存在一个 (x, ν, k)-集合，那么就能在 $O(m\sqrt{m})$ 时间内计算出极大 k-边连通子图。事实上，即使稍微放宽算法的保证，他们的归约仍然成立：算法“要么报告不存在 (x, ν, k)-集合，要么返回一个 $(x, O(\nu k)\ k)$-集合”。Chechik 等人 [CHI+17]在 $O(\nu k^{O(k)})$ 时间内解决了这个放宽后的问题，而 Forster 等人 [FNY+20] 将其改进到 $\tilde{O}(\nu k^2)$ 时间，由此分别得到运行时间为 $\tilde{O}(m^{3/2}k^{O(k)})$ 和 $\tilde{O}(m^{3/2}k^{3/2})$ 的最终算法。[4] 遗憾的是，当 k 较大时，这些算法速度太慢。

我们首先观察到，归约框架 [CHI+17] 可以调整为与定理 1.5 中运行时间独立于 k 的局部算法兼容。更具体地说，尽管“要么报告不存在 (x, ν, k)-集，要么返回一个 $(x, O(\nu k)\ k)$-集”这一较宽松的保证与问题 1.4 不兼容，但我们证明，问题 1.4 的局部算法仍可用于突破 $O(mn)$ 屏障。问题 1.4 的算法之所以足够，其基本思路源于这样一个事实：如果一个图不是 k-边连通的，那么对于某个顶点 x，要么必定存在一个 (x, ν, k)-极端集（而不仅仅是一个 (x, ν, k)-集）；要么所有大小小于 k 的割都具有至少 ν 的体积，也就是说，这些割相当平衡，对我们来说是理想的情况（形式化证明见引理 5.2）。

由于 [CHI+17] 的框架可以与针对问题 1.4的局部算法兼容，因此我们现在只需应用运行时间为 $\tilde{O}(\nu\sigma^2) = \tilde{O}(\nu^3)$ 的局部算法（来自 定理 1.5(1)），或运行时间为 $\tilde{O}(\sigma^4)$ 的局部算法（来自 定理 1.5(2)），从而分别得到最终算法，其运行时间为 $\tilde{O}(mm^{3/4})$ 或$\tilde{O}(mn^{4/5})$。至此，定理 1.1得证。

局部割算法。首先，我们回顾基于最小生成树的随机收缩技术的等价视角（正如 Karger 的原论文 [Kar93]中所述）。暂且假设输入图 $G = (V, E)$ 是无权图。

定义随机秩函数 $r : E \to [0, 1]$，其中每条边 e 的秩 $r(e)$ 是从 $[0, 1]$ 中独立抽取的均匀随机数。令 MST_r 为关于秩函数 r的最小生成树。

引理 2.1 ([Kar93])。 对于 $(S,\overline{S})$ 中的任意一个最小割， G其概率为 $\Omega(1/n^2)$，

$$r(e') > r(e)$$

对于所有 $e' \in E(S, \overline{S})$以及 $e \in$最小生成树$_r \setminus E(S, \overline{S})$.[5]

接下来，我们概述如何证明 定理 1.5。假设存在一个 (x, ν, σ, k)-极端集 S，它位于G中。我们的目标是返回某个 (x, ν, σ, k)-集。注意， S根据极端集的定义，它是该图的唯一最小割$G/\overline{S}$ 。只需将引理 2.1 应用于图 $G/\overline{S}$，该图至多包含 σ 个顶点，即可得到以下结论：

引理 2.2. 概率为 $\Omega(1/\sigma^2)$，

$$r(e') > r(e) \tag{1}$$

对所有 $e' \in E(S, \overline{S})$以及 $e \in G[S] \cap$最小生成树$_r \setminus E(S, \overline{S})$。

[4]他们在无向图中改进运行时间时，使用了针对无权图的特定技术和论证。因此，这些技术不适用于我们的情况。[5] 对于熟悉 Karger 分析的读者，边的秩对应于边收缩的顺序。这个条件等价于：不断进行收缩，直到只剩下两个顶点，此时收缩后的图中唯一剩下的割恰好是最小割（$S, \overline{S}$）。

当秩函数 r 满足此条件时，我们称它相对于集合 S 是良好的，或者说它 S 尊重 r。根据引理 2.2，我们将生成 $\tilde{O}(\sigma^2)$ 个彼此独立的秩函数，使得其中一个以高概率是良好的。现在，假设 r 是良好的，并假设我们从顶点 x 开始运行 Prim 最小生成树算法，以构建一棵“部分”最小生成树 $T_r \subseteq$最小生成树$_r$。根据公式 (1)，我们知道 T_r 必须先覆盖 S，然后才能跨越割 $(S,\bar{S})$。此时，$V(T_r) = S$ 是一个 (x,ν,σ,k)-集合，我们可以返回 $V(T_r)$。

令 $X = V(T_r)$。我们的算法很简单：每当 X 是一个 (x,ν,σ,k)-集合时，直接返回 X。我们可以通过维护 $\mathrm{vol}(X)$、$|X|$ 和 $\delta(X)$ 来进行检查。如果 r 不够理想，那么 X 可能永远不会成为一个 (x,ν,σ,k)-集合，但我们会在 X 太大时直接终止，即 $\mathrm{vol}(X) \geq \nu$ 或 $|X| \geq \sigma$。

在以下情况下，前提是可以访问 G 的邻接表。我们还可以通过对 G 进行一些简单的预处理，在 $\tilde{O}(\sigma^2)$ 时间内维护这些信息（详见 第 6 节 ）。由于有 $\tilde{O}(\sigma^2)$ 个秩函数，总运行时间分别变为 $\tilde{O}(\nu\sigma^2)$ 和 $\tilde{O}(\sigma^4)$。当图为无权图时，我们由此得到定理 1.5 。

最后，我们去掉图是无权的这一假设。我们注意到，只有引理 2.1 和引理 2.2 的正确性依赖于图是无权图这一事实。对于带权图，秩函数需要采用不同的定义。给定一个带权图$G = (V, E, w)$，如果我们将 G 视为无权多重图，并为 $[0,1]$ 中的每条边生成一个独立的随机秩，那么引理 2.1 就成立，但这种模拟速度太慢。我们需要的是：若边 e 的权重为 $w(e)$，则应将秩 $r(e)$定义为 $[0,1]$ 中 $w(e)$个独立随机数的最小值。我们推导出这一公式的显式表达式（见断言 4.6），并证明它可以高效计算且数值稳定（见附录 A）；这也是唯一需要边权重为整数的地方。这样一来，我们便能高效处理带权图。定理 1.5 的证明概要到此结束。

条件下界。 对于定理 1.6的第一条陈述，我们证明了 t团问题可以归约为强化版的问题 1.4，即：对于给定的种子，检查是否存在一个（x，ν，σ，k）-集合的问题，且即使 $x \in V$仍然只依赖于 σ。详见第 8.3 节。类似的归约已在 [FGK13] 中给出，但在他们的归约中，$\sigma = \Omega(n)$ 相当大，因为他们在无权图中进行研究。

接下来，我们基于 OMv 猜想[HKNS15]解释定理 1.6 第二个结论的证明。在 [HKNS15] 中，他们证明了：假设该猜想成立，给定一个具有n 个顶点的图 $G = (V, E)$，并且在 G上的预处理时间为 $\mathrm{poly}(n)$，那么对于给定的查询 $S \subset V$，不存在能在 $n^{2-\Omega(1)}$时间内检查 S 是否为独立集的算法。我们将说明如何利用问题 1.4 的子程序来检查 S 是否独立。给定 S，设想我们人为添加一个环 C_S，使其覆盖 S，且每条边 $e \in C_S$ 的权重均为无限。令 G_S 为所得图。令$\sigma = |S| + 1$，并令 $k = \mathrm{vol}(S)$。固定任意一个顶点 $x \in S$。如果 S 不是独立的，那么 S 是 G_S 中的一个 (x,σ,k)-极端集，因为 $|S| < \sigma$，$\delta(S) < \mathrm{vol}(S) = k$，并且任何真子集 $S' \subset S$的割值都是无限大。另一方面，如果 S是独立的，那么 G_S 中不存在 (x,σ,k)-集。反设存在一个 (x,σ,k)-集 T。我们知道 T不能与 S交叉，否则，T的割值将是无限大。由于 $x \in T$，所以 $S \subseteq T$。但 $|T| \leq \sigma - 1 = |S|$，所以 $T = S$。现在，由于 S 是独立的，我们有 $\delta(T) = \delta(S) = \mathrm{vol}(S) = k$，这就产生了矛盾。最后请注意，给定 G 的邻接表和矩阵，我们可以轻松模拟 G_S 的邻接表和矩阵；对于任意查询集 S，这只需额外增加$O(\sigma)$的开销，而该开销可被 $O(\sigma^{2-\varepsilon})$时间吸收。因此，若有一个能够访问 G 的邻接表和矩阵、运行时间为 $\sigma^{2-\Omega(1)}$的问题 1.4 子程序，我们就能在 $n^{2-\Omega(1)}$内检查 S 是否独立，从而推翻 OMv 猜想。

3 预备知识

我们将上文中问题陈述之前的段落称为问题 1.4 的基本符号约定。给定一个带权无向图 $G = (V, E, w)$，其中$w \in \mathbb{Z}^E_{\geq 0}$，$G$ 是 k-边连通的，如果全局最小割值 $\min_{\emptyset \neq S \subset V} w(\delta(S)) \geq k$。我们在附录C中给出以下简单观察的证明。

命题 3.1。 任意图 G 的极大 k-边连通子图 $\{V_1, \ldots, V_z\}$ 都是唯一的，并构成对 V 的划分。

根据 命题 3.1，我们通常将所需的解称为 $\{V_1, \ldots, V_z\}$ 的极大 k-边连通划分，以强调它构成一个划分。

3.1 递归最小割算法

本小节中，我们介绍一种朴素算法（例如参见 [CKN20] 的附录），用于寻找图的极大 k-边连通划分。这有助于读者熟悉其中涉及的思想，也为后续证明做铺垫。

假设图 G 中有一个小于 k 的割，那么任何极大的 k-边连通顶点集都应包含在割的一侧。否则，我们可以用权重至多为 k 的边，将由该顶点集诱导的图分开；因此，它不是 k-边连通的，也就不是极大 k-边连通的。

因此，一个直接的算法可以找到大小小于 k 的割（如果存在），并在该割所诱导的两个图上递归执行。如果不存在大小小于 k 的割，那么该图就是 k-边连通的，我们将剩余顶点加入划分，即可完成。

算法 1： RecMincut (G,k)

数据： $G = (V, E, w)$ k

结果： 顶点集 V的极大 k-边连通划分

计算图 G 的最小割 $(S, V \setminus S)$;

如果 $w(S, V \setminus S) \geq k$，**则**

  返回 $\{V\}$

否则

  返回 RecMincut$(G[S], k) \cup$ RecMincut$(G[V \setminus S], k)$

引理 3 .2. 算法 *1* 的输出是 k-边连通划分，它是 G 的极大划分。

证明。 设 $\mathcal{P}$ 为算法的输出。注意，算法 1 输出的集合构成顶点集 V 的一个划分，因为没有顶点重复，且每个顶点都包含在内；递归最终至少会以单顶点为底，因为每个顶点显然都是 k-边连通的。

现在，我们证明由每个集合 $S \in \mathcal{P}$ 诱导的子图是一个极大 k-边连通子图。事实上，根据我们停止递归的方式，$G[S]$ k-边连通。接下来，我们证明 S的极大性。反设存在一个集合 $T \supset S$，使得 $G[T]$ 也是 k-边连通的。在算法开始时，集合 T 包含于 V中。设 V' 是算法在 G 上运行期间最后一个包含 T的集合。因此，$G[V']$的最小割大小小于 k，将集合 T分成两部分。因此，$G[T]$ 也存在一个大小小于 k的割，这与我们假设 $G[T]$ 是 k-边连通的矛盾。因此，我们得出结论：$G[S]$ 是一个极大 k-边连通子图。

□

引理 3.3. 算法 *1* 最多需要 $O(mn)$ 时间。

证明。 算法 1 的递归深度显然至多为 n，因为每次递归都会使最大子图中的顶点数至少减少一个。

我们使用 [Kar00] 中的 Karger 算法，以近线性运行时间 $O(m\log^3 n)$ 在具有 m 条边和 n 个顶点的图中计算最小割。由于特定递归深度上的所有子图彼此不相交，且总共至多有 m 条边。设 G_i 是其中一个具有 m_i 条边和 n_i 个顶点的子图。因此，在所有子图中计算最小割的总时间为 $\sum_i O(m_i \log^3 n_i) \leq \sum_i O(m_i \log^3 n) \leq \tilde{O}(m)$。所以，所有 ˜ 个递归层的总运行时间至多为 $O(mn)$。 □

4 小体积割的局部算法

本节旨在证明定理 1.5 (1)。本节安排如下：我们在第 4.1 节介绍局部算法的关键子程序 LocalPrim。给定一个相对于极端集 S 的良好秩函数 r（回顾引理 2.2），LocalPrim 将找到一个 (x, ν, k)-集。在第 4.2 节中，我们介绍随机秩函数的思想，并利用 [Kar93] 中的随机收缩思想证明引理 2.2。由引理 2.2 可知，随机秩函数以 $\Omega(1/\sigma^2)$ 的概率是良好的。在第 4.3 节中，我们给出局部算法：通过使用多个随机秩函数重复执行 LocalPrim 子程序，提高找到良好秩函数的概率；随后，该算法以高概率解决问题 1.4，并证明定理 1.5 (1)。这里，我们重述定理 1.5 的第一部分。

定理 4.1。 存在一种随机化算法，能够以高概率解决问题 *1.4*，运行时间为 $O(\nu\sigma^2\log^2 n)$ 给定图的邻接表即可。

4.1 局部 Prim 算法

在本小节中，我们将介绍局部算法所需的关键子程序。为此，我们需要先给出一些定义。秩函数 r：$E \to [0,1]$ 是一个将每条边映射到 $[0,1]$ 中的某个数的函数。我们称 $r(e)$ 为边 e 的秩。令 $\mathrm{MST}_r(G)$表示相对于秩函数 r的 G 的最小生成树。

由于我们始终关注秩 $r(e)$相对于其他边的取值，而非某条边的绝对秩，因此在本文余下部分，我们将秩值简称为秩。

定义 *4.2*（集合遵循秩函数）。给定一个图 $G=(V,E,w)$和秩函数r：$E \to [0,1]$，若集合 $S \subset V$满足：对于所有 $e \in \mathit{MST}_r(G) \cap G[S]$，$\mathit{MST}_r(G) \cap E(S,\overline{S}) = e'$ 且 $r(e') > r(e)$，则称集合 $S \subset V$遵循秩函数 r。

换言之，集合 S 尊重 r，如果最小生成树$_r$ 只有一条边 e' 与该割相交$\delta(S)$，这使得最小生成树$_r \cap G[S]$ 成为 $G[S]$的最小生成树，并且该最小生成树中包含于 $G[S]$ 的每条边的秩都应小于边 e'的秩。回顾引理 2.2之后的讨论。我们称秩函数 r 关于集合 S 是 良好的，当且仅当 S 尊重 r。

该定义对于我们的算法的正确性至关重要。请注意，用于分析的最小生成树是相对于秩函数 r 而言的。它可能与使用权函数 w 计算出的最小生成树大不相同。

我们考虑以下问题。

问题 4.3 (LocalMST)。 给定一个图 $G=(V,E,w)$、一个顶点 x、一个秩函数 r 以及参数 ν, σ, k，寻找一个(x,ν,σ,k)-集合 S，使其满足秩函数 r；如果不存在满足条件的(x,ν,σ,k)-集合，则返回 $\perp$。

我们陈述 用于解决 问题 4.3 并证明其正确性的局部 Prim 算法 s.

算法 2: LocalPrim

输入: $G=(V,\ E,\ w)$ $x,\ \nu,\ \sigma,\ k,\ r$ 其中 r 是一个秩函数 $r\colon E\to[0,1]$

结果: $(x,\ \nu,\ \sigma,\ k)$-集; 若 r 存在则满足其要求, 否则为 $\perp$

```
X = {x};
当 w(δ(X)) ≥ k 且 vol(X) < ν 且 |X| < σ 执行
    找到秩最小的边, e = (u, v) ∈ E(X, V \ X) = δ(X);
    更新 X = X ∪ {v};
如果 w(δ(X)) < k 则
    返回 X
否则
    返回 ⊥
```

我们从单顶点割 X 开始扩展集合 $\{x\}$。由于我们只需要满足秩函数的集合, 因此考虑秩最小的割边 (u,v), 其中 $u\in X$且 $v\notin X$, 并通过设定 X 来扩展 $X\leftarrow X\cup\{v\}$。我们沿秩最小的边扩展集合 X, 直到违反集合 X 的体积或基数约束, 或找到割大小小于 k的集合。如果存在满足秩函数 x,ν,σ,k)-集合 S, 则算法 2会找到满足秩函数 r的最小集合, 使得它的任何真子集都不满足秩函数 r。我们将在下面的引理中证明这一点。

引理 4.4. 如果在 $(x,\ \nu,\ \sigma,\ k)$-集合 S 中存在 $G=(V,\ E,\ w)$ 满足秩函数 r, 且其为极小集合, 即没有真子集满足 r, 那么 算法 2将返回 S, 并且只有在不存在此类满足条件的集合时才返回 $\perp$。 S

证明。 令 S' 为 算法 2 返回的集合。我们声称 S' 不可能包含一个不属于S 的顶点。由于我们从 $X=\{x\}\subset S$ 开始, 因此要获得一个不属于 S 的顶点, 就需要沿着一条来自 X 的边扩展集合 $\delta(S)$。

在沿着一条来自 $\delta(S)$ 的边扩展之前, 令 X 为当前集合。如果 $X\subsetneq S$, 那么 $\delta(X)$ 中存在一条属于 $G[S]$ 的最小生成树的边, 其秩小于 $\delta(S)$ 中任意一条边的秩, 因为 S 满足 r。因此 $X=S$, 在我们收缩一条来自 $\delta(S)$ 的边之前; 但如果 $X=S$, $w(\delta(X))=w(\delta(S))<k$, 由于 S 是一个 (x,ν,σ,k)-集合, 因此 LocalPrim 会在沿着一条来自 $\delta(S)$ 的边扩展之前终止, 并返回 S。所以 S' 不可能包含 S 之外的顶点, 因此 $S'\subseteq S$。由于 S 的任何子集也是 (ν,σ)-集合, 并且根据算法, X 始终包含 x。因此, 我们仅在割大小小于 k 时终止, 所以返回的任何集合都是一个 (x,ν,σ,k)-集合。

剩下要证明的是, LocalPrim 不可能返回 $\perp$。假设它返回了 $\perp$。只有在无法找到任何 $(x,\ \nu,\ \sigma,\ k)$-集时, 我们才返回$\perp$。然而, 由于 S遵循 r, 沿秩最小的边进行扩展, 无非就是从 x 开始应用 Prim算法。根据遵循秩函数的集合的定义, 我们沿着$G[S]$ 中最小生成树的所有边进行扩展, 而不会沿着来自 $\delta(S)$ 的任何边扩展集合 X, 因为这些边的秩都更高, 因此得到$X=S$。因此, LocalPrim 要么在退出 while 循环时返回 S, 要么在找到一个同样遵循 r 、但无法返回 $\perp$ 的子集时提前终止。 □

现在我们来界定 LocalPrim 算法的运行时间。

引理 4.5。 LocalPrim 子程序最多需要 $O(\nu\log\nu)$ 时间。

证明。 这里, 我们描述如何实现 LocalPrim。我们创建一个字典 D, 其不变式为: $D=E(X,\ V\setminus X)$ 包含所有跨越 X 的割边。有了这个字典, 我们可以

获取最小秩边 $e = (u, v) \in E(X, V \setminus X)$，用于 第 3 行的 算法 2。此查找操作最多执行 $|X| < \sigma$ 次，因为每当 $|X| \geq \sigma$ 时，我们就会终止。

每当我们扩展 $X \leftarrow X \cup \{v\}$ 时，就会通过扫描 O 中 v 的所有边，在 $1 + \deg(v)$ 时间内更新字典。我们从 D 中删除 v 与 X 之间的所有边，并将 v 与 $V \setminus X$ 之间的所有边插入 D。在遍历 v 的边时，我们还会更新 X 的割值：减去已经与 X 相接的边的权重，并加上之前未与 X 相接的边的权重。

插入和删除操作的总数为 $O(\sum_{v \in X}(1 + \deg(v))) = O(\sigma + \nu)$，因为当 $\mathrm{vol}(X) \geq \nu$ 或 $|X| > \sigma$ 时我们就会终止。由于字典的每次操作都需要对数时间，因此总时间至多为 $O((\sigma + \nu) \log \nu) = O(\nu \log \nu)$。

□

根据上述引理 4.4，给定存在一个（ν，σ，k）-集合 S 在图 G中。如果我们找到一个秩函数 r 使得它 S 遵循 r，那么通过运行 LocalPrim ，并以任意顶点 $x \in S$ 作为输入，使用该秩函数 r，即可在 O（$\nu \log \nu$）时间内找到一个（ν，σ，k）-集合。

下一小节将展示我们如何构造秩函数，以帮助我们找到（ν，σ，k）-集。

4.2 良好秩函数

本小节将展示如何构造上文介绍的良好秩函数，并解释它与使用 [Kar93] 的随机收缩技术来寻找最小割这一思想之间的关系。

构造一个被任意特定集合 S 尊重的秩函数很容易。然而，我们不知道哪个子集是（ν，σ，k）-极端集，因此无法显式构造一个被未知极端集尊重的秩函数。我们提出了一个秩函数上的分布 R，并证明从该分布中选出的任意秩函数以较高概率是良好的。构造该分布 R 所采用的关键思想是随机收缩。

为了理解 R 的构造，我们需要从不同的角度考察 Karger 的随机收缩过程。我们先从 Karger 的最小割算法中随机边的采样过程开始。根据 [KS96] 中的引理 6.2.1，该过程也可以等价地表述为。

断言 4.6 (引理 6.2.1（见 [KS96]）。 如果我们选择一个秩函数，将每条边 e 映射到$r(e) = 1 - (1 - t)^{1/w(e)}$，其中该值通过对一个值 $t \in [0, 1]$ 进行均匀采样计算得到，且 $w(e)$ 是该边的权重，那么边 e 的秩在所有边中最小的概率为 $\frac{w(e)}{\sum_{e \in E} w(e)}$。

为完整起见，我们在此给出 断言 4.6 在我们形式体系下的证明，并将证明推迟到 附录 A。本节其余部分将说明随机秩函数以较高概率是一个良好秩函数。

根据 断言 4.6，我们知道，从所有边中选取秩值最小的边（$r(e)$），等同于按权重比例选取一条边。因此，Karger 的随机收缩 [Kar93] 算法不过是选取秩最小的边 $r(e)$并将其收缩。这个过程等同于在图 $G = (V, E, w)$ 上应用 Kruskal 算法，其中 r 是权函数。因此，Karger 随机收缩过程中收缩的所有边构成该图的最小生成树r of the graph G。

在下面的引理中，我们证明，任意 (x, ν, σ, k)-极端集以适当的概率满足上述随机秩函数。

注 4.7. 任意(x,ν,σ,k)-集都不会以适当的概率满足随机秩函数，因此，要使一个集合满足随机秩函数，我们还需要更多结构。

我们正式定义所抽取的秩函数的分布。对于图 $G=(V,\ E,\ w)$ 的每条边 $e\in E$，我们定义一个随机变量 R_e，其累积分布函数（CDF）为 $\Pr(R_e\le t)=1-(1-t)^{w(e)}$。分布 R 是图 E，$R=(R_{e_1},\ R_{e_2},\ \cdots)$ 中所有边对应的随机变量的联合概率分布。注意，当 $e\neq f$ 时，每个 R_e 都与另一条边的随机变量 R_f 独立。因此，从分布 R 中选取一个随机秩函数 r，无非就是让每条边 $e\in E$ 根据 R_e 选取一个值，该值就是这条边的秩值 $r(e)$。

现在我们证明一个关键引理，它对于证明局部算法的正确性至关重要。

引理 4.8。 设 r 是从分布 R 中抽取的秩函数，S 是图 $G=(V,E,w)$ 中的任意一个 (x,ν,σ,k)-极端集，则 S 遵循 r 的概率至少为 $\sigma^{2(\sigma-1)}$。

证明。我们需要给出极端集 S 遵循秩函数 r的概率下界。

令 $G/\overline{S}=(V_S,\ E_S,\ w_S)$ 表示将集合 $V\setminus S$ 收缩为一个超级节点 t后得到的图。$V_S=S\cup\{t\}$。根据定义，$(x,\ \nu,\ \sigma,\ k)$-极端集 $(S,\ \{t\})$ 是 G/S的唯一最小割。

在图 G/S上使用 Karger 随机收缩算法，成功收缩并找到最小割的概率至少为 $\sigma^{2(\sigma-1)}$，因为图 G/S 中的顶点数有上界 σ。

如 断言 4.6所述，Karger 的随机收缩过程中对边的采样可以通过从 R 中选取一个秩函数，并将其限制在图 G/S 上来模拟，即只考虑 G/S中的边。因此，秩函数 r 以至少 $\sigma^{2(\sigma-1)}$ 的概率成功找到唯一的最小割 $(S,\ \{t\})$；也就是说，集合 S 中的所有顶点都被收缩为一个节点，例如 s，最终得到割 $(\{s\},\ \{t\})$，因为它是唯一的最小割。

现在我们证明，从 R 中选取的此类成功秩函数会被图 G/S 中的集合 S 所遵循。由于秩函数 r 应当收缩整个 S，且不收缩最小割 $\delta_{G/S}(S)$ 中的任何边，因此相对于秩函数 r，G/S 的最小生成树应当在 G/S 中仅跨越最小割 S 一次，并且在跨越集合 S 之前也应覆盖 $G[S]$。

因此我们得出结论：G/S 的最小割 $(S,\ \{t\})$ 中的所有边，其秩都高于限制在 $G[S]$ 上的生成树中的边。因此，只要随机秩函数在 G/S 中找到唯一的最小割，S 就会在 G 中遵循 r。

由于随机收缩过程在 G/S 中找到最小割的成功概率至少为 $\sigma^{2(\sigma-1)}$，因此 S 遵循随机秩函数的概率也至少为 $\sigma^{2(\sigma-1)}$。 □

4.3 用于(ν,σ,k)-集的局部子程序

在第 4.1 节中，我们已经了解如何使用 LocalPrim 子程序，寻找一个遵循给定秩函数 r 的（x，ν，σ，k）-集。第 4.2 节表明，任何（ν，σ，k）-极端集都以较高概率遵循随机秩函数。在本小节中，我们将两者结合，给出一个能够解决问题 1.4并证明定理 1.5(1) 的算法。

根据 引理 4.8，我们知道，随机秩函数以 $\Omega(1/\sigma^2)$ 的概率被一个 (x,ν,σ,k)-极端集尊重。为了找到一个以高概率被尊重的秩函数，我们需要独立重复这一过程 $O(\sigma^2\log n)$ 次。

因此，求解问题 1.4 的算法主要是遍历（$O(\sigma^2\log n$）个彼此独立的随机秩函数。由于（x,ν,σ,k）-极端集至少满足 $O(\sigma^2\log\ n$）个随机秩函数中的一个，因此根据引理 4.4，我们可以保证返回一个（x,ν,σ,k）-集。

我们将上述论证形式化，并给出一种求解问题 1.4的算法。我们不失一般性地假设，在算法开始之前，所有随机秩函数 $r^i, i \in [O(\sigma^2 \log n)]$ 均已固定。

算法 3： LocalKCut

数据： $G = (V, E, w)$ x, ν, σ, k

结果： (x, ν, σ, k)-集，或者如果不存在 (x, ν, σ, k)-极端集，则为 $\perp$

```
对于 i ∈ [O(σ² log n)]，执行
    X = LocalPrim(G, x, ν, σ, k, r^i);
    如果 X ≠ ⊥，则
        返回 X;
返回 ⊥;
```

引理 4.9。 *算法 3 能够正确解决问题 1.4，且成功概率很高。*

证明。 根据引理 4.8，我们知道，任意 (x, ν, σ, k)-极端集至少以 $\sigma^{2(\sigma-1)}$ 的概率符合一个随机秩函数。若存在一个 (x, ν, σ, k)-极端集 S，那么 S 至少会符合 $\Theta(\sigma^2 \log n)$ 个随机秩函数中的一个。它们都不被 S 符合的概率，

$$\Pr(S \text{ does not respect any } r^i) \leq \left(1 - \frac{2}{\sigma(\sigma-1)}\right)^{\Theta(\sigma^2 \log n)} \leq \frac{1}{n^{\Theta(1)}}$$

由于存在一个被该 $(x, \nu, \sigma, k$<style id='13'>)-极端集符合的秩函数。根据引理 4.4，我们找到一个 $(x, \nu, \sigma, k$<style id='29'>)-集合，结论得证。

然而，如果我们未能通过任何一个随机秩函数找到 (x, ν, σ, k)-集合，那么我们就知道不存在 (x, ν, σ, k)-极端集，并且我们出错的概率 $\frac{1}{n^{\Theta(1)}}$ 可以忽略不计。 □

引理 4.10。 *算法 3 的运行时间为*

$$T(\text{LocalKCut}) = O(\sigma^2 \log n) \cdot T(\text{LocalPrim}). \tag{2}$$

证明。 for 循环的每次迭代需要 T(LocalPrim) 时间，因此所需的总时间为LocalKCut $O(\sigma^2 \log n) \cdot T$(LocalPrim)。 □

现在我们证明 定理 4.1，这与证明定理 1.5(1)相同。

定理 4.1 的证明：由引理 4.5 可得 T(LocalPrim) $= O(\nu \log \nu)$，其至多为$O(\nu \log n)$。根据引理 4.10， T(LocalKCut) $= O(\nu\sigma^2 \log^2 n)$。由引理 4.9 可知，算法 3 能以高概率在至多$O(\nu\sigma^2 \log^2 n)$ 的运行时间内解决问题 1.4，从而证明定理 4.1。 □

由于任意一个 (ν, σ, k)-极端集的体积至多为 ν，而其中的顶点需要彼此连通，因此其顶点数不可能超过 ν，故该集合所含顶点数的一个平凡上界为 ν。将顶点数的平凡上界 σ 代入公式 (2)，得到T(LocalKCut) $= O(\nu^3$ 对数$^2 n)$。现在我们可以从 σ 中消去定理 4.1中的参数，从而得到以下定理。

定理 4.11。 *给定一个图 $G = (V, E, w)$、一个顶点 x 以及参数 ν、k，存在一种随机化算法，以高概率找到一个 (x, ν, k)-集，或保证不存在 (x, ν, k)-极端集，其运行时间为 $O(\nu^3 \log^2 n)$。*

5 $\tilde{O}(m^{1.75})$ 用于最大 k-连通划分的算法

在上一节中，我们介绍了小型局部割的概念。我们还给出了一种局部算法，可以找到体积较小且割大小小于给定参数 k 的割。本节介绍一种利用局部子程序来降低递归深度的算法，并在 $\tilde{O}(m^{1.75})$ 的运行时间内找到极大 k-边连通划分。尽管当图较为稠密时，该算法可能比 $O(mn)$ 更慢，但在下一节中，我们将介绍另一种算法，以改进寻找极大 k-边连通划分的运行时间 $O(mn)$。

本节的主要目标是证明定理 1.1中给出的第一项运行时间。为完整起见，我们在此重述该定理。

定理 5.1。存在一种随机化算法，以高概率在至多 $O(m^{1.75}\log^{3.75} n)$ 时间内，给定一个含有 m 条边和 n 个顶点的图 $G=(V,E,w)$，找到最大 k-边连通划分。

5.1 使用 LocalKCut 的 GlobalAlgorithm

下面来看如何使用上文所述的 LocalKCut 子程序来降低递归深度。在本节余下部分，我们省略基数约束 σ。核心思路是利用 LocalKCut尽可能多地找到 (ν,k)-集。然而，移除一个 (ν,k)-集有时可能会在图中产生新的极端集。因此，在每个阶段，我们都会维护一个候选顶点列表（L，如 MaximalPartition中所述），其中的顶点可能位于某个 (ν,k)-极端集中。我们使用 LocalKCut 移除所有这些极端集，并将候选列表中的顶点 x 作为输入；该子程序要么找到一个包含 x 的 (ν,k)-集，要么保证不存在包含 x 的极端集。

穷尽候选列表可确保剩余图中不存在小型极端集，而这将意味着存在一个 ν-平衡的最小割。下面我们证明这一点。

引理 5.2。如果图 $G=(V,E,w)$ 中不存在 (ν,k)-极端集，那么以下两种情况之一成立：

- 最小割至少为 k，此时图 G 是 k-边连通的。
- 最小割小于 k，此时它是 ν-平衡的。

证明。第一部分 是显然的；如果图的最小割至少为 k，则它是 k-边连通的。

如果图的最小割小于 k，且每个包含意义下极小割都是 ν-平衡的，那么我们就完成了。假设存在一个包含意义下最小割 $(A,\bar{A})$，它不是ν-平衡的。不失一般性 $vol(A)<\nu$。由于 $\bar{A}$ 是包含意义下的最小割，A 是 $G/\bar{A}$中的唯一最小割。因此 A 是一个 (ν,k)-极端集，这与假设矛盾。因此$vol(A)\geq\nu$。所以每个包含意义下极小割的体积都为 $\geq\nu$，因此每个最小割都是ν-平衡的。

□

由上述 引理 5.2 可知，若图中不存在（ν，k）极端集，且该图不是 k-边连通的，那么每个最小割都是 ν-平衡的，因此会从图中移除至少 ν 体积。最大规模子图最多经过 $O(m/\nu)$ 层递归，就会缩减到原始体积的一半 $m/2$。鉴于这一

算法，我们现在将在下文对其进行正式定义和分析。

算法 4: kCutPartition
数据： $G = (V, E, w)$ ν, σ, k, L, m
结果： k-割划分 G

```
G = G;
R = {};
当 存在 x ∈ L 时
    S = LocalKCut( Ĝ, x, ν, σ, k);
    如果 S ≠ ⊥ 则
        R = R ∪ {S};
        L = (L ∪ V(δ_Ĝ(S))) \ S;
        Ĝ = Ĝ \ S;
    else
        L = L \ {x};
(A, B) = Mincut( Ĝ);
如果 λ(G) ≥ k 则
    R = R ∪ {V̂};
否则如果 vol( Ĝ[A]) > m 则
    R = R ∪ kCutPartition( Ĝ[A], ν, σ, k, V(δ_G(A)) ∩ A, m)
否则，如果 vol( Ĝ[B]) > m 则
    R = R ∪ kCutPartition( Ĝ[B], ν, σ, k, V(δ_G(B)) ∩ B, m);
否则
    R = R ∪ {A, B};
返回 R;
```

kCutPartition 将图划分为 $G = (V, E, w)$，所用割的大小小于 k，但无法得到最大的 k-边连通划分。它将问题缩减为规模更小的子图。以下引理 5.3证明了这一点。

引理 5.3. 令 $\mathcal{P}$ 为 k的最大边连通划分，其中 G 且 $\mathcal{R}$ 为算法 *4*返回的结果。对于所有 $U \in \mathcal{P}$，$\exists U' \in \mathcal{R}$ 使得 $U \subset U'$。换言之，$\mathcal{P}$是划分 $\mathcal{R}$的一个细化。对于所有 $U' \in \mathcal{R}$，体积$(G[U'])$ =体积$(G)/2$。

证明。 如引理 3.2的证明所示，任何最大 k-边连通顶点集 U 都不能被大小小于 k的割分开。在 kCutPartition中，用于划分该图的每个割的大小都小于 k。因此， k的每个最大边连通集 G 都必须严格包含于属于 $\mathcal{R}$的某个集合中。

由于我们总是在最小割体积较大的一侧递归，因此较小一侧的体积不到原体积的一半。通过局部割形成的任何子图，其体积都小于 ν，而我们之后会将其选为小于 $m/2$。因此，所有子图的体积至多为 vol$(G)/2$。 □

注意，原图中的每个极大 k-边连通集合，也是由 kCutPartition 得到的较小子图中的极大 k-边连通集合。因此，自然地，算

寻找最大 k-边连通划分的算法如下。

算法 5: MaximalPartition

数据: $G=(V,\ E,\ w)$ k

结果: 最大 k-边连通划分 G

$\mathcal{R}=$ kCutPartition$(G,\ \nu(|E|)\ \sigma(|V|)\ k,\ V,\ |E|)$;

$\mathcal{P}=\{\}$;

</style}对 $U\in\mathcal{R}$ 执行

  $\mathcal{P}=\mathcal{P}\cup$ MaximalPartition$(G[U],\ k)$;

返回 $\mathcal{P}$;

我们找到极大 k-边连通划分的图，其规模至多为原图体积的一半。因此，经过至多 $O(\log m)$ 次 kCutPartition递归调用 k，我们就能找到每个极大 $U\in\mathcal{P}$-边连通集合 k。由于任何 kCutPartition-边连通图都无法再被k进一步拆分，否则最终会得到一个显然是引理 5.4 的单个顶点。下面我们证明算法 5的正确性。

引理 5.4。 *算法 5 计算图 G 的极大 k-边连通划分。*

证明。 根据 引理 5.3，我们已经看到，任何极大 k-边连通集合都严格包含于 kCutPartition 返回的划分中的某个集合。然而，根据 kCutPartition ，每个连通分量的体积都会缩减为原图体积的一半。因此，在递归深度为 $m/2^i$ 时，每个连通分量的体积至多为 i。所以，任何特定集合 S，只要来自极大 k-边连通划分且满足 $vol(G[S]),\in[\frac{m}{2^{i+1}},\frac{m}{2^i}]$ 就会在递归深度至少为 $i+1$ 时被识别，因为没有任何割会切开集合 S。因此，经过 $O(\log n)$ 层递归后，每个连通分量要么已成为 k-边连通，并因此成为划分的一部分 $\mathcal{P}$；要么体积变为 0，即成为单顶点，而单顶点显然是 k-边连通。

□

5.2 运行时间分析

在本小节中，我们分析算法 4 以及算法 5的运行时间。

我们将 kCutPartition的运行时间分为两部分：调用局部子程序，以及调用全局最小割算法。对于第一部分，由于我们只会在图 kCutPartition 中使用来自候选列表 L 的种子顶点来调用局部子程序，因此可以通过界定加入 L 的顶点数量来界定运行时间。我们最初将候选列表设为 $L=V$，因为 (ν,k)-极端集可以包含 V 中的任意顶点。

找到大小小于 k 的割（使用 LocalKCut 或 Mincut）后，我们会从图 G 中移除较小的一侧，并在剩余图中寻找割。由于我们从图 G中移除了一些边，因此可能产生之前不存在的新极端集。所以，我们需要更新 L，以考虑因移除割边而可能出现的新极端集。遍历所有顶点来寻找新的极端集既不必要，也很耗时。与 [CHI+17]中的思路类似，我们证明，因从图中移除一条边而产生的任何新极端集都至少包含该边的一个端点。我们将此顶点加入候选列表，以便在新极端集为 (ν,k)-极端集时，找到一个围绕该顶点的 (ν,k)-集。

断言 5.5。 *通过移除一条边而在图中引入的任何新极端集，都必须至少包含该边的一个端点。*

证明。 令 G 表示移除边 e 之前的图， H 表示移除该边之后的图。只有当移除边 e 属于新极端集 A 的割边时，才可能形成新的极端集

，也就是说， $e \in \delta_G(A) = \delta_{G/A}(A)$。如果它是新极端集的内部边，也就是说，$e \in G[A]$，那么在移除该边之前，它就已经是一个极端集，因此并非通过移除边 e形成的。如果它属于 $G[A]$，那么这不会影响 A 是否为极端集，因为我们在判断 A 是否为极端集时会考虑图 G/A 。既然它是该极端集的割边，那么边 e 必有一个端点位于该集合中，由此证明了这一论断。 □

候选列表耗尽后，我们可以确信（以高概率）剩余图中不存在极端集；因此，我们对 ν应用全局最小割，得到一个平衡割，并递归处理较大的一侧，如第 15 行和第 17 行所示。我们现在在下面的引理中界定kCutPartition 的递归深度。

断言 5.6. 算法 4 的递归深度至多为 $O(m/\nu)$ ，对于一张有 m 条边的图而言。

证明。 while 循环结束时，我们可以保证不存在 (ν, k)-极端集；根据引理 5.2，这意味着最小割至少为 k ，此时只需将当前图的顶点集加入 $\mathcal{R}$即可完成。否则，最小割是ν-平衡的。因此，调用全局最小割时，最大的连通分量的体积至少减少 ν 。初始体积为 m，且每次递归深度增加时，最大连通分量的体积至少减少 ν，因此最多经过 $O(m/\nu)$次递归，最大连通分量的体积就会小于 $m/2$。一旦所有连通分量的体积都不超过原始体积的一半，我们就停止递归，只需将最小割的两侧都加入 $\mathcal{R}$。 □

我们通过将 kCutPartition 的总运行时间分为调用 LocalKCut局部子程序的部分和其余部分，来分析其运行时间。

断言 5.7。 算法 4 在整个算法的运行过程中，即所有 LocalKCut 个 $O(m)$ 递归层数内，最多调用 $O(m/\nu)$ 次。

证明。 局部子程序 LocalKCut 总是在属于 L 的顶点上调用。因此，方便的做法是对整个算法执行过程中加入候选列表的顶点数进行界定。

我们最初从 $L = V$ 开始，即大小为 n 的整个顶点集。每当一个 (ν, k)-集与 G 分离时，我们就将较大一侧割边的端点加入候选列表。根据 *Claim* 5.5，我们知道，移除割边后引入的任何新极端集都应包含割边的端点。因此，只需在割边端点附近检查 (ν, k)-极端集。根据 LocalKCut 的保证，如果存在任何 (ν, k)-极端集，我们就能以高概率找到 (ν, k)-集。由于我们从图中移除了 (ν, k)-集，该极端集不会出现在剩余图中。

一旦从图中移除一条割边，它就不可能属于后续递归中出现的任何子图。因此，每条边都会在列表 L 中引入一个新顶点，并且没有边会再次成为割边。因此，加入候选列表的顶点数量的一个平凡上界是边数。于是， LocalKCut 子程序的总调用次数至多为 $m + n$ ，即 $O(m)$。 □

由于对 LocalKCut 的调用次数为 $O(m)$，我们可以对 LocalKCut 的所有调用的失败概率应用并集界，得到 $O(m)\ n^c$ ，总错误概率仍至多为$\frac{1}{\text{poly}(n)}$ ，其中常数 c足够大。类似地，我们可以界定全局最小割算法被调用 $O(m/\nu)$次时的错误概率。由于算法被调用 $O(m/\nu)$次期间，图的规模至少为 $\Omega(n^{O(1)})$，如果全局最小割算法的错误概率至多为 $\frac{1}{n^c}$ ，其中常数 c 足够大，那么所有 $O(m/\nu)$的错误概率

调用的错误概率至多为$\frac{1}{\text{poly}(n)}$，符合要求。这证明了算法 4 的错误概率至多为 $\frac{1}{\text{poly}(n)}$。

引理 5.8。 *算法 4* 的运行时间至多为 $O(m^{1.75}\log^{2.75} n)$ 对于一个 m 边 n 顶点加权图。

证明。 根据 断言 5.7，我们已界定对 LocalKCut 子程序的调用次数，至多为 $O(m\nu^3\log^2 n)$，此结论来自 定理 4.11。

在 kCutPartition 的每个递归层，我们都会在一个边数少于 m 且顶点数少于 n 的图上应用全局最小割算法。因此，显然可以将运行时间上界为 $O(m/\nu)\cdot O(m\log^3 n)$，这是由于全局最小割算法的运行时间至多为该值。

因此， kCutPartition 的总运行时间至多为 $O(m\nu^3\log^2 n + m^2\log_\nu^3 n)$。通过使用 $\nu = O((m\log n)^{1/4})$ 进行优化， kCutPartition 的运行时间将为 $O(m^{1.75}\log^{2.75} n)$。 □

现在，我们来分析 算法 5 的运行时间。

引理 5.9。 *算法 5* 的运行时间至多为 $O(m^{1.75}\log^{3.75} n)$，对于一个有 m 条边和 n 个顶点的加权图。

证明。 根据 算法 4的保证，属于 $\mathcal{R}$ 的每个集合所诱导的图的体积都是原始体积的一半。因此，该MaximalPartition 子程序的递归深度至多为 $\log m = O(\log n)$。

接下来，我们来界定 MaximalPartition 在每个递归层所需的运行时间。划分后各集合所诱导的图的总体积不超过 m。设 $m_i(< m/2)$ n_i 表示应用kCutPartition 子程序后，顶点划分中 i^{th} 诱导子图的边数和顶点数。由于这些集合构成原图顶点的一个划分，并且我们只考虑这些划分所对应的诱导子图 $\sum_i n_i = n$，总体积不超过 m，即 $\sum_i m_i \le m$。

因此，我们得到以下递推方程。

$$T(m) = \sum_i T(m_i) + O(m^{1.75}\log^{2.75} n)$$
$$m_i \le m/2 \text{ for all i}$$

由于 $\sum_i O(m_i^{1.75}\log^{2.75} n_i) \le O(m^{1.75}\log^{2.75} n)$。因此，在每个递归层需要 $O(m^{1.75}\log^{2.75} n)$，且最多有 $O\left(\frac{n}{m^{1.75}}\log^{3.75} n\right)$ MaximalPartition 个递归层。因此，该子程序的运行时间至多为 $(\log)$。 □

证明 *定理 5.1*。 根据 引理 5.4，我们有 算法 5，它计算图 k-边连通划分 $G = (V, E, w)$，并以参数 k 作为输入，其运行时间至多为 $O(m^{1.75}\log^{3.75} n)$，由 引理 5.9 证明了 定理 5.1。 □

6 小基数割的局部算法

本节的主要目标是证明 定理 1.5 (2)。在本节中，我们只关注 (σ, k)-集，即割大小较小的小基数割。然而，当顶点的度较大时，它们的体积可以任意大。因此，应用算法 3将导致较长的运行时间。为此，我们通过进行额外的记账来克服这一问题，从而在 $\tilde{O}(\sigma^4)$ 时间内找到 (σ, k)-集。

我们将在第 6.1 节中介绍一种称为排序邻接表的数据结构，用于记账，随后利用它加速 LocalKCut 在第 6.2 节中的操作。

6.1 排序邻接表

给定一个图 $G=(V,\ E,\ w)$ 及其秩函数 $r\colon\ E\to[0,1]$ 在进行边删除的过程中，我们维护每个顶点的邻接表，并按照秩对其进行排序。除了排序邻接表之外，我们还维护一个邻接矩阵，以便在 $O(1)$ 时间内检查图中是否存在某条边。初始时，我们从空的邻接表 $\mathcal{L}$ 和空的邻接矩阵 $\mathcal{M}$开始，并可以通过以下操作修改数据结构。

- Insert ($e=(u,\ v)$):

  1. 将边添加到邻接矩阵 $\mathcal{M}$。2. 将边插入 $\mathcal{L}[u],\ \mathcal{L}[v]$的邻接表中。由于列表按秩排序且大小至多为 Δ，因此所需时间至多为 $O(\log\Delta)$。

- Delete ($e=(u,\ v)$):

  1. 在邻接矩阵 $\mathcal{M}$中将边标记为已删除。2. 我们可以在邻接矩阵中存储指向边 e 的指针，这样就能在 $O(1)$时间内找到它；如果将 $\mathcal{L}[u],\ \mathcal{L}[v]$ 维护为双向链表，也能在 $O(1)$时间内从邻接表 $\mathcal{L}[u],\ \mathcal{L}[v]$ 中删除它。

- NextEdge(X):

  1. 返回 $\min\{x\in X\colon\ \min(\mathcal{L}[x])\}$，时间为 $O(|X|)$。

我们通过将每条边 Insert 插入数据结构，构造图 G 关于秩函数 r 的排序邻接表，所需总运行时间至多为 $O(m\log\ \Delta)\ =O(m\log\ n)$。

如 LocalKCut 所示，为了提高找到极端集的概率，我们使用独立的秩函数，将 LocalPrim 子程序重复执行 $O(\sigma^2\log n)$ 次。因此，我们需要为所有 $O(\sigma^2\log n)$ 个秩函数分别维护上述数据结构。

针对所有秩函数构造数据结构，总共需要 $O(m\log\ n)\cdot O(\sigma^2\log\ n)\ =O(m\sigma^2\log^2 n)$ 的时间。在下一小节中，我们将介绍如何使用该数据结构来加速 LocalPrim。

6.2 加速LocalPrim

我们将在下文形式化说明如何使用排序邻接表来改进 LocalPrim 的运行时间。在算法中扩展集合 X 时，我们省略体积约束 ν，因为我们只关心

小基数割。算法的其余部分保持不变。

算法 6: LocalPrim

数据： $G=(V,E,w)$ x,σ,k,r，其中 r 是秩函数 $r\colon E\to[0,1]$

结果： (x,σ,k)-集合（若存在）满足 r，否则为 $\perp$

$X=\{x\}$;
当 $w(\delta(X))\geq k$ 且 ~~$vol(X)<\nu$ 且~~ $|X|<\sigma$ **时执行**
  找到秩最小的边，$e=(u,v)\in E(X,V\setminus X)=\delta(X)$;
  更新 $X=X\cup\{v\}$;
如果 $w(\delta(X))<k$ **则**
  返回 X
否则
  返回 $\perp$

在断言 6.1中，我们分析使用 LocalPrim 排序邻接表的运行时间。

断言 6.1。 LocalPrim 使用排序邻接表所需时间至多为 $O(\sigma^2)$。

证明。 在 while 循环的每次迭代中，我们使用 NextEdge 找到沿着其扩展集合 X 的下一条边，这需要 $O(|X|)$ 时间。令 v 为该边上不在 X 中的端点。为了维护 $w(\delta(X))$，我们需要加入 $\deg(v)$（加权度），这至多需要 $O(1)$ 时间，并删除当前存在于 X 中的顶点与 v 相连的边的权重。因此，在收缩 v 后计算割大小至多需要 $O(|X|)$。

通过加入顶点 v扩展集合后，我们需要删除两个端点都在 X 中的边，因为接下来用于扩展集合 X 的边不应来自集合X的内部边。用 v扩展集合 X 后，至多有 $|X|$ 条边变为内部边（即两个端点都位于 X中的边），需要从数据结构中删除（Delete）。因此，所需时间至多为 $O(|X|)$。

因此，while 循环的每次迭代需要 $O(|X|)$ 时间。由于 $|X|$ 的取值范围是从 1 到至多 σ，总运行时间至多为 $O(\sigma^2)$。 □

注意，我们针对特定的秩函数r修改排序邻接表数据结构。因此，我们会记录对数据结构所做修改的日志；由于我们只删除内部边，修改次数至多为 $O(\sigma^2)$。这样，在完成子程序 LocalPrim后，我们就可以撤销这些修改。因此，运行子程序仍需 $O(\sigma^2)$ 时间，并且能够保持数据结构不变。

类似于 LocalPrim，我们修改 LocalKCut 以寻找割大小较小的小基数局部割。请注意，我们在此处使用的 LocalPrim 是用于寻找小基数割的算法。

算法 7: LocalKCut

数据： $G=(V,E,w)$ x，σ，k

结果： (x,σ,k)-集；如果不存在 (x,σ,k)-极端集，则为 $\perp$

对于 $i\in[O(\sigma^2\text{对数}\,n)]$，**执行**
  $X=\text{LocalPrim}(G,x,\sigma,k,r^i)$;
  如果 $X\neq\perp$，**则**
    返回 X;
返回 $\perp$;

由 引理 4.10 可得以下 定理 6.2。

定理 6.2。 给定一个图 $G=(V,E,w)$、一个顶点 x 和参数 σ、k，存在一种随机化算法，可以找到一个(x、σ、k)-集，或以高概率保证不存在(x、σ、k)-极端集，并且运行时间为 $O(\sigma^4 \log n)$。

证明。 由于使用 σ, k-集的概率仅取决于该集合中的顶点数，且至少为 $\sigma^{-2(\sigma-1)}$，如 引理 4.8所示。因此，我们可以重复执行LocalPrim 子程序 $O(\sigma^2 \log n)$ 次来提高成功概率，这样总运行时间为 $O(\sigma^4 \log n)$。

□

这就证明了 定理 1.5(2)。

7 $\tilde{O}(mn^{0.8})$最大 k-连通划分算法

我们在 第 5 节 中看到，如何使用局部子程序 LocalKCut来降低递归深度，从而缩短运行时间，至少在图稀疏时如此。然而，当 $\tilde{O}(m^{1.75})$ 处于 m 稠密情形时，它仍然很高。下面我们通过使用 LocalKCut 中的快速 第 6 节，并将参数改为 σ 而非 ν，在保持算法总体框架不变的情况下，介绍上述算法的改进版本。这便得到了 定理 1.1中所述的第二个运行时间。本节证明的主要结果是以下定理。

定理 7.1。 存在一种随机化算法，对于一个图 $G=(V、E、w)$，其中包含 m 条边和 n 个顶点，该算法能够以高概率在至多 $O(mn^{0.8}\log^{3.6} n)$ 时间内找到最大 k-边连通划分。

7.1 关于 n 的参数化

在前面的局部子程序中，我们以局部割 ν 的体积作为参数，因此递归深度至多为 $O(m/\nu)$。当我们以 σ 作为参数时，可以改变递归深度，使其至多为 $O(n/\sigma)$。我们将在下面的引理中对其进行形式化证明。

引理 7.2。 使用更新后的 LocalKCut、 LocalPrim子程序时， kCutPartition 的递归深度至多为 n/σ。

证明。 在每一层中，当移除所有（σ， k）-极端集后，如果剩余图仍然不是 k-边连通的，则使用 Karger 的 Mincut 找到的最小割在两侧各至少包含 σ 个顶点。否则，我们将找到一个（σ， k）-极端集，这与不存在（σ， k）-极端集相矛盾（以高概率成立）。由于最小割是一个 σ-平衡割，大连通分量中的顶点数至少减少 σ。因此，至多经过 $O(n/\sigma)$ 的递归深度，就能将最大连通分量中的顶点数减少到其原始大小的一半，即 $n/2$。所以， kCutPartition 的递归深度至多为 $O(n/\sigma)$。 □

7.2 运行时间分析

分析方法与前一个算法的分析非常相似。

引理 7.3。 kCutPartition 的运行时间至多为 $O(mn^{0.8}\log^{2.6} n)$。

证明。 如 断言 5.7 所证明的，我们调用 LocalKCut 至多 $O(m)$ 次，每次耗时$O(\sigma^4 \log n)$ 时间。因此，局部子程序调用总共耗时 $O(m\sigma^4 \log n)$ 时间。

kCutPartition 的递归深度至多为 $O(n/\sigma)$，而在每个递归层级应用 Karger 的 Mincut 所需的运行时间至多为 $O(m\log^3 n)$。

我们还需要计算构建所有 $O(\sigma^2 \log n)$ 个秩函数所需的时间，总计至多为 $O(m\sigma^2 \log^2 n)$。

因此，总运行时间为 $O(m\sigma^2 \log^2 n) + O(m\sigma^4 \log\ n) + O(m \log^3 n) \cdot O(\binom{n}{\sigma})$。通过使用 $\sigma = O((n \log^2 n)^{1/5})$ 进行优化，总运行时间至多为 $O(mn^{0.8} \log^{2.6} n)$。 □

由于 MaximalPartition 调用 kCutPartition，它将图划分为若干部分，每个集合的大小至多为原始大小的一半。因此， MaximalPar-tition 的递归深度至多为 $\log n$。通过类似于 引理 5.9 中所述的论证，我们得出，寻找极大 k-边连通划分所需的运行时间至多为 $O(mn^{0.8} \log^{3.6} n)$。

这就证明了定理 1.1。注意，在满足 $\tilde{O}(mn^{4/5}$ 的稠密情形下，这优于 $\tilde{O}(m^{1.75}$，其中 $m = \Omega($ $n^{16/15})$。

8 条件下界

如前文所示，在我们的框架中寻找极大 k-边连通划分时，主要瓶颈在于这样一个子程序：寻找割大小小于 k 、且基数或体积较小的集合。一个自然的问题是："对于这些问题，我们能否改进运行时间对参数 ν, σ 的依赖？"本节将给出问题 LocalMST和 LocalKCut的下界。我们利用 OMv 问题的猜想困难性，给出这些问题的运行时间下界。

下面陈述 OMv 猜想。设 M 是一个维度为 $n \times n$ 的布尔矩阵。我们必须以在线方式，依次计算 $v_1, \cdots v_n$ 的矩阵-向量乘积。计算所有向量的向量乘积的朴素运行时间为 $O(n^3)$。猜想认为，即使允许多项式量级的预处理时间，也无法在朴素运行时间的基础上实现多项式级的改进。

猜想 8.1 (OMv 猜想)。 不存在一种算法，能够在总时间为 $n^{3-\Omega(1)}$ 内，以不超过 $1/3$ 的错误概率，在线计算依次到来的 n 个向量的矩阵-向量乘积。

特别是，我们利用 [HKNS15]证明的独立集查询问题的困难性，为 问题 1.4 和 问题 4.3给出下界。具体而言，通过结合 [HKNS15],中的定理 2.7 和 2.12，我们得到以下困难性结果：

定理 8.2 ([HKNS15])。 假设 猜想 *8.1*成立，则不存在算法能够在 $S \subset V$ 时间内判断图 $G = (V,\ E,\ w)$ 中的任意子集$1/3$ 是否为独立集，且错误概率至多为 $O(|S|^{2-\Omega(1)})$，即使允许对图进行多项式时间的预处理也不例外 G。

本节其余部分安排如下。在 第 8.1 节 中，我们证明包含 x 的基数约束小割$x,\ \sigma,\ k)$-LocalKCut 问题的下界。在 第 8.2 节 中，我们说明，即使是包含 x 并满足给定秩函数 r 的简单基数约束小割问题 $(x,\ \sigma,\ k)$-LocalMST，当仅提供图 G 的邻接表和矩阵时，也无法获得更快的算法。最后，在 第 8.3 节 中，我们将 [FGK13] 中给出的 NP 难归约扩展为 $W[1]$-困难性，不过是在加权情形下。

8.1 LocalKCut的下界

回顾一下 (x, ν, σ, k)-LocalKCut 问题。

问题 1.4 (LocalKCut)。 给定一个图 $G = (V, E, w)$、一个顶点 x以及参数 ν、 σ 和k，要么找出一个 (x, ν, σ, k)-集，要么返回 $\perp$，表示不存在(x, ν, σ, k)-极端集。

我们分别在第 4 节 和第 6 节 中针对 LocalKCut的不同参数变体给出了两种不同的局部算法，其运行时间分别为 $\tilde{O}(\nu\sigma^2)$ 和 $\tilde{O}(\sigma^4)$。我们将证明一个针对 (x,σ,k)-LocalKCut 的条件下界，如下所述。

定理 8.3。 假设 猜想 *8.1*成立，则不存在这样的算法：在不进行任何其他预处理的情况下，给定对 G 的邻接表和矩阵的访问权限，能在 (x,σ,k)-LocalKCut 时间内求解 $O(\sigma^{2-\Omega(1)})$。

在证明上述定理之前，我们先展示这一归约，它将寻找 (σ,k)-极端集的问题与判断一个集合是否为独立集的问题联系起来。

给定一个图 $G=(V,E,w)$ 和一个大小为 σ 的集合 S，我们可以在 $O(\sigma)$ 时间内求出 S 的体积。令 $k=\mathrm{vol}_G(S)$ $\sigma=|S|$。定义一个新图 $G_S=(V,E',w')$，它是 G 的完全副本，但还额外包含一个加权环，该环由 S 中的所有顶点构成，且每条边的权重为 $k+1$。如果新添加的环中有边已存在于 E 中，则只需将该边在 w' 中的权重增加 $k+1$，以确保 G_S 是简单图。我们用 $k+1$ 替代引言中的无穷值，因为这样就足以得到所需的结果。我们证明以下等价关系：

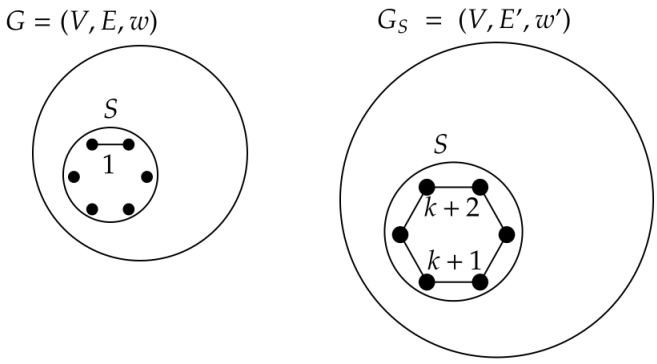

图1：$(x,\ \sigma,\ k)$-LocalKCut 的下界

引理8.4。 $E(S,S)\neq\phi$ 当且仅当 S 是 G_S 中的 $(\sigma+1,k)$-极端集。

证明。 ($\Longleftarrow$) 如果 S 是 G_S 中的 $(\sigma+1,\ k)$-极端集，那么根据 $(\sigma+1,\ k)$-极端集的定义，我们有 $\delta_{G_S}(S)<k$。根据我们对 G_S 的构造，我们有 $\delta_G(S)=\delta_{G_S}(S)$。因此，$\delta_G(S)=\delta_{G_S}(S)<k=\mathrm{vol}_G(S)$。由于 $\delta_G(S)<\mathrm{vol}_G(S)$，我们有 $E(S,\ S)\neq\phi$。

($\Longrightarrow$) 如果 $E(S,\ S)\neq\phi$，那么 $\delta_G(S)<k$ 且 $|S|<\sigma+1$，因此 S 是一个 $(\sigma+1,\ k)$-集。 S 也是 G_S 中的极端集，因为它是 G_S/S 的唯一最小割： G_S/S 的任何其他割都将包含来自该环的一条权重为 $k+1$ 的边，因此该割的权重将大于 k。 □

观察 8.5。 对于任意 $x\in S$, S，在 G_S 中，只有 $(x,\sigma+1,k)$-集同时也是极端的。

观察 8.5可由上述 G_S 的构造推出。现在，我们证明 (x,σ,k)-LocalKCut 问题的下界。

定理 定理 *8.3* 的证明。反设存在一个算法 $\mathcal{A}$，能够求解（$x,\ \sigma,\ k$）-LocalKCut，且运行时间为 O（$\sigma^{2-\varepsilon}$），其中 $\varepsilon>0$ 是某个常数。我们可以使用该算法，在运行时间 S 内判断给定集合是否独立 O（$\sigma^{2-\varepsilon}$），方法如下。

给定集合 S，要判断它在图 G 中是否独立，可在 $O(\sigma)$ 时间内按上述方法构造图 G_S。任意选取一个顶点 $x \in S$，并使用 $\mathcal{A}$ 在 G_S 中找出一个包含 x 的 $(\sigma+1, k)$-集。根据该算法的运行时间保证，这至多需要$O((\sigma+1)^{2-\varepsilon}) = O(\sigma^{2-\varepsilon})$，并可判断是否存在一个 $(x, \sigma+1, k)$-集；若不存在 $(x, \sigma+1, k)$-极端集，则返回 $\perp$。

如果 $\mathcal{A}$返回一个 $(x, \sigma+1, k)$-集，且输入中包含任意 $x \in S$，那么根据 S，它必定是观察 8.5中的结果。如果 S 是 $(\sigma+1, k)$-集，那么 S 不是独立的。

如果 $\mathcal{A}$ 返回 $\perp$，那么这意味着 G_S 中不存在 $(x, \sigma+1, k)$-极端集。根据观察 8.5，唯一的 $(x, \sigma+1, k)$-集是 S，它同时也是极端集。由于 $\mathcal{A}$ 意味着不存在 $(x, \sigma+1, k)$-极端集，因此不存在 $(x, \sigma+1, k)$-集，这意味着 S 不是 $(x, \sigma+1, k)$-集。由于 S 包含x 且大小小于 $\sigma+1$，唯一违反的条件是割大小约束，因此 $\delta_G(S) = \delta_{G_S}(S) \geq k = \mathrm{vol}_G(S)$。因此 S 是独立的。

因此，我们可以在 S 时间内使用 $O(\sigma^{2-\varepsilon}$，借助 $\mathcal{A}$计算其是否独立，这与猜想 8.1相矛盾。因此，对于任意常数 $O(\sigma^{2-\varepsilon}$，假设存在一个用于问题 1.4 的算法是错误的。 $\varepsilon > 0$

□

尽管定理 8.3 排除了在一般 $O(\sigma^2$情况下，比 σ快多项式倍的任何算法，但它仍未排除一种仅在集合 S 的大小相对于顶点集 V较小时才有效的算法。这一点很重要，因为根据我们的参数设置 $\sigma = \tilde{O}(n^{1/5})$。因此，即使在这个较小的参数范围内，我们也必须排除更优的问题 1.4算法。下面我们给出一个强下界，它排除了即使在已知查询集大小为 $O(\sigma^2$的情况下，仍能比 $|S| = \sigma \ll n$快多项式倍的算法。

定理 8.6。 假设 猜想 *8.1*成立，则不存在这样的算法：给定 G 的邻接表和矩阵（不进行进一步预处理），在问题 *1.4* 中以 $O(\sigma^{2-\Omega(1)})$ 求解问题，即使$\sigma \leq n^\gamma$ 对于每个常数 $\gamma \in [0, 1)$。

证明。 设 $\mathcal{A}$ 是一个算法，可在问题 1.4 中以 $O(\sigma^{2-\varepsilon})$求解问题，其中某个常数为 $\varepsilon > 0$ 且$\sigma \leq n^\gamma$。我们可以构造一个算法，使其解决独立集查询问题的速度在多项式意义上快于 $O(|S|^2)$，适用于任意大小为 n的图中包含 S 个顶点的集合。

给定一个图 $G = (V, E)$，构造一个任意图 $H = (V_H, E_H)$，其顶点集大小为$|V_H| = n^{1/\gamma}$，并将图 G 嵌入到 H 中任意选取的一个顶点数至多为 $|V_H|^\gamma$ 的子集中。

给定图 G 中的任意集合 $S \subseteq V$，构造对应的顶点子集 S_H，该子集位于 H 中，图 G 嵌入于此，并沿顶点 S_H 添加权重为 $\mathrm{vol}_G(S) + 1$ 的环，如定理 8.3 中所构造。现在我们可以使用 $\mathcal{A}$ 在 H 中寻找 $(\sigma+1, k)$-极端集，因为查询集合 S 的大小为 $|S| \leq n = |V_H|^\gamma$。因此，正如定理 8.3 中所证明的，我们可以判断集合S_H 在 H 中是否为独立集，这等价于判断 S 在 G中是否独立，运行时间为 $O(|S|^{2-\varepsilon})$，从而与猜想 8.1 矛盾。 □

8.2 LocalMST的下界

回顾问题 LocalMST。在本小节中，我们给出该问题的一个下界。

问题 4.3 (LocalMST)。**给定**一个图 $G = (V, E, w)$、一个顶点 x、一个秩函数 r 以及参数 ν, σ, k，求一个(x, ν, σ, k)集合 S，使其遵循秩函数 r；如果不存在遵循 r 的(x, ν, σ, k)集合，则返回 $\perp$。

在第 4 节中，我们介绍了一种用于解决问题 4.3的算法，该算法可以在ν、 σ、 k-集合，并使其满足秩函数 $O(\nu \log \nu)$ 时间内。我们在第 6 节中介绍了另一种算法，它

可以在σ、k-集合，并使其满足给定的秩函数 $O(\sigma^2)$ 时间内。下面我们将证明，即使是这个简单的问题 4.3，即寻找x、σ、k-集合并使其满足给定秩函数 r 的问题，也无法在运行时间上以多项式级别优于 $O(\sigma^2)$。

和前面的下界一样，我们将证明，对于任意参数范围 σ，不存在能以快于 $O(\sigma^2)$ 的多项式时间解决问题 4.3的算法。我们可以采用与定理 8.6 相同的论证，证明任何仅适用于更小参数范围 σ 的算法都存在下界。下面给出该定理。

定理 8.7。假设猜想 *8.1*成立。给定一个图 $G=(V,\ E,\ w)$，其邻接表和矩阵的参数 $\sigma,\ k$ 一个顶点 $x\in V$ 以及一个秩函数 r，不存在能在时间 $O(\sigma^{2-\Omega(1)})$内解决问题 *4.3* 的算法。

给定一个图 $G=(V,\ E,\ w)$ 和一个查询集 S，我们通过添加一条权重为 $k+1$ 的环来构造 G_S，其中 $k=\mathrm{vol}_G(S)$，如针对 LocalKCut 的下界证明中所示。我们定义秩函数 r_S：环中新增的边的秩值为 0，其余所有边的秩值均为 1。根据构造可知，S 在 G_S 中遵循秩函数 r_S，因为它有一棵生成树，其中每条边的秩值均为 0；而属于割 $E(S,\ V\setminus S)$ 的任意边的秩值均为 1，大于 0。

定理定理 *8.7*的证明。 假设我们有一个算法 $\mathcal{A}$，它能够在 问题 4.3 上用时 $O(\sigma^{2-\varepsilon})$，其中 $\varepsilon>0$。给定任意图 G，我们不进行预处理；每当收到一个查询集 S 时，就构造 G_S 及其对应的秩函数 r_S（隐式地）。

因此，我们可以使用算法 $\mathcal{A}$，在图 G_S 上，针对任意的 $x\in S$ 和秩函数 r_S，在 $O(\sigma^{2-\varepsilon})$ 时间内找到一个 $(x,\sigma+1,k)$-集，其中 $\sigma=|S|$，且 $k=\mathrm{vol}_G(S)$。

如果 $\mathcal{A}$找到一个$(x,\sigma+1,k)$-集，并且该集合在 r_S中满足 G_S，那么它必定是 S，因为根据 观察 8.5，我们知道 S是唯一的 $(x,\sigma+1,k)$-集。因此，S不是独立集。

否则，$\mathcal{A}$ 不会找到任何集合，并确认 S 是独立的。这只需 $O(|S|^{2-\varepsilon})$ 时间，从而与猜想 8.1矛盾。因此，不存在这样的算法 $\mathcal{A}$。 □

8.3 小型局部分离问题的下界

[FGK13] 证明问题 8.8在无权图上是 NP 完全的，并给出了从 Clique 正则图问题出发的归约。我们扩展了该定理，并证明问题 8.8 在以$W[1]$为参数时，对于带权图是 σ困难的。需要注意的是，问题 8.8的保证强于问题 1.4。我们称其为 SmallLocalSeparation。

问题 8.8 (SmallLocalSeparation)。**给定图** $G=(V,E,w)$、参数 σ,k和顶点 $x\in V$，要么找到一个 (x,σ,k)-集，要么保证不存在这样的集合。

值得注意的是，LocalKCut 问题是对 SmallLocalSep-aration的松弛。尽管在这两个问题中，我们都要找到一个（x，σ，k）-集，但在未能找到这样的集合时，LocalKCut 给出的保证较弱，因为我们只能排除（x，σ，k）-极端集；而 SmallLocalSeparation 给出的保证较强，因为我们必须排除所有（x，σ，k）-集。这种松弛对于 LocalKCut 的可处理性至关重要，因为我们利用了极端集所具有的结构。

引理 8.9。 问题 *8.8* 是 $W[1]$在参数 σ下为困难问题。

从高层次来看，该证明直接扩展自 问题 8.8 在无权情形下的 NP 完全证明，见 [FGK13], ，具体做法是将 dn 个顶点的基团替换为单个顶点，并将其通过权重为 d的边直接连接到原图的所有顶点。基团

在他们的构造中是保持图无权所必需的；不过，我们选择证明加权情形，代价是获得更强的困难性，即证明其为 $W[1]$-困难。

我们知道，即使在正则度为 d 的图上，以 t 为参数时， t-Clique 问题也是 $W[1]$-难的，其中 t 是团中的顶点数。

我们定义从 G' 到 G 的图构造，该构造将 t-Clique 问题归约为 问题 8.8，并用于证明 引理 8.9。

给定一个 d-正则无权图 $G = (V, E)$，作为 k-团问题的输入。 SmallLocalSeparation $G' = (V', E', w')$ 的输入构造如下。顶点集 V' 包含一个将在 SmallLocalSeparation 中用作终端的顶点 x ，因此我们称其为“终端节点”；还包含与图 G 的顶点和边相对应的节点，分别称为“顶点节点”和“边节点”。因此， $|V'| = 1 + n + m$。从 x 向所有顶点节点添加一条权重为 d 的边。因此， x 的度为 dn。对于每条边 $e = (u, v) \in E$，从边节点向 V' 中对应的顶点节点各添加一条权重为一的边。下面我们证明，图 G' 存在大小为 σ 的小型局部割，当且仅当 G 含有一个 t-团，并且 $\sigma =$poly(t)，这正是证明固定参数难解性所需的。

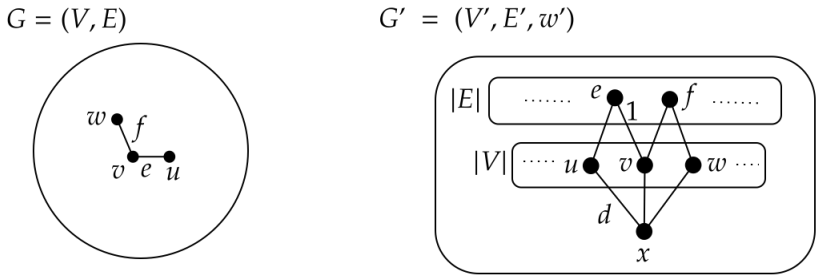

图 2： $W[1]$-困难性（ SmallLocalSeparation）

引理 8.10. 当且仅当 G 中存在一个大小为 t 的团时， $\sigma = 2 + t + \binom{t}{2} = \mathrm{poly}(t)$ 且 $k = dn - 2\binom{t}{2} + 1$ 的 (x, σ, k)-集存在。

证明. ($\Longleftarrow$) 设 S 是 G 中一个大小为 t 的团。按如下方式构造集合 S' ：它由终端节点 x、对应于 V' 中集合 S 的顶点节点，以及对应于边 $E(S, S)$ 的所有边节点组成。由于 S 是 G 中的一个团， S' 的大小为 $1 + |S| + |E(S, S)| = 1 + t + \binom{t}{2} < \sigma$。

割 $\delta_{G'}(S')$ 的大小是从 S' 发出的边的总数。与 x 相 incident 的割边权重为 $d(n - |S|) = d(n - t)$。根据我们的构造，排除连接终端节点的边后，顶点节点的度为 d 。因此，顶点节点的总度为 $d|S|$。然而，由于所有边节点 $E(S, S)$ 都包含在集合 S' 中，因此需要减去与它们相连的度。由于每条这样的边都有两个端点，所以被计算了两次。由顶点节点产生的割边权重为 $d|S| - 2|E(S, S)| = dt - 2\binom{t}{2}$。由于所有来自边节点的边都位于集合 S' 内，因此它们不会对割边作出贡献。因此， S' 的割大小为 $d(n - t) + dt - 2\binom{t}{2} = dn - 2\binom{t}{2} < k$。这证明了 S' 是一个（x， σ， k）-集合。

($\Longrightarrow$) 根据定义，若存在一个 x, σ, k-集合 S' in G'，那么它包含 x。令 V_o, E_o 分别表示集合 S' 中的顶点节点和边节点。 V_o 不能为空，因为每个边节点都会增加 2 的度数，而割大小将为 $dn + 2|E_o|$，这违反了割大小约束。因此，

$V_o \neq \emptyset$。假设 $E_o = \emptyset$。割大小将为 $|\delta_{G'}(V_o \cup \{x\})| = d(n - |V_o|) + d|V_o| = dn$，这违反了割大小约束。因此 $E_o \neq \emptyset$。

相对于所选顶点集 V_o，G 中的边分为三类：内部边 $E_G(V_o, V_o)$、割边 $E_G(V_o, V \setminus V_o)$ 和外部边 $E_G(V \setminus V_o, V \setminus V_o)$。（注意，我们对记号 V_o 的使用并不严谨；我们实际指的是 G 中对应的顶点集。）

向 S' 添加一个内部边节点会使割大小减少 2，添加一个割边节点到 S' 不会改变割大小，而向 S' 添加一个外部边节点会使割大小增加 2。因此，要将割大小从 dn 降至低于 $k = dn - 2\binom{t}{2} + 1$，我们至少应添加 $\binom{t}{2}$ 条内部边。为了满足 $|E_G(V_o, V_o)| \geq \binom{t}{2}$，$V_o$ 的大小必须至少为 t。由于集合 S' 中的节点数至多为 $1 + t + \binom{t}{2}$，V_o 中的顶点数不能超过 t，因此其数量恰为 t。这意味着 $|E(V_o, V_o)| = \binom{t}{2}$。因此，$G$ 中与 V_o 对应的顶点集必须诱导出一个大小为 t 的团。 □

根据上述归约，我们现在可以证明 $W[1]$-困难性。观察可知，集合 S' 在 G' 中不是极端的，因为任意边节点的度割大小均为 2，小于割 $\delta_{G'}(S')$ 的大小。因此，这种困难性证明不适用于问题 1.4。

引理 *8.9* 的证明。给定 t-Clique 的一个实例，d-正则无权图 $G = (V, E)$ 以及参数 t，我们按上文所示构造一个加权图实例 $G' = (V', E', w')$，并设置参数 $x, \sigma = \mathrm{poly}(t)$ k。

根据引理 8.10，任何以 σ 为参数的问题 8.8的 FPT 算法都可以转化为针对 t-Clique 的 FPT 算法，因为 $\sigma = \mathrm{poly}(t)$。由于 t-Clique 是 $W[1]$-困难的，问题 8.8 关于参数$W[1]$也是 σ-困难的。 □

9 近似边强度

在本节中，我们证明推论 1.2。

定义 9.1（边强度）。图 G 中边 $e =$（u, v）的强度，是满足以下条件的最大 k：存在一个包含 u, v 的集合 $S \subseteq V$，使得 $G[S]$ 是 k-边连通的。

因此，如果一条边的两个顶点属于最大 k-边连通划分中的同一个集合，那么该边的强度至少为 k。我们知道，任意一条边可能具有的最大强度为 $\Delta \leq (n-1)W = \mathrm{poly}(n)$，其中 W是图中一条边的最大权重。由于移除 $n-1$ 条边即可使任意图不连通，假设 W有 $\mathrm{poly}(n)$ 的上界，则边强度也有 $\mathrm{poly}(n)$ 的上界。

引理 9.2。给定顶点的一个最大 k-边连通划分：

1. 子图中包含的边，其强度至少为 k。
2. 跨越两个子图的边的强度小于 k。

证明。令 $\mathcal{R} = \{V_1, V_2, \cdots\}$ 为顶点的极大 k-边连通划分。第一部分由定义可直接得出。

任何跨越两个划分的边 $e = (u, v)$，都不可能被一个 k-边连通连通分量包围。反设存在一个 k-边连通的子图 H，同时包含u, v。

由于这条边连接了极大 k-边连通划分中的两个连通分量，令 V_i 为包含顶点 u 的连通分量。由于 H 是 k-边连通的，且 $G[V_i]$ 是 k-边连通的，

因此 $H \cup G[V_i]$ 也是 k-边连通的，这与由 V_i 诱导的 k-边连通子图的极大性矛盾。因此，这样的子图 H 不存在。因此，任何跨分区边的强度都小于 k，从而证明了该引理的第二部分。

□

因此，我们先近似每条边的强度，再将其限定在 $(1+\varepsilon)$ 的区间内。

推论 *1.2* 的证明推论 *1.2*： 图所能达到的 k -边连通的最大值 k有一个 poly(n) 的上界。因此，将空间按 $(1+\varepsilon)$ 的幂次划分，可得到 $O(\frac{1}{\varepsilon}\log n)$ 个值。

我们调用 *Algorithm 5* 中给出的子程序，令 k 取 $(1+\varepsilon)$ 的所有幂次。根据引理 9.2，我们可以求出每条边强度的上界和下界，误差至多为 $(1+\varepsilon)$ 倍。

因此，总运行时间至多为 $\tilde{O}(m \cdot \min(m^{3/4}, n^{4/5})/\varepsilon)$。 □

致谢

我们感谢 感谢 Bundit Laekhanukit 在本项目早期阶段进行富有洞察力的讨论 ct.

秩函数及其保证

证明 断言 *4.6*。 我们先从一个热身例子开始。设 e_1、e_2、e_3 是三个边，其样本值分别为 x、y、z，在 $[0,1]$中。那么 x 在这三个值中取最小值的 x、y、z概率为

$$\Pr(x = \min(x,y,z)) = \int_0^1 \int_x^1 \int_x^1 dz\, dy\, dx = \frac{1}{3}$$

因此，在 x、y、z 中选择秩最小的边，等同于选择一条均匀随机边。现在，让我们看看如何将其推广到带权重的边。

由于权重为整数，现在每条边根据其权重选择相应数量的样本，并像热身例子中那样选择随机值最小的边。因此，与热身例子类似，某个特定样本取最小值的概率为

$$\text{Probability of any sample value is minimum} = \frac{1}{\sum_f w(f)}$$

然而，由于每条边都有多个样本，只要某条边的 w 个（e）样本中的任意一个取最小值，我们就选择边 e 。因此，选择边 e 的概率为

$$\text{Probability of any } w(e) \text{ samples being minimum} = \frac{w(e)}{\sum_f w(f)}$$

.

这等同于按加权概率选择一条边。请注意，所需的随机样本数量为 $\sum_f w(f)$；当权重很大时，这个数量可能非常大。

我们可以通过采样单个随机变量，来模拟多个随机样本中最小值的随机分布，从而解决这个问题。设 $X = \min(S_1, S_2, \cdots, S_w)$ 为需要为权重为 w 的边计算的随机变量，其中 S_i 是均匀

分布于 $[0,1]$ 中的随机样本。 X 的累积分布函数如下 .

$$\begin{aligned}\Pr(X > t) &= \Pr(S_1 > t \text{ and } S_2 > t \text{ and } \cdots S_w > t)\\ &= (1-t)^w\\ \Pr(X \le t) &= 1 - \Pr(X > t)\\ &= 1-(1-t)^w\end{aligned}$$

我们可以如下模拟权重为 w 的边的 X。令 R 为 $[0,1]$ 中的均匀随机变量。对于任意 $p \in [0,1]$, $\Pr(R \le p) = p$。定义 $Y = f(R) = 1-(1-R)^{1/w}$，我们有

$$\begin{aligned}\Pr(R \le p) = p = \Pr(Y \le 1-(1-p)^{1/w}) &\qquad (\text{as } Y = f(R))\\ \Pr(Y \le t) = 1-(1-t)^w &\qquad (\text{取 } 1-(1-p)^{1/w} = t)\end{aligned}$$

因此，我们得到了一个新的随机变量 Y，其累积分布函数与 X 相同，并且只需将单个随机样本 R 代入给定函数，就能用一个随机样本生成该变量；这个函数其实就是随机变量 X 的逆概率分布。 □

我们已经证明，可以使用秩函数对随机加权边进行采样。然而，我们考虑的是在区间 $[0,1]$ 中对实数进行随机采样，而这无法通过计算机实现，因此我们需要对空间进行离散化，并在 $[0,1]$ 中对离散值进行采样，由此产生采样误差。

我们证明，将空间划分为非常细小的离散部分 $\frac{1}{\text{poly}(n)}$，可以以高概率限制秩的误差。由于边的秩误差很小，任意两条边的秩次序因这一误差而改变的概率也很小。通过界定所有边对发生错误的概率，我们证明，离散化空间 $[0,1]$ 导致边的秩次序改变的概率非常小。

首先，我们将空间 $[0,1]$ 离散化为 n^{10} 个部分，在 $t \in [0, n^{10}]$ 中采样一个数，并返回 $\frac{t}{n^{10}}$。回想一下，我们对均匀随机值 $x \in [0,1]$ 进行采样，并使用函数 $r(e) = 1-(1-x)^{1/w(e)}$。由于将空间离散化，样本 x 的误差至多为 $\frac{1}{n^{10}}$。因此，$dx = \frac{1}{n^{10}}$

$$\begin{aligned}\frac{dr(e)}{dx} &= \frac{1}{w(e)} \cdot (1-x)^{\frac{1}{w(e)}-1}\\ &= \frac{1-r(e)}{w(e)(1-x)}\\ &\le \frac{1}{1-x} &\qquad (w(e) \ge 1 \text{ and } r(e) \in [0,1])\\ &\le n^3 &\qquad (\text{when } x \le 1-\tfrac{1}{n^3})\\ dr(e) &\le \frac{1}{n^7}\end{aligned}$$

为了界定 $r(e)$中的误差，我们忽略 x的某个范围。出现这种 x 的概率至多为$\frac{1}{n^3}$。对于所有边所选取的均匀样本，这一点都必须成立。

其值至多为 $O(n^2)$。因此，对所有这些边应用并集界，$r(e)$中的误差超过$\frac{1}{n^7}$ 的概率至多为 $\frac{m}{n^3} \leq \frac{1}{n}$。因此，高概率下，我们可以假设，由于离散化空间， $r(e)$中的误差非常小。

我们还没有完成。尽管秩的误差很小，但我们仍需界定这样一个事件发生的概率：由于边的秩存在误差，边按秩排序的顺序发生改变。我们证明这种情况发生的概率极低。总体思路是，由于秩误差很小，若要让另一条边改变排序，它的取值就必须落在误差所覆盖的小范围内，而这种情况发生的概率极低。

设 e, f 是两条边。我们已在上文证明， $r(e)$ 的误差至多为 $\frac{1}{n^7}$。若要使 f 相对于 e 改变顺序， $r(f)$ 必须落在 $\frac{1}{n^7}$ 的一个范围为 $r(e)$ 的区间内。因此，我们需要界定范围较小的定义域大小。对于某个常数 c，有 $dr(f) \leq \frac{c}{n^7}$。

$$\begin{aligned} \frac{dr(f)}{dx} &= \frac{1}{w(f)} \cdot (1-x)^{\frac{1}{w(f)}-1} \\ dx &= w(f) \cdot (1-x)^{1-\frac{1}{w(f)}} \cdot dr(f) \\ &\leq \frac{c}{n^5} \qquad (1 \leq w(f) \leq n^2) \end{aligned}$$

对所有可能的边对应用并集界，边对数量至多为 $O(n^4)$，因此错误概率至多为 $O(n^4) \cdot \frac{c}{n^5} \leq \frac{c}{n}$。所以，以高概率，不会有边对因离散化空间而改变其秩顺序。

因此，秩误差较大或任意一对边因秩误差而交换顺序的事件，其总错误概率至多为 $O(\frac{1}{n})$。因此，我们认为，以高概率，边的秩顺序不会因空间离散化而改变。由于在选择要收缩的边时，我们只关心边的秩顺序，所以该过程以高概率输出一条随机边。

B 极小极端集的局部算法

根据 [CLP22] ，我们知道图中的所有极端集构成一个层状族。因此，所有包含某个特定顶点（例如 x ）的极端集彼此包含。于是，我们将包含特定顶点 k的极小 x -极端集定义如下。

定义 B.1 (极小 (x,k)-极端集)。 包含顶点 x 的 k-极端集 S 称为极小 (x,k)-极端集，如果不存在包含 x 的 S 的真子集是 k-极端集。

请注意，如果图 G 不是 k-连通的，即 G 的最小割为 $< k$，则可保证其存在。

类似地，我们可以定义极小 (x, ν, σ, k)-极端集 。然而，这不一定能保证存在，因为 k-极端集的体积不一定是 $< \nu$，其基数也不一定是 $< \sigma$。

在介绍如何寻找最小的（x, ν, k）极端集的算法之前（如果它存在），我们需要考察极端集的一些性质，这些性质有助于证明算法的正确性。根据 [CLP22] ，我们知道图中的极端集构成一个层状族。因此，所有包含 x 的极端集彼此包含。对于所有可能包含 x 的极端集，我们考虑其中满足体积、基数和割大小约束的（ν, σ, k）极端集。如果至少存在一个这样的极端集包含

x，那么在 x， ν， σ， k）个随机秩函数中，存在一个秩函数与该（ $O(\sigma^2 \log n)$）极端集相容。因此，我们可以使用类似于 算法 3的算法找到该集合。

仙人掌图是一种图，其中每条边至多属于一个简单环。根据 [DKL76] ，我们知道，图的所有最小割 G 都可以表示为仙人掌图的形式， H 其中仙人掌图的每个顶点都映射到原图中一个与其他集合不相交的顶点集， G 也可以映射到空集。删除一条非环边，或删除同一环上的两条边，在仙人掌图中产生的每个割，都会在原图中产生相应的割。每个这样的割都是原图的最小割。仙人掌图是原图所有最小割的简洁表示。当图中存在唯一的最小割时，该图的最小割的仙人掌表示就是一条边。根据 [PK09] ，我们知道，可以在近线性时间内找到图的仙人掌表示。因此，我们可以在近线性时间内判断图是否具有唯一的最小割。令IsExtreme 为一个用于检查图是否为极端集的子程序。

我们将问题正式定义如下。

问题 B.2. 给定一个图 $G = (V, E, w)$、参数 ν, σ, k 以及一个顶点 $x \in V$，寻找(x, ν, σ, k)-极小极端集；如果不存在这样的集合，则返回 $\perp$。

现在我们将介绍用于求解问题 B.2的局部算法。该算法与算法 3 类似，只是我们还会检查返回的集合是否为极端集。

算法 8： MinimalExtremeSet

数据： $G = (V, E, w)$ x， ν， σ， k

结果： (x, ν, σ, k)-极小极端集；如果不存在 (x, ν, σ, k)-极端集，则返回 $\perp$

```
S = V;
对 i ∈ [O(σ^2 对数 n)] 执行
    X = {x};
    当 vol(X) < ν 且 |X| < σ 时执行
        如果 w(δ(X)) < k 且 IsExtreme(G/X) 且 X ⊂ S 则
            S = X;
            跳出 while 循环;
        找到秩最小的边 r^i, e = (u, v) ∈ E(X, V \ X) = δ(X);
        更新 X = X ∪ {v};
如果 S ≠ V，则
    返回 S;
否则
    返回 ⊥;
```

引理 B.3。 算法 *8* 以高概率解决问题 *B.2* 。

证明。 如果不存在 (x, ν, σ, k)-极端集，那么该算法就不会将 S 从 V 更新；我们会在最后返回 $\perp$。

如果存在一个 (x, ν, σ, k)-极端集 S，那么正如引理 4.8 中所证明的， S 至少会满足 $O(\sigma^2 \log n)$ 个随机秩函数中的一个，不妨设为 r^*。因此，集合 S 将是针对 r^* 使用随机收缩得到的 $\sigma - 1$ 个集合之一。由于极端集构成一个层状族，任何其他 (x, ν, σ, k)-极端集都严格包含 S 。我们会找到 S 并返回它，而不会被任何其他极端集替换。 □

引理 B.4. 算法 *8* 的运行时间至多为 $\tilde{O}(\nu\sigma^3)$ 。

证明。 对于每个 $O(\sigma^2 \log n)$ 个秩函数，我们执行 σ 次边收缩，并且每次都检查该集合是否为极端集，这一步的耗时相对于 G/X 中存在的边数 $\tilde{}$ 为近线性，而边数至多为 $O(\nu)$。此外，寻找最小秩边还需要 $O(\nu \log \nu)$。$\tilde{}$ $\tilde{}$因此，对于每个秩函数，总耗时至多为 $O(\nu\sigma)$ $+ O(\nu \log \nu) = O(\nu\sigma)$。由于我们不知道极小极端集会遵循哪个秩函数，因此必须对每个秩函数都运行一次，使运行时间至多为 $\tilde{O}(\nu\sigma^3)$。

□

C 极大 k-边连通划分的唯一性

本节的主要目标是证明命题 3.1。

断言 C.1. 任意图 G 的极大 k-边连通子图 $\{V_1, \ldots, V_z\}$ 都是唯一的，并构成 V 的一个划分。

证明。 首先证明极大 k-边连通子图构成一个划分。任意极大k-边连通子图 H 总是 G 的一个诱导子图。也就是说，对于某个$S \subseteq V$， $H = G[S]$ 。否则，存在一条边 $e \in G[S] \setminus E$（H），且 $H \cup e$ 是 H 的真超图，并且是k-边连通的，这与 H的极大性矛盾。

因此，若两个极大 k-边连通图 $G[A]$, $G[B]$ 有交集，即 $A \cap B \neq \emptyset$。由于 $G[A]$ 和 $G[B]$ 都是 k-边连通的，因此 $G[A \cup B]$ 也是 k-边连通的，因为 $A \cap B \neq \emptyset$。这与 $G[A]$ 是极大 k-边连通的事实矛盾。因此这些集合两两不相交。每个顶点单独来看都是 k-边连通的，因此应属于某个极大 k-边连通子图。由此，它们构成一个划分。

现在我们证明它是唯一的。反设至少存在两个这样的划分。那么至少存在一个顶点 v ，它在这两个极大连通划分中属于不同的集合。再次设 A, B 为来自两个不同划分、包含 v 且都 k- $G[A]$, $G[B]$ 边连通的两个集合。使用上述相同的论证，我们证明该划分是唯一的。

□

D 边强度估计的反例

$\tilde{}$Benczur 和 Karger [BK02] 提出了一种算法，可对边强度 k_e 进行估计k_e ，使得 $\tilde{k}_e \leq k_e$ 对于所有 $e \in E$ 以及$\sum_{e \in E} \frac{1}{\tilde{k}_e} = O(n)$。这些估计是通过运行ESTIMATION($G$,1) 子程序获得的，该子程序在 [BK02]的引理 4.9 中有描述。要获得边强度的 c-近似，我们需要 $\tilde{k}_e \geq \frac{k_e}{c}$。然而，上面给出的第二个条件有所不同，并不一定能提供良好的近似保证。下面我们给出一个反例，在该例中，这些估计值与实际值相差很大。

考虑以下反例：一个棒棒糖图 G，它包含一条长度为 $n - n^{1/3}$ 的路径，并连接到一个大小为 $n^{1/3}$ 的团。注意， G 包含的边少于 2($n - 1$) 条。因此，过程 ESTIMATION(G, 1) 会将所有边的边强度估计设为 1。然而，团中的所有边的强度至少为 $n^{1/3}$。因此，团内边的边强度估计误差达到 $n^{1/3}$ 倍。

参考文献

[ACL06] Reid Andersen、Fan Chung 和 Kevin Lang。使用 PageRank 向量进行局部图划分。载于 *2006* 年第 *47* 届 *IEEE* 计算机科学基础年会（*FOCS'06*），第 475–486 页。IEEE，2006 年。4

[AIY13] Takuya Akiba、Yoichi Iwata 和 Yuichi Yoshida。通过随机收缩在线性时间内枚举大型网络中的极大 k-边连通子图。载于第 22 届 ACM 信息与知识管理国际会议论文集，第 909–918 页，2013 年。3[AKL+21] Amir Abboud、Robert Krauthgamer、Jason Li、Debmalya Panigrahi、Thatchaphol Saranurak 和 Ohad Trabelsi。次立方时间内求 Gomory-Hu 树。arXiv 预印本 arXiv:2111.04958，2021 年。6[BK02] Andr´as A. Bencz´ur 和 David R. Karger。容量图中割与流的随机近似方案。CoRR，cs.DS/0207078，2002 年。4，33[CCPS21] Ruoxu Cen、Yu Cheng、Debmalya Panigrahi 和 Kevin Sun。通过割平衡对有向图进行稀疏化。载于自动机、语言与编程国际会议，2021 年。4[CDK+21] Parinya Chalermsook、Syamantak Das、Yunbum Kook、Bundit Laekhanukit、Yang P Liu、Richard Peng、Mark Sellke 和 Daniel Vaz。用于边连通性的顶点稀疏化。载于 2021 年 ACM-SIAM 离散算法研讨会（SODA）论文集，第 1206–1225 页。SIAM，2021 年。4[CFK+15] Marek Cygan、Fedor V. Fomin、Lukasz Kowalik、Daniel Lokshtanov、D´aniel Marx、Marcin Pilipczuk、Michal Pilipczuk 和 Saket Saurabh。《参数化算法》。Springer，2015 年。5[CHI+17] Shiri Chechik、Thomas Dueholm Hansen、Giuseppe F Italiano、Veronika Loitzenbauer 和 Nikos Parotsidis。稀疏有向图中计算极大 2-连通子图的更快算法。载于第二十八届 ACM-SIAM 离散算法年会论文集，第 1900–1918 页。SIAM，2017 年。3，6，7，17[CKN20] Yu Chen 、 Sanjeev Khanna 和 Ansh Nagda。近线性规模的超图割稀疏化器。载于 2020 年 IEEE 第 61 届计算机科学基础年会（FOCS）论文集，第 61–72 页。IEEE，2020 年。4，9[CLN+22] Ruoxu Cen、Jason Li、Danupon Nanongkai、Debmalya Panigrahi、Thatchaphol Saranurak 和 Kent Quanrud。通过部分稀疏化求解有向图中的最小割。载于 2021 年 IEEE 第 62 届计算机科学基础年会（FOCS）论文集，第 1147–1158 页。IEEE，2022 年。3[CLP22] Ruoxu Cen、Jason Li 和 Debmalya Panigrahi。近线性时间内的边连通性增强，2022 年。31[CX18] Chandra Chekuri 和 Chao Xu。超图中的最小割与稀疏化。《SIAM 计算杂志》，47(6):2118–2156，2018 年。4[DKL76] E. Dinic、Alexander Karzanov 和 M. Lomonosov。图中的最小边割系统。载于：《离散优化研究》，A.A. Fridman 编，Nauka，莫斯科，第 290–306 页，俄文，1976 年 1 月。32[ES80] Ya M Erusalimskii 和 GG Svetlov。有向图的双连接点、双桥与双块。《控制论》，16(1):41–44，1980 年。3

[FGK13] Fedor V. Fomin、Petr A. Golovach 和 Janne H. Korhonen。关于从图中删除少量顶点的参数化复杂性。CoRR，abs/1304.6189，2013。8，23，26[FHHP19] Wai-Shing Fung、Ramesh Hariharan、Nicholas JA Harvey 和 Debmalya Panigrahi。图稀疏化的通用框架。SIAM 计算杂志，48(4)：1196–1223，2019。4[FNY+20] Sebastian Forster、Danupon Nanongkai、Liu Yang、Thatchaphol Saranurak 和 Sorrachai Yingchareonthawornchai。通过快速局部割算法，以近线性时间和查询计算及测试小连通度。载于第十四届 ACM-SIAM 离散算法研讨会论文集，第2046–2065页。SIAM，2020。3，4，6，7[Gab95] Harold N Gabow。一种用于寻找边连通度和打包树形结构的拟阵方法。计算机与系统科学杂志，50(2)：259–273，1995。3[GR97] Oded Goldreich 和 Dana Ron。有界度图中的性质测试。载于第二十九届 ACM 计算理论研讨会论文集，第406–415页，1997。4，5[HKL15] Monika Henzinger、Sebastian Krinninger 和 Veronika Loitzenbauer。以二次时间寻找2-边和2-顶点强连通分量。载于自动机、语言与编程国际研讨会论文集，第713–724页。Springer，2015。3[HKNS15] Monika Henzinger、Sebastian Krinninger、Danupon Nanongkai 和 Thatchaphol Saranurak。通过在线矩阵向量乘法猜想统一并加强动态问题的困难性。载于第四十七届 ACM 计算理论研讨会论文集，第21–30页，2015。5，8，23[HRG00] Monika R Henzinger、Satish Rao 和 Harold N Gabow。计算顶点连通性：基于旧技术的新界。算法杂志，34(2)：222–250，2000。3[HRW20] Monika Henzinger、Satish Rao 和 Di Wang。用于加速边连通度计算的局部流划分。SIAM 计算杂志，49(1)：1–36，2020。4[Jab16] Raed Jaberi。关于计算有向图的2-顶点连通分量。离散应用数学，204：164–172，2016。3[JHSZ22] Han Jiang、Shang-En Huang、Thatchaphol Saranurak 和 Tian Zhang。用于超边连通度的顶点稀疏器，2022。4[Kar93] David R. Karger。RNC 中的全局最小割，以及简单最小割算法的其他推论。载于Vijaya Ramachandran编，第4届 ACM/SIGACT-SIAM 离散算法研讨会论文集，1993年1月25日至27日，美国得克萨斯州奥斯汀，第21–30页。ACM/SIAM，1993。5，7，10，12[Kar00] David R Karger。近线性时间的最小割。ACM 杂志（JACM），47(1)：46–76，2000。3，10[KS96] David R Karger 和 Clifford Stein。一种求解最小割问题的新方法。ACM 杂志（JACM），43(4)：601–640，1996。12

[LNP+21] Jason Li、Danupon Nanongkai、Debmalya Panigrahi、Thatchaphol Saranurak 和 Sorrachai Yingchareonthawornchai。多对数最大流中的顶点连通性。载于第53届 ACM SIGACT 计算理论研讨会论文集，第317–329页，2021。3[Mak88] Seishi Makino。一种寻找有向图中所有 k-分量的算法。《国际计算数学杂志》，24(3-4)：213–221，1988。3[NSWN17] Danupon Nanongkai、Thatchaphol Saranurak 和 Christian Wulff-Nilsen。具有次多项式最坏情况更新时间的动态最小生成森林。载于2017年 IEEE 第58届计算机科学基础年度研讨会（FOCS），第950–961页。IEEE，2017。4[NSY19] Danupon Nanongkai、Thatchaphol Saranurak 和 Sorrachai Yingchareonthawornchai。突破小顶点连通度问题的二次时间界及一种近似方案。载于第51届 ACM SIGACT 计算理论研讨会论文集，第241–252页，2019。4，6[OR11] Yaron Orenstein 和 Dana Ron。测试有向稀疏图的欧拉性和连通性。《理论计算机科学》，412(45)：6390–6408，2011。4，6[OZ14] Lorenzo Orecchia 和 Zeyuan Allen Zhu。用于局部图聚类的基于流的算法。载于第二十五届 ACM-SIAM 离散算法研讨会论文集，第1267–1286页。SIAM，2014。4，6[PK09] Debmalya Panigrahi 和 David Karger。一种用于构造最小割仙人掌表示的近线性时间算法，2009年1月。32[PR02] Michal Parnas 和 Dana Ron。测试图的直径。《随机结构与算法》，20(2)：165–183，2002。4，5[ST13] Daniel A Spielman 和 Shang-Hua Teng。一种适用于大规模图的局部聚类算法及其在近线性时间图划分中的应用。SIAM 计算杂志，42(1)：1–26，2013。4[ST14] Daniel A Spielman 和 Shang-Hua Teng。用于预条件和求解对称对角占优线性系统的近线性时间算法。《SIAM 矩阵分析与应用杂志》，35(3)：835–885，2014。4[SW19] Thatchaphol Saranurak 和 Di Wang。扩张子分解与剪枝：更快、更强、更简单。载于第三十届 ACM-SIAM 离散算法研讨会论文集，第2616–2635页。SIAM，2019。4[Tar72] Robert Tarjan。深度优先搜索与线性图算法。SIAM 计算杂志，1(2)：146–160，1972。3[YI10] Yuichi Yoshida 和 Hiro Ito。测试有向图的 k-边连通性。《系统科学与复杂性杂志》，23(1)：91–101，2010。4，6[YI12] Yuichi Yoshida 和 Hiro Ito。图的 k-顶点连通性性质测试。《算法学》，62(3)：701–712，2012。4

[ZLY+12] Rui Zhou、Chengfei Liu、Jeffrey Xu Yu、Weifa Liang、Baichen Chen 和 Jianxin Li。从大型图中寻找最大 k 边连通子图。载于 第*15*届数据库技术拓展国际会议，第 480–491 页，2012 年。 3