通过稀疏化在幂图上实现分布式对称性破缺

Yannic Maus · yannic.maus@ist.tugraz.at · 格拉茨工业大学

Saku Peltonen · saku.peltonen@gmail.com · 阿尔托大学

Jara Uitto · jara.uitto@aalto.fi · 阿尔托大学

arXiv:2302.06878v1 [cs.DS] 14 Feb 2023

摘要。 本文介绍了幂图中经典对称性破缺问题——极大独立集（MIS）和支配集——的高效分布式算法。我们采用标准的 CONGEST 分布式消息传递模型，其中通信网络被抽象为图 G。通常，CONGEST 中的问题实例与通信网络 G相同，也就是说，我们在 G中执行对称性破缺。本文考虑的问题实例对应于幂图 G^k的情形，其中通信网络 G 中的每个节点都与其 k跳邻居相连。

β-支配集是一个由非邻接节点组成的集合，使得 G 中的每个节点都能在 β 跳内找到一个支配邻居；它是极大独立集的一种自然推广。支配集（在幂图中）不仅构成一类自然的问题，也因其在强大的碎裂框架中的应用而具有充分的研究动机 [BEPS《ACM 汇刊》'16、Ghaffari ACM-SIAM 离散算法研讨会'19]（以及其他工作）。我们提出了计算 G^k 的极大独立集和支配集的随机化算法，其运行时间基本与在 G中计算它们所需的时间相同。本文的主要贡献是一种用于计算 G^k的k-支配集的确定性多对数时间算法；对于 $k > 1$，它将当前最先进算法的运行时间呈指数级缩短。该结果的核心技术是一个可能具有独立研究价值的确定性稀疏化过程。

我们还重新审视了用于求极大独立集的碎裂算法 [BEPS J'ACM'16] ，并提出了后碎裂阶段的不同方法。与现有方案相比，我们的方法在算法设计和分析上都更简单（在 LOCAL 模型中亦如此），且达到了与 [Ghaffari SODA'16] 相同的运行时间。

目录

1 引言 **1**

1.1 我们的贡献.2 1.2 为什么要关注幂图上的问题？.4 1.3 其他相关工作.5

2 记号与 k 元独立随机变量 **7**

3 技术概述 **9**

4 通信工具 13

5 幂图的稀疏化 **16**

5.1 通过抽样进行随机化稀疏化.17 5.2 通过去随机化实现确定性稀疏化.19 5.3 幂图中的稀疏化.22 5.4 不依赖直径的稀疏化.25

6 确定性（$k+1, k^2$）-支配集（定理 1.1） **27**

7 图 G上的极大独立集：重新审视碎裂（定理 1.4 ） **29**

7.1 碎裂基础.30 7.2 后碎裂.31 7.2.1 方法一：两个预碎裂阶段.32 7.2.2 方法二：一个预碎裂阶段.33 7.3 [BEPS16、arXiv] 中的方法（或许 [Gha16、Gha19]中也采用了这一方法）. 35

8 幂图上的随机对称性破缺（定理 1.2，推论 1.3）37

8.1 幂图上的 Luby 算法.37 8.2 G^k 的极大独立集（定理 1.2）.37 8.3 G^k 中的支配集（推论 1.3）.42

幂图的网络分解 49

B 第 5.3 节的伪代码 53

1 引言

本文为幂图上的对称性破缺问题提供了高效的确定性和随机化算法，也就是说，对于整数 k，我们在幂图 G^k 上计算极大独立集和支配集，其中 G 是输入图。为说明这一设定，我们先定义本文的核心问题。图 G 的极大独立集（*MIS*） S 是由两两不相邻的节点组成的集合，并且图 G 中的每个节点 v 都由集合 S 中的某个节点支配，也就是说， S 中存在一个与 v 距离至多为 1 的节点。支配集通过放宽支配距离的限制，推广了这一概念。在幂图 G^k 中， G 中任意两个相距至多 k 跳的节点都相连。因此，在 G^k 的极大独立集中，任意两个节点的距离至少为 $k+1$，并且 G 中的每个节点都由 S 中距离至多为 k 的某个节点支配。极大独立集和支配集已在经典的分布式计算消息传递模型（即 LOCAL 和 CONGEST 模型）中得到广泛研究，例如参见 [Lub86、ABI86、BEPS16、AGLP89、Gha16、Gha19、SEW13、KMW18、BBKO22]。

理解幂图上的对称性破缺至关重要，因为幂图自然地出现在各种场景中。频率分配问题就是一个经典例子。为了避免无线发射器网络中的干扰，我们希望为通信网络中的节点分配频率，使每个节点的所有邻居都接收不同的频率。这个问题是幂图 G^2 [KMR01, HKM20, HKMN20] 上的顶点着色问题。解决 G 的问题时，也会将幂图问题作为子程序。一个例子是最先进的随机化算法，它通过计算幂图的支配集来求解极大独立集（属于 G）；该算法适用于 LOCAL 模型 [BEPS16, Gha16] 和 CONGEST 模型[Gha19]。另一个例子是当前最先进的计算极大独立集的确定性算法 [FGG+22]。第三个将此类支配集作为子程序的例子，是计算 [EM19] 中生成器的算法。在第 1.2 节中，我们将进一步举例，并进一步阐释幂图上的（对称性破缺）问题。本文的主要模型是分布式计算的 CONGEST 模型。

在 CONGEST 模型中处理幂图的挑战。 在 LOCAL 和 CONGEST 模型中，通信网络被抽象为一个具有 n 个节点的图G G，其中节点代表计算实体，边代表通信链路 [Lin92, Pel00]。节点配有 $O(\log n)$ 位标识符。为了在网络中解决某个问题，节点会通过同步轮次彼此通信。在每一轮中，节点可以执行任意本地计算，并向 G 中的每个邻居发送消息。在 LOCAL 模型中，消息大小不受限制，而在 CONGEST 模型中，消息大小被限制为 $O(\log n)$ 位。算法的时间复杂度 是指每个节点计算出自身解决方案部分所需的轮次数，例如判断自己是否属于某个独立集。传统上，文献中的算法（例如解决最大独立集问题的算法）都假设通信网络本身就是问题实例。相比之下，在我们的工作中，幂图 G^k 是问题实例，而 G 仍然是通信网络。在 LOCAL 模型中，这种设定不会带来重大困难，因为为 G设计的算法（例如计算 G 的极大独立集），可以在 G^k 上运行，乘法开销为 k 轮（以计算 G^k 的极大独立集）。然而，在 CONGEST 模型中，这一结论并不成立。事实上，在 CONGEST 模型中，节点甚至不知道自己在问题实例 G^k 中的度；如果许多顶点都想向 G^k 中的邻居发送不同的消息，通信网络就会出现拥塞，消息也无法高效传递。这看似微小的差异影响巨大，使得构造算法更具挑战性。

1.1 我们的贡献。

图G G 的 r-支配集 S 是一个由非邻接节点构成的集合，并且对于图G G 的每个节点 v，在 S 中都存在一个与之距离至多为 r 的节点 v。因此，在 r-支配集里， G^k 中任意两个节点的距离至少为 $k+1$，且支配距离为 $r \cdot k$。文献中常将其表示为 (α, β)-支配集，其中 α 指定节点之间的最小距离， β 指定支配距离。由于支配集放宽了极大独立集的支配保证，因此通常更容易计算。在算法应用中，支配集往往足以取代极大独立集计算，作为子程序使用。我们的主要贡献是提出首个用于计算 G^k 的 k-支配集的高效确定性算法。本文中， $\widetilde{O}(x)$ 表示省略了关于 x 的对数因子，而 Δ 表示图G G 的最大度，即使我们求解的是幂图 G^k 上的问题也是如此。

定理 1.1 (k-距离支配集 G^k)。设 $k \geq 1$ 为一个整数（可能是 n的函数）。存在一种确定性分布式算法，可在 CONGEST 模型中用多对数时间计算 G^k 的一个 k-距离支配集。更具体地说，其轮复杂度为 $\widetilde{O}(k^2 \cdot \log^4 n \cdot \log \Delta)$ 轮。

对于常数 $k > 1$，定理 1.1 将此前最先进水平的复杂度提升了指数级；此前的算法需要 $O(n^{1/k})$ 轮 [EM19]。该先前算法是针对 G^k 的扩展，源自一种经典确定性算法；该算法可在 $O(B \cdot \log_B n)$ 轮内计算 G的 $(\log_B n)$-距离支配集 [AGLP89、SEW13、HKN21、KMW18、EM19], ，其中 B 是一个可用于权衡支配能力与运行时间的参数。

对于 $k = 1$，定理 1.1 计算一个 (2,1) 支配集（也称为极大独立集）；对于 $k = 2$，它计算一个 (3,4) 支配集；对于 $k = 3$，它则是一个 $(4, 9)$ 支配集。为证明定理 1.1，我们的主要技术贡献是一种新颖的稀疏化过程。从非常高的层面来看，我们将 G^k 转化为一种稀疏得多的表示 $\bar{G}$，使得 $\bar{G}$ 的一个极大独立集能够成为原始 G^k上的一个良好支配集。¯的好处是，我们可以在 G上更高效地通信。

幂图中的随机对称性破缺。我们的第二个结果是一个用于计算 G^k 的极大独立集的算法，其运行时间与当前计算 G的极大独立集的最佳结果基本相同。

定理 1.2。存在一种随机分布式算法，可以在 $(k^2 \log \Delta \cdot \log\log n + k^4 \log^5 \log n)$ 轮次内，在 CONGEST 模型中以高概率计算出 G^k 的极大独立集。

通过先前工作能够实现的最佳结果，是一种基于 Luby 算法、用于计算极大独立集的 $O(k \log n)$ 轮次算法（详见第 8.1 节） [Lub86, ABI86]。定理 1.2 相较于当前在 CONGEST 模型中计算 G 的极大独立集所能达到的最佳结果更具优势，该结果需要 $O(\log \Delta \log\log n + \text{poly}\log\log n)$ 轮次 [Gha19]。在 LOCAL 中， G^k 上极大独立集的随机复杂度为 $O(k^2 \log \Delta + k \,\text{poly}\log\log n)$，这只需在幂图 [上运行已知的用于计算 G 上极大独立集的最佳算法即可 16]Gha。结合定理 1.2 与 [KP12] 和 [BKP14] 的稀疏化方法，我们得到以下推论。

推论 1.3。存在一种随机分布式算法，可在 β轮内计算 G^k 中$\widetilde{O}(\beta \cdot k^{1+1/(\beta-1)}(\log \Delta)^{1/(\beta-1)} + \beta \cdot k \log\log n + k^4 \log^5 \log n)$，得到 G^k 的 β-支配集。

此前唯一用于计算 G^k 支配集的随机化算法，是一种 $O(k \log\log n)$-支配集算法，可在 $O(k^2 \text{ loglog } n)$轮次内运行 [Gha19]。这种技术本质上无法计算 $O(1)$-支配集 G^k。

简化 G **上的** 极大独立集和支配集碎裂问题（适用于 LOCAL **和** CONGEST）。**幂图支配集对于当前最先进的随机化算法至关重要，这些算法用于在 LOCAL 模型中计算极大独立集 BEPS16, Gha，以及在 CONGEST 模型中计算极大独立集 Gha。在我们的最后一项贡献中，我们重新审视当前** 最先进的算法，这些算法用于在 G 中、LOCAL 模型下计算极大独立集 [BEPS16, Gha16], 以及在 CONGEST 模型下计算极大独立集 [Gha19]。这些结果基于如今已成为标准的碎裂技术：该技术利用随机过程解决图中大部分区域的问题，使剩余未解决的部分仅由（许多）小型连通分量构成。所谓连通分量小，是指单个连通分量的某个幂图支配集至多包含对数数量的节点。利用这一性质，可以高效地解决这些小型连通分量中剩余的问题。难点在于，若要通过算法利用这些支配集规模较小这一性质，就必须计算出这些支配集。直观而言，人们只需在每个剩余连通分量上并行运行经典的 $(5, O(\log\log n))$-支配集算法即可。关键问题是，计算支配集规模时，必须在 G 中度量支配集节点之间的距离，而不能只在小型连通分量内部度量。但节点无法高效判断 G 中距离为 5 的节点是否与自身属于同一个小型连通分量。因此，节点难以判断它们是否都能包含在支配集中，也就无法直接将不同的小型连通分量独立处理。

因此， [BEPS16] 的 arXiv 版本提出了一个复杂的两阶段碎裂过程，并进行了繁复的分析。事实证明，其中关于支配集大小上界的分析存在一个小但关键的错误（详情见第3节和第7节）。相关工作的期刊版本通过更加复杂的分析修正了这一问题 [BEPS16]。后续工作[Gha16, Gha19] 以 [BEPS16]的 arXiv 版本为基础。虽然我们认为这两项工作都可以调整为以 [BEPS16], 期刊版本中的论证为基础，但这无法通过黑盒方式实现。这两项工作采用了与 [BEPS16] 不同的支配集算法，而支配集算法的内部细节对于 [BEPS16]中的论证至关重要。

作为最后一项贡献，我们重新审视碎裂框架，并提出两种不同的算法和分析方案，用于在 LOCAL 和 CONGEST 模型中计算极大独立集。虽然这些方案并未降低 [BEPS16, Gha16, Gha19], 的复杂度，但我们的方法更简单，并修正了基于 [BEPS16]的 arXiv 版本的结果。

定理 1.4。 存在随机化的 LOCAL 和 CONGEST 算法，能够以高概率在任意具有 n 个节点、最大度为 Δ 的图上计算极大独立集；它们的运行轮次分别为 $O(\log\Delta)+\text{poly}\log\log n$ 轮和 $O(\log\Delta\log\log n)+\text{poly}\log\log n$ 轮。

尽管定理 1.4 讨论的是经典情形，即计算图G G的极大独立集（而非其幂图），但当前论证中的缺陷与如何恰当地处理幂图中的支配集密切相关。文献中的其他一些研究建立在 [BEPS16], 的 arXiv 版本的思想之上，例如 Δ着色 [GHKM21] 的算法，或 Lov´asz 局部引理[FG17], 的算法，因此也存在类似问题。不过，我们注意到，前述幂图支配集只有在 $\Delta\geq\text{poly}\log n$时才有意义；否则，可以采用更简单的方法来利用连通分量较小这一特点。但对于 $\Delta\geq\text{poly}\log n$的情形，无需修正这两篇论文中的缺陷。对于 Δ着色问题，最近的一篇论文提出了一种截然不同的算法，可在 $\Delta\geq\text{poly}\log n$时解决该问题，因此无需修正 [FMH22]。对于 Lov´asz 局部引理问题，可以使用 Chung、Pettie 和 Su 的算法来

[1]我们重新审视了这些算法的细节，因为它们依赖于我们也用于证明定理 1.2 的碎裂框架。不过，定理 1.2 中的情形更为复杂。

在 $O(\log\ n)$轮次内解决该问题 [CPS17b], , 这比 $\Delta \geq \text{poly}\log\ n$时 [FG17] 所需的 $O(\Delta^2)\ +$ $\text{poly}\log\log\ n$轮次更快。最后需要指出的是，当前最先进的随机化$\Delta+1$着色算法不受这一缺陷影响，因为它们仅在 $\Delta \leq \text{poly}\log\ n$ [时才依赖碎裂21]CLP20、HN。关于定理 1.4 的背景及我们的解决方案，详见第3节和第7节。请注意，定理 1.4 对文献中的无数结果也至关重要，因为该算法属于这一领域中使用最广泛的子程序。定理 1.2 和推论 1.3 也依赖于对用于定理 1.4 的算法思想的扩展。

1.2 为什么我们要关注幂图上的问题?

我们已经提到，幂图上的支配集是重要的子程序。本节将进一步介绍研究幂图上的 CONGEST 算法的各种理由。

去随机化与学习远距离信息。处理幂图时的主要挑战在于，尽管我们的目标是解决 G^k 上的问题，但通信网络本身只是 G ，节点无法直接与 G^k 中的邻居通信。举例来说，考虑这样一个简单任务：从每个邻居处获取一小段信息。在 G中，这个问题轻而易举，显然只需一个 CONGEST 轮次即可完成。然而，一旦转到 G^2，轮次数就会大幅增加。最坏情况下，完成此任务需要 Ω(Δ) 轮次，其中 Δ是图的最大度，可能非常大。

但是，获取并非存储在 G 的直接邻居处的大量信息，是近期高效 LOCAL 算法中的一个重要环节。这类做法的典型例子包括 LOCAL 模型中的一般去随机化结果 [GKM17、GHK18、RG20] 。请注意，除了获取大量信息之外，这些结果的另一个关键环节是计算幂图 G^k 的所谓网络分解，通常针对非常数 k。该领域一个重要的开放问题是，确定哪些类型的问题可以在 CONGEST 模型中实现此类去随机化。

积极的一面是，对于图 G上的一些特定问题，存在高效的 CONGEST 算法，它们都基于限制为高效地对（简单的）随机过程进行去随机化而必须汇总的信息量。例如，$(\Delta+1)$-着色 [BKM20、GK21], 极大独立集 [CPS17a、FGG+22], 以及最小支配集近似 [DKM19、FGG+22]的算法都属于此类。我们的稀疏化结果同样基于去随机化（但处于更极端的情形），因此扩展了可在 CONGEST 模型中高效去随机化的问题类别。第3节将详细说明，我们如何绕过稀疏化过程在 LOCAL 模型中所需的大量信息的获取。

将幂图和虚拟图问题作为子程序。近期许多研究成果的另一项常见要素，是在虚拟图上解决中间问题。例如，在 LOCAL 和 CONGEST 中最先进的确定性极大独立集算法里，我们必须在 G^2 [FGG+22] 上模拟一个算法（用于某个中间问题）。因此，尽管我们考虑的是通信网络等于输入图的情形，中间步骤仍然需要解决幂图上的问题。在 LOCAL 中，无界的消息大小使我们能够以常数开销进行通信，因此，虚拟图上的中间步骤算法 G^2 可以直接实现，且运行时间只增加常数开销。相比之下，在 CONGEST 模型中处理这些中间步骤要复杂得多。总的来说，我们认为，对幂图问题的研究有助于探讨一个普遍而重要的主题：将输入图与通信网络分离。幂图上的问题

为开发工具提供了一种清晰的抽象；当问题实例与通信网络并不完全相同时，这些工具便可派上用场。

算法和技术的鲁棒性。 更广泛地说，研究幂图上的问题也提供了一种简洁的抽象，可用于开发具有鲁棒性 的算法技术，即这些技术能够在通信限制更严格的环境中发挥作用。长远来看，我们也期望这类研究能带来模型依赖性更低的算法。例如，许多近期的 MPC 算法都是通信高效的 LOCAL 或 CONGEST 算法的快速实现 [GU19, CDP20, CFG+19, CDP21, BBF+22]。通信效率的极端例子是在蜂鸣模型中的算法；该模型类似于 CONGEST 模型，但消息只有 1 位。一个出色的例子是 Ghaffari 极大独立集算法中的预碎裂阶段：在为 G 计算极大独立集时，该阶段可在蜂鸣模型中运行 [Gha17]。由于设计鲁棒，这一方法经过少量调整后，已被用于在其他环境（如 MPC 和 LCA）中取得最先进的结果 [GU19, Gha22]。定理 1.2 也基于这一结果的扩展，该扩展适用于 G^k，并可在更强的蜂鸣模型版本中运行，允许使用大小为 $O(\log\log n)$ 的消息。详情请参见第8节。

使幂图算法更稳健的另一个方面是，这些算法必须适用于节点不知道自身度数的情形（在 G^k 中）。尽管有办法从算法中去除已知自身度数这一要求 [KSV13], , 但这些办法并不能在所有情形下都以黑盒方式应用。

1.3 其他相关工作

近年来，针对 CONGEST 模型中的幂图问题出现了若干结果，涵盖了高效验证解（或证明这无法实现） [FHN20], , 以及回答该模型下的若干计算问题，例如前文讨论过的距离-2着色问题 [HKM20,HKMN20]或优化问题 [BCM+20]。

支配集。我们首先关注 LOCAL 模型中的结果。我们已经讨论过 [SEW13、HKN21、KMW18] 提出的确定性支配集算法。事实上，该结果更为一般：如果图配备了一个使用 C 种颜色的顶点着色，那么它能在 $O(B\cdot\log_B C)$ 轮内求得一个 $O(\log_B C)$-支配集，其中 G。通过适当选择参数 B，并将其与计算 $O(\Delta^2)$-着色的 Linial算法结合，可在 $O(\log^* n)$ 轮内求得一个 β-支配集$O(\beta\cdot\Delta^{2/\beta}+\log^* n)$ 轮。

Gfeller 和 Vicari [GV07] 提出了一种随机化稀疏化算法，用于计算一个 $O(\log\log n)$-支配集[2] $S\subseteq V$，所需轮次为 $O(\log\log n)$，且 $G[S]$ 的最大度为 $O(\log^5 n)$。将其与前述方法结合，可得到一种随机化的 $O(\log\log n)$-支配集算法，其轮复杂度为 $O(\log\log n)$。Ghaffari 将这一方法扩展到 G^k CONGEST 模型，并在 $O(k^2\log\log n)$轮内得到一个 $O(k\log\log n)$-支配集 [Gha19]。

Kothapalli 和 Pemmaraju [KP12] 提出了另一种稀疏化方法，可在 $O(\log\Delta/\log\Delta')$轮内计算一个度为 $O(\Delta'\log n)$的支配集。Ghaffari 使用这种稀疏化方法进行多次迭代，在 LOCAL 模型中于 $O(\beta\log^{1/\beta}\Delta)+\text{poly}\log\log n$ 轮内计算一个 β-支配集 [Gha16]。他还在 CONGEST 模型中给出了类似但略弱的结果 [Gha19]。

[2]支配集在 [GV07] 中的定义有所不同：我们所称的 t-支配集在 [GV07], 中称为 t-支配集，而 $(2,t)$-支配集则称为独立的 t-支配集。

论文	结果		模型	运行时间
[R图G20]	MIS	det.	LOCAL	$O(\log^7 n)$
[CPS20] + [RG20]	MIS	det.	CONGEST	$O(\log^8 n)$
[FGG+22]	MIS	det.	CONGEST	$\widetilde{O}(\log n \cdot \log^2 \Delta)$
[AGLP89], [HKN21]	$(k+1,\ k\cdot$对数 $n)$	det.	CONGEST	$O(k\cdot$日志$\,n)$
[SEW13], [KMW18]*	$(k+1,\ k\lceil$对数$_B\, n\rceil)$	det.	CONGEST	$O(k\cdot B\cdot$对数$_B\, n)$
[SEW13], [KMW18]	$(k+1,\ kc)$	det.	CONGEST	$O\left(k\cdot c\cdot n^{1/c}\right)$
New	$(k+1,\ k^2)$	det.	CONGEST	$\widetilde{O}\left(k^2$对数$^4\, n\cdot$对数$\,\Delta\right)$
[上界86]	G^k的极大独立集	随机。	CONGEST	$O(k\cdot$日志$\,n)$
[BEPS16] + [图G20]	MIS	随机。	LOCAL	$O($对数$^2\,\Delta+$多项式 对数对数 $n)$
[图Gha16] + [R图G20]	MIS	随机。	LOCAL	$O($对数 $\Delta+$多项式 对数对数 $n)$
[Gha19] + [图GGR21]	MIS	随机化。	CONGEST	$O(\log\ \Delta\cdot\log\log\ n+\text{poly}\log\log\ n)$
New	G^k的极大独立集	随机化。	CONGEST	$\widetilde{O}(k^2$对数 $\Delta\cdot$对数对数 $n+k^4$对数^{5}对数 $n)$
[图G V07]	$(-,\ O($对数对数 $n))^\dagger$	随机。	CONGEST	$O($对数对数$\,n)$
[KP12]	$(-,\ 2)^\ddagger$	随机。	CONGEST	$O\left(\log\ \Delta/(\log n)^\epsilon\right)$
[BKP14]+[Gha16]+[R图20]	$(2,\ \beta)$	随机。	LOCAL	$O(\beta\cdot(\log\Delta)^{1/\beta}+$多对数 $\log\log\ n)$
[BKP14]+[Gha19]+[GGR21]	$(2,\ \beta)$	随机。	CONGEST	$O\left(\beta\cdot(\text{对数 }\Delta)^{\,1/(\beta-1)}+\text{多项式对数对数 }n\right)$
[Gha19]	$(k+1,\ O(k^2$对数对数 $n))$	随机。	CONGEST	$O(k^2\log\log\ n)$
New	$(k+1, k\beta)$	随机。	CONGEST	$O\left(\beta k^{1+1/(\beta-1)}(\text{对数 }\Delta)^{1/(\beta-1)}\right)$ $+\widetilde{O}\left(\beta k\log\log\ n+k^4\log^5\log\ n\right)$

表 1：CONGEST 和 LOCAL 模型中极大独立集与支配集问题相关工作的总结。文中明确指出了可用于对任意 G^k 求解相应问题的 CONGEST 算法。对于 LOCAL，所有算法都可以直接用于 G^k，减速因子为 poly k （运行时间中未显示）。（*）此外，给定图 G^k 的 γ 着色，存在一种确定性 $(k+1,\ k\lceil\log_B\gamma\rceil)$-支配集算法，运行 $O(k\cdot B\cdot\log_B\gamma)$ 轮次。（†，‡）相应算法不满足支配集的独立性条件，但会计算具有相应支配性质的有界度子图。其度上界分别为 (†) $O(\log^5 n)$ 和 $O(\log\ n\cdot 2^{(\log n)^\epsilon})$，对应 (‡)。

近期发展出的强大轮消去技术 [Bra19] 已被用于证明 LOCAL 中计算支配集的下界 [BBO22, BBKO22]。以节点数为参数，β-支配集的下界对于确定性算法为 $\Omega(\log n/(\beta \log\log n))$，对于随机化算法为 $\Omega(\log\log n/(\beta \log\log\log n))$，前提是 β 分别至多为 $\approx \sqrt{\log n/\log\log n}$ 和 $\approx \sqrt{\log\log n/\log\log\log n}$。作为最大度 Δ 的函数，下界为 $\Omega(\beta \cdot \Delta^{1/\beta})$。表 1 概述了已知的支配集和极大独立集算法，并将其与我们的结果进行了比较。

图稀疏化的相关工作。我们最复杂的结果（定理 1.1）的一个基本构件是图稀疏化方法 [AGM12, HPP+15, Gha17, GU19, ACK19, CDP20, CDP21]。粗略地说，其思路是将输入图转化为更稀疏的表示，同时仍能据此求解给定问题。稀疏化已应用于许多不同的场景和计算模型，而我们希望稀疏化具备的确切性质往往因场景而异。

全局稀疏化。在具有 $O(n)$ 内存的模型中，例如流式计算、草图、拥塞团模型和线性内存 MPC，目标是减少图中的总（全局）边数。如果稀疏表示大约有 n 条边，那么就可以在常数时间内进行本地处理。例如，目前连通性/最小生成树 [JN18、Now21]，极大独立集 [ACG+21、GGK+18]，以及 $(\Delta+1)$-顶点着色（和列表着色） [CFG+19、ACK19、AA20、CDP21] 的最佳结果，都基于这一方法。

局部稀疏化。另一种稀疏化方法是对图进行局部稀疏化。为了直观理解，假设每个节点的输出 v 只取决于某些局部信息，即可以通过检查节点 v 周围的小范围 T-跳邻域中的图来决定节点 v 的输出。那么，局部稀疏化会减少该邻域中的边数，同时保留这样一种性质：在稀疏化后的图中，仍然可以根据 v 的局部邻域确定 v 的正确输出。结合耗费大量内存的图指数化技术 [LW10]，，该方法已成功用于低空间 MPC 模型等次线性模型中。例如，目前极大独立集 [GU19、CDP20]以及 $(\Delta+1)$-顶点着色 [CFG+19、CDP21] 的最佳结果，都基于这一方法。

在 CONGEST 模型中，对通信图进行局部稀疏化是避免拥塞的一种自然方法。在此前的一项工作中，这一思路被用于顶点着色 G^2 [HKM20]。然而，在着色问题中，可以将该问题 拆分为使用不相交颜色集的独立实例。这一性质使着色与其他对称性破缺问题（例如极大独立集和支配集）之间存在根本差异，后者无法享受拆分为不相交实例的便利。稀疏化图本身也具有独立的研究价值，例如在寻找距离保持生成器的背景下，无论是在线性内存模型 [DN20, DFKL21] 中，还是在局部稀疏化 [FGdV22]中。

2 记号和 k重独立随机变量

我们始终用 $G=(V, E)$ 表示原始输入图。类似地，在我们的算法中，顶点的度 $d(v)$ 和（不含自身的）邻域 $N(v)$ 始终不变。对于 $s \geq 0$，幂图 G^s 是这样一种图：其中 $V(G^s)=V$ 且 $E(G^s)=\{\{v,w\} \in V \times V: \mathrm{dist}_G(v, w) \leq s\}$。由节点集 X 诱导的幂图子图记为 $G^s[X]$。注意，这不同于 $(G[X])^s$：后者只包含由路径构成、且路径仅使用.....中的节点的边

X。 $N^s(v)$ 称为 v 的距离-s 邻域，即 v 在 G^s 中的邻域。令$d^s(v) := |N^s(v)|$。对于任意 $X \subseteq V$，令 $N^s(X) := \cup_{v\in X} N^s(v)$。我们用距离-$s$ X-邻域表示 $N^s(v, X) := N^s(v) \cap X$。距离-$s$ X-度定义为 $d^s(v, X) := |N^s(v, X)|$。

$\mathcal{I} \subseteq V$ 在 G中是 α-独立的，当且仅当对于所有不同的 v、$w \in \mathcal{I}$，$\text{dist}_G(v, w) \geq \alpha$。对于 $\alpha = 2$，我们简称 $\mathcal{I}$ 在 G中是独立的。对于任意 $S \subseteq V$，如果对于所有 $u \in S$，都存在某个 $v \in Q$ 使得$\text{dist}_G(u, v) \leq \beta$，则 $Q \subseteq S$ 是 S的 β-支配集。当 $S = V$时，我们称 Q 为 β-支配集。图 $G = (V, E)$的(α, β)-支配集是一个子集 $Q \subseteq V$，满足 Q是 α-独立的且是 β-支配的。 α 和 β 通常分别称为独立参数和支配参数。注意，$(2, 1)$-支配集是G 的极大独立集，而$(k + 1, k)$-支配集是 G^k的极大独立集。如果对于所有 $S' \subset S$ 都有$\text{dist}_G(S', S \setminus S) \leq k$，则集合 $S \subseteq V$在 G中是 k-连通的。等价地， S 是 k-连通的，当且仅当 $G^k[S]$ 是连通的。

以 $r \in V$ 为根、深度为 $s \geq 0$ 的广度优先搜索树（BFS 树） $T \subseteq G$ 是一棵满足以下条件的树： $V(T) = N^s(v) \cup \{v\}$，且 $\forall v \in V(T)$： $\text{dist}_T(v, r) = \text{dist}_G(v, r)$。如果在分布式环境中，每个 $v \in V(T)$都知道其直接祖先 $(T, v) \in N(v)$、后代 $(T, v) \subseteq N(v)$ 以及根节点的 ID root $(T) \in V$，我们就称 BFS 树 $T \subseteq G$在分布式环境中是已知的。深度为 s 的 BFS 树包含根节点距离为 s 的邻域中的所有节点。当底层图不是树时，BFS 树并不唯一。当 $V(T) = V(G)$ 时，我们称 BFS 树 T是生成的。

定义 2.1 （网络分解）。图G 的 (c, d)-网络分解是将 V 划分为互不相交的 d-直径簇。簇可以进行 c 着色，使得相邻簇的颜色不同。簇直径可以用两种方式衡量。如果簇 $C \subseteq V$ 中的任意两个顶点在 $G[C]$ 中的距离至多为 d，则称该簇具有强直径 d。在弱直径簇中， C 中的任意两个顶点在 G 中的距离至多为 d。为了在簇中的节点之间通信，我们为 C 关联一棵斯坦纳树 T_C。这是一棵 G 的子图，包含终端节点和非终端节点的集合。终端节点即簇中的节点，而非终端节点可以是图中的任意节点。斯坦纳树的直径为 $O(d)$。不同簇的斯坦纳树可能彼此重叠。如果每条边至多属于任意单一颜色的 τ 棵簇的斯坦纳树，我们就称网络分解的拥塞为 τ。在幂图中， G^k 个簇的网络分解将顶点划分开，分离度为 $k+1$。对于任意两个颜色相同的簇 C、C'，要求 距离$_G(C, C') > k$。

k-路独立变量。 一组离散随机变量 X_1、.....、X_n 如果对于任意 $I \subseteq [n]$ 满足 $|I| \leq k$ 以及任意取值 x_i，都有 $\mathbb{P}(\wedge_{i\in I} X_i = x_i) = \prod_{i\in I} \mathbb{P}(X_i = x_i)$，则称其为 k-路独立。我们可以通过从一个由 k-路独立哈希函数组成的函数族中随机选取一个随机哈希函数来模拟这类变量:

定义 2.2。 对于满足 $k \leq N$ 的 N、L、$k \in \mathbb{N}$，函数族 $\mathcal{H} = \{h : [N] \to [L]\}$若对于所有互不相同的 x_1、.....、$x_k \in [N]$，随机变量 $h(x_1)$.....、$h(x_k)$ 在 $[L]$ 中相互独立且均匀分布，则称其为 k-路独立，其中 h 从 $\mathcal{H}$ 中均匀随机选取。

引理 2.3 (推论 *3.34*（见 [Vad12]）。对于每个 a、b、k，都存在一个由 k-重独立哈希函数组成的族 $\mathcal{H} = \{h : \{0, 1\}^a \to \{0, 1\}^b\}$，从 $\mathcal{H}$ 中随机选取一个函数需要$k \cdot$至多$\{a, b\}$ 个随机比特，而计算 $\mathcal{H}$ 中的一个函数需要 $\text{poly}(a, b, k)$时间。

3 技术概述

确定性稀疏化。我们的主要技术手段是一个确定性稀疏化过程。从整体上看，我们逐步稀疏化输入图，即计算一系列更稀疏的节点集 $V \supseteq Q_0 \supseteq Q_1 \supseteq \ldots \supseteq Q_k$，同时确保 V中的每个节点始终与该集合保持常数距离。更精确地说，对于每个 $1 \leq s \leq k$， Q_{s-1} 中的任意节点到 Q_s 的距离在 G^s 中至多为 2（在 G中至多为 $2s$ ），并且集合 Q_s 在 G^s中是稀疏的，也就是说，每个节点在 G^s中只有数量有界的 Q_s-邻居。我们利用 Q_s 的稀疏性，高效地限制计算更稀疏集合 Q_{s+1}时的拥塞。最后， Q_{k-1} 足够稀疏，可以高效地模拟在 $G^k[Q_{k-1}]$上的任意算法；也就是说，该子图由 G^k 中 Q_{k-1} 所诱导。例如，要计算 G^k 的一个 k- 支配集，我们可以利用稀疏化找到 Q_{k-1} ，然后在幂图 G^k上计算 Q_{k-1} 的一个极大独立集，其中只有 Q_{k-1} 中的节点可以加入该极大独立集。

我们认为，这一稀疏化过程本身具有独立的研究价值，也可能有助于解决其他问题，以及应用于其他计算模型。我们在下一个引理中对其进行概述。回顾一下，节点 v 的距离-k Q-度是 $N^k(v) \cap Q$ 中邻居的数量。

引理 3.1 (幂图中的稀疏化)。 设 $k \geq 1$ （可能是 n 的函数）。存在一种确定性分布式算法，给定子集 $Q_0 \subseteq V$，可以找到一个顶点集合 $Q \subseteq Q_0$，使得对于所有 $v \in V$，

- （有界距离-k Q-度）： $d^k(v, Q) \leq 72$ 对数 $n = O($对数 $n)$
- （支配）：距离$_G(v, Q) \leq k^2 + k +$ 距离$_G(v, Q_0)$

该算法在 $($直径$(G) \cdot k \cdot$ 对数$^2 n \cdot$ 对数 $\Delta + k^2 \cdot$ 对数 $\Delta)$ 轮次内运行，采用 CONGEST 模型。

利用网络分解，我们可以将引理 3.1 中的直径因子替换为关于 n 的多对数项（更多细节请参见第 5.4 节，尤其是引理 5.8 中的精确表述）。对于 $k = 1$ （如果使用 $Q_0 = V$ 初始化），我们可以获得一个多对数轮算法，用于计算一个支配距离为 2 的集合，使得每个 $v \in V$ 在 Q 中至多有 $O(\log n)$ 个邻居。请注意，度上界是一个局部性质。如果希望计算具有相同支配距离且全局地最小化其大小的集合，那么已知即使是常数近似算法，在 CONGEST [BCM+20] 中也需要近二次时间。

稀疏化是如何实现的? 我们确定性稀疏化的核心思想，是对以下随机采样算法进行去随机化：首先假设图 G^k 是 Δ^k-正则的。我们可以以 $O(\log n/\Delta^k)$ 的概率将每个节点采样到集合 Q 中。这样，以高概率，V 中的每个节点都能获得 $\Theta(\log n)$ 个距离为 k Q 的邻居。尽管这是一个简单的 0 轮算法，但对其进行去随机化并非易事，因为节点施加的约束取决于距离 $\Theta(k)$ 内节点的决策。一般而言，直接且确定性地找到能够满足所有节点约束的良好随机比特（又称良好条件）希望渺茫。相反，我们采用一种更细粒度的稀疏化方法，逐步对图进行稀疏化，每次都使支配性质略有损失。具体而言，这可以看作先针对 $k = 1$ 计算集合 Q_k ，再针对 $k = 2$ 计算集合，依此类推。

继续在随机化设定下，我们解释如何在 G中寻找 Q_1 。采样算法有两个目标。对于所有 $v \in V$， Q_1 中的邻居数量应至多为 $O(\log n)$，而到 Q_1 的距离应至多为 2（或至多为某个常数）。由于各节点的度并不均匀，因此很难找到能让所有节点都满足这两个目标的合适采样概率。因此，进程被放慢至 $O(\log \Delta)$ 阶段。 Q_0 中的所有节点一开始

时都处于活跃状态（在我们的应用中，$Q_0 = V$处于活跃状态）。在每个阶段，我们对活跃节点进行抽样，以得到 Q_1。最初，采样概率较低，因此高阶节点至少能获得一个被采样邻居，同时又不会超过 $O(\log n)$ 的上限。每个阶段结束时，采样节点的距离为 2 的邻域会被停用。这确保高阶节点在后续阶段不会获得更多被采样邻居，从而有效降低最大活跃度。活跃度降低意味着我们可以在下一阶段采用略高的采样概率。最终，只剩下低活跃度节点，这些节点可以纳入采样集合。结果 Q_1 是一个 2-支配集，而所有 $v \in V$的 Q_1-邻居数量至多为 $O(\log n)$。

稀疏化 G. 我们针对 G 的确定性稀疏化基于对先前采样算法的去随机化。随机化分析采用 $O(\log n)$-路独立性。这使得我们可以用一个 $O(\log^2 n)$位的随机种子来模拟一个阶段中的随机性。去随机化按阶段进行，使用条件期望法。在每个阶段，我们需要保证两个事件：(1) 每个节点至多获得 $O(\log n)$个被采样邻居；(2) 每个活跃度较高的节点至少有一个邻居被采样。这两个事件都只取决于图G中直接邻域内发生的事件，因此每个节点都能轻松获得所需信息。种子的各个位依次确定。为了确定种子的某一位 b （取0或1），每个节点都要针对两种取值 b，分别计算其两个事件的条件期望。计算条件期望时，节点需要知道已确定的种子位的取值，以及其活跃邻居的标识符。在 G 中，只需一轮即可获知相关标识符，因为这两个事件由直接邻域内活跃邻居的决策决定。随后，所有节点的条件期望会汇总（相加）到一个领导节点，由它决定该位 b应采用哪个更优的取值。然后，所有节点继续处理下一位。根据条件期望法，在这一过程结束时，所有节点的事件都会成立。

幂图稀疏化。 固定某个 $1 \le s \le k$。在幂图 G^s上模拟稀疏化的第 s次迭代 。上一次迭代的输出作为初始活跃节点集合 Q_{s-1}。第 s次迭代得到 $Q_s \subseteq Q_{s-1}$，其中 Q_s 在 G^s中是稀疏的，同时将 Q_{s-1} 在 G^s 中的支配距离最多增大2（或在 G中增大 $2s$ ）。在 s 次迭代中，距离增量之和为 $\sum_{j=1}^{s} 2j \le s^2 + s$，因此 Q_s 是 G 的一个（$s^2 + s$）支配集（初始化时使用 $Q_0 = V$）。

我们工作面临的一项主要挑战是，在对幂图进行稀疏化时，确保所有节点都能获得去随机化所需的信息（距离为s 的邻居的随机比特和标识符）。为了保证 G^s中的稀疏性，所有节点都必须继续充当观察者（并中继消息），参与去随机化。在这里，简而言之，关于上一次迭代的稀疏性有助于获取所需信息。更形式化地说，我们构建了通信工具，以便在 G^s上高效运行算法，并利用 Q_{s-1}的稀疏性。

通信工具。为了利用 Q:= Q_s的稀疏性，我们开发了通信工具（见引理 4.2），以便在幂图上执行基本通信原语。 G^s 中 Q 的稀疏性可用于高效地将消息从 Q 发送给其在 G^{s+1}中的邻居。通信算法包括：在 O轮次内，将 Q 中的节点发出的广播发送给其在 G^{s+1} 中的邻居，耗时为$s+$（n log n）；以及在 O轮次内模拟在 $G^{s+1}[Q]$ 上运行的 CONGEST 算法的一轮，耗时为$s+$（$^2 n$log $^2 n$）（因此，基本上可以假设该算法运行在通信网络 $G^{s+1}[Q]$ 上，开销为 O（$s + \log^2 n$））。其效率基于以下事实：对于 G中的所有节点，在 Q 中距离为s 的邻居数量 Δ 有界。这有效地限制了经过图中任意一条边转发的消息数量。例如，对于广播，

对于通信网络中的任意一条边 $\{v, w\}$，必须将广播^从 v 转发到 w 的节点 $x \in Q$ 数量至多为 d^s（v, Q）$\leq \Delta$，因为消息最多只会被转发 $s+1$ 跳。通信工具还用于得到稀疏化引理（引理 3.1）；其中，利用 G^s 中 Q_s 的稀疏性，可以高效地在 G^{s+1} 上执行第（$s+1$）次稀疏化迭代。在支配集应用中，我们使用通信工具来模拟在 G^k 上运行的最大独立集算法。

确定性 k-支配集（位于 G^k）。我们的确定性稀疏化算法用于计算一个稀疏子集 $Q := Q_{k-1} \subseteq V$，同时保持其到图中其余部分的支配距离为常数。对图进行稀疏化后，我们计算 $G^k[Q]$ 的极大独立集。借助我们的通信工具，我们可以用黑盒方式在 $G^k[Q]$ 上模拟任意极大独立集算法。一般而言，这种方法会得到图 G 的一个（$k+1$, $\beta+k$）-支配集，其中 β 是 Q 的支配距离（形式化表述见引理 6.3）。使用我们的稀疏化算法，结果是图 G 的一个（$k+1$, k^2）-支配集，或者等价地说，是一个 k-支配集。可以通过（$k + \log^2 n$）倍的减速，在 Q 中实现距离为 k 的邻居之间的消息传递。结合最先进的极大独立集算法 [FGG+22]，我们可以在 O 中用（poly log n）轮计算出一个（$k+1$, k^2）-支配集（定理 1.1）。当 $k > 1$ 为常数时，这比先前工作有指数级改进；先前工作需要 $O(k^2 \cdot n^{1/k})$ 轮[SEW13、KMW18、EM19]。

随机化极大独立集 G^k。对于随机化结果，我们采用碎裂 框架：使用随机化底层算法 高效地解决图的大部分区域中的问题。剩余的连通分量（在 G^k 中）规模较小，且这一点以高概率成立。小型连通分量则通过另一种算法解决。有关碎裂框架的更多详情，另请参见下一段关于定理 1.4 的内容。对于底层算法，我们无法将 [Gha16]，中 Ghaffari 极大独立集算法的模拟作为黑盒来使用，因为在 G^k 上进行模拟的代价会高得难以接受。因此，我们采用 [Gha17] 的 $BeepingMIS$ 算法（并作了一个细微但关键的修改）。该算法适用于简单的蜂鸣通信模型。不过，转发蜂鸣信号时必须谨慎：当通信网络中存在环时，如果 $k \geq 3$，节点可能会将邻居发出的蜂鸣信号误认为是自己发出的。为避免这种情况，我们为蜂鸣信号附上发出蜂鸣的节点的标识符。节点转发任意一个至多包含两个蜂鸣信号的子集；这足以让任何发出蜂鸣信号的节点判断是否存在发出蜂鸣信号的邻居。在碎裂后阶段，图中剩余的未解决部分构成了 G^k 中的小型连通分量，每个分量有 $N = O(\Delta^{4k} \cdot \log n)$ 个节点，且这一点以高概率成立。为了在这些剩余分量中找到解，我们并行运行 $O(\log_N n)$ 次 BeepingMIS 算法，保证至少有一次执行成功；在 n 中，这一保证以高概率成立。为限制拥塞，我们根据 $[N]$ 为连通分量中的节点分配唯一标识符。这样，总通信量被限制在 $O(\log_N n \cdot \log N) = O(\log n)$（用于模拟所有实例的一步）。这一方法实现的运行时间与 G [Gha19]，的最先进极大独立集算法相同，减速因子为 k。

定理 1.4：在 G 中进行碎裂，适用于 LOCAL 和 CONGEST。自 [BEPS16]的开创性工作以来，碎裂技术已成为该领域的重要工具。该技术分为两个阶段。在预碎裂阶段，通过一种非常高效的随机化过程，解决图中大部分区域的相关问题（例如最大独立集问题），使得之后高概率只剩下小连通分量——可以认为其规模为多对数规模。在后碎裂阶段，则采用另一种算法，通常是确定性算法，利用小连通分量的规模较小这一特点，高效地完成对它们的处理。

这些连通分量的规模实际上是 poly $\Delta \cdot \log n$，对于较大的 Δ 值而言，这并不算小。

高层解决方案。幸运的是，连通分量还具有其他有益性质：固定一个小组件 C 以及一些合适的 β 和 $\alpha \geq 5$（常数 5 取决于预碎裂阶段的性质，且适用于 BEPS16、Gha16、Gha19], 中的极大独立集算法；其他问题可能需要采用其他数值）。已知，任意一个 (α, β)-支配集 R_C 在 C 中至多有 $O(\log n)$ 个节点。如果我们能获得这样一个支配集，就可以利用其规模来设计算法，具体做法是创建如下虚拟图： R_C 中的每个节点都围绕自身形成一个由节点构成的连通球，使得 C 中的每个节点都归入唯一一个球。然后，可以构建虚拟图 H，为每个这样的球设置一个顶点。如果相应的球中包含在原图 G中彼此相邻的节点，那么这两个球就在 H 中相连。由于 $|V(H_C)| = |R_C| = O(\log n)$，且在 H_C 中进行一轮通信可以在 G 中用 $O(\beta)$轮（在 LOCAL 模型中）模拟，因此可以非常高效地计算 H 的网络分解（ND），例如采用 [RG20]的算法。此处无需详述具体细节，但已知 H_C 的网络分解蕴含 C 的网络分解，而后者可用于计算 C上的极大独立集。

挑战。 挑战在于，目前尚不清楚如何计算支配集 R_C。算法必须在所有小连通分量上并行运行，而在诱导子图上运行并不可行，因为得到的支配集需要在 G 中满足 5-独立性；仅在 $G[C]$ 中满足 5-独立性是不够的。另一个关键问题是，节点 $v \in C$，$v \neq R_C$ 是否会被其他某个小连通分量的支配集中的节点所支配 $C' \neq C$。为了说明这一难点，注意到一个节点无法高效判断距离为 5 的节点 G 是否与自己属于同一个小连通分量。因此，节点很难确定它们是否可以同时包含在支配集中。

解决方案。 受 [BEPS16], 不同版本的结合启发，我们提出以下解决方案。在预碎裂阶段之后，我们再执行第二个随机预碎片化阶段，该阶段在所有小连通分量上并行运行（它在 n 中以高概率成功，并且至多有 n 个连通分量，因此我们可以对不同连通分量的错误概率应用并合界）。它将每个连通分量划分为所谓的 微小连通分量。接下来，我们证明以下引理。

引理 （引理 7.5 的非正式版本）。 设 $\alpha = 5$，且令 β 为一个整数，并考虑一个小连通分量 C。那么，其微小连通分量的每个 （α、β）-支配集至多包含 O（log n）个节点。支配集的两种距离均在由 C 诱导的图中测量；即使一个微小连通分量中的节点由另一个微小连通分量中的节点支配，该大小上界仍然成立。

该引理的好处在于，我们可以在由小连通分量诱导的图上运行支配集算法（以支配微小连通分量中的所有剩余节点），从而使该算法在不同的小连通分量之间完全独立。无需担心 G 中距离为 5 的节点是否与自身位于同一个连通分量中，只需忽略所有不包含在 $G[C]$ 中的边，并在由小连通分量诱导的图上工作即可。

引理的证明。尽管整个设置相当复杂，与 [BEPS16], 中的设置非常相似，但引理的证明非常简单。如果我们固定一个特定的支配集 R_C，如引理中所述，那么 [BEPS16] 的核心碎裂论证指出，在 n 中，集合 R_C 的大小为 $O(\log n)$，且这一结论以高概率成立。直观上，这是因为每个节点仅以较小的概率仍未确定，而且对于间隔至少 5 跳的节点，这些事件相互独立。因此，以高概率，不会有超过 $O(\log n)$ 个两两距离为 5 的节点仍未确定。现在，对所有此类支配集应用并合界，即可得到该引理。这种解法的关键在于

可以利用 C 的规模较小这一点（在第一个预碎裂阶段之后，该结论以高概率成立），从而证明这类集合的数量只有 poly$n$，使得并合界成为一种可行的方法。[3]

以往的解决方案。arXiv 版本的 [BEPS16] 采用了非常相似的两阶段设置，但使用了不同（且有误[4]）的方法来证明一个与我们的引理相似的结论。 [BEPS16] 的（正确）期刊版本也采用了类似的两阶段结构，但随后使用了一种不同且更复杂的论证，来证明可以高效地计算虚拟图H 的网络分解。除了一些其他技术细节外，主要区别在于，他们证明了：如果在计算支配集时将节点即时分配给支配节点，那么每个球都满足特定的幂图连通性要求（如果节点只是加入最近支配节点的球，他们的证明则不成立）。因此，该证明需要了解所用支配集算法的内部机制，也就难以像后续工作那样将其形式化为黑盒。

我们的第二种解决方案。我们的第二种解决方案表明，利用当前最先进的支配集算法的内部机制，可以省去第二个预碎裂阶段。尽管这种证明比 [BEPS16], 期刊版本中的证明更简单，但我们认为，基于两阶段结构的解决方案更容易理解。不过，采用单个预碎裂阶段的版本更容易推广到 G^k，而这正是定理 1.2 和推论 1.3 所需要的。

4 通信工具

本节将介绍在稀疏表示的幂图上进行高效通信所需的通信工具。

引理 4.1（学习距离-（$s+1$） Q-邻域中的标识符）。设 $s \geq 1$ 和 $\Delta \geq 1$ 为整数。设 $Q \subseteq V$，使得所有 $v \in V$ 至多有 Δ 个距离-s Q-邻居。每个 $v \in V$ 都有一个唯一的 a 位标识符 $ID(v)$。已知所有 $v \in V$ 都知道 $N^s(v, Q)$ 中的标识符集合，所有 $v \in V$ 都可以在 $O(\widehat{\Delta} \cdot a/\text{带宽})$ 轮次内，使用 带宽 位消息，获知 $N^{s+1}(v, Q)$ 中的标识符集合。

此外，假设对于每个 $v \in Q$，都有一棵以 v 为根节点、深度为 s 的 *BFS* 树 T_v，并以分布式方式给出。每棵树都可以在额外的 $O(\Delta \cdot a/\text{带宽})$ 轮次内扩展至深度 $s+1$。

证明。 每个节点 $v \in V$通过流水线传输，将 $N^s(v, Q)$ 中的标识符集合发送给它的每个邻居 $w \in N(v)$。根据假设，对于所有 $v \in V$， $d^s(v, X) \leq \Delta$。因此，流水线传输这些标识符最多需要 $\Delta \cdot a/$带宽轮次。每个 $v \in V$将收到的标识符集合取并集，形成$\cup_{w \in N(v)} N^s(w, Q) = N^{s+1}(v, Q)$。

为了扩展广度优先搜索树，将每个 $v \in V$添加到距离恰好为 $s+1$ 的 Q 中各节点的树中。对于每个 $x \in N^{s+1}(v, Q) \setminus N^s(v, Q)$，我们将 v 作为叶节点添加到树 T_x 中。设 $w_x \in N(v)$ 是向 v 发送标识符(x) 的一个邻居。如果有多个这样的邻居，则任意选择其中一个 $w \in N(v)$。根据假设， w 已包含在 T_x 中。现在， v 向 w_x 发送确认，其中包含标识符(x)。 v 设置祖先$(T_x, v) := w_x$，并将 v 添加到后代(T_x, w) 中。额外的时间复杂度与发送标识符所需的时间相同。□

引理 4.2 (从 $Q \subseteq V$)发送消息。令 $s \geq 1$ 和 $\Delta \geq 1$ 为整数。令 $Q \subseteq V$满足条件：所有 $v \in V$的距离-Δ 至多为$(s-1)$ Q邻居。对于每个 $v \in Q$，令 T_v 为一棵以 v 为根节点、深度为 s的 *BFS* 树。假设每个 $v \in Q$ 都知道 $N^s(v, Q)$中的标识符集合，以及对于每个 $w \in N(v)$，还知道 $N^{s-1}(w, Q)$。存在一个确定性算法，可以完成以下任务：

[3]熟悉相关内容的读者会注意到，这与证明在第一个预碎裂阶段后出现小连通分量的标准证明不同（也更简单）。在那种情形下，类似的并合界并不可行。[4]详情请参见第7节。

- 广播：每个 $v \in Q$ 向 $w \in N^s(v)$ ^发送一条 m 位的消息 msg_v，耗时 $O(s + m\Delta/\text{带宽})$ 轮次。
- Q-消息：每个 $v \in Q$ 在 $O(s + (m + a)\widehat{\Delta}^2/\text{带宽})$ 轮次内，向每个 $w \in N^s(v, Q)$发送一条 m 位消息 $msg_{v,\ w}$。

每个节点都有一个 a 位标识符，该标识符在其 s 邻域内唯一。该算法使用^带宽位消息，其中 带宽 $\geq \Delta$。

证明。 两种算法都使用以每个 $v \in Q$ 为根节点的 BFS 树，将消息从 v 发送到^节点（位于 $N^s(v)$ 中）。首先，我们证明任意一条边至多属于 $P := 2\Delta$ 棵树。设 $e = \{v, w\} \in E$ 为任意一条边，并设 $x \in Q$ 满足 e 属于 x 的树。树 T_x 是通过从 x 开始、深度为 s 的广度优先搜索构造的。因此，要么 $x \in N^{s-1}(v,\ Q)$，要么 $x \in N^{s-1}(w,\ Q)$。任意节点的距离-$s-1$以^度至多为 Δ，所以 e 至多属于 2Δ 棵树。

广播：我们使用 [GGR21 第 5] 节中的广播算法。下面一段是对其算法的总结。每棵树 T_v 在其边上仅分配 $b' :=$带宽$/P$ 比特的通信带宽。根节点 v 将其消息拆分为 m/b' 个片段。消息从根节点开始逐片段发送，其中第 i 个片段在第 i 轮发送。树中的其他节点将片段转发给其后代，且不会产生拥塞。我们可以在所有树中并行执行该算法——总计而言，单条边每轮发送的比特数至多为$P \cdot b' =$ 带宽。无需包含每棵树的信息即可组装完整消息：给定一个片段集合，发送方 v 向每个 $w \in N(v)$形成一条消息，其中包含满足 T且其 $w \in$后代$(v,\ T)$的树的片段。发送方 v按照根节点的标识符对片段进行排序并将其串联起来。接收方 w 将消息拆分为 b' 比特的片段，并利用关于祖先$(w,\ T)$的知识，将它们映射到相应的树。消息的最后一个片段^到达所有叶节点所需的总时间为 $O(s + m/b') = O(s + mP/\text{带宽})$轮。在我们的应用中，$P = 2\Delta$，因此^广播算法的轮复杂度为 $O(s + m\Delta/\text{带宽})$。

Q-消息：考虑任意 $x \in Q$。它向每个邻居 $y \in N^s(x, Q)$发送一条 m位的消息 $\text{msg}_{x,y}$。来自 x 的消息总数至多为 $\Delta \cdot \widehat{\Delta}$，因为 $|N^s(x, Q)| = |\cup_{w \in N(v)} N^{s-1}(w, Q)| \leq \Delta \cdot \widehat{\Delta}$，这是因为根据假设，$d^{s-1}(w, Q) \leq \widehat{\Delta}$。该算法分为两步。第一步，我们将消息均匀分配给 x的直接邻居。每条消息 $\text{msg}_{x,y}$ 都被打包成如下形式的元组：$(\text{msg}_{x,y}, \text{ID}(y))$。现在，$x$向每个直接邻居 $w \in N(x)$发送一组消息 $S_{x,w} := \{(\text{msg}_{x,y}, \text{ID}(y)) : y \in N^{s-1}(w, Q)\}$，这里利用了如下假设：$x$知道每个 $w \in N(x)$对应的 $N^{s-1}(w, Q)$。这可能会将某些元组发送给多个邻居。不过，对于任意 w, $|S_{x,w}| \leq \Delta$，根据假设，$d^{s-1}(w, Q) \leq \widehat{\Delta}$。从 x发送给 w的元组集合可装入 $M := (m + a)\widehat{\Delta}$ 位。第一步需要 $O(M/\text{带宽})$轮次，并且图G中的所有节点都可以并行执行。

在第二步中，我们将每个 $x \in Q$ 的树 T_x 划分为 $O(\Delta)$棵子树 $T_{x、w}$，每棵子树以各自的 $w \in N(x)$为根。$T_{x、w}$ 是由 w 及其在 T_x 中的所有后代节点 w 诱导出的子树。对于任意 $x \in Q$，T_x 的子树两两边不相交且节点不相交。此前我们证明过，对于任意 $e \in E$，至多有 2Δ 个根节点 $x \in Q$ 满足 $e \in T_x$。由于 T_x 的子树两两边不相交，对于任意 $e \in E$，至多有 2Δ 棵子树 $T_{x、w}$ 满足 $e \in T_{x、w}$。现在，每棵子树 $T_{x、w}$ 中的任务是将消息 $\text{msg}_{x、y}$ 传递给每个 $y \in N^{s-1}(w、Q) \cap V(T_{x、w})$。具体做法如下：在所有 $x \in Q$ 的全部子树中并行执行。将完整的元组集合 $S_{x、w}$ 作为一条广播消息发送至 $T_{x、w}$，并使用前文所述的 GGR21] 广播算法。所有节点 $v \in T_{x、w}$ 都具备运行该广播算法所需的知识：每个非根节点都知道 ancestor$(v、T_{x、w})$ 和 descendants$(v、T_{x、w})$，因为它们与 T_x 中的相同。原始根节点 x 的标识符可以

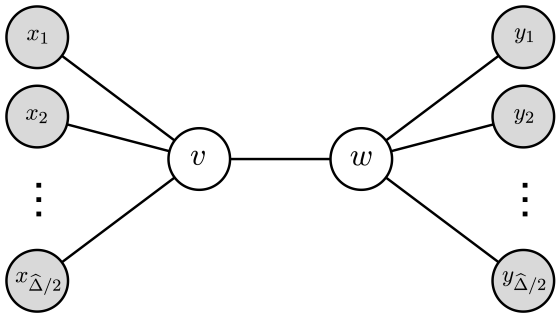

图1：引理 4.2 的示例图，灰色表示节点 Q。对于 $s=3$，在 $\{v,w\}$ 上的消息总数最多为 $\widehat{\Delta}$（当 Q 中的节点进行广播时），以及 $\widehat{\Delta}^2/4$（Q 条消息）。

用作根节点标识符，因为 T_x 的子树彼此节点不相交。运行广播算法需要 $O(s+M\Delta/\text{带宽})$ 轮，其中 M是广播消息的大小，且 $T_{x\text{的深度为}w}$ 至多 $s-1$。节点 $y \in N^{s-1}(w) \cap V(T_{x,w})$ 根据自身标识符，从接收元组集中找到 $\text{msg}_{x,y}$。总复杂度为 $O\left(\frac{M}{\text{带宽}} + s + \frac{M\widehat{\Delta}}{\text{带宽}}\right) = O\left(s + \frac{(m+a)\widehat{\Delta}^2}{\text{带宽}}\right)$ □

引理 4.2 的结果是紧的，这一点可从图1中看出。

引理 4.3（生成树上的汇聚广播）。设 T 是一棵给定的生成广度优先搜索树，其根节点是某个 $r \in V$。假设每个 $v \in V$ 都有一个 m 位的非负整数 x_v。存在一种分布式算法，可在根节点处计算总和 $\sum_{v\in V} x_v$。该算法运行

$$O\left(\text{diam}(G) + (m + \log n)/\text{bandwidth}\right)$$

轮，在 CONGEST 模型中使用 带宽为 *{bandwidth}* 位的消息。

证明。总和 $\sum_{v\in V} x_v$ 至多为 $n \cdot 2^b$。使用 [GGR21 的算法，引理 5.1]，通过$O(\log n+b)$ 位的消息，可在根节点处计算出 $\sum_{v\in V} x_v$，所需轮数为 $O(\text{diam}(G)+(m+\log n)/\text{bandwidth})$。

□

引理 4.3 所需的生成广度优先搜索树可以通过在 $O(\text{diam}(G))$ 时间内进行领导者选举来构造：从每个节点启动一个广度优先搜索令牌，并转发其树 T 的根节点具有最小标识符的令牌。在网络分解簇中使用引理 4.3 时，我们将该簇的斯坦纳树用作生成树。

使用通信网络 G模拟虚拟图

定义 4.4（虚拟图 $H=(V_H, E_H)$）。设 $G=(V,E)$ 为通信网络。设 $V_H \subseteq V$和 $E_H \subseteq \{\{v,w\} \mid v,w \in V_H, v \neq w\}$。$H=(V_H, E_H)$ 称为 G的虚拟图。

G 的任意子图都是 G的虚拟图，但该定义并不要求 H 的边存在于 G中。例如，对于某个 $s \geq 1$，我们可以将 H 定义为 G^s。

设 $\mathcal{A}$ 为任意 CONGEST 算法（确定性或随机化算法）。假设我们要在 H上运行 $\mathcal{A}$，但可用的通信网络是 G，而不是 H。我们一般性地定义通过通信网络 G在 H 上模拟 $\mathcal{A}$ 的含义如下：

定义 4.5（在虚拟图 H 上模拟 $\mathcal{A}$，通信网络为 G）。设 $\mathcal{A}$ 为任意CONGEST算法，且 $H=(V_H, E_H)$是 G的虚拟图。假设 V_H中的节点知道它们在 $\mathcal{A}$中的输入（如果有），也知道它们在 H 中的度（如果算法需要）。逐轮模拟 $\mathcal{A}$。要模拟 $\mathcal{A}$的一轮：

1. 每个 $v \in V_H$ 根据其在 $\mathcal{A}$中的状态，在零轮内形成消息集合 $\{msg_{v,\,w}:\ \ w \in N_H(v)\}$。*2.* H 中邻居之间的通信通过通信网络 G进行，并使用某种消息传递算法。该算法可以将完整的传入消息集合 $\{msg_{w,\,v}:\ \ w \in N_H(v)\}$中继给每个 $v \in V_H$，也可以传递这些消息的某种聚合，只要其中包含足够的信息，使 v 能够更新其在 $\mathcal{A}$中的状态，效果与收到完整消息集合完全相同。聚合类型会与被模拟的算法一并指定。*3.* 每个 $v \in V_H$ 根据其在 $\mathcal{A}$ 中的先前状态及收到的信息更新状态，不再进行进一步通信。

从现在起，　　通信网络假定为 G，不再明确指定。

在 H 上模拟 $\mathcal{A}$ 的效率，在很大程度上取决于算法类型、虚拟图 H 以及 H 与通信网络 G之间的关系。一般而言，G 的边可能不得不在 V_H中任意多对节点之间中继消息。此外，在许多算法中，无法压缩中继消息集合。稍后我们将进行针对具体算法的优化。目前，我们聚焦于虚拟图为 $G^s[Q]$的情形，其中$s \geq 2$ 和 $Q \subseteq V$。下一个引理表明，如果 Q 满足某些稀疏性条件，则任何CONGEST 算法都可以被高效模拟：

引理 *4.6*（通过通信网络 G 在 $G^s[Q]$ 上模拟 $\mathcal{A}$）。设 $\mathcal{A}$ 是任意 *CONGEST* $\widehat{\ }$ 算法。设 $s \geq 1$ 和 $\Delta \geq 1$ 为整数。设 $Q \subseteq V$ 满足：所有 $v \in V$ 的距离不超过 Δ 的 $s-1$ 邻居至多有 Q 个。对于每个 $v \in Q$，设 T_v 是一棵以 v 为根节点、深度为 s 的 *BFS* 树，且其信息以分布式方式已知。假设每个 $v \in Q$ 都知道 $N^s(v, Q)$ 中的标识符集合，以及对于每个 $w \in N(v)$，都知道 $N^{s-1}(w, Q)$。设 $T^{\mathcal{A}}(H)$ 为在输入图 H 上运行算法 $\mathcal{A}$ 的时间复杂度。存在一种确定性 *CONGEST* 算法，可在通信网络 G 上用 $O((s+\widehat{\Delta}^2) \cdot T^{\mathcal{A}}(G^s[Q]))$ 轮次，在 $G^s[Q]$ 上模拟 $\mathcal{A}$。

证明。我们使用引理 4.2 的 Q-消息 算法，该算法允许每个 $v \in Q$ 向每个 $w \in N^s(v,\ Q)$ 发送一条包含 $O(\log\ n)$ 位的消息，并在 $O(s+\widehat{\Delta}^2)$ 轮内完成。这些正是 v 在$G^s[Q]$中的邻居。注意，根据假设，运行 Q-消息所需的知识是可用的。模拟每一轮需要 $O(s+\widehat{\Delta}^2)$ 轮，因此总时间复杂度为 $O((s+\widehat{\Delta}^2) \cdot T^{\mathcal{A}}(G^s[Q]))$。

□

5 幂图稀疏化

本节的目标是证明引理 3.1，这是我们关于幂图的主要稀疏化结果。首先，在接下来的两节中，我们证明以下确定性稀疏化结果。引理 5.1 是独立的，不涉及幂图。在第 5.3 节中，我们迭代地使用它，得到一个序列 $V \supseteq Q_1 \supseteq Q_2 \supseteq \ldots \supseteq Q_k$，其中 Q_i 在幂图 G^i 中是稀疏的。

引理 5.1 (确定性稀疏化)。 设 $A \subseteq V$ 为初始活跃顶点的集合。存在一个确定性分布式算法，可以找到一个顶点集合 $Q \subseteq A$，使得对于所有 $v \in V$，

- （有界 Q-度）： $d(v, Q) \leq 72$对数 $n = O$（对数 n）
- （支配）：距离$_G$（v, Q） $\leq 2 +$ 距离$_G$（v, A）

令 $\Delta_A \geq$ 最大值$_{v \in V}\, d$（v, A）为提供给所有节点的输入参数，其至少等于活跃邻居的最大数量。该算法在 CONGEST 中运行 O（直径（G）$\cdot$对数$^2\, n \cdot$对数 Δ_A）轮次。

5.1 通过采样进行随机化稀疏化

我们首先介绍一种随机化稀疏化算法，用于寻找满足引理 5.1 所述性质的某个稀疏顶点集合，并且以高概率成立。参见算法 1。该算法由 $r :=$ ⌊对数 $\Delta_A -$ 对数对数 n⌋ $- 5$ 个阶段组成。令 $H_1 := A$，其中 $A \subseteq V$是任意一组初始活跃节点。对于 $1 \leq i \leq r$， H_i 是相应阶段中的活跃节点。在每个阶段 i，我们抽取一个集合 $M_i \subseteq H_i$。每个节点 $v \in H_i$ 以 $\frac{24 \cdot 2^i \cdot \text{对数}\, n}{\Delta_A}$ 的概率被纳入 M_i，其中节点的决策是 8 对数 n-重独立的。我们停用所有已抽取的节点，以及在 G 中两跳范围内存在已抽取节点的活跃节点。具体做法是从每个已抽取节点发送一个标志，并将其传播两跳；多个传入标志可以合并为一个进行转发。节点一旦被停用，就永久保持非活跃状态。令 $H_{i+1} = H_i \setminus (M_i \cup N^2(M_i))$ 表示剩余活跃节点。经过 r 个阶段后，算法返回 $Q := \cup_{i=1}^{r+1} M_i$，该集合由已抽取的集合 M_1、.....、M_r 以及剩余活跃节点 $M_{r+1} := H_{r+1}$ 组成。

算法 1： 随机化稀疏化

输入： $\Delta_A \geq$ 最大值$_{v \in V}\, d(v,\ A)$，
每个 $v \in V$都知道自己是否属于初始活跃节点的集合 $A \subseteq V$。
设置 $r := \lfloor \log \Delta_A - \log \log\ n \rfloor - 5$ 以及 $H_1 := A$
对于阶段 $i = 1,\ \ldots,\ r$：
  $M_i := \emptyset$
  对 $v \in H_i$ 并行执行：
    以概率加入 M_i $\frac{24 \cdot 2^i \cdot \text{对数}\ n}{\Delta_A}$
  $H_{i+1} := H_i \setminus (M_i \cup N^2(M_i))$
$M_{r+1} := H_{r+1}$
返回 $Q := \cup_{i=1}^{r+1} M_i$

定义 5.2 (活跃度)。 对于 $v \in V$，其在阶段 i 的活跃度定义为 $d(v, H_i)$。如果 $v \in V$ 在阶段 i 满足 $d(v, H_i) \geq \Delta_A / 2^i$，则称其具有较高的活跃度。

节点的活跃度在算法运行过程中会发生变化。注意，非活跃节点永远不会重新激活，因此对所有 $v \in V$，都有 $d(v, H_1) \geq d(v, H_2) \geq \cdots \geq d(v, H_{r+1})$。此外， v 自身处于活跃或非活跃状态，不会直接影响其活跃度。接下来，我们证明每个阶段结束后以高概率成立的性质。

定理 5.3 （ [SSS95]中的定理 2.5）。 设 X 是 p 重独立的 $[0, \lambda]$ 值随机变量之和，其期望为 $\mu = \mathbb{E}$（X），并设 $\delta \leq 1$。则 $\mathbb{P}$（$|X - \mu| \geq \delta\mu$） $\leq e^{-\lfloor \min\{p/2, \delta^2 \mu / (3\lambda)\} \rfloor}$。

引理 5.4（阶段 i 的随机化稀疏化）。固定某个阶段 $1 \le i \le r$，并令 $H_i \subseteq V$ 为活跃节点的集合，使得所有节点 $v \in V$ 在 H_i 中至多有 $\Delta_A/2^{i-1}$ 个邻居。算法 *1* 的第 i 个阶段返回 M_i 和 H_{i+1}，使得对所有 $v \in V$：

(i) $d(v, M_i) \le$ 72对数 n，概率至少为 $1 - 1/n^3$。

(ii) 如果 $d(v, H_i) \ge \Delta_A/2^i$，那么 $v \in M_i \cup N(M_i)$，概率至少为 $1 - 1/\quad n^3$.

(iii) $d(v, H_{i+1}) < \Delta_A/2^i$，概率至少为 $1 - 1/n^3$。

一个阶段需要 *2* 轮次。如果随机选择是 $8\log n$-路独立的，则这些结论成立 nt.

证明。对于一个活跃节点 $w \in H_i$，令 X_w 为事件“w 被采样”的指示变量。**（i）的证明**。令 $v \in V$ 为任意一个顶点。根据假设，v 至多有 $\Delta_A/2^{i-1}$ 个活跃邻居。令 W 为一个虚拟顶点集合，将其添加到 v 的活跃邻居集合中，使得 v 恰有 $\Delta_A/2^{i-1}$ 个活跃邻居。对于虚拟顶点 $w \in W$，以与真实活跃节点相同的方式定义指示变量 X_w。显然，当且仅当 $N(v, H_i) \cup W$ 中至多有 72 log n个顶点被采样时，v 至多有 72 log n 个真实的已采样邻居。利用定理 5.3，可以给出该事件概率的下界。令 $X = \sum_{w \in N(v, H_i) \cup W} X_w$ 为指示变量之和，其期望值为 $\mu := \mathbb{E}[X] = \frac{\Delta_A}{2^{i-1}} 24 \cdot 2^i \cdot$对数 $\Delta_A \quad ^n = 48 \log$ ~~n。令 $\delta =$~~ $\frac{1}{2}$ 为定理 5.3 中的参数：

$$\begin{aligned}\mathbb{P}(X \ge 72 \log n) &\le \mathbb{P}(|X - 48 \log n| \ge \frac{1}{2} 48 \log n) \\ &\le e^{-\lfloor \min\{8 \log n/2, 48 \log n/12\} \rfloor} = e^{-\lfloor 4 \log n \rfloor} \le n^{-3} .\end{aligned}$$

因此，v 在 M_i 中的邻居数至多为 72 log n，其概率至少为 $1 - 1/n \quad ^3$.

（ii）的证明。设 $v \in V$ 是一个至少有 $\Delta_A/2^i$ 个活跃邻居的节点。我们将计算 v 与一个已采样节点相邻的概率下界。任取活跃邻居的一个子集 $S \subseteq N(v, H_i)$，使得 $|S| = \Delta_A/2^i$。令 $X = \sum_{w \in S} X_w$ 为指示变量之和，其期望值为 $\mu := \mathbb{E}[X] = \frac{\Delta_A}{2^i} 24 \cdot 2^i \cdot \log_{\Delta_A} \quad ^n = 24 \log n$。当 $X > 0$ 时，至少有一个邻居被采样。~~利用定理~~ 5.3，可以给出 $X = 0$ 的概率上界。令 $\delta = \frac{3}{4}$：

$$\begin{aligned}\mathbb{P}(X = 0) &\le \mathbb{P}(|X - \mu| \ge (3/4) \cdot \mu) \\ &\le e^{-\lfloor \min\{8 \log n/2, 72 \log n/16\} \rfloor} = e^{-\lfloor 4 \log n \rfloor} \le n^{-3} .\end{aligned}$$

因此，v 的至少一个活跃邻居被采样的概率至少为 $1 - 1/n^3$。

（iii）的证明。首先，考虑 $v \in V$，其中 $d(v, H_i) < \Delta_A/2^i$。由于节点不会重新激活，v 在阶段 $i+1$ 中的活跃度也将小于 $\Delta_A/2^i$。现在，设 $v \in V$ 是阶段 i中任意一个活跃度较高的节点，也就是说，$d(v, H_i) \ge \Delta_A/2^i$。该算法会停用已采样节点的距离为 2 的邻域。因此，每当某个 $w \in N(v, H_i)$ 被采样时，所有 $N(v, H_i)$ 都会被停用。根据（ii），这种情况发生的概率至少为 $1 - \frac{1}{n^3}$。此时，在所有后续阶段 $i < j \le r$中，v 的活跃度$d(v, H_j)$ 均为零。 □

现在我们可以证明引理 5.1 的一个随机化版本，其运行轮次为 $O(\log \Delta)$：算法 1 的 $r = O(\log \Delta_A) = O(\log \Delta)$ 个阶段产生 $Q := \cup_{i=1}^{r+1} M_i$，由已采样集合 M_1、……、M_r 以及剩余活跃节点 $M_{r+1} := H_{r+1}$组成。引理 5.4 (i) 指出，在最大活跃

度在前一阶段降至 $\Delta_A/2^{i-1}$ 的条件下，高概率地，任意节点在任一 M_i 中都有 M_j 个邻居。根据引理 5.4 (iii)，高概率地，任意节点的最大活跃度都会降低。对于任意 $i \neq j$，集合 M_i 和 M_j 之间的距离至少为 2，因此任意节点在至多一个 M_i 中有邻居。对所有阶段应用并合界可知，任意节点在整个 $Q = \cup_{i=1}^{r+1} M_i$ 中至多有 $O(\log n)$ 个邻居。这也包括剩余活跃节点 $M_{r+1} := H_{r+1}$。接下来，我们通过对每个阶段中活跃节点的随机选择进行去随机化，构造一种确定性稀疏化算法。

5.2 通过去随机化实现确定性稀疏化

我们构造一种确定性稀疏化算法，以证明引理 5.1。该确定性算法（算法 2）称为 DetSparsification。DetSparsification 的结构与随机化稀疏化算法（算法 1）相同。与之前一样，输入是活跃节点集合 $A \subseteq V$ 和最大活跃度参数 $\Delta_A \geq \max_{v \in V} d(v, A)$，其中每个 $v \in V$ 都知道 $v \in A$，以及 Δ_A 的值。共有 $r = \lfloor \log \Delta_A - \log \log n \rfloor - 5$ 个阶段。对于 $1 \leq i \leq r$，H_i 表示第 i 个阶段中的活跃节点集合，其中 $H_1 := A$ and $H_i \subseteq H_{i-1}$。固定某个阶段 $1 \leq i \leq r$。在第 i 个阶段中，DetSparsification 通过对算法 1 第 i 个阶段的采样过程进行去随机化，选择集合 $M_i \subseteq H_i$，所需轮次为 $O(\text{diam}(G) \cdot \log^2 n)$。下一条引理将对此进行描述。算法的其余部分与随机化版本完全相同。剩余活跃节点为 $H_{i+1} = H_i \setminus (M_i \cup N^2(M_i))$。经过 r 个阶段后，算法返回 $Q := \cup_{i=1}^{r+1} M_i$，其中包括采样集合 $M_1, \ldots, M_r$ 和剩余活跃节点 $M_{r+1} := H_{r+1}$。

算法 2: DetSparsification

输入： $\Delta_A \geq$ 最大值$_{v \in V} d(v, A)$，
每个 $v \in V$ 都知道自己是否属于初始活跃节点集合 $A \subseteq V$。

$r := \lfloor \log \Delta_A - \log \log n \rfloor - 5$

$H_1 := A$

对于阶段 $i = 1, \ldots, r$ **：**

- 使用断言 5.6 找到 $M_i \subseteq H_i$，使得对于任意 $v \in V$，Ψ_v 和 Φ_v (1, 2) 都不会出现
- $H_{i+1} := H_i \setminus (M_i \cup N_G^2(M_i))$
- **对每个** $v \in V$ 并行执行**：**
  - 告知每个 $w \in N(v)$ 是否满足 $v \in H_{i+1}$

$M_{r+1} := H_{r+1}$

返回 $Q : = \cup_{i=1}^{r+1} M_i$

引理 5.5 (去随机化 i阶段)。一个阶段的轮复杂度为 $O(\text{直径}(G) \cdot \text{对数}^2 n)$。固定一个阶段 $1 \leq i \leq r$，并令 $H_i \subseteq V$ 为活跃节点的集合，使得所有节点 $v \in V$ 在 H_i中最多有 $\Delta_A/2^{i-1}$ 个邻居。*DetSparsification* 的第 i阶段返回 M_i 和 H_{i+1}，使得对于所有 $v \in V$:

(i) $d(v, M_i) \leq$ 72对数 n, , ***(ii)*** 如果 $d(v, H_i) \geq \Delta_A/2^i$, 那么 $v \in M_i \cup N(M_i)$, ***(iii)*** $d(v, H_{i+1}) < \Delta_A/2^i$.

证明。 我们对随机采样算法的一个阶段 $1 \le i \le r$ 进行去随机化（参见第 5.1 节和算法 1）。每个活跃节点 $v \in H_i$ 以概率 $24 \cdot 2^i$ 对数$_{\Delta_A}{}^{n}$ 被采样。完成去随机化后，我们希望对所有~~$v \in V$~~保证两类事件。首先，所有$v \in V$在 M_i中最多有 $72 \log n$ 个邻居。其次，在阶段 i 中具有高活跃度的节点至少有一个被采样邻居，或者节点自身被采样。这样做的目的是确保图中的最大活跃度降低，因为任何高活跃度节点的邻域都会被停用。

定义这些事件的补事件的指示变量。令 $Z = \{v \in V : d(v, H_i) \ge \Delta_A/2^i\}$ 为阶段 i中活跃度较高的节点集。对于每个 $v \in Z$，令 Φ_v 为一个指示变量，表示 v 未被采样且没有被采样邻居：

$$\Phi_v := \begin{cases} 1 & v \notin M_i \cup N(M_i) \\ 0 & \text{else} \end{cases} \tag{1}$$

为方便起见，我们还将 Φ_v 针对 $v \in V \setminus Z$ 定义为 $\Phi_v = 0$。根据引理 5.4 (ii)，对于所有 $v \in Z$，v 的邻居均未被采样或其自身未被采样的概率至多为$1/n^3$。因此，对于所有 $v \in V$，

$$\mathbb{P}(\Phi_v = 1) \le \frac{1}{n^3}$$

接下来，对于所有 $v \in V$（包括 Z），令 Ψ_v 为一个指示变量，表示 v 拥有超过 $72 \log n$ 个被采样邻居：

$$\Psi_v := \begin{cases} 1 & d(v, M_i) > 72 \log n \\ 0 & \text{else} \end{cases} \tag{2}$$

根据假设，阶段 i 开始时的最大活跃度至多为 $\Delta_A/2^{i-1}$。在此假设下，引理 5.4 (i)指出，对于所有 $v \in V$，

$$\mathbb{P}(\Psi_v = 1) \le \frac{1}{n^3}$$

对于每个 $v \in H_i$，令 X_v 为一个指示变量，表示其被采样到 M_i的事件。我们使用断言 5.6 来确定活跃节点的决策：

断言 *5.6*。固定某个阶段 $1 \le i \le r$。令 $H_i \subseteq V$，并令 $\Delta_A = O(n)$ 为一个整数参数。设 T 是一棵以 $r \in V$ 为根、以分布式方式给定的生成广度优先搜索树。令 $\{X_v\}_{v \in H_i}$ 为一组 $8 \log n$ 重独立的二元变量，且 $\mathbb{P}(X_v = 1) = 24 \cdot 2^i \cdot \log_{\Delta_A}{}^{n}$。对于每个 $v \in V$，令 Φ_v 和 Ψ_v 为事件，且每个事件的概率至多为 $\frac{1}{n^3}$。令 $vbl(v) \subseteq H_i$ 为节点的一个（唯一最小）集合，使得 X_w，$w \in vbl(v)$ 的取值能够确定 Φ_v 和 Ψ_v。假设每个 $v \in V$ 都知道 $vbl(v)$ 中的标识符。存在一种确定性 *CONGEST* 算法，为每个 $v \in H_i$ 选择 X_v 的取值，使得所有事件均不发生。所有通信均通过 T 中的汇聚操作完成。轮复杂度为 $O(diam(G) \cdot \log^2 n)$。

证明。 我们通过从由 $\mathcal{H}$ 个 $8 \log n$ 重独立哈希函数组成的函数族中均匀随机地选择一个哈希函数 $h: [n] \to [\Delta_A]$，来模拟活跃节点的随机选择。以概率 $24 \cdot 2^i \cdot \log_{\Delta_A}{}^{n}$ 设置 $X_v = 1$，等价于当 $h(v) \le 24 \cdot 2^i \cdot \log n$ 时设置 ~~$X_v = 1$~~。根据引理 2.3，从 $\mathcal{H}$ 中随机选择一个函数需要 $\gamma := 8\log^2 n$ 个随机比特。

如果一个哈希函数 $h \in \mathcal{H}$ 良好地作用于变量（Φ_v 或 Ψ_v），使其值为零，也就是说，相应事件不发生，我们就称它为良好哈希函数。每个事件发生的概率至多为 $1/n^3$，事件总数为 $2n$。因此，

$$\mathbb{E}\Big(\sum_{v\in V} \Phi_v + \Psi_v\Big) \le 2n \cdot \frac{1}{n^3} = \frac{2}{n^2}$$

当 n 足够大时，非期望事件的期望数量小于 1。因此，根据全期望定律，存在一个哈希函数，使得所有非期望事件都不会发生。

令 $B = (B_1, \ldots, B_\gamma)$ 为用于选择哈希函数的随机比特。我们使用条件期望法，逐个固定随机比特的值。现在，我们说明在已固定值 $B_1 = b_1, \ldots, B_{j-1} = b_{j-1}$ 的情况下，如何选择第 j 个随机比特 B_j 的值。考虑任意一个节点 $v \in V$。v 计算两个条件期望：

$$\alpha_{v,0} = \mathbb{E}[\Phi_v + \Psi_v \mid B_1 = b_1, \ldots, B_{j-1} = b_{j-1}, B_j = 0]$$
$$\alpha_{v,1} = \mathbb{E}[\Phi_v + \Psi_v \mid B_1 = b_1, \ldots, B_{j-1} = b_{j-1}, B_j = 1]$$

期望 $\alpha_{v,b}, b \in \{0,1\}$ 是在前缀为 $b_1, \ldots, b_{j-1}, b$ 的哈希函数中，$\Phi_v + \Psi_v$ 的平均结果。给定任意哈希函数 $h \in \mathcal{H}$，v 可以检查 h 对其变量而言是否良好。为此，v 确定 $X_w, w \in \mathsf{vbl}(v)$ 的值：当且仅当 $h(\mathsf{ID}(w)) \le 24 \cdot 2^i \cdot \log n$ 时，X_w 的值为 1。此过程需要零轮，因为根据假设，$\mathsf{vbl}(v)$ 中的这些标识符对 v 已知。因此，计算条件期望需要零轮。接下来，我们在 T 的根节点 r 中计算和 $\sum_{v\in V} \alpha_{v,b}$，$b \in \{0,1\}$。这通过并行运行引理 4.3 的两个实例来完成，每个比特对应一个实例。计算这些和的轮复杂度为 $O(\mathsf{diam}(G))$ 轮[5]。根节点选择 $b_j := \mathsf{argmin}_{b\in\{0,1\}} \sum_{v\in V} \alpha_{v,b}$，并将选定的值发送给所有节点。

一旦所有 γ 位都固定下来，哈希函数的选择就是确定性的。对于所有节点 $v \in V$，Φ_v 和 Ψ_v 的结果现在都由固定的种子决定。现在可以将发生的不期望事件数量写作 $\sum_{v\in V} \mathbb{E}[\Phi_v + \Psi_v \mid B_1 = b_1, \ldots, B_\gamma = b_\gamma]$。根据全期望定律，我们有

$$\sum_{v\in V} \mathbb{E}[\Phi_v + \Psi_v \mid B_1 = b_1, \ldots, B_\gamma = b_\gamma] \le \sum_{v\in V} \mathbb{E}[\Phi_v + \Psi_v] = \mathbb{E}\Big[\sum_{v\in V} \Phi_v + \Psi_v\Big] < 1$$

因此，不会发生任何不期望事件。

总时间复杂度为 $O(\mathsf{diam}(G) \cdot \log^2 n)$。这包括逐位固定 $\gamma = O(\log^2 n)$ 位。对于每一位，节点可以在零轮次内计算条件期望 $\alpha_{v,0}$，$\alpha_{v,1}$。将条件期望的总和汇集到某个根节点需要 $O(\mathsf{diam}(G))$ 轮次。根节点将该位的固定值发送给图中其余节点需要 $O(\mathsf{diam}(G))$ 轮次。所有 γ 位都固定后，活跃节点无需轮次即可从哈希函数中提取其决策。■

对于所有 $v \in V$，事件 Φ_v 和 Ψ_v 只取决于活跃邻居的决策。每个 $v \in V$ 都可以在一轮通信中获知 $N(v, H)$ 中的标识符。各事件发生的概率至多为 $\frac{1}{n^3}$。—

[5] 固定第 i 位时，$\alpha_{v,b}$ 是一个形如 $x_{v,b}/2^{\gamma-i+1}$ 的分数，其中 $x_{v,b} \in [2 \cdot 2^\gamma]$ 是 Ψ_v 和 Φ_v 在至多 $2^\gamma = n^2$ 个前缀为 $b_1, \ldots, b_{i-1}, b$ 的哈希函数中出现的总次数。所有节点的分母都相同，因此我们可以改为发送 $x_{v,b}$，占用 $O(\log n)$ 比特空间。

断言 5.6 提供了一种确定性算法，用于修正 X_v，$v \in H$ 的值，使得对于任意 $v \in V$，Φ_v 和 Ψ_v 都不会发生。令 $M_i = \{v \in H_i : X_v = 1\}$ 为采样节点的集合。

证明 (i)。所有节点的 $d(v, M_i) \leq 72 \log n$，因为 $\Psi_v = 0$ 对所有 $v \in V$ 均成立。**证明 (ii)**。任何至少有 $\Delta_A/2^i$ 个活跃邻居的 $v \in V$ 都属于 $M_i \cup N(M_i)$，因为 $\Phi_v = 0$ 对所有 $v \in Z$ 均成立。

证明 (iii)。与随机化算法一样，采样节点及其距离为 2 的邻域会从活跃节点集合中移除。令 $H_{i+1} \subseteq H_i$ 为剩余活跃节点的集合。任何满足 $d(v, H_i) \geq \Delta_A/2^i$ 的 $v \in V$ 的邻域都会被完全停用，因为 $N(v, H_i)$ 中至少有一个活跃节点被采样。因此，对所有 $v \in V$，$d(v, H_{i+1}) < \Delta_A/2^i$。

总时间复杂度为 $O(\text{diam}(G) \cdot \log^2 n)$：根据断言 5.6，去随机化需要 $O(\text{diam}(G) \cdot \log^2 n)$，而停用采样节点的距离为 2 的邻域需要 2 轮。□

引理 *5.1* 的证明。运行 DetSparsification 算法。阶段数为 $r = \lfloor \log \Delta_A - \log \log n \rfloor - 5$。注意，我们可以假设 $\Delta_A \geq 2^5 \log n$,[6]，这样 r 就是非负的。DetSparsification 返回 $Q = \cup_{i=1}^{r+1} M_i \subseteq A$，其中 $M_1, \ldots, M_r$ 是在阶段 $1 \leq i \leq r$ 中选出的集合，而 $M_{r+1} = H_{r+1}$ 是剩余活跃节点的集合。

支配性质：我们首先证明 Q 的支配性质，即 $\forall v \in V$：$\text{dist}_G(v, Q) \leq 2 + \text{dist}_G(v, A)$。令 $v \in V$ 为任意节点，并令 $w \in A$ 为距离 v 最近的任意初始活跃节点，即 $\text{dist}_G(v, w) = \text{dist}_G(v, A)$。在算法中，$w$ 有三种可能的结果：(1) $w \in Q$。此时 $\text{dist}_G(v, Q) = \text{dist}_G(v, A)$。(2) w 在某个阶段 i 被停用。根据定义，存在某个 $w' \in N^2(w)$ 使得 $w' \in M_i$。因此 $\text{dist}_G(v, Q) \leq \text{dist}_G(v, A) + 2$。(3) w 从未被停用。因此 $w \in H_{r+1}$。剩余活跃节点 $M_{r+1} := H_{r+1}$ 被包含在 Q 中。因此 $\text{dist}_G(v, Q) = \text{dist}_G(v, A)$。在所有情况下，$\text{dist}_G(v, Q) \leq 2 + \text{dist}_G(v, A)$。

稀疏性： 我们证明关于有界 Q-度的断言，$\forall v \in V$：$d(v, Q) \leq 72 \log n$。对于任意阶段 $1 \leq i \leq r$，引理 5.5 指出，假设阶段 i 开始时的最大活跃度至多为 $\Delta_A/2^{i-1}$，则任何 $v \in V$ 在 M_i 中的邻居至多有 $72 \log n$ 个。对于 $i = 1$，该假设成立，因为根据定义，$\Delta_A \geq \max_{v \in V} d(v, A)$。对于阶段 $i = 2$、.....、r，引理 5.5 (iii) 保证了这一点；该引理指出，阶段 $i - 1$ 结束时的最大活跃度至多为 $\Delta_A/2^{i-1}$。最后，$M_{r+1} = H_{r+1}$ 由经过 r 个抽样阶段后仍处于活跃状态的节点组成。根据引理 5.5 (iii)，在阶段 $r = \lfloor \log \Delta_A - \log \log n \rfloor - 5$ 之后，最大活跃度为

$$\frac{\Delta_A}{2^{\lfloor \log \Delta_A - \log \log n \rfloor - 5}} \leq \frac{\Delta_A}{2^{\log \Delta_A - \log \log n - 6}} = 64 \log n$$

因此，对于任意 $1 \leq i \leq r + 1$，M_i 中的任意 $v \in V$ 至多有 $72 \log n$ 个邻居。最后，M_i 中的节点与 $M_j, i \neq j \in 1, \ldots, r$ 中的节点没有任何共同邻居，因为采样节点的距离为 2 的邻域已被停用。出于同样的原因，M_{r+1} 中的节点与 $M_i, 1 \leq i \leq r$ 中的节点没有任何共同邻居。由此可知，Q 中每个节点至多有 $72 \log n = O(\log n)$ 个邻居。

DetSparsification 的运行时间由 $\lfloor \log \Delta_A - \log \log n \rfloor - 5$ 个阶段组成，根据引理 5.5，每个阶段需要 $O(\text{diam}(G) \cdot \log^2 n)$ 轮次。总运行时间为 $O(\text{diam}(G) \cdot \log \Delta \cdot \log^2 n)$。□

5.3 幂图的稀疏化

在本节中，我们为 G^k 提出一种确定性稀疏化算法，并证明引理 3.1。

[6]如果 $\Delta_A < 2^5 \log n$，那么根据 Δ_A 的定义，$\max_{v \in V} d(v, A) < 2^5 \log n$。返回初始活跃节点集合 A，此时它满足引理 5.1 的两个条件。

构造概要 </style>令 $Q_0 \subseteq V$ 为任意初始活跃节点集合。我们的算法由 k 次针对 G 的确定性稀疏化算法迭代组成（DetSparsification，算法 2）。第一次迭代在 G 中运行 DetSparsification，并将活跃节点初始化为 Q_0。在第 $2 \leq s \leq k$ 次迭代中，DetSparsification 在幂图 G^s 上进行模拟，其通信网络为 G。第 s 次迭代的活跃节点集合初始化为 $Q_{s-1} \subseteq V$，其中 Q_{s-1} 是上一次迭代的结果。第 s 次迭代的结果是某个 $Q_s \subseteq Q_{s-1}$。经过 k 次迭代后，算法输出 Q_k。

完整的算法描述以及如何在 G^s 上模拟 DetSparsification 的详细信息将在后文给出。每次迭代 s 中模拟的效率取决于前一次迭代中 DetSparsification 返回的活跃节点集合 Q_{s-1} 的性质。具体而言，我们维护以下不变量：

算法不变量 针对 G^k 的稀疏化算法会找出一系列节点集合$Q_0 \supseteq Q_1 \supseteq \cdots \supseteq Q_k$。在所有迭代 $1 \leq s \leq k$ 完成后，以下不变量确定性地成立：

I1.1（距离-s Q_s-度有界） $\forall v \in V$： $d^s(v, Q_s) \leq 72 \log\, n$

I1.2（距离-$(s+1)$ Q_s-度有界） $\forall v \in V$： $d^{s+1}(v, Q_s) \leq 72\Delta \log\, n$

I2（支配） $\forall v \in V$: $\mathrm{dist}_G(v, Q_s) \leq \sum_{j=1}^{s} 2j + \mathrm{dist}_G(v, Q_0) = s^2 + s + \mathrm{dist}_G(v, Q_0)$

I3（关于距离为 $s+1$ 的 Q_s 邻域的知识）所有 $v \in V$ 都知道 $N^{s+1}(v,\ Q_s)$ 中的标识符集合。此外，对于每个 $x \in Q_s$，都存在一棵以 x 为根、深度为 $s+1$ 的 BFS 树。每个 $v \in V$ 都知道对于其所属的每个 T_x，根节点 x 的标识符、祖先 $(T_x,\ v) \in N(v)$ 以及后代 $(T_x,\ v) \subseteq N(v)$。

算法描述 令 $k \geq 1$ 为一个整数，$Q_0 \subseteq V$ 为任意集合。该算法由 k 次 DetSparsification（算法 2）迭代组成，其中第 s 次迭代在幂图 G^s 上进行模拟。

在第一次迭代中，我们在 G 上运行 DetSparsification。活跃节点集合初始化为 Q_0，最大活跃度参数 Δ_A 设为 $\Delta_A^{(1)} := \Delta$。令 Q_1 为第一次迭代中由 DetSparsification 返回的节点集合。为了准备在下一次迭代中于 G^2 上进行模拟，我们必须维护不变量 (I3)。为此，每个 $v \in V$ 在 O 中用 $\log\, n$ 轮次将 $N(v,\ Q_1)$ 中的标识符集合发送给所有邻居 $w \in N(v)$，如引理 4.1 所述。因此，每个 $v \in V$ 都知道 $N^2(v,\ Q_1)$ 中的标识符集合，并且对于每个 $x \in Q_1$，都存在一棵深度为 2、根节点为 x 的 BFS 树。

固定某次迭代 $2 \leq s \leq k$。假设不变式在迭代 $s-1$中成立。我们在幂图 G^s 上模拟 DetSparsification，其通信网络为 G。活跃节点初始化为 Q_{s-1}，其中 Q_{s-1} 是上一次迭代的结果。最大活跃度参数 Δ_A 设为 $\Delta_A^{(s)} := 72\Delta \log\, n$（对所有 $2 \leq s \leq k$）。模拟的详细信息见引理 5.7。设 $Q_s \subseteq Q_{s-1}$ 为本次迭代中 DetSparsification 返回的节点集。在迭代结束时，我们必须再次维持 (I3)。每个 $v \in V$ 在 $N^s(v,\ Q_s)$ 中向其每个邻居 $w \in N(v)$ 发送标识符集合，并在 $O(\log\, n)$ 内完成，如引理 4.1 所述。现在，所有节点 $v \in V$都知道 $\cup_{w \in N(v)} N^s(w,\ Q_s) = N^{s+1}(v,\ Q_s)$ 中的标识符集合，并且 Q_s 中节点的 BFS 树扩展至深度 $s+1$。$Q_{s-1} \setminus Q_s$ 中节点的树不再使用。至此，算法的一次迭代完成。

经过 k 次迭代后，算法输出 Q_k。

证明　　变体。我们通过对迭代次数进行归纳法来证明这些不变量　　s.

基础情形 $s = 1$。(I1.1) 和 (I2) 可由引理 5.1 中对 DetSparsification 的分析得出。该引理指出，对于任意 $v \in V$，都有 $d_G(v, Q_1) \leq 72 \log n$，且 $\text{dist}_G(v, Q_1) \leq 2 + \text{dist}_G(v, Q_0)$。(I1.2) 是 (I1.1) 的直接推论，因为对于任意 $v \in V$，$d^2(v, Q_1) = |N^2(v, Q_1)| = |\cup_{w \in N(v)} N(w, Q_1)| \leq \sum_{w \in N(v)} |N(w, Q_1)| \leq \Delta \cdot 72 \log n$。(I3) 由引理 4.1 得出。

假设这些不变量对 $s-1$ 成立。在第 s 次迭代中，我们将 Q_{s-1} 设为活跃节点的初始集合。根据第 $s-1$ 次迭代的 (I1.2)，对于所有 $v \in V$，距离-s Q_{s-1}-度至多为 $72\Delta \log n$。DetSparsification 中的最大活跃度参数 Δ_A 被设为 $\Delta_A^{(s)} = 72\Delta \log n$。这满足引理 5.1 中的要求 $\Delta_A \geq \max_{v \in V} d(v, A)$（由于我们在 G^s 上模拟 DetSparsification，该引理的形式为 $\Delta_A \geq \max_{v \in V} d^s(v, A)$）。现在可以应用引理 5.1。对于 G^s，引理 5.1（有界- Q-度）指出，对所有 $v \in V$，$d^s(v, Q_s) \leq 72 \log n$，从而证明 (I1.1)。(I1.2) 同样是 (I1.1) 的推论，因为对于任意 $v \in V$，$d^{s+1}(v, Q_s) = |N^{s+1}(v, Q_s)| = |\cup_{w \in N(v)} N^s(w, Q_s)| \leq \sum_{w \in N(v)} d^s(w, Q_s) \leq \Delta \cdot 72 \log n$。对于 (I2)，我们使用引理 5.1（支配），该引理指出 $\forall v \in V$: $\text{dist}_{G^s}(v, Q_s) \leq 2 + \text{dist}_{G^s}(v, Q_{s-1})$。在 G^s 中，距离增加 2 对应于 G 中至多增加 $2s$。结合第 $s-1$ 次迭代的 (I2)，可得 $\text{dist}_G(v, Q_s) \leq 2s + \text{dist}_G(v, Q_{s-1}) \leq 2s + (s-1)^2 + s - 1 + \text{dist}_G(v, Q_0) = s^2 + s + \text{dist}_G(v, Q_0)$。最后，(I3) 根据构造成立（见引理 4.1）。□

引理 *3.1* 的证明。将活跃节点初始化为 $Q_0 \subseteq V$。使用 k 次迭代运行第 5.3 节的算法。令 $Q := Q_k$ 为最后一次迭代的结果。

算法的正确性由这些不变量推出。对于第 k 次迭代，(I1.1) 表明 $\forall v \in V$: $d^k(v, Q) \leq 72 \log n$，并且根据 (I2)，$\forall v \in V$: $\text{dist}_G(v, Q) \leq k^2 + k + \text{dist}_G(v, Q_0)$。

最后，我们分析算法的运行时间。共有 k 次迭代。在每次迭代 s 中，我们运行 DetSparsification，包含 $\lfloor \log \Delta_A - \log \log n \rfloor - 5$ 个阶段。参数 Δ_A 设置为 $\Delta_A^{(1)} = \Delta$，用于$s = 1$；对于 $2 \leq s \leq k$，则设置为 $\Delta_A^{(s)} = 72\Delta \log n$。因此，所有迭代中的阶段总数为 $O(\log \Delta)$。根据引理 5.7，在 G^s 上模拟 DetSparsification 需要 $O(\text{diam}(G) \cdot \log^2 n \cdot \log \Delta + s \log \Delta)$ 轮次。在每次迭代结束时，每个 $v \in V$ 都会将 $N^s(v, Q_s)$ 中的标识符集合发送给其每个邻居$w \in N(v)$。通过流水线传输标识符需要 $O(\log n)$ 轮次，因为根据 (I1)，$d^s(v, Q_s) = O(\log n)$。第 s 次迭代的时间复杂度为 $O(\text{diam}(G) \cdot \log^2 n \cdot \log \Delta + s \log \Delta + \log n) = O(\text{diam}(G) \cdot \log^2 n \cdot \log \Delta + k \log \Delta)$。因此，总时间复杂度为 $O(k \cdot \text{diam}(G) \cdot \log^2 n \cdot \log \Delta + k^2 \cdot \log \Delta)$。□

最后，我们展示如何在 G^s 上模拟 DetSparsification，适用于任意 $1 \leq s \leq k$，其中活跃节点初始化为前一次迭代 $s-1$ 的结果。该模拟的效率依赖于前一次迭代的不变式 (I1.1)。

引理 5.7（在 G^s 上模拟 DetSparsification）。考虑迭代 $2 \leq s \leq k$。令 $\Delta_A = 72\Delta \log n$。将活跃节点集合初始化为 Q_{s-1}，其中 $Q_{s-1} \subseteq Q_0$ 是在迭代 $s-1$ 中返回的节点集。假设迭代 $s-1$ 的不变量均成立，则存在一个确定性CONGEST 算法，可在通信网络 G 中于 G^s 上模拟 ***DetSparsification***，并在 $O(\text{直径}(G) \cdot \log^2 n \cdot \log \Delta + s \log \Delta)$ 轮内完成。

证明。我们模拟 DetSparsification，它由 $r = \lfloor \log \Delta_A - \log \log n \rfloor - 5 = O(\log \Delta)$ 个阶段组成。活跃节点初始化为 $H_1 := Q_{s-1}$ and $\Delta_A = 72\Delta \log n$。引理 5.5 描述了 DetSparsification 的一个阶段。我们展示如何在 G^s 上通过通信网络 G高效地模拟该阶段。固定任意阶段 $1 \leq i \leq r$。假设每个 $v \in V$都知道其活跃邻居的标识符，$N^s(v, H_i)$。根据迭代 $s-1$ 的 (I3)，第一阶段满足这一条件：每个 $v \in V$都知道以下集合的

标识符：$N^s(v,\ H_1) = N^s(v,\ Q_{s-1})$。稍后我们将说明如何在该阶段结束时获知 $N^s(v,\ H_{i+1})$ 中的标识符。

第 i 个阶段首先对活跃节点的选择进行去随机化。为清晰起见，我们先针对 G^s 重述引理 5.5 中事件的定义。对于 $v \in H_i$，令 X_v 为事件“v 已采样”的指示变量。令 $Z = \{v \in V :\ d^s(v,\ H_i) \geq \Delta_A/2^i\}$ 为 G^s 在第 i 个阶段中活跃度较高的节点集。对于每个 $v \in Z$，令 $\Phi_v := \mathbb{1}(v \notin M_i \cup N^s(M_i))$；对于每个$v' \in V \setminus Z$，令 $\Phi_{v'} := 0$。对于每个 $v \in V$，令 $\Psi_v := \mathbb{1}(d^s(v,\ M_i) > 72 \log n)$。注意，这些恰好是针对 G^s 重述后的引理 5.5 中的事件。鉴于我们正在 G^s 中模拟稀疏化算法的第 i 个阶段，引理 5.4 的随机化分析指出，对于所有 $v \in V$，$\mathbb{P}(\Phi_v = 1) \leq 1/n^3$ 以及 $\mathbb{P}(\Psi_v = 1) \leq 1/n^3$。

与 DetSparsification 中一样，我们应用断言 5.6 来确定活跃节点的决策。对于所有 $v \in V$，事件 Φ_v 和 Ψ_v 取决于它们在 G^s中的活跃邻居所作的决策。根据假设，每个 $v \in V$都知道 $N^s(v,\ H_i)$ 中的标识符。因此，根据断言 5.6，可以在 $O(\mathrm{diam}(G) \cdot \log^2 n)$ 轮内确定 $X_v,\ v \in H$的值，使得对于任意 $v \in V$，Φ_v和 Ψ_v都不会发生。请注意，断言 5.6 中唯一的通信，是通过在 G的生成树子图上聚合值来完成的。特别是，无需模拟 G^s中邻居之间的通信。令 $M_i = \{v \in H_i :\ X_v = 1\}$ 为已采样节点的集合。

每个 $v \in M_i$ 都会停用自身。我们还需要停用已采样节点在 G^s 中的距离为 2 的邻域（ G中的距离为 $2s$ 的邻域）。每个 $v \in M_i$ 都会发送一个标志 已采样，该标志会在 G中传播 $2s$ 跳。节点会转发收到的标志，同时跟踪剩余距离。多个标志可以合并为一个，因为它们不包含任何与发送方相关的特定信息，因此不会产生拥塞。轮复杂度为 $2s = O(s)$。任何收到标志 已采样 的活跃节点 $v' \in H_i$都会停用自身。令 $H_{i+1} \subseteq H_i$ 为剩余活跃节点。

为下一阶段做准备，每个 $v \in V$需要了解 $N^s(v,\ H_{i+1})$ 中的标识符 $\subseteq N^s(v,\ H_i)$。为此，每个 $v \in H_i \setminus H_{i+1}$ 都会通知其在 G^s 中的邻域，告知 v 已不再活跃。根据引理 4.2，每个已停用节点 v 广播 一条已停用消息，并将 ID(v) 一同发送给其在 G^s中的所有邻居。根据迭代 $s-1$的 (I1.1)，任意 $v \in V$至多拥有 $O(\log n)$ 个距离为 $s-1$的 Q_{s-1}邻居。根据迭代 $s-1$的 (I3)，每个 $x \in Q_{s-1}$ 周围深度为 s 的 BFS 树均已给出。因此，引理 4.2 可在 $O(s + \log n)$ 轮次内并行发送来自所有已停用节点（可能是 $Q_{s-1} \supseteq H_i$中的全部节点）的广播。节点 $w \in V$利用接收到的广播，通过从集合 $N^s(v,\ H_i)$ 中移除已停用节点，形成关于 $N^s(v,\ H_{i+1})$ 的知识。[7]至此完成了 DetSparsification 一个阶段的模拟。

单个阶段的时间复杂度为 $O(\mathrm{diam}(G) \cdot \log^2 n + 2s + s + \log n)$。共有 $r = O(\log \Delta)$ 个阶段，因此在 G^s 中模拟 DetSparsification 的时间复杂度为 $O(\mathrm{diam}(G) \cdot \log^2 n \cdot \log \Delta + s \log \Delta)$。 □

5.4 无直径依赖的稀疏化

我们将引理 3.1 的结果与网络分解算法相结合，将稀疏化的运行时间提升至 $O(\mathrm{polylog}\ n)$。网络分解的定义见定义 2.1。

[7]注意，在 G^s 中拥有被采样邻居的节点 $v \in V$ 可以在零轮次内完成更新：v 使用由固定随机比特确定的哈希函数 h，检查其活跃邻居 $w \in N^s(v,\ H_i)$ 中是否有被采样的节点。如果存在被采样的节点 $w \in N^s(v,\ H_i)$，那么 N^s中的所有剩余活跃节点 $(v,\ H_i)$ 都已被停用，因为它们位于 G^s中 w的距离为 2 的邻域内。然而，被采样的节点可能不在 $N^s(v,\ H_i)$ 中，而只位于某个活跃节点 $w \in N^s(v,\ H_i)$ 的距离为 $2s$ 的邻域内。在这种情况下，w已被停用，但 v尚未得知这一点。

利用网络分解，经典做法是按顺序遍历网络分解中的各个颜色，贪心地计算解。然而，我们需要谨慎处理，以免破坏后续颜色的簇中节点的解。计算簇 C 的解时，我们还必须考虑距离-k 邻域 $N^k(C)$ 中的节点。这个簇边界 在算法中充当不活跃的观察者，以确保它们在Q中距离-k 邻居的数量有界。此外，簇边界中的节点也很重要，因为 $G^k[C]$ 中的连接也可能经由 C 的邻居形成（距离最多为 $\lfloor k/2 \rfloor$）。因此，为了对同色的每个簇独立运行引理 3.1，我们需要确保簇之间具有 $2k+1$ 分离度。

引理 5.8。设 $k \geq 1$ （可能是 n 的函数）。设 T^{ND} 为确定性计算弱 (c, d) 网络分解的复杂度，其中簇间距离为 $2k+1$，拥塞为 τ。

存在一种确定性分布式算法，给定子集 $Q_0 \subseteq V$，可找到一个顶点集合$Q \subseteq Q_0$，使得对所有 $v \in V$，

- （有界距离-k Q-度）：$d^k(v, Q) \leq 72$ 对数 $n = O($对数 $n)$
- *(支配)*：距离$_G(v, Q) \leq k^2 + k +$ 距离$_G(v, Q_0) = O(k^2) +$ 距离$_G(v, Q_0)$

该算法在 CONGEST 模型中运行 $O\big(T^{ND} + c(d\tau k \cdot$ 对数$^2 n \cdot$ 对数 $\Delta + k^2 \cdot$ 对数 $\Delta)\big)$ 轮次。运行时间为 $\widetilde{O}(k^2 \cdot$ 对数$^4 n \cdot$ 对数 $\Delta)$，使用定理 *A.1* 的网络分解。

证明。我们首先确定性地计算一个弱 (c, d)-网络分解，其簇间距离为 $2k+1$，拥塞为 τ。定义全局活跃节点集合 全局活跃节点 $H_G := Q_0$。我们依次处理网络分解中的 c个颜色类。引理 3.1 的稀疏化算法在同一颜色的簇上并行运行。每处理完一个颜色类，就更新全局活跃节点集合。

固定一个颜色 $1 \leq j \leq c$，令 $C \subseteq V$为颜色 j的任意簇。C 的距离为 k 的邻域也参与该算法。我们将 $N^k(C)$ 称为簇边界。$2k+1$的分离度保证了同一颜色的各簇的簇边界两两不相交。令 T_C 为网络分解给出的 C的斯坦纳树。通过从每个 $v \in C$开始执行广度优先搜索，我们可以在 $O(k)$ 轮次内扩展 T_C，使其包含 $N^k(C)$。簇外的节点总共转发传入搜索中的一个，持续 k 跳。令 T_C 为更新后的斯坦纳树（此时不一定仍是一棵树）。由于簇边界互不相交，任何边上的拥塞至多增加一。

最终输出 Q 是所有颜色和簇 C对应的 Q_C 集合的并集。我们通过证明 Q满足引理 3.1 的两项性质来证明正确性。（有界距离-k Q-度）：考虑任意$v \in V$。令 C 为第一个包含这样一个距离-k 邻居的簇：该邻居属于 v，并被选中用于 Q。C 是唯一的，因为同一颜色的簇之间至少相距 $2k+1$ 跳。根据迭代 k的 (I1)，v的距离-k Q邻居数量至多为 $72 \log n$。此外，任何已采样节点都会停用其距离-$2k$ 邻域内的全局活跃节点。因此，之后的颜色中不会再有更多 v 的距离-k邻居加入 Q。（支配）：令 $x \in Q_0$ 为距离 v最近的任意初始活跃节点。存在三种可能性：（i）$x \in Q$，（ii）x 在其所属簇中一开始就是活跃的，但 $x \notin Q$ （iii）x 在其所属簇运行算法之前就已被停用。对于（i），支配性显然成立。在第二种情况下，Q 中还有另一个节点位于 x所属的簇中，因为 x 不在 Q中。根据引理 3.1，x 到该簇中 Q 内某个节点的距离至多为 $k^2 + k$。在情况（iii）中，x 被停用的原因是另一个簇中有一个被选中的节点，距离 x不超过 $2k$。因此，三种情况下支配性质均成立。

最后，我们分析运行时间。计算网络分解需要 T^{ND} 轮次。我们依次处理 c 种颜色。每个簇的直径为 d，而在

每个簇的斯坦纳树中进行通信会因拥塞而减慢 $O(\tau)$ 倍。计算簇边界并停用其他簇中的全局活跃节点，各自需要 $O(k)$ 轮次。因此，对单个颜色类运行稀疏化算法需要 $O(d\tau k \cdot \log^2 n \cdot \log\ \Delta + k^2 \log\ \Delta)$ 轮次。总运行时间为 $O(T_{\mathrm{ND}} + c(d\tau k \cdot \log^2 n \cdot \log\ \Delta + k^2 \log\ \Delta))$。使用定理 A.1 的网络分解算法 $\widetilde{}$，我们在 $\widetilde{O}(k \cdot \log^3 n)$ 轮次内计算出一个分离度为 $2k+1$、且具有 $\tau = 1$ 的 $(O(\log\ n)\ O(k \cdot \log\ n))$-网络分解。总运行时间为 $\widetilde{O}(k^2 \cdot \log^4 n \cdot \log\ \Delta)$。 □

6 确定性 $(k+1,\ k^2)$-支配集（定理 1.1）

本节将证明我们的主要结果（定理 1.1），即给出一个 $\mathrm{poly}\log n$ 时间的确定性 $(k+1,\ k^2)$-支配集算法。我们首先说明，已知最简单的确定性支配集算法可以扩展到幂图。这些算法的缺点是：它们要么只能提供较差的支配性质，要么比我们的主要结果慢一个指数级别。

幂图的现有支配集算法 有一个广为人知的确定性 $(k+1,\ k \cdot \log\ n)$-支配集算法 [，该算法由 [HKN21]改编至 CONGEST 模型。[KMW18]给出了该算法的一般形式：

定理 6.1 （[AGLP89、SEW13、HKN21、KMW18] ）。设 k 为一个正整数。给定 G 的一个距离-k 着色，其中使用了 γ 种颜色，则存在一种确定性分布式算法，对于任意 $B \geq 2$，可在 $(k+1,\ k \cdot \lceil \log_B \gamma \rceil)$-支配集，并在 CONGEST模型的 $O(k \cdot B \cdot \log_B \gamma)$ 轮次内计算得到该支配集。

证明概要。考虑输入着色以 B 为底数的表示，它可以用 $m := \lceil \log_B \gamma \rceil$ 位数字给出。令 $\chi_i(v)$ 为节点 $v \in V$ 的颜色以 B 为底数表示时的第 i 位数字。令 $U := V$ 为一个节点集，最终将作为支配集返回。初始时，U 是一个平凡的支配集，没有独立性保证。算法依次处理着色的 m 位数字。每次迭代 $1 \leq i \leq m$ 都会保持以下不变式：对于任意 $v,\ w \in U$，若 $\mathrm{dist}_G(v,\ w) \leq k$，则 v 和 w 的颜色在前 i 位数字上相同。这意味着经过 m 次迭代后，U 中距离不超过 k 的任意一对节点在所有位数字上都相同，也就是说，它们的颜色相同。这样的节点对不存在，因为输入是图 G^k 的一个着色，因此结果是 $(k+1)$-独立的。

固定某次迭代 $1 \leq i \leq m$。它包含 B 个步骤，依次遍历第 i 位数字的所有可能取值。在第 s 步，对于任意 $0 \leq s \leq B-1$，满足 $\chi_i(v) = s$ 的节点 $v \in U$ 都会向其距离为 k 的邻域发送蜂鸣信号。任何满足 $\chi_i(w) > s$ 的 $w \in U$，若收到蜂鸣信号，就会永久地从 U 中移除（注意，发出蜂鸣信号的节点在蜂鸣时无需监听其他节点的蜂鸣信号）。这保持了第 i 次迭代的不变式：经过 B 个步骤后，不存在距离为 k 的邻居 $v,\ w \in U$ 满足 $\chi_i(v) \neq \chi_i(w)$。第 s 次迭代最多会使 U 的支配性减弱 k：发出蜂鸣信号并导致其某些距离为 k 的邻居被移除的节点，在本次迭代剩余过程中仍会保留在 U 中。因此，最终支配集的支配性为 $k \cdot m = k \cdot \lceil \log_B \gamma \rceil$。一次迭代的运行时间为 $k \cdot B$ 轮。总运行时间为 $O(k \cdot B \cdot m) = O(k \cdot B \cdot \log_B \gamma)$ □

定理 6.1 的算法在更大的距离下仍具有良好的扩展性，但其支配参数并非常数，且取决于输入着色的大小。此外，幂图的确定性着色很困难。对于更大的距离，通常必须依赖由唯一标识符给出的着色，因此支配参数关于 n 呈对数级。通过选择非常数底数 B，可以实现常数支配，但运行时间会显著增加：

推论 6.2。 设 k 为正整数。存在一种确定性分布式算法，对于任意 $c \geq 1$，可在 CONGEST 模型中用 $O(k \cdot c \cdot n^{1/c})$ 轮计算图G的一个$(k+1, ck)$-支配集。

证明。 使用定理 6.1 的 $(k+1, k\lceil \log_B \chi \rceil)$-支配集算法，将节点的唯一标识符作为颜色，并使用 $B = \lceil n^{1/c} \rceil$。运行时间为 $O(k \cdot c \cdot n^{1/c})$ 轮次。所得支配集的支配度为 $k\lceil \log_{\lceil n^{1/c} \rceil} n \rceil \leq kc$。 □

据我们所知，对于 $k \geq 2$ 和 $k = o(\log n)$，这是唯一能计算出支配度在 n中为常数的支配集的确定性支配集算法。对于 $k = \Omega(\log n)$，可以使用定理 6.1，在 $O(k \cdot \log\ n)$ 时间内计算一个 $(k+1, O(k^2))$-支配集。

网络分解也可以得出支配集。给定一个 (c, d)-网络分解（弱直径或强直径），且同色簇之间的分离度为 $2k$，则可以在 $O(c(d+k))$ 轮次内计算一个 $(k+1, \max(d, k))$-支配集。初始时，所有节点均为活跃节点，且支配集为空。依次遍历各个 c 颜色类。在给定颜色类的各个簇中，任选一个活跃节点（若存在）加入支配集，例如可在 O(d) 时间内选择标识符最小的节点。加入支配集的节点会停用整个图中距离为 k 的邻域。同色不同簇的距离为 k 的邻域互不相交，因此所得集合是 $(k+1)$-独立的。支配度至少为 d，因为每个簇中都有一个支配集节点，除非该簇中的所有节点都已被支配。当前最先进水平的、具有更大分离度的网络分解（见定理 A.1）并未改进定理 6.1 的结果。此外，要找到支配度为常数的幂图支配集，我们需要簇直径为常数。但使用多对数数量的颜色无法做到这一点，因为常数直径意味着对于某个常数 λ [LS93]，至少需要 $\Omega(n^{1/\lambda})$ 个颜色类。

确定性$(k+1,\ k^2)$支配集我们的确定性支配集算法能够在任意 $k \geq 1$ 的多对数时间内计算一个（$k+1$，k^2）-支配集。该算法将稀疏化与诱导子图上的极大独立集过程相结合。稀疏化的目标是找到一个顶点集合 $Q \subseteq V$，使得所有 $v \in V$ 在 G 中都与 Q 距离较近，同时在 Q 中拥有数量有界的距离-$(k-1)$邻居。第二个性质使得我们能够在具有通信网络 G的 $G^k[Q]$ 上高效地模拟算法。特别地，我们可以以黑盒方式运行任意极大独立集算法，以计算 $G^k[Q]$的极大独立集。所得结果即为支配集。下面的引理将这一一般方法进行了形式化描述：

引理 6.3（通过稀疏化构造支配集）。 设 $k \geq 1$ （可能是 n的函数）。用以下问题在CONGEST 模型中确定性求解的复杂度表示为

- $T_{\text{极大独立集}}(n, \Delta)$：在有 n 个节点、最大度为 Δ 的图中求解极大独立集的复杂度
- $T_{\text{稀疏化}}($ k β $\widehat{\Delta}$ $Q \subseteq V$ $v \in V$ d^k v Q $\leq, ,)$：寻找一个子集的复杂度，使得所有元素都具有 $(,)\Delta$ 且 距离$_G(v, Q)$ $\leq \beta$ （在有 n 个节点、最大度为 Δ 的图中）

存在一种确定性分布式算法，给定一个有 n 个节点且最大度为 Δ 的图 $G = (V, E)$，可在 CONGEST 模型下的 $O(T_{\text{稀疏化}}(k-1, \beta, \widehat{\Delta}) + k\widehat{\Delta} + (k+\widehat{\Delta}^2) \cdot T^{MIS}(n, \Delta \cdot \widehat{\Delta}))$ 轮次内，计算出 G 的 $(k+1, \beta+k)$-支配集。

证明。 *</style>*令 和 为任意正整数。首先，找出节点的一个子集 ，使得所有 的距离为 的节点中，度不超过 的 ，且 $dist(, Q) \leq \beta$。找出 Q 的时间复杂度为 $T^{\text{sparsification}}(k-1, \beta, \widehat{\Delta})$。

我们使用引理 4.6，在通信网络 G中模拟于 $G^k[Q]$ 上计算极大独立集。为准备该模拟，我们需要构建深度为 k 、以 Q中的节点为根节点的广度优先搜索树，以供 Q中的节点通信使用。第 5.3 节的不变量 (I3)也提供了这些树，因此使用引理 3.1 的稀疏化算法时，可以跳过以下准备步骤。对于每个 $v \in Q$，令 T_v 为一棵广度优先搜索树，初始时仅包含 v。运行引理 4.1 的 $k-1$轮次。在第 s轮迭代中，对于任意 $1 \leq s \leq k-1$，每个 $v \in V$都会获知 $N^{s+1}(v, Q)$中的标识符集合，并且每个 $v \in Q$的树都会扩展至覆盖 $N^{s+1}(v)$。每个 $v \in V$都知道它所属的每棵树 T的根节点标识符、descendants$(T, v) \subseteq N(v)$以及 ancestor$(T, v) \in N(v)$。第 $\backsim s$轮迭代需要 $O(\Delta)$轮次，使用 $\Theta(\log n)$ 位的通信带宽和 $O(\log\ n)$ 位的标识符。经过 $k-1$ 轮迭代后，所需的广度优先搜索树均已构建完成，并已分布式地为各节点所知。此外，每个$v \in Q$ 都知道其在 $G^k[Q]$中的度以及其邻居的标识符。准备步骤需要 $O(k \cdot \widehat{\Delta})$轮次。

$G^k[Q]$ 的最大度至多为 $\Delta \cdot \widehat{\Delta}$，因为所有 $v \in V$ 在 G 中最多有 Δ 个直接邻居，而这些邻居各自最多有 Δ 个距离为 $k-1$ 的 Q 邻居。$G^k[Q]$ 中的节点数至多为 n。在输入图 $G^k[Q]$ 上计算极大独立集的复杂度为 $T_{\text{极大独立集}}(n, \Delta \cdot \widehat{\Delta})$。我们使用通信网络 G模拟在 $G^k[Q]$ 上计算极大独立集。根据引理 4.6，我们可以使用通信网络 G 模拟在 $G^k[Q]$ 上运行的任意 CONGEST 算法，减速因子为 $O(k + \widehat{\Delta}^2)$。因此，极大独立集步骤的运行时间为 $O((k + \widehat{\Delta}^2) \cdot T^{\mathsf{MIS}}(n, \Delta \cdot \widehat{\Delta}))$。令 $\mathcal{I} \subseteq Q$为 $G^k[Q]$的极大独立集。

关于正确性，注意到图 G^k 的极大独立集是图 G的一个（$k+1$，k）-支配集。图 $G^k[Q]$ 的极大独立集在 G^k中是独立的。它在图 G^k中并非极大，我们只能得出对于所有 $v \in Q$，$\text{dist}_G(v, \mathcal{I}) \leq k$。因此，对于所有 $v \in V$，$\text{dist}_G(v, \mathcal{I}) \leq \text{dist}_G(v, Q) + k \leq \beta + k$。总运行时间为 $T^{\text{sparsification}}(k-1) + k\widehat{\Delta} + (k + \widehat{\Delta}^2) \cdot T_{\text{极大独立集}}(n, \Delta \cdot \widehat{\Delta})$。□

我们使用引理 5.8 的稀疏化算法和 [FGG+22] 的极大独立集算法，以多对数 n时间计算一个（$k+1$，k^2）-支配集：

定理 1.1 （k-支配集（针对 G^k）。 设 $k \geq 1$ 为一个整数（可能是 n的函数）。存在一种确定性分布式算法，可在CONGEST 模型中以多对数时间计算 G^k 的一个 k-支配集。更具体地说，轮复杂度为 $\widetilde{O}$（$k^2 \cdot$对数$^4\, n \cdot$对数 Δ）轮。

证明。 应用引理 6.3。根据引理 5.8，我们有 $T^{\text{sparsification}}(k) = \widetilde{O}(k^2 \cdot \log^4 n \cdot \log\ \Delta)$。计算极大独立集时，我们使用 [FGG+22 定理 3.1] ，该定理可在任意最大度为 Δ 的 n 节点图中，用 $O(\log^2 \Delta \cdot \log \log\ \Delta \cdot \log\ n)$ 轮次计算出一个极大独立集。这给出 $T^{\text{MIS}}(n, \Delta \log\ n) = \widetilde{O}(\log\ n \cdot \log^2 \Delta)$。引理 6.3 的运行时间变为 $\widetilde{O}(k^2 \cdot \log^4 n \cdot \log\ \Delta + (k + \log^2 n) \cdot \log\ n \cdot \log^2 \Delta) = \widetilde{O}(k^2 \cdot \log^4 n \cdot \log\ \Delta)$。□

7 图G上的极大独立集：重新审视碎裂（定理 1.4）

在本节中，我们介绍一种在 LOCAL 和 CONGEST 模型中计算 G 的极大独立集的随机化算法。与以往工作一样，我们依赖碎裂框架，整体方法与这些工作相同，主要是 [BEPS16、BEPS16、Gha16、Gha19]。事实上，我们以黑盒方式使用 [Gha16、Gha19] 的随机预碎片化阶段，从而有效地将问题化简为小型未解决分量。接着，在 [BEPS16、BEPS16、Gha16、Gha19]

而在我们的工作中，这些小连通分量在后碎裂阶段得到解决。我们介绍两种不同的后碎裂方法，它们在算法上和/或分析上与以往工作有所不同。此外，它们比以往的方法更简单。

7.1 碎裂基础

由于我们依赖相同的一般碎裂框架 [BEPS16]以及[Gha16]的预碎裂阶段，本节与这些工作中的陈述非常相似。

预碎裂。 算法的第一部分称为预碎裂阶段。它由随机过程的 $O(\log \Delta)$个步骤组成，该随机过程简称为独立集算法。预碎裂阶段计算出一个独立集 $\mathcal{I} \subseteq V$。一些节点 $B := V \setminus (\mathcal{I} \cup N(\mathcal{I}))$仍未作出决定，因为它们没有加入 $\mathcal{I}$ ，并且没有邻居加入该集合。可以证明，对于任意5-独立集 $U \subseteq V$，U 中的所有节点仍未作出决定的概率很低：

引理 7.1 ([Gha16])。 设 $H = (V, E)$ 包含至多 n 个节点，且最大度至多为 Δ。设 $c > 0$ 为任意常数。设 $U \subseteq V$ 为 H中的任意5-独立集。在 H 上运行独立集算法 $\Theta(c \log \Delta)$ 个步骤后，$U \subseteq B$ 的概率至多为 $\Delta^{-c|U|}$。

引理 7.1 中的相同结论也适用于 [BEPS16] 算法，经过 $O(\log^2 \Delta)$ 步之后亦然。引理 7.1 的一个推论是，对于任意固定的、大小至少为 t 、且为 5-独立的 $U \subseteq V$：$= \log_\Delta n$，其中所有节点都仍未确定的概率至多为 $1/n^c$。这只对固定的 U 成立，而此类 U 的选择数量极大，无法通过取并合界来获得高概率保证。因此，我们增加了额外的连通性要求。回顾一下，如果对于所有 $U' \subset U$，均有 $\text{dist}_G(U', U \setminus U) \leq s$，则集合 $U \subseteq V$ 在 G 中是 s-连通的。我们还利用这样一个事实： s-连通集合 U 的一个 (5，4)-支配集是 5-独立的，且是 $(s + 8)$-连通的。

引理 7.2 （图G的支配集 s连通分量）。 令 $H =$ （V, E）。令 $\alpha, s \geq 1$ 并令 $\beta \geq \alpha - 1$。令 $U \subseteq V$为 H 中的一个 s-连通集合。任何相对于 H 中距离的 U 的 （α, β）-支配集 $R \subseteq U$，都是 α-独立的，并且在 H 中是（$s + 2\beta$）-连通的。

证明。证明过程的示意图见图 2。首先，根据定义， R 在 H 中是 α-独立的。为了证明 R 是 ($s+2\beta$)-连通的，我们需要证明对于所有 $R' \subset R$：$\text{dist}_H(R', R \setminus R') \leq s + 2\beta$。注意，根据 R 的支配性，对于任意 $v \in U$，存在某个 $w \in R$，使得 $\text{dist}_H(v, w) \leq \beta$。现在，任取 $R' \subset R$。令 $X = N_H^\beta(R', U) = \cup_{u \in R'} N_H^\beta(u, U)$ 为 H 中节点 R' 的距离-β U 邻域。由 U 的 s-连通性可知，$\text{dist}_H(X, U \setminus X) \leq s$。令 $x \in X$ 和 $y \in U \setminus X$ 为 $\text{dist}_H(X, U \setminus X)$ 的某些极小化元。根据 X 的定义，$\text{dist}_H(R', x) \leq \beta$。对于 y，存在某个 $w \in R$，使得 $\text{dist}_H(y, w) \leq \beta$。此外，由于 $y \notin X$，我们知道 y 不可能被 R' 中的节点支配，因此 $w \in R \setminus R'$。综上可得

$$\text{dist}_H(R', R \setminus R') \leq \text{dist}_H(R', x) + \text{dist}_H(x, y) + \text{dist}_H(y, R \setminus R') \leq \beta + s + \beta \qquad \square$$

这使得我们能够证明以下的打散保证([BEPS16，定理 3.3]，[Gha16，引理 4.2])。我们使用同一陈述的一个稍微更一般的版本：

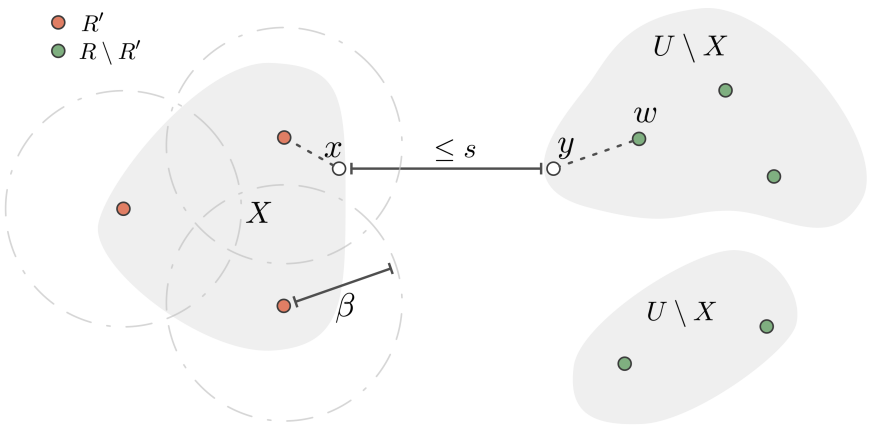

图 2: β-支配集 R（红色和绿色节点）中的 s-连通集 U（灰色）是 $(s+2\beta)$-连通的。 $X \subseteq U$ 是 $R' \subset R$ 的距离-β U邻域。从 X 到 $U \setminus X$ 的距离至多为 s。在此， $y \in U \setminus X$ 被另一个节点 $w \in R \setminus R'$ 支配。

引理 7.3 (碎裂**)。** 设 $s \geq 1$ 和 $H = (V, E)$ 是任意一个节点数至多为 n 、最大度至多为 Δ 的图。令 $t = \log_\Delta n$。在 H 上运行 $\Theta(s \log \Delta)$ 步独立集算法。令$\mathcal{I} \subseteq V$为计算得到的独立集，$B = V \setminus (\mathcal{I} \cup N(\mathcal{I}))$ 为未决节点。以 n 的高概率，

(P1) 不存在满足 $|U| \geq t$ 的 *5*-独立$(8+s)$-连通 $U \subseteq B$。

(P2) 所有 *s*-连通集 $C \subseteq B$ 至多包含 $O(t \cdot \Delta^4)$ 个节点。

证明。 (P1) 令 $s' = s + 8$。满足这些性质的任意 $U \subseteq B$ 都会在 $H^{[5,s']}$ 中形成一棵含 t 个节点的树，其中$H^{[5,s']}$是幂图，若 $v, w \in V$ 满足 $5 \leq$ 与 v, w 之间的$_H$距离不超过 $\leq s'$，则它们相邻。U 可能对应许多棵树。一般而言，含 t 个节点的无标号有根树的数量小于 4^t，因为这类树的欧拉游走可以用 $2t$ 位编码。将这类树嵌入$H^{[5,s']}$中的方式少于 $n \cdot \Delta^{s'}{}_{(t-1)}$。根据引理 7.1，对于任意常数 $c > 0$，经过 $\Theta((s+c) \log \Delta)$ 步后， $U \subseteq B$ 的概率至多为 $\Delta^{-(s+c)\cdot|U|}$。根据并合界，任意此类U 包含在未决节点中的概率至多为

$$4^t \cdot n \cdot \Delta^{s'(t-1)} \cdot \Delta^{-(s+c)\cdot t} = n^{\log_\Delta 4+1} \cdot \Delta^{s'(t-1)-(s+c)\cdot t} \leq n^{\log_\Delta 4+1} \cdot \Delta^{t(8-c)} \leq n^{10-c}$$

(P2) 假设存在一个 s-连通的集合 $C \subseteq B$，其中有 $t \cdot \Delta^4$ 个节点。我们展示如何构造违反 (P1) 的 $U \subseteq B$。贪心地构造 C 的一个 (5,4)-支配集 $U \subseteq C$，其中距离按 G 中的距离计算。首先，任取一个 $v \in C$，并令 $U := \{v\}$。然后迭代地选择满足 $\text{dist}_H(v, U) \geq 5$ 的 $v \in C \setminus U$，并设置 $U = U \cup \{v\}$。这样会将 C 中距离 v 不超过 4 的所有节点排除在考虑范围之外，这些节点至多有 Δ^4 个。因此，U 的大小至少为 $t \cdot \Delta^4 / \Delta^4 = t$。由于 C 是 s-连通的，且U 是 H 中按距离计算的 C 的一个 (5,4)-支配集，引理 7.2 表明， U 在 H 中是 5-独立且 $(s+8)$-连通的。这与 (P1) 矛盾。 □

7.2 后碎裂。

引理 7.3 指出，经过一个持续 $O(\log \Delta)$ 轮次的预碎裂阶段后，图会被打散为由未决节点构成的小型连通分量：形式化地说，引理 7.3（P2）指出，其中

$s = 1$ 指出，经过一个预碎裂阶段后，任意连通分量 C 中的节点数至多为 $O(\log_\Delta n \cdot \Delta^4)$，其中 $G[B]$，且此结论以高概率成立。由于 Δ 可能远大于多对数规模，我们更倾向于依靠性质（$P1$），以便通过算法利用连通分量规模较小这一特点。作为（P1）的推论，在任意连通分量 C中，以高概率不存在任何 $U \subseteq V$（C），使得 U 在 G 中为 5-独立，且 $|U| \geq t$。这是因为注意到，这样的 U 必定是 V（C）的某个（$5, 4$）-支配集 U^* 的子集。根据引理 7.2，作为一个连通分量的（$5, 4$）-支配集，U^* 是 5-独立且 9-连通的。因此，根据（P1），根据引理 7.3，U^* 不会仍然全部处于未决状态，且此结论以高概率成立。

如何以算法方式使用（$P1$）？ 为了直观了解如何在 MIS 算法的其余部分中使用 $P1$ 的界，假设给定一个支配集 $R_C \subseteq V$（C），其中 R_C 在 G 中是 5-独立的，并且在 C 中是 h-支配的。我们可以将每个未决节点收缩到最近的支配集节点，从而形成半径为 h 的球。这定义了一个虚拟的球图，以 R_C 为节点，并在 v、$w \in R_C$ 之间连边，当且仅当对应的球共享一条边。随后对球图计算网络分解（定义见第 2 节）。由此得到连通分量的网络分解：

断言 7.4。球图的网络分解可以转换为图G的网络分解。簇直径会按与球直径成比例的因子增大。

证明。每个簇（一组球）都以自然的方式扩展为图G中的一个节点集，即取这些球的并集。C 中位于同一个球内的任意相邻节点也属于同一个簇。如果它们位于不同的球中，那么这些球在球图中相邻。球图的网络分解会将这些球划分到同一个簇中，或划分到颜色不同的两个簇中。 □

在一个含有 t 个节点的图上，可以在 poly log t = poly log log n 时间内计算出网络分解。此外，由于通信网络中的每个节点都是一个半径为 h 的球，因此会带来 h 倍的性能下降。通过依次处理网络分解中的各个颜色，并在与簇直径成正比的时间内求解每个簇，即可计算出最终最大独立集（在 CONGEST 中，这需要更多工作）。

为什么标准方法无法 达到 （P1）给出的界? 一种方法是针对所有连通分量，同时根据图G中的距离计算一个 $(5, h)$-支配集 R。现在，C 中的节点可能会被另一个连通分量中的支配集节点支配 $C' \neq C$，也就是说，围绕支配集节点形成的球并不局限于单个连通分量，而可能包含距离为 h 的其他连通分量中的节点。若 G 中没有更强的连通性保证，引理 7.3（P1）就无法界定球图的一个连通分量中的球的数量。另一种方法是在每个诱导子图 C 中运行支配集算法，但这是错误的，因为引理 7.3 要求在 G 中满足 5-独立性，而不是在 C 中。

本节其余部分将介绍两种方法，以充分利用连通分量规模较小这一特点。

7.2.1 方法一：两个预碎裂阶段

这个版本的支配集论证与 [BEPS16 的]期刊版本中的原始证明最为相似。它在概念上很简单，而且不同于原始证明，不依赖于所使用的支配集算法的内部细节。

我们首先在 G上运行 $O(\log \Delta)$轮独立集算法。设 $\mathcal{I} \subseteq V$ 为计算得到的独立集，设 $B := V \setminus (\mathcal{I} \cup N(\mathcal{I}))$为未决节点。鉴于前面的讨论，我们再次运行 $O(\log \Delta)$轮独立集算法，这次是在 $G[B]$ 的每个连通分量 C 上并行运行。节点可以确定其在 $G[B]$中的关联边，因此可以在每个 $G[V(C)]$ 上并行运行该算法。设 $\mathcal{I}_C \subseteq V(C)$为在 C中计算得到的独立集，设 $B_C := V(C) \setminus (\mathcal{I}_C \cup N_C(\mathcal{I}_C))$为剩余的未决节点。在 C 上运行最大独立集算法，可保证彼此至少相距 5 跳的节点的执行在 C 中是相互独立的。这使我们能够将引理 7.1 应用于子图 C。我们使用 [Gha19 的引理 2.2]算法，针对 C 中的距离，在 B_C上计算一个 $(5, O(\log \log n))$-支配集R_C。该算法以高概率在 n中成功，事实上对所有连通分量都成功。

到目前为止，除了使用了不同的支配集算法外，这种方法与 [BEPS16]，相同。[8] 下一步是对每个连通分量的 $|R_C|$ 大小进行界定；在这方面，我们的解法不同于 [BEPS16]，而且明显更简洁。根据引理 7.3（P2），以高概率，在第一个预碎裂阶段之后，每个未决节点连通分量 C 至多包含 $t \cdot \Delta^4$ 个节点。我们可以利用这一点，通过并合界对可能的子集 $U \subseteq V$ （ C ）数量进行界定，从而界定 R_C 的大小：

引理 7.5。设 $G = (V, E)$ 是任意一个包含 n 个节点、最大度为 Δ 的图。设 C 是 G 的任意子图，至多包含 $t \cdot \Delta^4$ 个节点且最大度至多为 Δ。在 C 上运行 $\Theta(\log \Delta)$ 步独立集算法。设 $\mathcal{I}_C \subseteq V(C)$ 为计算得到的独立集， $B_C = V(C) \setminus (\mathcal{I}_C \cup N(\mathcal{I}_C))$ 为未决节点。以高概率，在 n 中不存在满足以下条件的 $U \subseteq B_C$： U 在 C 中是 *5*-独立的，且 $|U| \geq t$。

证明。回顾一下 $t = \log_\Delta n$。令 $U \subseteq C$ 为图G C 中的任意一个 5-独立集。根据图G C 的引理 7.1，经过 Θ（$c$log Δ）轮次后，仍然未决的概率至多为 $\Delta^{-c|U|}$，其中 c 是一个足够大的常数。我们利用这样一个事实：第一个预碎裂阶段显著减少了节点总数。这使得大小为 t 的子集的选择数减少至 $\binom{t \cdot \Delta^4}{t} \leq \left(\frac{e \cdot t \cdot \Delta^4}{t}\right)^t \leq \Delta^{5t}$。对不同的集合应用并合界，可得任意一个包含于未决节点中的 5- 独立 U （其中 $|U| = t$ ）出现的概率至多为 $\Delta^{5t} \cdot \Delta^{-c|U|} = \Delta^{5t-ct} = n^{5-c}$。 □

根据引理 7.5， R_C 的大小小于 t，且在 n 中以高概率成立。构造一个球图$\mathcal{B}$，其中每个 $v \in V$（C）都连接到距离图G C 中最近的支配集节点所对应的球。球图中，每对在图G G 中相邻的球之间都有一条边。 $\mathcal{B}$ 至多有t 个节点，且包含在小组件中。接下来，计算球图的网络分解。由此可得到 $\mathcal{C}$ 的网络分解（见断言 7.4）。根据所用的模型，剩余的最大独立集算法与 [Gha16] 或 [Gha19]，中的算法相同。

7.2.2 方法二：一个预碎裂阶段

作为上一节方法的替代方案，我们提出另一种方法，无需在每个由未决节点构成的连通分量上重新运行独立集算法。在 G 上的预碎裂阶段之后，我们根据距离计算未决节点的一个支配集

[8]由于 [BEPS16] 的总体运行时间目标较长，因此可以使用一种更简单的支配集算法，即计算一个 B_C 在 C 中距离意义下的 $(5, O(\log \Delta)$-支配集。不过，对于本证明而言，支配值并不重要。

（距离）在 G 中。我们证明，以支配集节点为中心构造的特定球具有良好的连通性，因此可以用来界定球图的一个连通分量中球的数量。

在 G 上运行 $\Theta(s\log\ \Delta)$ 轮独立集算法，参数为 $s = 8$。令 $\mathcal{I} \subseteq V$ 为计算得到的独立集，$B = V \setminus (\mathcal{I} \cup N(\mathcal{I}))$ 为未决节点集合。到目前为止，这基本上与前一种方法所做的相同。接下来，我们使用命题 7.6，根据图G G 中的距离，计算图G B 的一个 $(5, O(\log\log\ n))$-支配集 R。这使用了 [Gha19 的支配集算法，引理 2.2]。此外，该算法将图G B 划分为一组不相交的球 $\{\text{Ball}(v) \subseteq B : v \in R\}$，这些球以支配集为中心，使得每个球在图G G 中都是一个 8-连通集：

命题 *7.6*。图*G* B 的一个 $(k+1, O(k^2\ log\ log\ n))$-支配集 R 可在 $O(k^2\ log\ log\ n)$ 轮内以高概率计算得到，同时将图*G* B 划分为一组不相交的簇 $\{Ball(v) \subseteq B\ |v \in R\}$。对于每个 $v \in R$，$Ball(v)$ 都是 $2k$-连通的，且弱直径为 $O(k^2\ log\ log\ n)$。对于每个$v \in R$，都存在一棵以 $Ball(v)$ 为终端节点的斯坦纳树 T_v。图*G* E 中的每条边至多出现在 $O(k\ log\ log\ n)$ 棵树中。

证明概要。使用 [Gha19 的引理 2.2]中的算法计算支配集。支配集计算的正确性请参见原始证明。我们将说明如何使用该算法，将 B 划分为以支配集节点为中心的 $2k$-连通球。支配集算法以 $R := B$ 作为候选支配集。对于每个 $v \in R$，我们维护一个节点集 $\text{Ball}(v) \subseteq B$；对每个 $v \in R$，其初始值为 $\text{Ball}(v) = \{v\}$。$v \in R$ 的斯坦纳树初始化时仅包含 v。支配集算法分为两部分，节点逐步从 R中移除。粗粒度部分是一个随机采样过程，用于稀疏化 R，包含$O(\log\log\ n)$个阶段。在第二个细粒度部分，剩余的 R在 k轮次内变为$(k+1)$-独立。在第 i轮中，从 R中移除一些节点，使其变为$(i+1)$-独立。具体做法是首先计算 R的距离-i $O(\log^5 n)$着色。然后利用该着色，通过 [AGLP89] 的算法计算 R的$(i+1, O(i\log\log\ n))$-支配集（见定理 6.1）。每次支配集计算都包含 $O(\log\log n)$个阶段。

我们说明如何在每个阶段结束后，为每个剩余的 $u \in R$ 更新球和斯坦纳树，这对粗粒度部分和细粒度部分都适用。在这两部分中，每当节点 v 从R中被移除时，我们都能指出一个唯一的剩余节点 $u \in R$，该节点将 v 踢出，也就是说，该节点导致了 v被移除。发生这种情况时，会将 v 的球并入 u 的球。在粗粒度部分和细粒度部分中，v 到将其踢出的节点的距离分别至多为 $2k$ 和 k。此外，对于 G 中位于 v 与 u 之间最短路径上的任意 v'，可以保证：若 v' 也被踢出，则将它踢出的也是同一个节点 u。为了更新 u的斯坦纳树，会将树 T_v以及 v 与 u 之间的一条最短路径添加到 T_u中。

对两部分的各个阶段作简单归纳可知，每个剩余节点的球都是 $2k$-连通的。每当一个节点被移除，就会将它到某个剩余节点的最短路径添加到该剩余节点的斯坦纳树中。这条路径的长度至多为 $2k$。总共经过 $O(\log\log\ n + k\log\log\ n)$ 个阶段后，斯坦纳树直径将达到 $O(k^2\log\log\ n)$。在任意阶段，对于任意边 $e \in E$，至多有一个 $u \in R$ 会踢出其他节点，且 e位于 u 与被移除节点之间的最短路径上。因此，在所有阶段中，e 至多会被添加到 $O(k\log\log\ n)$ 棵树中。□

一般而言，命题 7.6 表明，使用 [Gha19 引理 2.2] 计算出的 $(k+1, O(k^2\log\log\ n))$-支配集周围形成的特定球在 G中是 $2k$-连通的。类似地，可以证明，使用 [AGLP89]的算法计算出的 $(k+1, k\log\ n)$-支配集周围形成的特定球是 k-连通的 [BEPS16 期刊版本，定理 2.4]。

命题 7.6 给出了 B 到球 Ball(v) $\subseteq B, v \in R$的一个划分。每个球在 G中都是 8-连通的。对于每个 $v \in R$，给出一棵以 Ball(v) 为终端节点的斯坦纳树 T_v。我们构造一个球图 $\mathcal{B}$，其节点为 R。在球图中，如果对应的球在 G中相邻，即 $\exists v' \in$ Ball(v) $\exists w' \in$ Ball(w) : $(v', w') \in E$，则 $v, w \in R$之间有一条边。极大独立集算法的其余部分在球图的每个连通分量上并行执行。设 $\mathcal{C}$ 为 $\mathcal{B}$的一个连通分量。设 $R_{\mathcal{C}}$ 为对应的支配集节点，并设 $B_{\mathcal{C}} := \cup_{v \in R_{\mathcal{C}}}$ Ball(v) 为包含在 $R_{\mathcal{C}}$各球中的所有未决节点组成的集合。

Claim 7.7. *$B_{\mathcal{C}}$ 在 G中是 8-连通的。*

证明。 $B_{\mathcal{C}} := \cup_{v \in R_{\mathcal{C}}}$Ball($v$)，其中每个 Ball($v$) 都是 8-连通的。由于 $\mathcal{C}$ 是球图的一个连通分量，对于任意 $v \in R_{\mathcal{C}}$，都存在 $w \in R_{\mathcal{C}}$，$v \neq w$ 使得 dist_G(Ball(v)，Ball(w)) $= 1$（除非 $\mathcal{C}$ 只包含一个节点）。因此，对于任意 $S \subset B_{\mathcal{C}}$，$\text{dist}_G(S, B_{\mathcal{C}} \setminus S) \leq \max(1, 8) = 8$。

□

我们证明 $\mathcal{C}$ 中的球的数量至多为 $t = \log_\Delta n$。根据引理 7.3（P1），我们通过证明 $R_{\mathcal{C}}$ 是某个 5-独立、16-连通集 $R'_{\mathcal{C}} \supseteq R_{\mathcal{C}}$的子集，来界定 $|V(\mathcal{C})| = |R_{\mathcal{C}}|$。集合 $R_{\mathcal{C}}$ 是 5-独立的，但其中的节点之间仍可能相距很远。为便于分析，构造另一个集合 $R'_{\mathcal{C}} \supseteq R_{\mathcal{C}}$，使得 $R'_{\mathcal{C}}$ 是 $B_{\mathcal{C}}$的一个（5，4）-支配集，其中的距离以G中的距离为准。根据引理 7.2，$R'_{\mathcal{C}}$ 在 G中是 16-连通的，这里使用了 $B_{\mathcal{C}}$ 是 G中的一个 8-连通集这一事实。回顾一下，我们在 G上运行了 $\Theta(s\log \Delta)$ 轮独立集算法，并满足 $s = 8$。因此，根据引理 7.3（P1），以高概率，不存在满足 $|U| \geq t$的 5-独立、16-连通集 $U \subseteq B$。如果不存在这样的集合，则 $R_{\mathcal{C}} \subseteq R'_{\mathcal{C}}$ 的大小至多为 t，从而证明了该结论。我们还可以界定连通分量 $\mathcal{C}$中的未决节点总数，这在 CONGEST 中很有用。集合$B_{\mathcal{C}}$ 是 8-连通的。引理 7.3（P2）指出，B 的任意 8-连通子集的大小至多为$O(t \cdot \Delta^4)$。

在球图的每个连通分量 $\mathcal{C}$ 中继续进行，我们计算 $\mathcal{C}$的网络分解。$\mathcal{C}$ 的网络分解可以给出底层节点集 $B_{\mathcal{C}}$ 的网络分解（见断言 7.4）。根据球图的定义，球图中任意两个连通分量之间的距离至少为 2（在 G中）。因此，可以对球图的每个连通分量独立地解决问题的其余部分。最终最大独立集通过遍历网络分解的各个颜色来计算，并在与簇直径成正比的时间内解决每个簇（在 CONGEST 中，这需要更多工作）。

7.3 {v1}中的方法 法：[BEPS16，arXiv]（并且推测也包括 [Gha16，Gha19]）

本节讨论我们的方法和证明与使用碎裂框架解决对称性破缺问题的算法之间的关系。我们此前已经讨论过，[BEPS16，期刊] 中的方法为碎裂框架所需的性质提供了正确的算法，其证明虽然非常复杂，但确实是正确的。然而，许多后续工作建立在 [BEPS16，arXiv]中早期但存在缺陷的方法之上。虽然我们的第一种方法为其中棘手的部分提供了不同的证明（见第 7.2.1 节），但我们认为，强调这些技术细节对学界是有益的，因为碎裂技术已经发展成为开发随机化算法的一项必备技术。

与我们的第一种方法类似，他们采用两个阶段的预碎裂[9]：第一阶段生成小连通分量，第二阶段则在所有小连通分量上独立执行

[9]当然，我们的工作受到了他们工作的启发，而不是反过来。

并行地将每个小连通分量划分为（许多）微小连通分量。为了说明他们工作中的问题，我们先约定一些记号。考虑一个小连通分量 C，令 B_C 表示C 中在第二个预碎裂阶段结束后仍未决的节点，即微小连通分量中的节点。为了利用这些连通分量规模小这一特点，他们计算[10] 一个针对某个非恒定 h的 (5，h)-支配集R_C。为了界定 R_C的大小，他们希望对图 C 使用引理 7.3 (P1)，其中 $s = 1$，并考虑第二个预碎裂阶段产生的随机性。尽管 (P1) 确实保证以高概率不存在满足 $|U| \geq t$ 的 5-独立、9-连通集 $U \subseteq B_C$，但所构造的 (5，h)-支配集 R_C 未必是 9-连通的。因此，作者采用一种贪心过程，不断将未决节点加入支配集，直至其变为 9-连通（同时始终确保其保持 5-独立）。事实上，该贪心过程应返回 B_C的某个 (5，4)-支配集$R'_C \supseteq R_C$，其中距离按 C中的距离计算。注意，该集合仅为分析之用。然而，这种构造并不总能使集合成为 9-连通集。如果微小连通分量彼此相距很远（在 C中），就无法再向集合中加入未决节点（ B_C中的节点），以得到连通性足够强的支配集。因此，无法应用引理 7.3。事实上， [BEPS16 的 arxiv v3] 工作中的问题出现在第 19 页的步骤 3 和 4，即执行这一贪心构造之处。在这里，他们也将 C 中已经确定（不属于微小连通分量）的节点加入支配集，使其成为 9-连通集。但这样一来，第二个预碎裂阶段的概率分析就无法界定 R'_C 是否存在，因为它只界定了在第二个预碎裂阶段中，5-独立节点集中的所有 节点都保持未决状态的概率。此外，也不能依赖第一个预碎裂阶段的概率分析，因为那样就需要在 R' 中测量节点之间的距离，而不是在小连通分量中测量。 G

贪心地扩展这个支配集实际上是所有已知证明中第一阶段预碎裂阶段不可或缺的一部分，尤其对于证明（$P2$）而言更是如此。这些证明的关键区别在于，证明者可以通过贪心方式创建支配集，使其变得 9-连通。在这种情况下，由于可以专注于未决节点的单个连通分量，这一过程就变得十分直接。

BEPS16 [期刊]版本的解决方案。 回想一下，在算法中利用连通分量规模较小这一性质时，难以聚焦于单个连通分量。使用某种算法以黑盒方式计算支配集，并通过将节点分配给最近的支配集节点来围绕每个支配节点构造球时，某个节点可能会被分配给一个支配节点，而该支配节点实际上位于另一个小连通分量中，而非该节点所属的连通分量。因此，BEPS 的期刊版本采用了不同的方法。它同样采用两阶段预粉碎方法，但利用支配集算法的内部机制将节点分配给支配集节点。这确保了每个球都保持 4-连通。不过，仍不能排除某个节点被分配给另一个连通分量中的支配集节点。因此，他们的解决方案将分析限制在所构造球图的一个连通分量上，而不是聚焦于未决节点的一个连通分量。如果按上述方式构造这些球，这种设置满足应用 (P1) 的要求。我们的第二种方法表明，在这种方法中，第二个预粉碎阶段并非必要。因此，我们的第二种方法是简化该框架的另一种方式（除第一种方法之外）。

[10]事实上，这三项工作使用了不同的支配集算法。 [Gha19] 的一个贡献实际上是提出了一种（5，$O(\log\log n)$-支配集算法，该算法适用于 CONGEST 模型，并且可以用于这种情况。

8 幂图上的随机化对称性破缺（定理 1.2，推论 1.3）

本节给出用于计算图G的极大独立集和支配集的随机化算法，也就是说，我们证明定理 1.2（见第8.2节）和推论 1.3（见第8.3节）。在第8.1节中，我们先简要说明为什么 Luby 算法可以扩展到图 G（这并不适用于该算法的所有版本）。

8.1 幂图上的 Luby 算法

Luby 算法 [Lub86, ABI86] 可扩展至在 O 的 k ·log n 轮内计算出 G^k 的一个极大独立集，且成功概率很高。所有节点最初均处于 未决 状态；一旦自身或其某个邻居（在 G^k 中）加入独立集，它们便会变为 已决 状态。在 G 中，算法的每一步包含两轮。每个未决节点 v 都会从 $[n^c]$ [MRSZ 11], 中选取一个随机数 x_v，其中 c 是一个足够大的常数。然后将该数值与未决邻居的数值进行比较。如果 v 的数值最小， v 就加入独立集并通知其邻居。算法能在 $O(\log n)$ 轮内计算出 G 的一个极大独立集，且成功概率很高（简单分析见 [MRSZ11] ）。在 G^k 上模拟该算法会产生 k 倍的减速，这是因为需要传递随机数的最小值，并在节点加入极大独立集时通知距离为 k 的邻居。重要的是，该算法不要求节点知道自己的度。

由于节点不知道自己在 G^k中的度，因此尚不清楚如何修改该算法的版本，使得 G^k 中的节点以取决于其度的概率标记自己。

G^k 的极大独立集（定理1.2）

与定理 1.4 的算法类似，我们针对 G^k 的极大独立集算法也基于碎裂框架。我们的预碎裂阶段和后碎裂阶段，都是对*BeepingMIS*算法的修改。该算法最初由 [Gha17 第 2 节2]提出。它可以看作是 [Gha16]预碎裂阶段的一种节省通信的变体。

通过蜂鸣求极大独立集（属于 G） [Gha17，第 2 节。2]: 所有节点最初均为未决状态；一旦它们自身或某个邻居加入独立集，它们就会变为已决状态。算法分步执行（最初称为轮次），每步包含两个通信轮次。在第一步中，未决节点 $v \in V$ 以某个概率 x_v标记自己。被标记的节点通过单比特蜂鸣通知其邻居。节点根据是否存在至少一个被标记的邻居来更新其标记概率 x_v。在第二步中，没有被标记邻居的已标记节点加入独立集 $\mathcal{I}$ 并通知其邻居。加入独立集的节点 $\mathcal{I}$ 及其邻居会被停用，并停止参与算法。

BeepingMIS 可用于在 O 中以高概率于 G 个 O(log n) 轮次内计算一个极大独立集。经过 O 个 O(log Δ) 轮次后，我们得到以下来自 [BEPS16 定理 3.3] 和 [Gha16 引理 4.2] 的 打散保证 。

引理 8.1 (碎裂)。 设 $s \geq 1$ 和 $G = (V, E)$ 是任意一个节点数至多为 n 、最大度至多为 Δ的图。令 $t = \log_\Delta n$。在图G G上运行 ***BeepingMIS*** 的 $\Theta(s\log\ \Delta)$ 步。令$\mathcal{I} \subseteq V$为计算得到的独立集， $B = V \setminus (\mathcal{I} \cup N(\mathcal{I}))$ 为未决节点。以 n 的高概率，

(P1) 不存在 $8+s$)-连通的 *5*-独立$(8+s)$集合 $U \subseteq B$，使得 $|U| \geq t$。

(P2) 所有 *s*-连通集 $C \subseteq B$ 至多包含 $O(t \cdot \Delta^4)$ 个节点。

证明。对于每个节点 v，经过 Θ（log $d(v)$ $+\log 1/\epsilon$）后，v 变为已决的概率至少为 $1-\epsilon$ [Gha17，定理 2.1]。此外，仔细追踪概率——当限制在单个步骤内时，相距超过 2 跳的节点的决策彼此独立——可以得到：对于任意 5-独立集 $U \subseteq V$，经过 Θ（$c$log Δ）步后，U 中的所有节点都仍未决的概率至多为 $\Delta^{-c|U|}$。这给出了引理 7.1 的一个版本。有了这个结果，剩余证明与引理 7.3 的证明相同。 □

计算 G^k 的极大独立集的概览。我们在 G^k 上运行 O（log Δ^k）步 BeepingMIS（其中 Δ^k 是 Δ（G^k）的一个上界）。之后，G^k 被打散为由未决节点组成的小连通分量。

后碎裂阶段的总体思路与用于 G 的算法类似。我们首先通过为连通分量的幂图计算一个合适的支配集来构造球图。然后计算该球图的网络分解，从而得到一个针对小连通分量的网络分解，该分解具有较少的颜色类和较小的簇半径。主要难点在于处理单个簇（独立于所有其他簇）。为了完成对一个簇的处理，我们并行运行 O(log n) 个 BeepingMIS 算法实例（用于 G^k）。我们证明，以高概率（在 n 中），其中一个实例会成功，并且簇中的所有节点都能在基本与簇直径成正比的时间内，就某个获胜实例达成一致。簇中的节点采用该实例的输出，然后继续执行。

我们将继续介绍这一进程的细节，并从模拟在 G^k 上运行 BeepingMIS 所需的工具开始。随后，我们将在 G^k设置下介绍几种处理球图的工具。最后，本节末尾将给出使用所有这些基本工具的定理 1.2 的证明。

在 G^k上模拟 BeepingMIS。为此，蜂鸣信号必须附带标识符，以免发出蜂鸣信号的节点将自己的蜂鸣信号与其他节点的蜂鸣信号混淆。请注意，运行算法以求解 G 时不会出现这一难题。如下所示（引理 8.2），BeepingMIS 可以在 G^k 上模拟，减速因子为 k。由于我们需要在后碎裂阶段并行运行多个 BeepingMIS 实例，因此无法承担使用 O(log n) 位来附加标识符的开销。我们可以利用距离为 k 的着色，来减少单个实例所用的带宽：

引理 8.2 (在 G^k上模拟蜂鸣)。设 $k \geq 1$ 和 $S \subseteq V$是任意节点集。若每个节点都有一个 a 位标识符，且该标识符在距离 k 内唯一，则每个 $v \in V$都可以在 $\widetilde{O}(k \cdot \lceil a/$带宽$\rceil)$ 轮内，使用带宽位消息获知是否存在某个 $w \in N^k(v,S)$（其中 $w \neq v$）。

证明。每个 $x \in S$ 通过发送一个元组来发出蜂鸣信号 (ID(x)，k)，其中第二个元素是距离剩余计数器，初始值为 k，每次转发该元组时都会递减。经过 k 轮后，每个$v \in V$ 都会向每个邻居 $w \in N$(v) 转发至多两个标识符各不相同的任意传入元组子集，并附上其中最大的剩余距离。经过 k 轮后，任何满足 $v \in V\, N^k$(v, S) 的 $\neq \emptyset$，无论 $v \in S$ 是否成立，都会至少收到来自某个 $w \in N^k$(v, S) 的一个蜂鸣信号。每轮都可以用 $\lceil(a + \log k)/$ 带宽轮次实现。总时间复杂度为 $\sim$ O($k \cdot \lceil a/$ 带宽$\rceil$)。 □

距离-k 球图。设 $B \subseteq V$ 是一个（未决）节点集合，$R \subseteq B$ 是 B 的某个支配集。假设给定未决节点的一个划分 $\{\text{Ball}(v) \subseteq B : v \in R\}$。回顾一下，$\{\text{Ball}(v) \subseteq B : v \in R\}$ 的球图 是一个虚拟图，其节点为 R。若对应的球 Ball(v) 和 Ball(w) 在 G 中相邻，则 v，$w \in R$ 之间存在一条边。

在后碎裂阶段，会为剩余的未决节点计算一个网络分解。在 G 中，为 $\{\text{Ball}(v) \subseteq B : v \in R\}$ 的球图计算网络分解，会导出 G 的网络分解（见断言 7.4 ）。这对图幂并不能直接奏效。我们改为计算 $\{\text{Ball}(v) \subseteq B : v \in R\}$ 的距离-k 球图的网络分解。节点集为 R 的图 $\mathcal{B}$ 是一个距离-k 球图，当且仅当对任意 v，$w \in R$，若 $\text{dist}_G(\text{Ball}(v), \text{Ball}(w)) \leq k$，则 $\text{dist}_{\mathcal{B}}(v, w) \leq k$。接下来，我们展示如何在通信网络 G中构造距离-k 球图：

引理 8.3 (构造距离-k 球图)。设 $R \subseteq B \subseteq V$，并假设给定了 B 的一个划分$\{\text{Ball}(v) \subseteq B : v \in R\}$。假设对于每个 $v \in R$，都有一棵弱直径为 $O(D)$ 的斯坦纳树 T_v，其终端节点为 $\text{Ball}(v)$，且 E 中的任意一条边至多属于 τ 棵树。存在一个 $O(k)$ 轮的确定性 CONGEST 算法，可构造 $\{\text{Ball}^+(v) \subseteq V : v \in R\}$，使得

- 球$^+$(v) $\subseteq V$和球$^+$(v) $\supseteq$ 球(v) 对于所有 $v \in R$，
- 球$^+$(v) $\cap$ 球$^+$(w) $= \emptyset$ 对于所有 $v \neq w \in R$，
- 球图 $\mathcal{B}$ 对于 $\{$球$^+$（v）：$v \in R\}$ 是一个距离-k 球图，对于 $\{$球（v）：$v \in R\}$。
- 对于每个 $v \in R$，存在一棵斯坦纳树 T_v^+，其弱直径为 $O(D+k)$，并且以 球$^+$（v）作为终端节点he。E 中的任意边至多属于 $\tau + 1$ 棵树。

证明。对于每个 $v \in R$，令边界(v) $\subseteq N^k$(v)为球(v)周围的节点集。边界仅由 $V \setminus B$中的节点组成，并且边界(v) $\cap$边界(w) $= \emptyset$ 对于所有 $v \neq w \in R$均成立。见图 3。最终，我们设球$^+$(v): $=$ 球(v) $\cup$ 边界(v)。

初始时，对于所有 $v \in R$，Border(v): $= \emptyset$。对于每个 $v \in R$，并行地从 Ball(v) 中的所有节点开始，在 G 中执行最多 k 跳的广度优先搜索。ID(v) 用作从 Ball(v) 发起的搜索的标识符。令 $z \in V \setminus B$。当 z 首次接收到一个或多个搜索时，它接受标识符最小的搜索（假设该搜索为 v），并加入 Border(v)。之后，z 将接受的搜索转发给其其他邻居，总共最多转发 k 跳。B 中的节点不会加入边界，也不会转发搜索。

令 Ball$^+$(v)：对于所有 $v \in R$，$=$ Ball(v) $\cup$ Border(v)。通过将广度优先搜索令牌的路径添加到 T_v，形成一棵斯坦纳树 T_v^+。每条边至多被添加到一棵树中。令 $\mathcal{B}$ 为 { 的球图，其中 Ball$^+$(v): $v \in R$}。为证明正确性，考虑任意 v、$w \in R$，其中 $\text{dist}_G(\text{Ball}(v), \text{Ball}(w)) \leq k$（见图 3）。我们证明 $\text{dist}_{\mathcal{B}^+}(v, w) \leq k$，即 v 和 w 在 $\mathcal{B}^k$ 中相邻。令 $(v', x_1, \ldots, x_s, w')$ 为 G 中 Ball(v) 与 Ball(w) 之间的一条最短路径，其中 $s \leq k-1$、$v' \in$ Ball(v)，且 $w' \in$ Ball(w)。从 Ball(v) 和 Ball(w) 出发的广度优先搜索沿两个方向各传播 k 跳，直到到达另一个球或边界。因此，每个 x_i、$1 \leq i \leq s$ 都属于某个 Ball$^+$(y_i)，其中 $y_i \in R$（可能为 $y_i = v$ 或 $y_i = w$）。根据路径上的节点数，互不相同的 Ball$^+$(y_i) 的总数至多为 $k-1$。这些节点在球图 $\mathcal{B}^+$ 中形成一条从 v 到 w 的路径，长度至多为 k。 □

令 $\mathcal{B}$ 为距离-k 球图，针对 $\{\text{Ball}(v) \subseteq R : v \in R\}$，并使用引理 8.3 构造。我们计算 $\mathcal{B}$的网络分解，其分离度为 $k+1$，由此得到 G^k 中未决节点的网络分解：

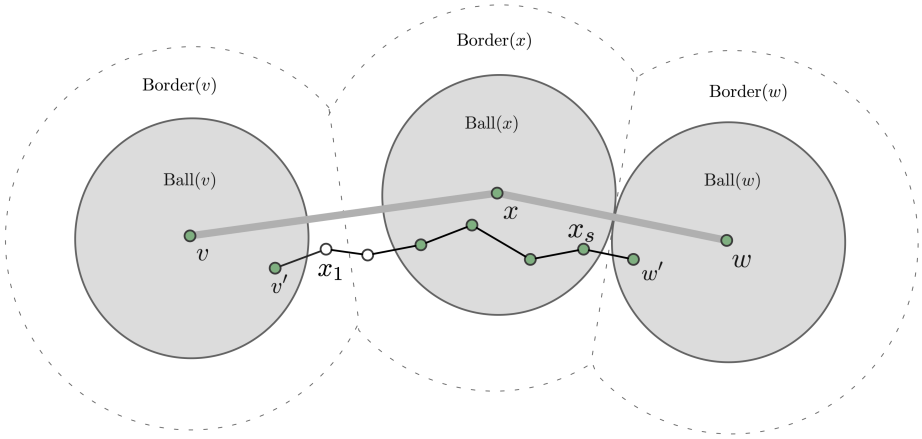

图 3：距离-k球图 $\mathcal{B}$，包含三个顶点 v，x，w以及底层图的部分结构G。 $\mathcal{B}$ 的边以较粗的线条表示。节点 v 和 x 在 $\mathcal{B}$ 中相邻，因为虽然 Ball(v) 和 Ball(x) 在 G中没有共用一条边，但它们的边界有共用边。

引理 8.4. 可以将 $\mathcal{B}^k$ 的网络分解转换为 G^k的网络分解，适用于 B中的节点。簇直径会增加一个与 k 及球的直径成正比的因子。

证明。 G^k 的网络分解按自然方式构造：对于每个 $v \in R$，Ball(v) 中的节点都加入 v 的簇。

G 中的簇直径增加了一个 $O(k+D)$ 倍，其中 D 是 {Ball(v) $\subseteq B : v \in R$}中球的弱直径，因为 $\mathcal{B}$ 中的节点（如引理 8.3 中所构造）实际上是直径为 $O(k+D)$ 的球。我们证明这些簇在 G^k 中恰当地彼此分离。反设有两个不同的同色簇在 G^k中相邻。设v，w是来自这两个簇的一些节点，且 $\text{dist}_G(v, w) \leq k$。这些节点分别位于两个不同的球 Ball($v'$) 和 Ball($w'$) 中（否则 v 和 w 会加入同一个簇）。根据 $\mathcal{B}$的定义，我们有 $\text{dist}_{\mathcal{B}}(v', w') \leq k$。由于我们计算了 $\mathcal{B}^k$的网络分解，v' 和 w' 必定属于 $\mathcal{B}$中的同一个簇。这与 v 和 w 属于 G中的不同簇相矛盾。

□

我们现在可以证明本节的主要结果。为了计算 G^k的极大独立集，我们使用碎裂框架，并以 BeepingMIS 作为底层算法。建议读者参阅第7节，其中对 G中的碎裂进行了更详细的描述。我们采用单阶段预碎裂方法（第 7.2.2 节）。在后碎裂阶段，会并行运行多个 BeepingMIS 实例。为了在 G^k中高效地做到这一点，我们为剩余未作决定的节点分配较短的标识符。结合引理 8.2中的模拟工具，这使运行时间达到与 G 的极大独立集相同的水平（运行时间相差 k 倍因子）。

定理 1.2。 存在一种随机分布式算法，能够在 G^k 的 $\widetilde{O}$（k^2 对数 Δ·对数对数 n+k^4 对数^5 对数 n）轮内，在CONGEST 模型中以高概率计算出极大独立集。

运行时间也可以表示为 $\widetilde{O}(k \log \Delta(G^k) \cdot \log\log n + k^4 \log^5 \log n)$。

定理 *1.2* 的证明：**预碎裂阶段**。我们模拟 [Gha17]on 的 *BeepingMIS* 算法，其通信网络为 G。运行该算法 $O(s\log \Delta^k)$ 步，其中 $s = 8$，且 Δ^k 是 $\Delta(G^k)$ 的上界。根据引理 8.2，每一步都可以在 G^k 上模拟，代价是因子为 k 的

减速，使用原始的 $O(\log n)$ 位标识符和完整的 $\Theta(\log\ n)$ 位通信带宽。预碎裂阶段的总运行时间为 $O(k^2 \log \Delta)$。BeepingMIS 算法计算出 G^k 的一个独立集 $\mathcal{I} \subseteq V$，它是最终结果的一部分。

后碎裂：支配集。令 $B = V \setminus (\mathcal{I} \cup N^k(\mathcal{I}))$ 表示在图 G^k上执行 BeepingMIS 的 $\Theta(\log \Delta^k)$ 轮后仍未作出决定的节点。我们计算图 B的一个支配集 R，使得 R 中的节点在预碎裂阶段的执行彼此独立。为此，图 G^k中 R 内的节点之间至少要相隔 5 跳，或者等价地，在图 G 中的距离至少为 $4k+1$（为方便起见，我们采用更大的独立性间距 $5k+1$）。使用 Gha19 的支配集算法，计算相对于图 G中的距离，图 B的一个 $(5k+1, O(k^2 \log\log\ n))$-支配集 R，参见引理 2.2]。这需要 $O(k^2 \log\log n)$ 轮，并以高概率成功。在并行运行支配集算法的同时，我们使用命题 7.6 将图 B 划分为集合 $\{\text{Ball}(v) \subseteq B :$ 以支配集节点为中心的不相交球。对于每个 $v \in R$，Ball(v) 是图 G中的一个 $10k$-连通集合。这意味着每个 Ball(v) 在图 G^k中都是 10-连通的。对于每个 $v \in R$，图 G中都存在一棵斯坦纳树 T_v，其终端节点为 Ball(v)，直径为 $O(k^2 \log\log\ n)$，且任意一条边最多属于 $O(k \log\log\ n)$ 棵斯坦纳树。

球图。我们使用引理 8.3，在 $O(k)$ 轮内为 $\{\text{Ball}(v) \subseteq B :\ v \in R\}$ 构造距离-k 球图 $\mathcal{B}$。$\mathcal{B}$ 中的节点集为 R。每个节点 $v \in R$ 对应一个集合 $\text{Ball}^+(v) \subseteq V$，其中 $\text{Ball}^+(v)$ 是 Ball(v) 的超集，并且对于任意 $v \neq w \in R$，$\text{Ball}^+(v)$ 与 $\text{Ball}^+(w)$ 不相交。对于每个 $v \in R$，令 T_v 为 $\text{Ball}^+(v)$ 的斯坦纳树，它由 Ball(v) 的原始斯坦纳树和引理 8.3 中添加的部分组合而成。G 中的任意一条边至多出现在 $O(k \log\log\ n)$ 棵树中（引理 8.3 不会增加拥塞）。$\mathcal{B}$ 定义为 $\{\text{Ball}^+(v) :\ v \in R\}$ 的球图，也就是说，如果 $\text{Ball}^+(v)$ 与 $\text{Ball}^+(w)$ 相邻，则 $v, w \in R$在 $\mathcal{B}$ 中相邻。根据引理 8.3，对于任意 $v \neq w \in R$，若 $\text{dist}_G(\text{Ball}(v), \text{Ball}(w)) \leq k$，则 $\text{dist}_{\mathcal{B}}(v, w) \leq k$。因此，球图中任意两个不同的连通分量在 G中的距离至少为 $k+1$。MIS 算法的其余部分在球图的每个连通分量上独立执行。

固定 $\mathcal{B}$ 的一个连通分量 $\mathcal{C}$。我们对 $\mathcal{C}$ 中球的数量，以及包含在 $\mathcal{C}$ 的球中的未决节点总数进行界定。令 $\mathcal{C}(\subseteq R)$ $R_{\mathcal{C}} = V$ 表示 $\mathcal{C}$ 中的节点，并令 $= \cup_{v \in R_{\mathcal{C}}}$:vBall($R_{\mathcal{C}}$) 表示包含在 $B_{\mathcal{C}}$ 中节点的球里的未决节点集合。集合 $10k$ 在 G 中是 $10k$-连通的，因为每个球都是 $2k$ -连通的，且相邻球在 G 中的距离至多为 $B_{\mathcal{C}}$ 跳（参见 Claim 7.7）。等价地，G^k 在 $|R_{\mathcal{C}}|$中是 10-连通的。首先界定节点 G^k 的数量。对于 $s = 10$，引理 8.1 指出，以高概率，在 G^k 中不存在满足 $|U| \geq t = \log_\Delta n$ 的 5-独立、18-连通集合 $U \subseteq R$（其中 Δ 是 $\Delta(G^k)$ 的下界）。这一结论不能直接使用，因为集合 $R_{\mathcal{C}}$ 在 G^k 中是 5-独立的，但未必是 18-连通的。为便于分析，我们构造另一个集合 $R'_{\mathcal{C}} \supseteq R_{\mathcal{C}}$，其中 $R'_{\mathcal{C}} \subseteq B_{\mathcal{C}}$。$R'_{\mathcal{C}}$ 是根据 G^k 中的距离，从 $B_{\mathcal{C}}$ 中任意贪心选取的一个 (5, 4)-支配集。根据引理 7.2，$R'_{\mathcal{C}}$ 在 G^k 中是 5-独立且 18-连通的，这里我们利用了 $B_{\mathcal{C}}$ 在 G^k 中是 10-连通的这一事实。鉴于以高概率，G^k 中任意 5-独立且 18-连通集合的大小都小于 t，可得 $|R_{\mathcal{C}}| < t$。对于第二个界，回顾一下，$B_{\mathcal{C}}$ 是 G^k 中的一个 10-连通集合。鉴于引理 8.1 (P1) 成立，(P2)（同样针对 G^k 和 $s = 10$）指出，$|B_{\mathcal{C}}|$ 至多为 $t \cdot \Delta^{4k}$，其中 Δ^k 是 $\Delta(G^k)$ 的上界。

网络分解。我们计算图 $\mathcal{C}$的网络分解。我们使用定理 A.1~，确定性地计算图 $\mathcal{C}$ 的网络分解，其颜色数为 $O(\log\log\ n)$，弱直径为$O(k \cdot \log\log\ n)$，同色簇之间的分离度为 $2k+1$。网络分解算法可以在球图上模拟，这一点已在 Claim A.4 中得到验证。由于每个节点都由一棵弱直径为 $O(k^2 \log\log\ n)$、拥塞为 $O(k \log\log\ n)$的斯坦纳树模拟，该模拟的运行时间会增加 $O(k^3 \log^2 \log n)$倍。因此，运行时间

为 $\widetilde{O}(k^4 \log^5 \log n)$。根据引理 8.4，我们可以为 G 提取一个适用于 $B_{\mathcal{C}}$的网络分解： $B_{\mathcal{C}}$ 中的每个节点都加入其所属球对应的簇。从现在起，网络分解中的簇均视为图 $B_{\mathcal{C}}$中的节点集合。在 G中，簇的实际直径为 $O(k^3 \log^2 \log n)$。每个簇都对应一棵斯坦纳树，最初位于球图 $\mathcal{C}$中。将球图中的每个节点 $v \in R_{\mathcal{C}}$替换为相应球 T_v的斯坦纳树，即可得到图 G中该簇的一棵斯坦纳树。定理 A.1 生成的斯坦纳树之间没有拥塞，而图 E 中的每条边至多属于 $O(k \log \log n)$棵球的斯坦纳树。因此，为簇构造的斯坦纳树中的拥塞至多为 $O(k \log \log n)$。

我们分别处理网络分解中的每种颜色。固定一种颜色 j，设 S 是颜色为 j 的一个簇。回顾一下， $N := t \cdot \Delta^{4k}$ 是 $|B_{\mathcal{C}}|$的一个上界。我们为每个节点 $v \in S$ 分配一个来自 $[N]$ 的标识符，使得簇内节点的标识符各不相同；这些标识符使用 $O(\log N) = O(\log \log n + k \log \Delta)$ 位。标识符通过簇的斯坦纳树分配，耗时 $O(k^3 \log^2 \log n)$，具体做法是根据子树所包含的 S 中节点数量，为每棵子树分配一个标识符范围。

最终最大独立集。现在，我们可以让 S 中节点的独立集成为极大独立集。我们运行 BeepingMIS $O(\log N) = O(k \log \Delta + \log \log n)$ 步，并行运行 $O(\log_N n)$ 次算法执行。每次执行分配 $\Theta(\log N)$ 位通信带宽，对应于新标识符的大小。因此，并行执行所用的总带宽为 $O(\log N \cdot \log_N n) = O(\log n)$，可以装入一条消息。由于 BeepingMIS 在 G^k 中的一步需要在 G 中进行 $O(k)$ 个通信轮次，因此并行执行 $O(\log N)$ 步，总共需要 $O(k \log N) = O(k^2 \log \Delta + k \log \log n)$ 轮。相同颜色簇的执行可以彼此独立地进行，因为消息从未决节点出发最多传播 k 跳，而簇之间至少相隔 $2k+1$ 跳。每次执行以至少 $1 - 1/N^c$ 的概率输出簇内 G^k 的一个极大独立集，其中 c 是某个常数。至少有一次执行成功的概率不低于 $1 - 1/N^{c \cdot \log_N n} = 1 - 1/n^c$。为找出成功的执行，我们在簇的斯坦纳树中汇总每次执行是否成功（若执行不成功，则其结果不是极大的，即存在未决节点），每次执行用一个比特指示符，耗时 $O(k^4 \log^3 \log n)$ 轮。簇首领选出一次成功的执行，并通知簇内其余节点。选出的独立集被加入输出。无论 $B_{\mathcal{C}}$ 中的节点是否位于颜色为 j 的簇内，只要它们距该独立集不超过 k 跳，也会变为已决节点。每个颜色类的运行时间为 $O(k^2 \log \Delta + k^4 \log^3 \log n)$。遍历所有 $\widetilde{O}(\log \log n)$ 种颜色后，运行时间为 $\widetilde{O}(k^2 \log \Delta \cdot \log \log n + k^4 \log^4 \log n)$。包括预粉碎、计算支配集和网络分解在内的总运行时间为 $\widetilde{O}(k^2 \log \Delta \cdot \log \log n + k^4 \log^5 \log n)$。

□

推论 8.5. 定理 *1.2* 的最大独立集算法在使用子集 $Q \subseteq V$ 时，能够以高概率计算出 $G^k[Q]$ 的一个最大独立集。运行时间为 $\widetilde{O}(k \log \Delta_Q \cdot \log \log n + k^4 \log^5 \log n)$，其中$\Delta_Q = \Delta(G^k[Q])$。

证明。 只有 Q 中的节点可以加入独立集。其他节点也会参与算法，但在算法开始时会被设为已决 状态。与其他已决节点一样，它们仍会参与在剩余未决节点之间转发消息，因为它们可能属于 $G^k[Q]$ 的边。预碎裂阶段和后碎裂阶段都可以在这种设置下执行。

□

8.3 图G G^k 中的支配集（推论 1.3）

本节证明定理 1.2 的以下推论。

推论 1.3。 存在一种随机分布式算法，可在 β-支配集上计算 G^k 中$\widetilde{O}(\beta \cdot k^{1+1/(\beta-1)}(\log \Delta)^{1/(\beta-1)} + \beta \cdot k \log\log n + k^4 \log^5 \log n)$ 轮次。

我们可以将极大独立集 G^k 算法与 [KP12]和 [BKP14] 的稀疏化过程结合起来，计算幂图的支配集。我们先简要介绍 G中 [KP12]的算法。设 $H = (V_H, E_H)$为任意一个最大度为 Δ_H的图，并设 $f \geq 2$为一个参数。 [KP12] 的算法会在 O轮（$\log_f \Delta_H$）内采样一个节点集 $Q \subseteq V_H$，使得：(1) 以高概率， $H[Q]$ 中的最大度为 $O(f \cdot \log n)$；(2) Q 是 V_H的支配集。所有通信都通过蜂鸣信号完成：采样节点发出蜂鸣信号，通知其邻居自己已被采样。这可以在 G^k 中用 k 轮模拟，并使用引理 8.2（事实上，无须包含标识符，因为发出蜂鸣信号的节点不需要监听其他蜂鸣信号）。该算法也不要求节点知道自己的度（参见 [BKP14]的算法 Sparsify-GG）。因此，我们可以在通信网络为 G的 G^k 中模拟该算法，代价是增加 $O(k)$倍的时间。

给定 $\beta \geq 2$，通过迭代 [KP12]$\beta - 1$ 次算法，并结合在最终已采样子图 [上计算极大独立集，可以计算出一个 $(2, \beta)$-支配集 BKP14, Gha19]。图经过迭代稀疏化，产生一系列子集 $V \supseteq Q_1 \supseteq \cdots \supseteq Q_{\beta-1}$，其中 Q_s 是第 s 次迭代的结果，通过在 $G[Q_{s-1}]$ 上运行 [KP12] 算法计算得到。设 $f_1 > \cdots > f_{\beta-1}$ 为每次迭代中使用的参数。对于每个 $1 \leq s \leq \beta - 1$，高概率地有：(1) $G[Q_s]$ 的最大度 Δ_s 为 $O(f_i \cdot \log n)$；(2) Q_s 是 G 的一个 s-支配集。第 s 次迭代需要 $O(\log_{f_s} \Delta_{s-1}) = O(\log f_{s-1} + \log\log n \log f_s)$ 轮。通过令 $f_s = 2^{(\log \Delta)^{1-s/(\beta-1)}}$，可以平衡各次迭代的运行时间， $\beta-1$ 次迭代总共需要 $O((\beta-1) \cdot \log^{1/(\beta-1)} \Delta)$ 轮。 $G[Q_{\beta-1}]$ 的最大度为 $O(f_{\beta-1} \cdot \log n) = O(\log n)$。现在，可以使用极大独立集算法计算 $G[Q_{\beta-1}]$ 的极大独立集，从而得到 G 的一个 $(2, \beta)$-支配集。

推论 *1.3* 的证明。 我们使用[KP12]的算法，对 G^k 进行 $\beta - 1$ 次迭代的稀疏化。对于 $1 \leq s \leq \beta - 1$，令 $Q_s \subseteq V$为第 s次迭代的结果。初始时， $Q_0 := V$。在第 s次迭代中，我们设置 $f_s = 2^{(\log \Delta^k)^{1-s/(\beta-1)}}$。我们在 $G^k[Q_{s-1}]$上模拟 [KP12] 的算法，并令 Q_s 为本次迭代中已采样的节点。根据 [KP12] 针对 G^k的分析，以高概率有：(1) $G^k[Q_s]$ 的最大度为 $O(f_s \cdot \log n)$；(2) Q_s 是 G^k中 Q_{s-1} 的一个支配集。这意味着 Q_s 是 G中 Q_{s-1} 的一个 k-支配集。 [KP12] 的算法运行于

$$O\left(\log_{f_s} \Delta_{s-1}\right) = O\left(\frac{\log f_{s-1}}{\log f_s} + \log\log n\right) = O\left((\log \Delta^k)^{1/(\beta-1)} + \log\log n\right)$$

G^k 中的模拟运行速度减慢了 k倍。

最后一次迭代的结果是一个集合 $Q_{\beta-1}$，使得 $G^k[Q_{\beta-1}]$ 中的最大度为$O(f_{\beta-1} \cdot \log n) = O(2^{(\log \Delta^k)^0} \cdot \log n) = O(\log n)$。我们使用定理 1.2 的算法计算 $G^k[Q_{\beta-1}]$ 的一个极大独立集 $\mathcal{I}$（见推论 8.5）。鉴于 $\Delta\big(G^k[Q_{\beta-1}]\big) = O(\log n)$，计算该极大独立集需要 $\widetilde{O}(k^4 \cdot \log^5 \log n)$ 轮次。其结果在 G 中是 $(k+1)$-独立的。支配性为 $(\beta - 1) \cdot k + k = \beta \cdot k$。因此，该结果是 G 的一个 $(k+1, \beta \cdot k)$-支配集。总运行时间为

$$\begin{aligned}
&\widetilde{O}\left((\beta-1) \cdot k \cdot \left((\log \Delta^k)^{1/(\beta-1)} + \log\log n\right) + k^4 \cdot \log^5 \log n\right) \\
&= \widetilde{O}\left(\beta \cdot k^{1+1/(\beta-1)} \cdot (\log \Delta)^{1/(\beta-1)} + \beta \cdot k \cdot \log\log n + k^4 \cdot \log^5 \log n\right)
\end{aligned}$$ □

参考文献

[AA20] Noga Alon 和 Sepehr Assadi。《超越 (Δ+1) 顶点着色的调色板稀疏化》。载于《近似、随机化与组合优化：算法与技术，*APPROX/RANDOM 2020*》，第6:1–6:22页，2020年。[ABI86] Noga Alon、L´aszl´o Babai 和 Alon Itai。《用于极大独立集问题的快速简易随机并行算法》。《算法学报》，7(4):567–583，1986年。[ACG+21] Kook Jin Ahn、Graham Cormode、Sudipto Guha、Andrew McGregor 和 Anthony Wirth。《数据流中的相关聚类》。《*Algorithmica*》，83(7):1980–2017，2021年。[ACK19] Sepehr Assadi、Yu Chen 和 Sanjeev Khanna。《(Δ + 1) 顶点着色的次线性算法》。载于 Timothy M. Chan 编，《第*30*届*ACM-SIAM*离散算法研讨会论文集，*SODA 2019*，美国加利福尼亚州圣迭戈，*2019*年*1*月*6*日至*9*日》，第767–786页。SIAM，2019年。[AGLP89] Baruch Awerbuch、Andrew V. Goldberg、Michael Luby 和 Serge A. Plotkin。《分布式计算中的网络分解与局部性》。载于《计算机科学基础研讨会（*FOCS*）论文集》，第364–369页，1989年。[AGM12] Kook Jin Ahn、Sudipto Guha 和 Andrew McGregor。《图摘要：稀疏化、生成器和子图》。载于《第*31*届*SIGMOD-SIGACT-SIGAI*数据库系统原理研讨会（*PODS*）论文集》，第5—–14页，2012年。[BBF+22] Alkida Balliu、Sebastian Brandt、Manuela Fischer、Rustam Latypov、Yannic Maus、Dennis Olivetti 和 Jara Uitto。《在具有最优内存的MPC中超越局部性的指数级加速》。载于《分布式计算国际研讨会（*DISC*）论文集》，第9:1–9:21页，2022年。[BBKO22] Alkida Balliu、Sebastian Brandt、Fabian Kuhn 和 Dennis Olivetti。《分布式 Δ-着色玩捉迷藏》。载于 Stefano Leonardi 和 Anupam Gupta 编，《*STOC '22*：第*54*届*ACM SIGACT*计算理论年会，意大利罗马，*2022*年*6*月*20*日至*24*日》，第464–477页。ACM，2022年。[BBO22] Alkida Balliu、Sebastian Brandt 和 Dennis Olivetti。《支配集的分布式下界》。《*SIAM*计算杂志》，51(1):70–115，2022年。[BCM+20] Reuven Bar-Yehuda、Keren Censor-Hillel、Yannic Maus、Shreyas Pai 和 Sriram V. Pemmaraju。《幂图上的分布式近似》。载于《第*39*届分布式计算原理研讨会论文集》，PODC '20，第501–510页，美国纽约州纽约，2020年。计算机协会。[BEPS16] Leonid Barenboim、Michael Elkin、Seth Pettie 和 Johannes Schneider。《分布式对称性破缺的局部性》。《*ACM*学报》，63(3)，2016年6月。http://arxiv.org/abs/1202.1983。

[BKM20] Philipp Bamberger、Fabian Kuhn 和 Yannic Maus。小带宽下高效的确定性分布式着色。在 Yuval Emek 和 Christian Cachin（编）主编的 PODC '20：ACM分布式计算原理研讨会论文集，线上活动，意大利，2020年8月3日至7日，第243–252页。ACM，2020年。[BKP14] Tushar Bisht、Kishore Kothapalli 和 Sriram V. Pemmaraju。简短公告：超快速 t-支配集。在 Magn´us M. Halld´orsson 和 Shlomi Dolev （编）主编的 ACM 分布式计算原理研讨会，PODC '14，法国巴黎，2014年7月15日至18日，第379–381页。ACM，2014年。[Bra19] Sebastian Brandt。分布式问题的自动加速定理。在 Peter Robinson 和 Faith Ellen（编）主编的《2019年 ACM 分布式计算原理研讨会论文集》（PODC），第379–388页。ACM，2019年。[CDP20] Artur Czumaj、Peter Davies 和 Merav Parter。利用图稀疏化在低空间下对大规模并行计算进行去随机化。在《第32届 ACM 算法与体系结构并行性研讨会论文集》，SPAA '20，第175–185页，美国纽约州纽约，2020年。计算机协会。[CDP21] Artur Czumaj、Peter Davies 和 Merav Parter。低空间 MPC 中改进的确定性（$\Delta+1$）着色。在 Avery Miller、Keren Censor-Hillel 和 Janne H. Korhonen（编）主编的 PODC '21：ACM分布式计算原理研讨会，线上活动，意大利，2021年7月26日至30日，第469–479页。ACM，2021年。[CFG+19] Yi-Jun Chang、Manuela Fischer、Mohsen Ghaffari、Jara Uitto 和 Yufan Zheng。拥塞团模型、大规模并行计算和集中式局部计算中的（$\Delta + 1$）-着色复杂度。在 ACM分布式计算原理研讨会（PODC）论文集，第471–480页，2019年。[CLP20] Yi-Jun Chang、Wenzheng Li 和 Seth Pettie。通过超快速图碎裂实现分布式（$\Delta+1$）-着色。《SIAM 计算杂志》，49(3)：497–539，2020年。[CPS17a] Keren Censor-Hillel、Merav Parter 和 Gregory Schwartzman。带宽限制下局部分布式算法的去随机化。在第31届国际分布式计算研讨会（DISC）上，2017年。[CPS17b] Kai-Min Chung、Seth Pettie 和 Hsin-Hao Su。lov´asz 局部引理和图着色的分布式算法。《分布式计算》，30(4)：261–280，2017年。[CPS20] Keren Censor-Hillel、Merav Parter 和 Gregory Schwartzman。带宽限制下局部分布式算法的去随机化。《分布式计算》，33(3-4)：349–366，2020年。[DFKL21] Michal Dory、Orr Fischer、Seri Khoury 和 Dean Leitersdorf。拥塞团模型和 MPC 中的常数轮生成器与最短路径。在《2021年 ACM 分布式计算原理研讨会论文集》（PODC）中，第223–233页，2021年。[DKM19] Janosch Deurer、Fabian Kuhn 和 Yannic Maus。CONGEST 模型中确定性分布式支配集近似。在 Peter Robinson 和 Faith Ellen（编）主编的《2019年 ACM 分布式计

算原理研讨会论文集》，PODC 2019，加拿大安大略省多伦多，2019年7月29日—8月2日，第94–103页。ACM，2019年。[DN20] Michael Dinitz 和 Yasamin Nazari。大规模并行近似距离草图。在第23届分布式系统原理国际会议（OPODIS 2019）上，第35:1–35:17页，2020年。[EM19] Michael Elkin 和 Shaked Matar。低多项式确定性 CONGEST 时间中的近加性伸展子。在 Peter Robinson 和 Faith Ellen（编）主编的《2019年 ACM 分布式计算原理研讨会论文集》，PODC 2019，加拿大安大略省多伦多，2019年7月29日—8月2日，第531–540页。ACM，2019年。[FG17] Manuela Fischer 和 Mohsen Ghaffari。用于 lov´asz 局部引理的次对数分布式算法及复杂度层次。在 Andr´ea W. Richa（编）主编的第31届国际分布式计算研讨会，DISC 2017，2017年10月16日至20日，奥地利维也纳，第91卷 LIPIcs，第18:1–18:16页。Schloss Dagstuhl - Leibniz-Zentrum f¨ur Informatik，2017年。[FGdV22] Sebastian Forster、Martin Gr¨osbacher 和 Tijn de Vos。用于生成器和低直径分解的改进随机移位算法。在第25届分布式系统原理国际会议（OPODIS 2021）上，第16:1–16:17页，2022年。[FGG+22] Salwa Faour、Mohsen Ghaffari、Christoph Grunau、Fabian Kuhn 和 V´aclav Rozho˘n。局部分布式舍入：推广至极大独立集、匹配、集合覆盖及其他问题，2022年。[FHN20] Pierre Fraigniaud、Magn´us M. Halld´orsson 和 Alexandre Nolin。距离-k 着色的分布式测试。在 Andrea Werneck Richa 和 Christian Scheideler（编）主编的《结构信息与通信复杂度——第27届国际学术讨论会》，SIROCCO 2020，德国帕德博恩，2020年6月29日至7月1日，论文集，计算机科学讲义第12156卷，第275–290页。Springer，2020年。[FMH22] Manuela Fischer、Yannic Maus 和 Magn´us M. Halld´orsson。快速分布式 Brooks 定理。CoRR，abs/2211.07606，2022年。[GGH+22] Mohsen Ghaffari、Christoph Grunau、Bernhard Haeupler、Saeed Ilchi 和 V´aclav Rozho˘n。通过去随机化改进网络分解、击中集和生成器，2022年。[GGK+18] Mohsen Ghaffari、Themis Gouleakis、Christian Konrad、Slobodan Mitrovic 和 Ronitt Rubinfeld。用于极大独立集、匹配和顶点覆盖的改进大规模并行计算算法。在 ACM分布式计算原理研讨会（PODC）论文集，第129–138页，2018年。[GGR21] Mohsen Ghaffari、Christoph Grunau 和 V´aclav Rozhon。改进的确定性网络分解。在 D´aniel Marx（编）主编的《2021年 ACM-SIAM 离散算法研讨会论文集》，SODA 2021，线上会议，2021年1月10日至13日，第2904–2923页。SIAM，2021年。

[Gha16] Mohsen Ghaffari。改进的极大独立集分布式算法。载于 Robert Krauthgamer（编），第*27*届年度 *ACM-SIAM* 离散算法研讨会论文集（*SODA 2016*），美国弗吉尼亚州阿灵顿，*2016*年*1*月*10*日至*12*日，第270–277页。SIAM，2016年。[Gha17] Mohsen Ghaffari。通过全对全通信实现分布式极大独立集。载于*ACM*分布式计算原理研讨会论文集（*PODC*），第141–149页，2017年。[Gha19] Mohsen Ghaffari。使用小消息计算分布式极大独立集。载于 Timothy M. Chan（编），第*30*届年度 *ACM-SIAM* 离散算法研讨会论文集（*SODA 2019*），美国加利福尼亚州圣迭戈，*2019*年*1*月*6*日至*9*日，第805–820页。SIAM，2019年。[Gha22] Mohsen Ghaffari。极大独立集的局部计算。载于第*63*届 *IEEE*计算机科学基础年会（*FOCS 2022*），美国科罗拉多州丹佛，*2022*年*10*月*31*日至*11*月*3*日，第438–449页。IEEE，2022年。[GHK18] Mohsen Ghaffari、David G. Harris 和 Fabian Kuhn。关于局部分布式算法的去随机化。载于 Mikkel Thorup（编），第*59*届 *IEEE*计算机科学基础年会（*FOCS 2018*），法国巴黎，*2018*年*10*月*7*日至*9*日，第662–673页。IEEE计算机学会，2018年。[GHKM21] Mohsen Ghaffari、Juho Hirvonen、Fabian Kuhn 和 Yannic Maus。改进的分布式 Δ着色。《分布式计算》，34(4):239–258，2021年。[GK21] Mohsen Ghaffari 和 Fabian Kuhn。确定性分布式顶点着色：更简单、更快速，且无需网络分解。载于计算机科学基础研讨会论文集（*FOCS*），第1009–1020页，2021年。[GKM17] Mohsen Ghaffari、Fabian Kuhn 和 Yannic Maus。局部分布式图问题的复杂度。载于 Hamed Hatami、Pierre McKenzie 和 Valerie King（编），第*49*届 *ACM SIGACT* 计算理论年会论文集（*STOC 2017*），加拿大魁北克省蒙特利尔，*2017*年*6*月*19*日至*23*日，第784–797页。ACM，2017年。[GU19] Mohsen Ghaffari 和 Jara Uitto。稀疏化分布式算法及其在大规模并行计算和集中式局部计算中的影响。载于*ACM-SIAM*离散算法研讨会论文集（*SODA*），2019年。[GV07] Beat Gfeller 和 Elias Vicari。增长有界图上极大独立集问题的随机分布式算法。载于第*26*届 *ACM*分布式计算原理研讨会论文集，PODC '07，美国纽约州纽约，2007年，第53–60页。计算机协会。[HKM20] Magn´us M. Halld´orsson、Fabian Kuhn 和 Yannic Maus。CONGEST 模型中的距离-2着色。载于 Yuval Emek 和 Christian Cachin（编），*PODC '20*：*ACM*分布式计算原理研讨会，线上活动，意大利，*2020*年*8*月*3*日至*7*日，第233–242页。ACM，2020年。

[HKMN20] Magn´us M. Halld´orsson、Fabian Kuhn、Yannic Maus 和 Alexandre Nolin。无需了解邻居的颜色即可快速着色。载于 Hagit Attiya（编），第34届国际分布式计算研讨会（DISC 2020），2020年10月12日至16日，线上会议，LIPIcs 第179卷，第39:1–39:17页。Schloss Dagstuhl - Leibniz-Zentrum f¨ur Informatik，2020年。[HKN21] Monika Henzinger、Sebastian Krinninger 和 Danupon Nanongkai。用于近似单源最短路径的确定性近乎紧界分布式算法。SIAM计算杂志，50(3)，2021年。[HN21] Magn´us M. Halld´orsson 和 Alexandre Nolin。通过高效颜色抽样，在 CONGEST 中实现超快速着色。载于 Tomasz Jurdzinski 和 Stefan Schmid（编），《结构信息与通信复杂性——第28届国际学术研讨会》（SIROCCO 2021），波兰弗罗茨瓦夫，2021年6月28日至7月1日，论文集，计算机科学讲义第12810卷，第68–83页。Springer，2021年。[HPP+15] James W. Hegeman、Gopal Pandurangan、Sriram V. Pemmaraju、Vivek B. Sardeshmukh 和 Michele Scquizzato。迈向拥塞团模型中的最优界：图连通性与最小生成树。载于 ACM分布式计算原理研讨会论文集（PODC），第91–100页，2015年。[JN18] Tomasz Jurdzi´nski 和 Krzysztof Nowicki。拥塞团模型中的 $O(1)$轮最小生成树。载于 ACM-SIAM 离散算法研讨会论文集（SODA），第2620–2632页，2018年。[KMR01] Sven Oliver Krumke、Madhav V. Marathe 和 S. S. Ravi。无线电网络中的信道分配模型与近似算法。《无线网络》，7(6):575–584，2001年。[KMW18] Fabian Kuhn、Yannic Maus 和 Simon Weidner。线图的确定性分布式支配集。载于 Zvi Lotker 和 Boaz Patt-Shamir（编），《结构信息与通信复杂性——第25届国际学术研讨会》（SIROCCO 2018），以色列马阿莱哈哈米沙，2018年6月18日至21日，修订精选论文，计算机科学讲义第11085卷，第193–208页。Springer，2018年。[KP12] Kishore Kothapalli 和 Sriram V. Pemmaraju。超快速3统治集。载于 Deepak D'Souza、Telikepalli Kavitha 和 Jaikumar Radhakrishnan（编），IARCS软件技术基础与理论计算机科学年会（FSTTCS 2012），印度海得拉巴，2012年12月15日至17日，LIPIcs 第18卷，第136–147页。Schloss Dagstuhl - Leibniz-Zentrum f¨ur Informatik，2012年。[KSV13] Amos Korman、Jean-S´ebastien Sereni 和 Laurent Viennot。迈向更局部化的局部算法：消除有关全局知识的假设。《分布式计算》，26(5-6):289–308，2013年。[Lin92] Nati Linial。分布式图算法中的局部性。SIAM计算杂志，21(1):193–201，1992年。[LS93] Nathan Linial 和 Michael Saks。低直径图分解。Combinatorica，13:441–454，1993年12月。

[Lub86] Michael Luby。极大独立集问题的一个简单并行算法。*SIAM*计算杂志，15:1036–1053，1986年11月。[LW10] 克里斯托夫·伦岑和罗杰·瓦滕霍费尔。简短公告：利用非局部通信加速局部算法。在第*29*届*ACM SIGACT-SIGOPS*分布式计算原理研讨会（*PODC*）论文集中，第295–296页，2010年。[MRSZ11] Yves M´etivier、约翰·罗布森、纳赛尔·萨赫布-贾赫罗米和阿卡·泽马里。最优位复杂度的随机化分布式极大独立集算法。《分布式计算》，23:331–340，2011年1月。[MU21] Yannic Maus和Jara Uitto。用于 Lov´ász 局部引理的高效 CONGEST 算法。在塞斯·吉尔伯特（编）所编的第*35*届分布式计算国际研讨会（*DISC 2021*），*2021*年*10*月*4*日至*8*日，德国弗赖堡（线上会议）中，LIPIcs第209卷，第31:1–31:19页。Schloss Dagstuhl - Leibniz信息学中心，2021年。[Now21] Krzysztof Nowicki。拥塞团模型中常数轮次解决最小生成树问题的确定性算法。在*ACM*计算理论研讨会论文集（*STOC*）中，第1154—-1165页，2021年。[Pel00] 戴维·佩莱格。分布式计算：一种局部性敏感方法。SIAM，2000年。[RG20] V´aclav Rozho˘n和Mohsen Ghaffari。多对数时间确定性网络分解与分布式去随机化。在第*52*届*ACM SIGACT*计算理论年会论文集中，STOC 2020，第350–363页，纽约州纽约，美国，2020年。计算机协会。[SEW13] Johannes Schneider、Michael Elkin和罗杰·瓦滕霍费尔。依赖于色数或邻域增长的对称性破缺。理论计算机科学，509，2013年10月。[SSS95] 珍妮特·P·施密特、艾伦·西格尔和阿拉温德·斯里尼瓦桑。有限独立性应用的切尔诺夫–霍夫丁界。*SIAM*离散数学杂志，8(2):223–250，1995年。[Vad12] 萨利尔·P·瓦丹。伪随机性。《理论计算机科学基础与趋势》。Now Publishers，2012年。

幂图的网络分解

我们的算法将幂图的网络分解作为子程序。更正式地说，对于某个参数 k，我们要求同一颜色类中的簇彼此至少相距 $k+1$ 跳。这些相应的分解可以通过 [MU21]的算法计算得出。然而，由于近期关于网络分解的结果 [GGH$^+$22] 同样适用于 CONGEST，且速度显著快于 [MU21], 的结果，并能提供更好的保证，我们将简述如何调整他们的算法

以提供所需的分解。[11] 关于网络分解的定义，请参见定义 2.1。

定理 *A.1*（ G^k的网络分解）。令 $k \geq 1$ （可能是 n的函数）。存在一种确定性 *CONGEST* 算法，可在 $\widetilde{O}(k \cdot \log^3 n)$ 轮次内，在 G^k 中计算出一种网络分解，其具有 $O(\log n \cdot \log\log n)$ 种颜色，且弱直径为 $O(k \cdot \log n)$。对于每种颜色 c和每个簇 $C \subseteq V$，存在一棵斯坦纳树 T_C，其在 G中的半径为 $O(k \cdot \log n)$，并以 C作为终端节点。 G 的每条边至多属于颜色为 c的簇所对应的一棵树。

引理 A.2 （低度聚类。改编自 [GGH$^+$22，定理 4。1]）。 令 $s \geq 2$ （可能是 n的函数）。令 $A \subseteq V$ 为存活节点的集合。存在一种确定性 CONGEST 算法，可在 $\widetilde{O}(s \cdot \log^2 n)$ 轮次内，计算出一个聚类 $\mathcal{C}$，其中 $C \subseteq A$ 对于每个簇$C \in \mathcal{C}$，且

1. 弱直径为 $O(s \cdot \log n)$，在 G 中。对于每个簇 $C \in \mathcal{C}$，都存在一棵半径为 $O(s \cdot \log n)$ 的斯坦纳树，该树以 C 作为终端节点。 G 的每条边至多出现在一棵树中。

2. s-跳度 [12] 在 $\mathcal{C}$ 中至多为 $\lceil 100 \log\log n \rceil$，该度是相对于斯坦纳树定义的；*3.* 已聚类节点的数量至少为 $|A|/2$。

证明概要。 我们首先概述 [GGH$^+$22 的算法，定理 5.1]。他们的算法由两个主要部分组成：基于给定延迟的聚类，以及计算合适的延迟函数。延迟函数为每个 $v \in V$ 分配一个正整数延迟 del(v)，范围从 1 到 $O(s\log n)$。令 $\kappa = \lceil 100 \log\log n \rceil$ 为一个常数（在 [GGH$^+$22] 中替代 k，注意不要与定理 A.1 中 G 的幂所对应的参数混淆）。延迟函数与分离参数 s 一起确定一种聚类，该聚类通过以下方式计算：每个节点等待由延迟指定的若干步后，启动一次广度优先搜索。每个 $v \in V$ 每步转发至多 κ 个令牌。 v 被划入最先到达的广度优先搜索令牌所对应节点的簇（若出现并列，则选择标识符最小的节点）。第一个令牌到达后， v 会继续将后续到达的令牌转发给其他节点，持续 $2s$ 步，或直到总共转发了 κ 个令牌为止。统计从第一个令牌到达起的 $2s$ 步间隔内收到的令牌总数。在此间隔内令牌到达 v 的节点集记为 frontier^{2s}(v)。每个满足 frontier^{2s}(v) $\leq \kappa$ 的 $v \in V$，都会加入令牌最先到达的节点所在的簇。总的来说， $\widetilde{}$ 的广度优先搜索过程需要 $O(s \cdot \log n)$ 轮。簇直径受最大延迟 $O(s\log n)$ 限制，因为在此之后，任何节点都会自行发起令牌。该算法的第二个主要部分是计算合适的延迟函数。延迟函数的计算方式使得一个常数比例的节点满足 frontier^{2s}(v) $\leq \kappa$，也就是说，一个常数比例的节点会被聚类。延迟函数从指数分布中选取，通过逐步对两两独立抛硬币过程进行去随机化得到。我们对这两个主要部分作如下修改。

基于给定延迟的聚类：只有存活节点会加入簇。失效 节点 $V \setminus A$ 不会发起或加入簇。失效节点会正常转发收到的广度优先搜索令牌。这样做是必要的，因为距离需要在输入图中测量，而不只是 $G[A]$。生成的簇具有弱直径而非强直径，因为相应的强直径簇内部可能存在失效节点

[11] [GGH$^+$22] 的方法本质上直接给出了所需的结果。此外，我们还与 [GGH$^+$22]的一位作者确认了这一事实。[12]参见 [GGH$^+$22] ，了解聚类的 s跳度定义

。簇的广度优先搜索令牌所经过的路径构成斯坦纳树，其半径为 $O(s\log\ n)$。请注意，只有存活节点到最近簇首领的距离有界（距离至多为 $O(s\log\ n)$，因为至多经过 $O(s\log\ n)$步后，任何存活节点都会自行发起一个簇）。为确保算法对失效节点也能终止，我们在 $5sR + 2s = O(s\log\ n)$步后停止执行；这是为所有存活节点计算簇和前沿所需的最大步数。

计算延迟函数：最初，对延迟函数的选择进行去随机化，以保证至少有 $|V|/2$ 个节点被聚类。对于我们的应用，势函数是对存活节点求和计算，而不是对图中的所有节点求和。这保证了被聚类节点的数量至少为 $|A|/2$。□

引理 A.3 (从低度到隔离。改编自 [GGH+22，定理 5。1])。令 $s \geq 2$（可能是 n 的函数）。给定一个存活节点集合 $A \subseteq V$，以及引理 *A.2* 中的聚类 $\mathcal{C}$，存在一个确定性的 CONGEST 算法，可以在 $\widetilde{O}(s \cdot \log^2 n)$ 轮次内计算出一个聚类 $\mathcal{C}^{out}$，其中对于每个簇 $C \in \mathcal{C}^{out}$，$C \subseteq A$，且

1. 弱直径为 $O(s \cdot \log n)$。对于每个簇 $C \in \mathcal{C}^{out}$，存在一棵斯坦纳树 T_C，其半径为$O(s \cdot \log n)$，位于 G 中，且 C 为终端节点。G 的每条边至多属于一棵树。
2. 分离度 s
3. 已聚类节点的数量 A 至少为 $\frac{|A|}{1000^{|\cup\mathcal{C}|}\log\log\ n}$

证明。参见 [GGH+22 定理 5.1] 的证明。与原始表述唯一的区别是簇具有弱直径，这并不重要，因为不存在拥塞。注意，$\mathcal{C}^{\text{out}}$ 中的簇仅由 A中的节点组成，因为每个 $C \in \mathcal{C}^{\text{out}}$ 都是某个 $C' \in \mathcal{C}$的子集，其中$C \subseteq A$ 由引理 A.2 的定义给出。

□

定理 *A.1* 的证明。固定 $G = (V,\ E)$。令 $A := V$为存活节点的集合。该结果在 $O(\log\ n \cdot \log\log n)$ 次迭代内得到。每次迭代通过结合引理 A.2 和引理 A.3 的聚类过程，形成网络分解的一个颜色类。

固定某次迭代 i。首先使用引理 A.2 和 $s = k+1$，对 G 中的存活节点进行低度聚类 $\mathcal{C}$。运行时间为 $\widetilde{O}(k \cdot \log^2 n)$ 轮，且已聚类节点的数量至少为 $|A|/2$。以计算得到的聚类作为输入，我们应用引理 A.3，在 $\widetilde{O}(k \cdot \log^2 n)$ 轮内计算一个 $(k+1)$-分离的聚类 $\mathcal{C}^{\text{out}}$，其中至少聚类了 $|A|/2000$ $\log\log\ n$ 个节点。我们将 $\mathcal{C}^{out}$ 中的簇染为第 i 种颜色。令 $A = A \setminus (\cup\mathcal{C}^{\text{out}})$ 为剩余的存活节点。

每次迭代至少将剩余节点中的 $\log\log\ n$ 比例划分为簇，因此 $O(\log\ n \cdot \log\log\ n)$ 次迭代就足够了。总运行时间为 $\widetilde{O}(k \cdot \log^3 n)$。□

对于第 8.2 节中的极大独立集算法，我们还证明了 [GGH+22] 的网络分解算法（因此定理 A.1 也如此）可以在球图上模拟。该模拟专用于 [GGH+22]的算法，而对于一般的 CONGEST 算法则无法高效实现。球图中节点发给其邻居的所有消息都沿着该球的斯坦纳树发送。通信必须由广播和汇聚广播等简单基本工具构成，才能实现高效模拟。

Claim A.4（球图的网络分解）。令 $R \subseteq V$，并令 $\{ball(v) \subseteq V : v \in R\}$ 为一个由互不相交的球组成的集合。令 $\mathcal{B}$ 为 $\{ball(v) \subseteq V : v \in R\}$的球图，其节点为 R；若 *Ball(v)* 和 *Ball(w)* 在 G 中相邻，则 v 与 $w \in R$ 之间有一条边。假设对于每个 $v \in R$，都存在一棵弱直径为 $O(r)$ 的斯坦纳树 T_v，其中 *Ball(v)* 为终端节点，并且 E 中的任意边至多属于 $O(\tau)$ 棵树。定理 *A.1* 可在 $\mathcal{B}$ 上模拟，通信网络为 G，减速因子为 $O(r \cdot \tau)$。

证明。 定理 A.1 的证明由反复应用引理 A.2 和引理 A.3 构成。这些引理分别基于 [GGH$^+$22、定理 4.1 和定理 5.1], 。尤其是，通信方面保持不变。 [GGH$^+$22 、定理 4.1 和定理 5.1] 中的通信分别基于 [GGH$^+$22、引理 4.7 和引理 5.2], 中的通信原语。我们证明，利用各球的斯坦纳树，可以在球图上实现这些通信原语，减速因子为 $O(r \cdot \tau)$。

[GGH$^+$22，引理 4.7]: 球图中的每个节点 $v \in R$ 都会在给定延迟后启动广度优先搜索。广度优先搜索令牌包含 ID(v)，以及 v 的延迟是否仍在递减（即 $v \in V$ 是否处于活动状态 $_i$）。令 $\kappa := \lceil 100 \log \log n \rceil$ 为常数。在整个广度优先搜索过程中，球图中的每个节点 $v \in R$ 至多向其在 $\mathcal{B}$ 中的邻居转发 κ 个令牌。广度优先搜索的每一步中，转发的令牌子集至多包含 κ 个令牌，并优先处理节点 $x \notin V_i^{\text{active}}$。该过程的一步可在球图中用 $O(\kappa \cdot r \cdot \tau)$ 个 CONGEST轮次实现，即减速因子为 $O(r \cdot \tau)$。从 $v \in R$ 发送的一组令牌沿着 T_v 流水线传输至 Ball(v) 中的节点，这可在 $O(\kappa \cdot r \cdot \tau)$ 轮次内完成。每个 $w \in$ Ball(v) 在 $O(\kappa)$ 轮次内将这组令牌发送给其邻居 $w' \in N(w) \setminus$ Ball(v)。每个 $w \in$Ball(v) 都会接收传入的令牌。至多 τ 个令牌组成的子集会沿着 T_v 转发至根节点 v，并优先处理处于活动状态 $x \notin V$ 的节点 $_i$；这同样可在 $O(\kappa \cdot r \cdot \tau)$ 轮次内完成。

令牌所经过的路径可用于实现所需的通信原语。对于每个 $v \in R$，进程会在球图中定义一棵树。根据 [GGH$^+$22] 中对广度优先搜索过程的原始分析，由令牌路径构成的树包含球图中所有必要的节点（即 $M_{i-1}(v)$）。通过为树中的每个球在斯坦纳树内选择一条路径，可以将球图中的树扩展为 G。由于任意一条边最多属于 $O(\tau)$ 棵球的斯坦纳树，且球图中的每个节点 $v \in R$总共最多转发 κ 个令牌，因此任意一条边最多属于 $O(\kappa \cdot \tau)$ 棵扩展树。利用扩展树，每个 $v \in R$ 都可以向 $M_{i-1}(v)$ 中的节点发送一条 $O(\log n)$ 位消息。按照广度优先搜索过程，该过程的一步需要 $O(\kappa \cdot \tau \cdot r)$ 轮。反向操作时，我们可以将消息从$M_{i-1}(v)$ 聚合到 $v \in R$，每一步需要 $O(\kappa \cdot \tau \cdot r)$ 轮。因此，减速因子为 $O(\tau \cdot r)$。

[GGH$^+$22，引理 5.2]: 该引理同样基于一个广度优先搜索过程，由两个阶段组成。在第一阶段，球图中每个簇（ R的子集）的中心都启动一次广度优先搜索，并在球图中最多传播 s 跳（s 是 [GGH$^+$22 定理 5.1] 中的分离参数）。每个节点 $v \in R$ 每步最多向其在 $\mathcal{B}$ 中的邻居转发 κ 个令牌。在第二阶段，属于某个簇的球沿簇的一棵树从根节点向叶节点传播收到的令牌集合。根据聚类过程（已聚类节点的 s 跳度至多为 κ），转发的令牌数量至多为 κ。该过程的两个阶段都可以在球图上实现，代价是增加 $O(\tau \cdot r)$ 倍的延迟，原理与 [GGH$^+$22 引理 4.7] 相同。该过程在球图中定义了一棵树；对于树中的每个球，在斯坦纳树中选取一条路径，即可将其扩展到 G。所需的通信原语在原理上类似，包括从簇中心发送单条消息，以及向簇中心进行汇聚广播。这些操作在扩展后的树中以与上述相同的方式实现。

□

B 第 5.3 节伪代码

算法 3： G^k的稀疏化

输入： $k \geq 1$；每个 $v \in V$都知道自己是否属于初始活跃节点 $A \subseteq V$的集合。

$Q_0 := A$
对于迭代 $s = 1 \quad , \ldots, k$：
  $H_1 := Q_{s-1}$
  $\Delta_A^{(s)} :=$ Δ 如果 $s = 1$ 则 72Δ记录日志 n
  $r :=$ ⌊记录日志 Δ_A − 记录日志 记录日志n⌋ − 5
  遍历 阶段 $i = 1, \ldots, r$：
    使用断言 5.6 找到 $M_i \subseteq H$，使得对于任何情况，Φ_v 和 Ψ_v（引理 5.7）都不会出现 $v \in V$
    对每个 $v \in H_i$ 并行：
      如果 $v \in M_i$：
        发送标志 已采样，传播至距离为 $2s$ 的邻域（距离为 2 邻域在 G^s 中）
      如果 $v \in M_i$ 或收到 已采样：
        从 H_i 中移除 v
        向距离为 s 的邻域广播（已停用，标识符(v)）（距离为 1 邻域中 G^s），使用引理 4.2
    对每个 $v \in V$ 并行：
      形成关于 $N^s(v, H_{i+1}) \subseteq N^s(v, H_i)$ 的知识（移除发送了停用信息的节点 $w \in H_i$ 已停用）
  $M_{r+1} := H_{r+1}$
  $Q_s := \cup_{i=1}^{r+1} M_i$
  对每个 $v \in V$ 并行执行：
    将 $N^s(v, Q_s)$ 发送给所有邻居 $w \in N(v)$，以获知 $N^{s+1}(v, Q_s)$ 并扩展广度优先搜索树（引理 4.1）
返回 Q_k