arXiv:2305.06120v1 [cs.DS] 10 May 2023

极大独立集与匹配的平均唤醒复杂度

Mohsen Ghaffari JulianPortmann MIT 苏黎世
联邦理工学院 ghaffari@mit.edu
pjulian@inf.ethz.ch

摘要

Chatterjee、Gmyr 和 Pandurangan [最近在 2020] PODC 上引入了分布式算法中的 *awake complexity* 概念，该概念用于衡量一个节点处于 *awake* 状态的轮次数。在其他轮次中，该节点处于 *sleeping* 状态，不执行任何计算或通信。衡量唤醒轮次的数量在分布式计算的许多场景中都具有重要意义，例如在需要关注能源消耗的传感器网络中。

在该论文中，Chatterjee 等人为极大独立集（MIS）问题提出了一种优雅的随机化算法，实现了 $O(1)$ *node-averaged awake complexity*。也就是说，节点的平均唤醒时间为 $O(1)$ 轮次。然而，为了实现这一点，该算法牺牲了更为标准的 *round complexity* 度量。该度量源自 Luby 在 STOC'85], 上提出的 MIS 的著名 $O(\log n)$ 界，并将其改进为 $O(\log^{3.41} n)$ 轮次。

我们的第一项贡献是提出一种简单的随机化分布式 MIS 算法，该算法以高概率具有 $O(1)$ 的节点平均唤醒复杂度和 $O(\log n)$ 的最坏情况轮复杂度。我们的第二项，也是更具技术性的贡献，是为最大匹配的 $(1+\varepsilon)$-近似和最小顶点覆盖的 $(2+\varepsilon)$-近似展示具有相同 $O(1)$ 节点平均唤醒复杂度和 $O(\log n)$ 最坏情况轮复杂度的算法，其中 ε 表示任意小的正数。

1 引言

过去四十年来，针对局部图问题的分布式算法——例如极大独立集、极大匹配以及顶点着色和边着色——已得到广泛研究。这些研究主要关注算法的时间复杂度，例如所有节点终止前所需的轮次数。当然，除了时间之外，还存在其他资源，而仅凭这一度量无法足够准确地反映这些资源的消耗。近期，Chatterjee、Gmyr 和 Pandurangan[CGP20] 提出了唤醒复杂度这一概念，将其作为衡量其他某些资源的更优指标，其中最重要的是能量（例如，在传感器网络中，能量是一种至关重要的资源）。下文将回顾该模型及这一度量，在介绍已有研究成果后，进一步概述我们的贡献。

1.1 模型与动机

基准设定是 将网络抽象为同步消息传递模型。具体地，我们采用 CONGEST 模型 [Pel00]。

这种 CONGEST 模型 将网络抽象为一个 n-节点图 $G=(V,E)$，每轮中每个节点都可以向其每个邻居发送一条 $O(\log n)$ 位消息。消息大小无界的变体称为 LOCAL 模型 [Lin87，Pel00]。初始时，节点不知道网络的拓扑结构，但知道一些全局参数的估计值，即 n 和最大度 Δ的多项式紧约束上界。最终，每个节点都应知道输出中属于自己的部分；例如，在极大独立集问题中，节点应知道自己是否属于计算得到的极大独立集。

休眠轮次/唤醒轮次与唤醒复杂度在他们提出的模型变体中，Chatterjee 等人允许每个节点在每一轮中处于唤醒或休眠状态。休眠节点不会接收任何通信，也不执行任何计算。节点 A_v 的唤醒复杂度 v，简单来说，就是节点 v 在终止之前处于唤醒状态的轮次数量。

研究动机 除此之外，研究唤醒复杂度的一个关键动机是，它能够对能耗给出更精确的界。值得注意的是，能耗是各种分布式系统中的关键资源之一，尤其是在实际应用场景中，例如可能部署局部图问题算法的传感器网络和自组织网络。由于休眠节点不会进行任何计算或通信，因此将休眠轮次的能耗建模为零（即可忽略）是合理的。从这个意义上说，节点 v 消耗的能量可以建模为与其唤醒复杂度 A_v成正比。

节点平均唤醒复杂度 研究能耗时，除了每个节点消耗的最大能量——我们将其称为 最坏情况唤醒复杂度——了解整个网络消耗的能量也很有意义（甚至可以说更加重要）。与上文类似，网络消耗的总能量可以建模为与 $\sum_{v\in V} A_v$成正比，即对所有节点 v 在 v 处于唤醒状态的轮次数求和。因此，要衡量网络消耗的总能量，我们可以关注节点平均唤醒复杂度，即 $\frac{1}{n}\sum_{v\in V} A_v$. 有关节点平均指标的更多内容，请参阅 第 1.5 节。

1.2 研究现状

极大独立集（MIS）问题以及匹配问题——极大匹配和最大匹配的近似——是分布式图算法中的核心问题。自 20 世纪 80 年代以来，研究人员对此进行了大量研究。我们先回顾标准度量——轮复杂度——的已知最佳界限，然后讨论唤醒复杂度方面的已知结果。

轮复杂度的研究现状——随机化算法 Luby [Lub86] 以及 Alon、Babai 和 Itai [ABI86], 提出的经典随机化 MIS 算法，以高概率达到 $O(\log n)$ 的轮复杂度[1]。这也直接给出了一个用于求解极大匹配的 $O(\log n)$ 轮算法，因为极大匹配可以轻易归约为线图上的 MIS。由此还可在 $O(\log n)$ 轮内得到最大匹配的 2-近似。Lotker、Patt-Shamir 和 Pettie [LPSP15] 展示了如何在保持相同 $O(\log n)$ 轮复杂度[2]的同时，将最大匹配的近似比改进为 $(1+\varepsilon)$，其中常数 $\varepsilon > 0$可以任意小。最后，Bar-Yehuda、Censor-Hillel、Ghaffari 和 Schwartzman [BYCHGS17] 将 $(1+\varepsilon)$-近似的轮复杂度改进至 $O(\log \Delta/\log\log \Delta)$，其中 Δ 表示最大度；在最坏情况下，$\Delta = \Theta(n)$ 会使轮复杂度达到$O(\log n/\log\log n)$。总而言之，所有这些问题已知的最佳轮复杂度仍大致为 $O(\log n)$，但匹配近似稍快，为 $O(\log n/\log\log n)$ 轮。

轮复杂度研究现状——确定性算法这些问题的确定性算法要慢得多。为确保内容完整，我们仍然回顾一下当前最先进的研究成果。对于 MIS，长期以来，已知最好的确定性算法一直停留在 Panconesi 和 Srinivasan [PS92] 提出的 $2^{O(\sqrt{\log n})}$ 界限，但最近 Rozhoˇn 和 Ghaffari [RG20] 将其改进为 $O(\log^7 n)$，随后 Ghaffari、Grunau 和 Rozhoˇn [GGR21]又进一步将其改进为 $O(\log^5 n)$。对于极大匹配，尽管 MIS 的相同界限也适用，但此前已有速度更快的算法。Hanckowiak、Panconesi 和 Karonski [HKP01] 提出了一种轮复杂度为 $O(\log^4 n)$ 的确定性极大匹配算法。随后，Fischer [Fis20], 将轮复杂度改进为 $O(\log^2 \Delta \log n)$，而且该算法还能够在 $O(\log^2 \Delta + \log^* n)$ 轮次内提供最大匹配的 $(2+\varepsilon)$-近似。

[1]按照惯例，我们用短语 *with high probability* （w.h.p.）表示某事件发生的概率至少为$1-1/n^c$，其中 $c \geq 2$是一个足够大的常数。[2]由于本文仅研究无权匹配，因此不讨论最大 *weighted* 匹配的近似问题的相关大量文献。

特殊图的改进算法不区分随机化算法和确定性算法时，对于任意图，已知的最佳轮复杂度上界仍约为 $O(\log n)$。然而，在某些特殊情况下，算法性能已有所改进，尤其是在最大度不太大的情况下。

对于最大度为 Δ （具体而言为 $\Delta \leq O(\log n)$）的图，可以采用 Panconesi 和 Rizzi 的 $O(\Delta + \log^* n)$ 轮确定性算法 [PR01]，得到更快的算法。对于中等偏小的 Δ（具体而言，当 $\log \Delta = o(\log n)$ 且 $\Delta \geq \text{poly}(\log \log n)$ 时），已有更快的随机化算法，其轮复杂度为 $O(\log \Delta) + \text{poly}(\log \log n)$。该算法源自 Ghaffari 的工作 [Gha16], ，并结合了 Rozho˘n 和 Ghaffari 的近期成果（及其描述）[RG20]。这建立在 Barenboim 等人提出的碎裂方法 [BEPS16]之上。但需注意：该算法要求使用大消息。对于消息长度为 $O(\log n)$ 位的情形， [PPP+17、Gha19、GP19] 给出了改进的界，而目前已知的最佳轮复杂度为 $O(\log \Delta \cdot \sqrt{\log \log n}) + 2^{O(\sqrt{\log \log n})}$。对于极大匹配，结合 [Gha16] 和[Fis20]的工作，目前已知的最佳复杂度为 $O(\log \Delta + \log^3 \log n)$。对于匹配近似，Bar-Yehuda 等人上述的工作 [BYCHGS17] 为 $(1+\varepsilon)$-近似实现了 $O(\log \Delta / \log \log \Delta)$ 的复杂度；对于 $\Delta = O(\sqrt{\log n})$，这是最优轮复杂度，与 Kuhn 等人的下界相匹配 [KMW04, KMW16]。

对于其他受限图族，极大独立集和匹配还有许多其他算法结果，例如有界树杂度的图 [BE10，BE13], 增长有界图 [SW08], 等。

下界 $\Omega(\log^* n)$轮的下界源于 Linial 的开创性工作 [Lin87], ，即使在网络为环的情况下也成立。这一结论适用于极大独立集（MIS）和极大匹配，并且也扩展到了随机化算法 [Nao91]。Kuhn、Moscibroda 和 Wattenhofer 提出了一个 $\Omega(\min\{\log \Delta \log \log \Delta, \sqrt{\log n \log \log n}\})$ 的下界 [KMW04，KMW16] ，该下界同样适用于随机化算法，并适用于极大独立集、极大匹配，甚至最大匹配的任意常数近似。

最后，Balliu 等人 [BBH+19] 最近证明，对于极大独立集或极大匹配，不存在轮复杂度为 $o(\Delta) + o(\log \log n / \log \log \log n)$ 的随机化算法，也不存在轮复杂度为 $o(\Delta) + o(\log n / \log \log n)$ 的确定性算法。上述所有下界均适用于允许任意大小消息的 LOCAL 模型。

1.3 超越最坏情况

总而言之，关键结论是：对于任意图，已知最佳轮复杂度仍约为 $O(\log n)$ 轮 [Lub86, ABI86], ，但在最大匹配近似的情形下需额外乘上一个 $\log \log n$ 因子。我们还知道，对于 MIS 或匹配问题（最大匹配的任何常数近似，因此也包括极大匹配），最坏情况轮复杂度不可能优于 $\Omega(\min\{\sqrt{\log n \log \log n}, \log \Delta \log \log \Delta\})$ [KMW16], 。此外，在一项近期独立完成的工作中，Balliu、Ghaffari、Kuhn 和 Olivetti [BGKO22] 证明，对于这些问题的任何算法，即使按节点取平均的轮复杂度也存在 $\Omega(\min\{\sqrt{\log n \log \log n}, \log \Delta \log \log \Delta\})$ 下界（关于特殊图族中节点平均轮复杂度的一些改进下界，见第 1.5 节）。

面对轮复杂度乃至节点平均轮复杂度方面的这一障碍，剩下的一线希望或许是节点处于唤醒状态的轮次数能大幅减少。例如，这意味着能耗可以显著降低。不过，由于节点在休眠轮次中不进行任何计算或通信，唤醒复杂度能否大幅降低，事先并不明朗。事实上，据我们所知，目前还没有任何算法能在 MIS 或任意常数匹配近似中实现 $O(\log n / \log \log n)$ 的最坏情况唤醒复杂度。但对于节点平均唤醒复杂度，至少在 MIS 问题上已有更好的结果，下面我们将对此进行回顾。

节点平均唤醒复杂度的最新进展 Chatterjee 等人 [CGP20] 提出了一种随机化极大独立集算法，其节点平均唤醒复杂度为 $O(1)$。此外，该算法的最坏情况唤醒复杂度为 $O(\log n)$，也就是说，每个节点最多唤醒 $O(\log n)$ 轮次。但遗憾的是，该算法最终牺牲了最常用的指标，即轮复杂度。具体而言，该算法的轮复杂度为 $O(\log^{3.41} n)$，且以高概率成立，甚至节点平均

轮复杂度为 $O(\log^{3.41} n)$。这明显高于上文讨论的通常 $O(\log n)$界 [Lub86, ABI86]。这引出了一个自然的问题：

问题：在保持轮复杂度为 $O(\log n)$ 的同时，我们能否实现理想的 $O(1)$ 节点平均唤醒复杂度？

此外，MIS 与（极大）匹配之间的标准联系并不能提供节点平均唤醒复杂度。请注意，如果我们在线图上应用 MIS 算法来求解极大匹配，那么 MIS 问题中的节点就是匹配问题中的边，而节点平均唤醒复杂度只能说明每条边的平均活跃时间。这显然无法对（极大）匹配的节点平均唤醒复杂度提供任何有用的信息。因此，一个自然的问题是：

问题：对于匹配，具体来说，对于极大匹配，或者最大匹配的任意常数近似，我们能否也实现 $O(1)$ 的节点平均唤醒复杂度？理想情况下，我们还希望保持轮复杂度为 $O(\log n)$。

1.4 我们的贡献

MIS 方面的贡献。我们的第一个贡献回答了上文提到的第一个关于 MIS 的问题。具体来说，我们证明：

定理 1.1. 对于任意由 n 个节点组成的网络，在 CONGEST 模型中存在一个随机化（拉斯维加斯）分布式极大独立集算法，其性能如下，且该性能以高概率成立：

- $O(1)$ 节点平均唤醒复杂度，以及
- $O(\log n)$ 最坏情况轮复杂度。

这改进了 Chatterjee 等人近期的工作 [CGP20]，他们实现了 O（1）节点平均唤醒复杂度，以及 O（$\log^{3.41} n$）最坏情况轮复杂度。我们的结果有效地表明，为了实现节点平均唤醒复杂度，我们无需牺牲轮复杂度。顺带一提，我们的算法相当简单；事实上，算法描述及其分析都比 Chatterjee 等人的工作短得多 [CGP20]。

关于匹配和顶点覆盖的贡献。如前所述，与最坏情况复杂度不同，一个具有良好节点平均（唤醒）复杂度的极大独立集（MIS）算法，并不一定能够导出一个具有良好节点平均（唤醒）复杂度的（极大）匹配算法。我们的第二项也是技术性更强的贡献，是提供能够计算最大匹配的 $(1+\varepsilon)$-近似解和最小顶点覆盖的 $(2+\varepsilon)$-近似解的算法，其节点平均唤醒复杂度为 $O(1)$，最坏情况轮复杂度为 $O(\log\ n)$。

定理 1.2. 在任意包含 n 个节点的网络中，以高概率存在随机化分布式算法，可在 CONGEST 模型中计算最大匹配的 $(1+\varepsilon)$-近似解和最小顶点覆盖的 $(2+\varepsilon)$-近似解，并具有以下性能：

- $O(1)$ 节点平均唤醒复杂度，以及
- $O(\log n)$ 最坏情况轮复杂度。

<escape>我们指出，在上述陈述中，清醒复杂度和轮复杂度的界限以高概率成立，即其概率至少为 $1-1/n^c$，其中 $c \geq 2$ 是一个足够大的常数。近似界限则在期望意义下成立。这是讨论随机近似算法时对近似保证的一种标准表述。事实上，我们的近似结果在该期望值附近具有更强的集中性：获得更差近似结果的概率随最大匹配规模的增加大致呈指数级衰减（这种行为在某些用于匹配近似的随机分布式/并行算法中较为常见，参见例如 [BYCHGS17、GGK+18]）。我们以两个版本给出这一结果：一个是较易证明的期望近似因子版本，另一个是更强的、以高概率成立的近似因子版本。对于后者，我们假设最大匹配规模至少为 poly(log n)。我们将这一假设的进一步放宽留待完整版讨论。</escape>

1.5 节点平均轮复杂度的其他相关工作

值得注意的是，研究节点平均唤醒复杂度可以被视为沿着 Feuilloley 开创的方向继续探索[Feu20]，，他研究的是节点平均轮复杂度。在其他研究成果中，Feuilloley 证明了 Linial 关于环的 3-着色的 $\Omega(\log^* n)$ 轮下界同样适用于节点平均轮复杂度。他还展示了一些其他稀疏图，在这些图中，3-着色的节点平均轮复杂度为 $O(1)$，而最坏情况下的轮复杂度则必须达到 $\Omega(\log^* n)$。Barenboim 和 Tzur [BT19]针对各种问题在这一研究方向上取得了进一步进展。对于极大独立集（MIS）问题，他们提出了一种确定性算法，其节点平均轮复杂度为 $O(a + \log^* n)$，其中 a 表示图的稀疏度（arboricity）。对于密切相关的 $\Delta + 1$ 着色问题，Johansson 的经典随机算法 [Joh99] 实现了 $O(1)$ 的节点平均轮复杂度，以及 $O(\log n)$ 的最坏情况轮复杂度。前者的原因很简单：在每一轮中，每个尚未着色的节点都有一个常数概率获得颜色（并从问题中移除）。如前所述，在极大独立集和匹配问题方面，Alkida 等人 [BGKO22] 通过同期独立开展的研究证明，Kuhn、Moscibroda 和 Wattenhofer 关于最坏情况复杂度的 $\Omega(\min\{\sqrt{\log n \log\log n}, \log\Delta\log\log\Delta\})$ 下界可以加以调整，使其同样适用于节点平均轮复杂度。

2 预备知识

我们会用到一些基本的集中不等式:

定理 2.1（霍夫丁界）。设 $X_1, \ldots, X_N$ 是相互独立的随机变量，且每个变量的取值都在范围 $[a_i, b_i]$ 内。设 $\bar{X} = \sum_{i=1}^{N} X_i$。那么，对于任意 $t \geq 0$，有

$$\Pr\left(|\bar{X} - \mathbb{E}[\bar{X}]| \geq t\right) \leq 2exp(-\frac{2N^2t^2}{\sum_{i=1}^{N}(b_i - a_i)^2}).$$

定理 2.2（Chernoff 紧约束版本）。设 $X_1, \ldots, X_n$ 为相互独立的随机二元变量，且 $a_1, \ldots, a_n$ 为 $[0, 1]$ 中的系数。设 $X = \sum_{i=1}^{n} a_i X_i$。那么，对于任意 $\mu \geq \mathbb{E}[X]$ 和 $\delta > 0$

$$\Pr[X > (1 + \delta)\mu] \leq \left(\frac{e^{\delta}}{(1 + \delta)^{1+\delta}}\right)^{\mu}.$$

除此之外，我们还需要利用 Chernoff/Hoeffding 界的一种扩展形式，以处理随机变量之间存在较弱依赖关系的情况。下面将对此进行形式化描述。

定理 2.3 （切尔诺夫界在有界依赖情形下的推广 [Pem01]）。设 $X_1, \ldots, X_N$是二元随机变量，并设 $a_1, \ldots, a_n$ 是 $[0, 1]$ 中的系数。假设每个 X_i 变量与除至多 Δ' 个变量之外的所有其他变量都相互独立。令 $\bar{X} = \sum_{i=1}^{N} a_i \cdot X_i$，并且 $\mu = \mathbb{E}[\bar{X}]$. 那么，对于任意 $\delta \geq 0$，有

$$\Pr\left(|\bar{X} - \mathbb{E}[\bar{X}]| \geq \delta\mathbb{E}[\bar{X}]\right) \leq 8(\Delta' + 1) \cdot exp(-\frac{\mathbb{E}[\bar{X}]\delta}{3(\Delta' + 1)}).$$

方便的是，令 $\Delta' = 1$，我们便可以在常数因子的意义下，将标准切尔诺夫界用于独立随机变量。

3 MIS 算法

这里，我们介绍用于证明 定理 1.1 的 MIS 算法，即实现 $O(1)$ 的节点平均唤醒复杂度和 $O(\log n)$ 的轮复杂度。

算法概述 该算法分为三个部分：在每个部分中，我们都会向集合S中添加一些节点。该集合最初为空，并始终保持为一个独立集，从而确保最终 S 是一个极大独立集。第一部分将最大度降低至 $\text{poly}(\log n)$。第二部分是新算法的技术核心；它运行 $O(\log\log n)$ 次迭代，每次迭代需要 $O((\log\log n)^3)$ 轮次，并且每次迭代都会移除剩余节点中的一个常数比例。同时，该过程也确保节点平均唤醒复杂度保持为 $O(1)$。第三部分是一个简单的清理步骤，用于处理所有剩余节点。接下来，我们将逐一介绍这三个部分。

3.1 第一部分

算法，第一部分 第一部分中，我们使用随机贪心极大独立集算法的一次部分运行[BFS12，FN18]。回顾一下，在随机贪心极大独立集算法中，每个节点 v 都会选取一个随机数 r_v ，该随机数属于$I = \{1, 2, \ldots, n^{10}\}$，并且每轮中，所有随机数严格小于其邻居随机数的节点都会被加入独立集 S。随后，这些节点及其邻居都会从图中移除。与 Luby 算法 [Lub86] 的关键区别在于，我们不会在每轮使用新的随机数。

我们只对随机数的前 $p = 1/\log n$ 部分运行此算法。具体而言，我们只对随机数属于 $I_p = \{1, 2, \ldots, n^{10}/\log n\}$ 的节点运行该算法。在此期间，其余节点保持休眠。最后，我们再用一轮从图中移除所有随机数属于 I_p 的节点，以及 $V \setminus I_p$ 中所有与计算得到的独立集 S 相邻的节点。

第一部分分析 我们现在说明，算法的这一部分将剩余图的最大度降低至 $\text{poly}(\log n)$，并且其节点平均唤醒复杂度为 $O(1)$。

引理 3.1. 以高概率，算法第一部分（如上所述）具有以下性质：

(A) 该算法运行 $O(\log n)$ 轮次，

(B) 该算法对节点平均唤醒复杂度 $\frac{1}{n}\sum_{v\in V} A_v$ 的贡献为 $O(1)$，

(C) 在由剩余节点诱导的子图中，每个节点的度至多为 $O(\log^2 n)$。

证明。根据[FN18], 随机化贪心算法会在 $O(\log n$轮次内终止。注意， I_p 中的随机数对应的节点最迟会在此时终止，并且它们不会受到随机数大于此范围的其他节点的影响。因此，即使仅在编号位于 I_p 内的节点上运行算法，也会在 $O(\log n$轮次内终止。

在此期间，每个节点以 $|I_p|/|I| = 1/\log n$ 的概率处于唤醒状态。因此，根据切尔诺夫界，在算法运行期间，处于唤醒状态的节点数量至多为 $O(n/\log n)$。考虑到算法运行需要 $O(\log n)$ 轮次，这对所有节点的唤醒复杂度之和 $\sum_v A_v$ 的贡献至多为 $O(n/\log n) \cdot O(\log n) = O(n)$。换言之，这使节点平均唤醒复杂度 $\frac{1}{n}\sum_v A_v$ 增加了一个加性项 $O(1)$。

最后，根据 [BFS12，引理 4.3], ，我们可以得到如下性质：以高概率，在由剩余节点诱导的子图中，每个节点的度至多为 $O(\log^2 n)$。论证很简单，因此我们在此重述一遍，以使本文内容完整。算法计算出的独立集，与按随机数递增的顺序逐个处理节点、每次将下一个节点加入独立集 S 并移除其所有邻居所得到的独立集完全相同。现在考虑任意节点 v。每处理一个节点时，该节点在所有剩余节点中均匀随机选取。因此，如果 v 的度至少为 d，那么下一个被处理的节点是 v 的邻居之一的概率至少为 d/n。在这种情况下，v 会从图中被移除。因此，在单步中，节点 v仍然存在且度至少为 d的概率至多为 $(1 - d/n)$。经过 $\Theta(pn)$ 步后，我们可以得出结论： v 仍然存在的概率至多为 $(1 - d/n)^{pn}$。令 $d = 10\log n/p = \Theta(\log^2 n)$，则度大于 $\Theta(\log^2 n)$ 的节点 v仍然存在的概率至多为 $1/n^2$。对所有顶点应用并合界可知，以至少 $1 - 1/n$ 的概率，没有节点仍然存在且度至少为 $d = \Theta(\log^2 n)$。

□

3.2 第二部分

第二部分是我们算法的核心创新。在这一部分开始时，我们得到一个最大度至多为 $d = O(\log^2 n)$ 的图，且这一点以高概率成立。对于第二部分，我们有一个算法，其最坏情况轮复杂度为 $O((\log\log n)^4)$。该算法由 $O(\log\log n)$ 次迭代组成，每次都在剩余节点上重复同一个 $O((\log\log n)^3)$ 轮的过程。该过程使得每次迭代的节点平均唤醒复杂度为 $O(1)$，平均范围是执行该次迭代的节点。此外，每次迭代剩余节点的数量 i 会随迭代次数 i呈指数级递减。因此，第二部分的节点平均唤醒复杂度总体为 $O(1)$。

算法，第二部分我们在剩余节点上重复执行以下过程，共进行 $O(\log\log n)$ 次相同的迭代。每次迭代的算法包含 $\log d = O(\log\log n)$ 个阶段，其中，在阶段$i \in [1,\log d]$中，我们进行 $C(\log d - i)^2$ 轮次：在 i^{th} 阶段的每一轮中，我们以概率$2^i/d$标记每个节点。随后，所有没有相邻标记节点的已标记节点都会被加入独立集。这里， C 是一个足够大的常数，我们将在后面讨论这一点。在每次迭代结束时，我们还会额外进行一轮，在这一轮中，每个孤立节点（即没有剩余邻居的节点）都会加入独立集 S，并从图中移除。

现在我们解释节点处于唤醒或休眠状态的时间：在每次迭代开始时，每个节点v 都会确定本次迭代中 v 被标记的所有轮次。这是根据上述标记概率来完成的。任何节点 v 都会一直休眠，直到本次迭代中 v 首次被标记的时刻，然后从该时刻起一直保持唤醒状态，直到本次迭代结束。如果不存在 v 被标记的轮次，那么 v在整个迭代过程中都会保持休眠状态，但最后一轮除外，因为在最后一轮中，我们移除了孤立节点。

最后，我们需要确保移除那些已经加入独立集 S 的节点的邻居，即使这些邻居可能处于休眠状态。为此，我们将每一轮划分为两个子轮次。对于在第 r 轮加入独立集 S 的任意节点 v，在任意第 $r' \geq r+1$ 轮中，在第 r' 轮的第一个子轮次中，节点 v 会将节点 v 已经属于所计算出的独立集 S 这一信息通知给第 r' 轮中所有被标记（因此处于唤醒状态）的邻居 u。在这种情况下，节点 u 将终止整个算法，并输出自己不属于独立集 S。第二个子轮次用于声明节点 w 是否被标记。在此过程中，只有那些已被标记且没有任何已加入独立集 S 的邻居的节点 w（如果存在这样的邻居，节点 w 就会在第一个子轮次中获知这一情况）才会向其邻居声明自己已被标记。我们指出，将每一轮替换为两个子轮次的这一调整，不会影响轮复杂度或唤醒复杂度的渐近界。为简单起见，我们将忽略这一 2 倍因子，仅使用轮数进行表述。

第二部分分析

引理 3.2. 以高概率，算法第二部分（如上所述）具有以下性质：

(A) 算法运行 $O(\log^4 \log n)$ 轮。(B) 算法对节点平均唤醒复杂度 $\frac{1}{n}\sum_v A_v$ 的贡献为 $O(1)$，并且 (C) 最后一次迭代后至多剩下 $O(n/\log n)$ 个节点。

证明。根据算法的结构，显然共有 $O(\log\log n)$ 次迭代，每次迭代需要 $\sum_{i=1}^{\log d} C(\log d - i)^2 = O((\log\log n)^3)$ 轮。因此，该算法此部分的最坏情况轮复杂度为 $O((\log\log n)^4)$。

在每次迭代中，每个节点都会休眠，直到它第一次被标记的轮次，然后在该次迭代剩余时间内保持唤醒。节点 v在阶段 i唤醒的概率至多为 $\frac{2^i}{d}\cdot C(\log d - i)^2$。如果 v在阶段 i唤醒，那么在该次迭代中，它会在此后所有剩余轮次保持唤醒，而剩余轮次至多为 $\left(\sum_{j=i}^{\log d} C(\log d - j)^2\right)$ 轮。因此，每个节点的预期唤醒轮数

参与某次迭代的节点至多为

$$\sum_{i=1}^{\log d} \frac{2^i}{d} \cdot C(\log d - i)^2 \cdot \Big(\sum_{j=i}^{\log d} C(\log d - j)^2\Big) \le \sum_{i=1}^{\log d} 2^{-(\log d - i)} \cdot O((\log d - i)^5)$$
$$= \sum_{x=1}^{\log d} 2^{-x} \cdot O(x^5) = O(1).$$

由上可知，参与一次迭代的节点的预期唤醒轮数为 $O(1)$。设 V_k 为开始迭代 k的节点集合。我们将在下一段中很快看到，$|V_k| \le O(n/2^k)$以高概率成立。此外，不同节点的唤醒轮数是相互独立的随机变量，且每个值的最大值至多为 $O((\log\log n)^4)$。因此，利用 Hoeffding 界（参见 定理 2.1），可得以高概率，对于每个 $k \in [1, O(\log\log n)]$，参与该次 k^{th}迭代的节点的唤醒轮数总和 $\sum_{v\in V_k} A_v$ 至多为 $O(n/2^k)$。这意味着，以高概率，所有这些迭代对节点平均唤醒复杂度的贡献 $\frac{1}{n}\sum_v A_v$ 为 $\sum_{k=1}^{O(\log\log n)} O(1/2^k) = O(1)$。

最后，我们证明每个节点 v 在每次迭代中被移除的概率至少为某个正的常数。这将使我们能够完成引理性质 (B) 的证明，同时也能证明性质 (C)。

我们证明一个更一般的结论：任取一个在阶段 k开始时存在的节点 v。我们证明，在该次迭代的各个阶段中，以一定概率，v 的度始终被一个指数递减函数所上界。事实上，我们不仅会对 v 证明这一点，还会以一定概率对 v周围某个半径范围内的节点证明这一点。随着我们不断推进，这个半径会逐渐减小，最终达到零，也就是说，只剩节点 v 本身。此外，我们保证成立的失败概率也会随时间增加，但最终至多达到 $1/2$。下面我们将具体说明。

设 $\mathcal{E}_i$ 表示这样一个事件：在阶段 i结束时，距 v不超过 $r_i = (\log d - i)$ 的范围内，至少有一个节点 u满足 u的度至少为 $d/2^i$。我们通过对 i进行归纳，证明 $Pr[\mathcal{E}_i] \le 2^{-2(\log d - i + 1)^2}$。对于基例 $i = 0$，该结论显然成立，因为其中所有节点的度都至多为 d，因此 $Pr[\mathcal{E}_0] = 0$。

考虑一个阶段 $i \in [1, \log d]$。根据归纳假设，我们知道 $Pr[\mathcal{E}_{i-1}] \le 2^{-2(\log d - i + 2)^2}$。此外，假设 $\mathcal{E}_{i-1}$ 没有发生。我们已经得到了 $\mathcal{E}_{i-1}$ 概率的上界，并通过并集界将其纳入考量。

考虑任意一个距离 $r_i = (\log d - i)$ 以内的节点 u，其中 v。由于 $\mathcal{E}_{i-1}$ 没有发生，我们知道，在阶段 i 开始时，节点 u 及其任意邻居的度都小于 $d/2^{i-1}$。如果 u 的度至少为 $d/2^i$，那么在该阶段的每一轮中，节点 u 或其某个邻居被标记且没有被标记的邻居的概率至少为一个常数 $c > 0$。原因很简单：考虑这样一轮。我们逐个检查 $N(u) \cup \{u\}$ 中的节点，每次揭示该节点是否被标记，并在找到一个被标记的节点 $w \in N(u) \cup \{u\}$时停止。这里，$N(u)$ 表示在这一轮中 u 的邻居。我们以至少 $(1 - 2^i/d)^{d/2^i + 1} \ge 1/4$ 的概率找到至少一个未被标记的节点。现在，节点w 的度至多为 $d/2^{i-1}$。因此，w 的邻居（不在 $N(u) \cup \{u\}$ 中）均未被标记的概率至少为 $(1 - 2^i/d)^{d/2^{i-1}} \ge 1/16$。所以，总体而言，存在一个被标记的节点 $w \in N(u) \cup \{u\}$，且其没有被标记的邻居的概率至少为 $1/4 \cdot 1/16 = 1/64$。也就是说，节点 u 或其某个邻居以至少一个常数概率 $c > 0$ 加入独立集 S。这意味着节点 u 实际上已被移除（即，它会在首次唤醒时被移除）。此外，由于该阶段的 $C(\log d - i)^2$ 轮中的随机性相互独立，阶段结束时节点 u 仍具有至少 $d/2^i$ 的度（即，其闭邻域内没有独立集节点，因此未被实际移除）的概率至多为 $2^{-5(\log d - i + 1)^2}$。只要根据上述常数概率 $c = 1/64$ 将 C 选得足够大，这一结论就成立。此外，我们还可以对 v 的 $r_i = (\log d - i)$ 跳邻域中的所有此类节点 u 作联合界。由于 $\mathcal{E}_{i-1}$ 没有发生，该邻域中每个节点的度至多为 $(d/2^{i-1})$，这意味着该邻域中的节点数至多为

$(d/2^{i-1})^{\log d-i} = 2^{(\log d-i+1)(\log d-i)}$。因此，我们有

$$\begin{aligned}\Pr[\mathcal{E}_i] &\le \Pr[\mathcal{E}_{i-1}] + 2^{(\log d-i+1)(\log d-i)} \cdot 2^{-5(\log d-i+1)^2} \\ &\le 2^{-2(\log d-i+2)^2} + 2^{(\log d-i+1)(\log d-i)} \cdot 2^{-5(\log d-i+1)^2} \\ &\le 2^{-2(\log d-i+2)^2} + 2^{-5(\log d-i+1)^2} \\ &\le 2^{-2(\log d-i+1)^2}/2 + 2^{-2(\log d-i+1)^2}/2 \\ &= 2^{-2(\log d-i+1)^2}.\end{aligned}$$

以上便完成了归纳证明。现在，将该结论应用于 $i = \log d$，可知到第 $\log d$ 阶段结束时，节点 v 的度至少为 1 的概率至多为 $1/2$。因此，以至少 $1/2$ 的概率，节点 v 在 $\log d$ 个阶段结束时处于孤立状态，从而最迟会在该次迭代的最后一轮中被移除。

最后，由于不同迭代使用相互独立的随机性，我们得到，节点 v 在经过 k 次迭代后仍然存在的概率至多为 $1/2^k$。因此，经过 k 次迭代后，剩余节点的期望数量至多为 $n/2^k$。此外，我们可以应用针对低依赖随机变量的切尔诺夫界扩展（参见定理 2.3），证明以高概率成立类似的上界：注意，对于彼此距离大于 $O(\log 4 \log n)$ 的两个节点 v 和 u，其对应的事件是相互独立的，因为该算法即使包括全部 $O(\log \log n)$ 次迭代，其轮复杂度仍为 $O(\log^4 \log n)$。因此，对于每个节点 v，除至多有 $\Delta' = (O(\log^2 n))^{O(\log^4 \log n)} = 2^{O(\log \log^5 n)}$ 个与其距离在 $O(\log^4 \log n)$ 跳以内的节点外，节点 v 仍然存在的事件与所有其他节点的事件相互独立。也就是说，不同节点仍然存在的这些事件的依赖度至多为 Δ'。因此，利用定理 2.3，我们可以得出结论：对于任意 $k \in [1, O(\log \log n)],$，经过 k 次迭代后，剩余节点数量超过 $O(n/2^k)$ 的概率至多为

$$\begin{aligned}O(\Delta') \cdot exp\big(-\Theta(\frac{n}{2^k\Delta'})\big) &\le exp\big(O(\log^5 \log n) - \frac{n}{2^{O(\log^5 \log n)}}\big) \\ &\ll exp(-n^{0.9}) \ll 1/\operatorname{poly}(n).\end{aligned}$$

因此，以高概率，在 k^{th} 次迭代后，最多剩下 $O(n/2^k)$ 个节点。这正是我们在上文计算节点平均唤醒复杂度并证明引理性质 (B) 时所用的性质。

此外，通过设置 $k = O(\log \log n)$ 次迭代，我们可以得到：以高概率，在最后一次迭代后最多剩下 $O(n/\log n)$ 个节点，这证明了引理性质 (C)。 □

3.3 第三部分

算法第二部分结束时，图中最多剩下 $O(n/\log n)$ 个节点，即满足以下条件的节点：v 使得 v 在其闭邻域 S 中没有节点 $N(v) \cup \{v\}$。在算法第三部分中，我们执行一个简单的清理过程：对剩余节点运行标准的极大独立集算法，例如 Luby 算法[Lub86]。该算法以高概率在 $O(\log n)$ 轮次内运行完毕。此外，它会向独立集 S 中添加额外节点，并确保 S 是一个极大独立集。最后，由于我们只对剩余节点运行该算法，而这些节点至多有 $O(n/\log n)$ 个，因此这 $O(\log n)$ 轮次只会给整体节点平均唤醒复杂度增加 $O(1)$ 轮次。

3.4 总结

最后，我们将展示如何组合前面各部分的内容，并给出定理 1.1的正式证明。

定理 1.1的证明。如引理 3.1所述，第一部分需要 $O(\log n)$ 轮次。此外，在此期间只有$O(n/\log n)$ 个节点处于唤醒状态，这意味着求和中有一项 $O(n)$ 贡献 $\sum_v A_v$。

此外，正如引理 3.2中所述，第二部分运行 $O((\log \log n)^2)$ 轮次；此外，参与该部分的节点的平均唤醒复杂度为 $O(1)$。因此，该部分对求和 $O(n)$ 的贡献同样为 $\sum_v A_v$。最后，正如引理 3.2所示，第二部分结束后，至多剩下 $O(n/\log n)$ 个节点。

第三部分只需在所有剩余节点上运行 Luby 的 $O(\log n)$ 轮算法 [Lub86] 。由于第三部分开始时至多剩下 $O(n/\log n)$ 个节点，因此该部分对求和 $O(n)$ 的贡献也至多为 $\sum_v A_v$。

总体而言，综合这三个部分，我们得到一个轮复杂度为 $O(\log n)+O((\log\log n)^4)+O(\log n) = O(\log n)$ 的算法。此外，总唤醒复杂度为 $\sum_v A_v = O(n) + O(n) + O(n) = O(n)$。因此，节点平均唤醒复杂度为 $\frac{1}{n}\sum_v A_v = O(1)$。□

4 匹配近似

在本节中，我们首先给出一种计算最大匹配的 $(2+\varepsilon)$-近似算法，并证明它也能帮助我们获得一种求解顶点覆盖的 $(2+\varepsilon)$-近似算法。最后，我们证明，对于任意常数 ε，只需在轮数和平均唤醒复杂度上增加常数开销，就可以将匹配算法的近似比提升至 $(1+\varepsilon)$。

4.1 近似分数匹配的基础算法

我们首先介绍以下用于计算分数匹配的简单算法。该算法或其近似变体已在以往的研究中得到广泛应用 [KMW16, GGK+18, Fis20]（不过，该结果通常被作为一种已知过程提及，我们不确定它最早出现于何处）。该算法本身的平均唤醒复杂度为 $\Theta(\log\Delta)$，因为它要求节点始终处于活跃状态，因此无法满足我们实现平均唤醒复杂度为 $O(1)$ 的目标。稍后我们将讨论如何利用精心设计的随机抽样和估计方法，在相同框架的基础上开发一种替代算法，从而实现平均唤醒复杂度为 $O(1)$。

基础算法 每条边 e 初始值为 $x_e = \frac{1}{\Delta}$。最初，所有边都处于活跃状态。随后，在每一轮中，满足 $c_v = \sum_{e\ni v} x_e \geq 1-\varepsilon$ 的节点 v 称为紧约束节点，我们冻结所有与紧约束节点相邻的边。最后，将所有仍处于活跃状态（即未被冻结）的边的值 x_e 更新为 $(1+\varepsilon)x_e$，然后进入下一轮。

分析 注意到，该过程会在 $O(\log_{1+\varepsilon}\Delta)$ 轮次内结束，因为边所能达到的最大值为 1。对于计算所得解的质量，我们有如下保证。

命题 4.1. 设 M^* 是 G 中的一个最大匹配，对于每条边 e，设 x_e 为通过上述算法计算得到的其值。那么，我们有 $|M^*| \leq (2+4\varepsilon)\sum_{e\in E} x_e$，即所有分数值之和是最大（分数）匹配的一个 $(2+4\varepsilon)$-近似，并且 x_e 的值构成一个匹配，即对于每个顶点 v，我们都有 $c_v = \sum_{e\ni v} x_e \leq 1$。

证明。首先，我们来说明计算得到的值 x_e 构成一个有效的分数匹配。第一步中，我们设置 $x_e = \frac{1}{\Delta}$，因此所有顶点都显然满足条件 $c_v \leq 1$ 。在后续迭代中，只有当一条边的两个端点都满足 $c_v \leq 1-\varepsilon$时，我们才会增加该边的权重；因此，将所有未冻结边的权重乘以因子 $1+\varepsilon$，仍能保证所有顶点 v都满足 $c_v \leq (1+\varepsilon)\cdot(1-\varepsilon) = 1-\varepsilon^2 \leq 1$。第一部分到此结束。

关于分数匹配为何是一个 $(2+4\varepsilon)$-近似算法，需要注意的是，最终没有任何边仍处于活跃状态，因此每条边都至少与一个满足 $c_v > 1-\varepsilon$ 的顶点 v 相 incident。我们可以将其视为一个分摊过程：首先在最大（分数）匹配 M^* 上设置权值，然后将这些权值局部转移到我们计算得到的分数匹配上，使得每条边 e 接收到的分摊值至多为 $x_e\cdot 2(1+2\varepsilon)$，从而证明 $|M^*| \leq (2+4\varepsilon)\sum_{e\in E} x_e$。我们从最大（分数）匹配 M^* 开始。将 M^* 中一条边 $e=(u,v)$ 的所有分数值转移到其紧约束端点，记为 v。由于 M^* 是一个合法匹配，转移到任意节点 v 的所有分数值之和至多为 1，因此不超过 $\frac{1}{1-\varepsilon}c_v$。随后，我们将这些权重分摊到已计算出的分数匹配上，因此，每条边 e 从其每个端点接收到的值至多为 $x_e\cdot\frac{1}{1-\varepsilon}$，这意味着其总值至多为 $x_e\cdot\frac{2}{1-\varepsilon} \leq x_e\cdot 2(1+2\varepsilon)$。形式化地，这一观察可以表示为

$$|M^*| \leq \frac{1}{1-\varepsilon}\cdot\sum_{v\in V} c_v = 2(1+2\varepsilon)\cdot\sum_{e\in E} x_e.$$

□

随机舍入给定一个分数匹配，众所周知，可以使用一种简单的随机舍入方法来获得一个整数匹配，其大小在一个常数因子范围内相同（在期望意义下，或者事实上具有随匹配大小呈指数变化的集中性）。我们采用 Ghaffari 等人提出的一种已知方法。 [GGK+18]为确保自包含性，我们概述舍入过程并讨论期望的整数匹配大小，但关于概率集中性论证，请参阅 [GGK+18] 。

引理 4.2 (Ghaffari 等人 [GGK+18])。 存在一个 $O(1)$ 轮算法，给定一个大小为 M的分数匹配 $|M| = \sum_{e\in E} x_e$，以至少 $|M|/50$的概率计算出一个大小至少为$1 - exp(-\Theta(|M|))$。

引理 4.2 的证明概要。每个节点 v 要么以概率 $x_e/10$ 选择其一个邻居 u 并向其发出提议，其中 $e = \{v, u\}$；要么不向任何邻居发出提议，其发生概率为$1 - \sum_{e\ni v} x_e$。如果 v向 u发出了提议，或者 u向 v发出了提议，则边 $e = \{v, u\}$ 会被标记。我们保留那些不与任何其他已标记边关联的已标记边。显然，这一过程可以通过 $O(1)$ 轮消息传递来实现。还容易看出，边 e 被标记的概率至少为 $x_e/10$；在此条件下，不与e 相连的其他任何边被标记的概率为 $1 - (2 \cdot 2/10) = 0.6$ 。这已经表明，匹配的期望大小至少为 $3|M|/50$。Ghaffari 等人 [GGK+18] 通过应用 McDiarmid 不等式证明，该期望附近确实存在指数集中现象，特别是，匹配大小低于$|M|/50$的概率至多为 $exp(-\Theta(|M|))$。

□

4.2 将平均唤醒复杂度改进至 $O(1)$

上述基础算法要求所有尚未冻结的顶点在每一轮都处于活动状态。因此，我们无法直接得到平均唤醒复杂度较低的算法。为了将唤醒复杂度改进至$O(1)$，我们修改算法：在每次迭代中，算法不再考虑所有邻居来精确计算c_v，而只将一小部分顶点视为唤醒状态，并让每个顶点根据其处于唤醒状态的邻居所做的决策来估算 c_v，并据此做出自己的决策。

先来看一种算法初步尝试中的前几轮，以突出其主要思路。初始时，所有边权都设为 $\frac{1}{\Delta}$，与基础算法相同，因此，是否冻结一个节点的决策与估算其度几乎相同。因此，如果以概率 $p_1 = 10\log n\frac{1}{\Delta}$对节点进行采样，那么每个节点 v都能以高概率区分其度是大于$(1-\varepsilon)\Delta$还是小于$(1-20\varepsilon)\Delta$，例如，可以将采样到的邻居数量与 $p_1 \cdot(1-10\varepsilon)\Delta$进行比较。

对于第二轮，我们必须先考虑两种情况来估算节点 v的值：首先， v 的邻居中有些在第一轮被冻结，其与 v 相连的边的值为 $\frac{1}{\Delta}$；另一些邻居仍处于活跃状态，其关联边的权重为 $\frac{1+\varepsilon}{\Delta}$。为了解决这个问题，我们进行两次采样，分别估算这两类邻居的数量。令 S_1 为以概率 $p_1 = 10\log n\frac{1}{\Delta}$ 采样节点得到的集合，令 S_2 为以概率 $p_2 = 10\log n_{(1+\varepsilon)\Delta}$ 采样得到的集合。现在，我们可以估算节点经过两轮后的值：

$$\tilde{c}_v = \frac{|S_1 \cap N(v)|}{p_1} \cdot \frac{1}{\Delta} + \frac{|S_2 \cap N(v)|}{p_2} \cdot \frac{\varepsilon}{\Delta}.$$

这对应于第一轮中处于活跃状态的节点的贡献，以及在第二轮中仍保持活跃状态的节点的贡献。第一轮结束后，活跃节点的最大度只能大致为$\frac{\Delta}{1+\varepsilon}$。因此，采样概率 p_2 被设置为与度成反比，并额外乘以 $10\log n$，以确保估计以高概率（w.h.p.）保持准确。

然而，这种方法有一个关键问题：它仍然无法达到我们所需的唤醒复杂度。一旦继续进行后续轮次，我们就会得到 $p_i = 10\log n\frac{(1+\varepsilon)^i}{\Delta}$，对所有 i 求和后，唤醒复杂度仍为 $O(\log n)$。[3] 要解决这个问题，唯一的办法是降低

[3]只需稍加改进，这个思路实际上就能转化为一个唤醒复杂度为 $O(\log\log n)$ 的算法。

采样概率（尤其是在后续轮次中），这就要求我们分析节点采取与基础算法中不同决策时会发生什么。

对于该算法，我们将分阶段进行，并为每个阶段设定一个采样概率。在第 1 阶段，该概率为 $p_1 = \frac{C}{(\log n)^4}$。只要活跃节点的度为 $(\log n)^5$，，该概率就能保证算法以高概率（w.h.p.）表现得与基础算法相同。对于第 2 阶段，我们将设定 $p_2 = \frac{C}{(\log \log n)^4}$，但这样也会降低我们将节点取值正确估计到 $\mathrm{poly}(\frac{1}{\log n})$ 的概率，只要度至多为 $(\log \log n)^5$。最后一个技术细节是，我们不考虑活跃节点的度，而是使用活跃边权重的倒数来近似度。

完整算法 更一般地，从节点 v 的视角来看，算法如下：我们从第 $j = 0$ 轮开始，此时所有节点均处于活跃状态。令 $w_j = (1+\varepsilon)_{\Delta}^{\ j}$ 表示第 j 轮活跃边的权重。若 i 是满足 $w_j \leq \frac{1}{(\log^{(i)} n)^5}$ 的最小值，则称第 j 轮处于阶段 $i = i(j)$。此外，令 $p_i = \frac{C}{(\log^{(i)} n)^4}$。在第 j 轮，对于每个之前的轮次 $h \leq j$，我们采样一个集合 S_h^j，其中每个节点都以概率 $p_{i(h)}$ 独立地纳入集合。然后，令 $A_h^j \subseteq S_h^j$ 表示同时处于活跃状态的节点集合。需要指出的是，即使 h 的值相同，只要 j 的值不同，我们也会采样不同的集合，以保持各轮之间的独立性。因此，$\left|A_h^j \cap N(v)\right|$ 表示节点 v 的邻居中，在第 h 轮处于活跃状态且被采样到（概率为 $p_{i(h)}$）的节点数。对于第 j 轮，节点 v 现在将值 $\tilde{c}_v$ 估计为

$$\tilde{c}_v = \sum_{h=1}^{j} \frac{|A_h \cap N(v)|}{p_{i(h)}} \cdot (w_h - w_{h-1}).$$

然后，如果 $\tilde{c}_v > (1 - 10\varepsilon)$，我们就冻结顶点 v，也就是说，它在后续迭代中不会处于活跃状态。否则，我们继续下一次迭代。

当到达阶段 i 时，我们停止算法的这一部分，其中 $\frac{1}{\log^{(i)} n} > \varepsilon \cdot \frac{}{\log(1+\varepsilon)\ 1000}$。此时，我们切换到执行基础算法，也就是说，从现在起，在所有后续轮次中，我们都将采样概率设为 1。

算法结束后，也就是所有边都被冻结时，我们按如下方式设置最终边权重：对于边 $e = (u, v)$，令 j 为 u 和 v 仍同时处于活跃状态的最后一轮。然后我们设定 $x_e = (1+\varepsilon)_{\Delta}^{\ j}$。最后，每个顶点都会计算其最终值 $c_v = \sum_{e \ni v} x_e$，我们只保留满足 $c_v \leq 1$ 的顶点。也就是说，对于所有满足 $c_v > 1$ 的顶点，我们将其所有关联边的权重设为 0。

分布式实现 我们从以下角度来理解该算法：在每个分布式轮次k中，每个决定变为活跃的节点 v 都应能够模拟上述算法的所有轮次 $j \leq k$。因此，它需要知道所有 $h \leq j$ 和 $j \leq k$对应的集合 A_h^j。这可以通过递归实现：所有节点$u \in S_h^j$ 都需要知道自己在第 h 轮是否处于活跃状态，因此需要知道自己是否在第 $h-1$ 轮（或更早）被冻结。所以，对于每个 $h' < h$，它们都需要知道所有集合 $A_{h'}^j$，以便做出上述判断；我们通过递归来完成这一点。

因此，分布式实现如下（以自底向上的方式构造递归）：首先，我们对所有集合 S_h^j 进行采样，使得每个节点 v 都知道 j 和 h 的所有取值，且满足 $v \in S_h^j$。在第 0 轮中，所有属于任意 j 对应的 S_0^j 的节点都变为活跃状态。由于它们在第 0 轮开始时均处于活跃状态，因此它们都会判定自己也属于 A_0^j，并在这一轮及之后的所有轮次中，将这一信息连同所有满足该条件的 j 的取值一起发送给自己的所有邻居。同时，在第 0 轮中，所有属于任意 j 对应的 S_1^j 的节点都会监听其邻居中哪些节点属于 A_0^1，并利用这些信息判断自己在算法的第 0 轮中是否被冻结，即是否也属于 A_1^j。从第 1 轮开始，属于任意 A_1^j 的节点随后会将这一信息连同 j 的取值一起发送给自己的所有邻居。更一般地，在第 $h-1$ 轮中，所有满足以下条件的节点 v 都会变为活跃状态并接收有关 $N(v)$ 的信息：存在（至少一个）j，使得 $v \in S_h^j$，其中 $l \leq h'$ 和 $h' < h$。然后，节点 v 利用集合 $A_l^{h'}$ 来模拟算法的第 h' 轮，即判断对于所有 $h' < h$，自己是否在第 h' 轮被冻结；如果在模拟过程中某一轮被冻结，则停止模拟。如果 v 始终未被冻结，那么它在第 h 轮中就是活跃的，因此，对于所有满足 $v \in S_h^j$ 的 j，v 都属于 A_h^j。

总而言之，对于节点 v，令 h 为满足 $v \in S_h^j$ 的最小值。那么， v 从第 $h-1$ 轮开始一直处于活跃状态，直到算法结束。最后，在某个时刻，所有节点都会变为活跃状态；此后，我们只需以分布式方式执行基础算法。

4.3 路线图

在给出正式分析之前，我们先概述并解释证明策略。具体来说，我们会将算法与基础算法进行比较，看看二者有何不同。

估计节点 v 的值时可能出现两种误差；(1) c_v 的值被低估，这意味着即使增加相邻边的权重就能使 v 的值超过 1，某个节点仍可能进入下一轮。(2) c_v 的值被高估，即尽管 c_v 尚未达到值 $1-\Theta(\varepsilon)$，节点 v 仍可能被错误地冻结。这些误差会以不同方式影响最终解的质量：(1) 会破坏输出为有效分数匹配这一性质；(2) 则使得难以论证所计算解的质量。

我们先来关注问题 (1)，并将满足以下条件的节点称为 $c_v > 1$ 重节点。算法中已经通过移除所有重顶点来处理这个问题；然而，在分析中，我们还需要界定这种做法会损失多少值。请注意，仅仅界定重节点的数量是不够的，因为c_v 可能增长到 $\Theta(\Delta)$。因此，我们考察与这些重节点相邻的值，并将这些相邻边上的值标记为受损。相较于考察节点，考察边能让我们更容易界定每轮中额外受损的分数值，因为边上的值至多为 1。我们在引理 4.5 中对此进行了形式化，并用它证明每轮受损的当前分数赋值仅占很小一部分。

对于第二个问题 (2)，我们将那些在 $c_v < 1-\Theta(\varepsilon)$ 轻量条件下被冻结的节点称为轻量节点。相当容易证明，以高概率而言，所有节点中只有很小一部分是轻量节点。然而，这些节点中的每一个都可能阻挡多达 $\Theta(1)$ 的总分数值。例如，如果最大分数匹配的规模仅为 $\sqrt{n}$，这意味着由于轻量节点而损失的分数值将远远大于所计算出的分数分配本身的规模。为了解决这一问题，我们将更仔细地考察一个值为 c_v 的轻量节点过早被冻结的概率，特别是，我们将证明该概率与 c_v 成正比（见引理 4.8）。通过这种方式，我们可以证明，在任意给定轮次中被冻结的轻量节点的期望数量，仅占当前分数分配规模的一小部分（引理 4.9），并且我们将证明，当前分数分配的规模不可能远大于最大分数匹配的规模。

最后，我们将在定理 4.11中整合所有内容，以证明：与命题 4.1的证明中类似的收费论证，可以用来说明算法计算出的分数匹配的大小在期望意义下是最优分数匹配的（$2+\varepsilon$）近似值。对于最大匹配足够大的图，这一结论以高概率也成立。

4.4 theMatching 算法分析

我们先介绍一些预备知识：我们假设 ε 足够小，即 $\varepsilon \leq 1/1000$，因为这会简化一些计算。此外，我们将使用以下定义：

定义 4.3. 考虑算法第 j 轮中仍处于活动状态的节点 v。设其每条关联边 $e=(v,u)$ 的值为 $x_e = w_k$（如果 u 在第 $k<j$ 轮被冻结），如果 u 在第 j 轮仍处于活动状态，则其值为 $x_e = w_j$。我们称 v 在第 j 轮开始时具有（真实）值 $c_v = \sum_{e \ni v} x_e$。

请注意，这等同于在第 j 轮之前停止算法，并计算算法结束时的各项值。根据这个定义，我们现在可以证明，节点变为重节点，或虽为重节点但尚未冻结的概率很小。

引理 4.4. 在阶段 i 的第 j 轮中，值为 $c_v > 1-\varepsilon$ 的节点 v 将其值估计为 $\tilde{c}_v < 1-10\varepsilon$ 的概率至多为 $\frac{1}{1000(\log^{(i-1)} n)^{10}}$。

现在我们可以利用这个结论，界定算法每轮中新标记为损坏的分数值总量。回顾一下，如果边 e 的关联节点 e 是重节点，我们就称该边的分数值已损坏。因此，要界定第 j 轮额外损坏的值，只需考察该轮中变为重节点或仍为重节点的所有节点 j。

引理 4.5。在阶段 i 的每一轮 j 中，值为 $c_v > 1-\varepsilon$ 的节点所增加的额外分数值（即额外损坏的量）至多为 $100\,(\log^{(i)} n)^2 \cdot \max\{(\log n)^{18}, |M_j|\}$，其概率至少为 $1-n^{-9}$。其中，~~M_j 是第~~ j 轮开始时的分数分配。如果该轮处于阶段 $i>1$，我们假设剩余活跃顶点中的度至多为 $(\log n)^5$。

现在，为了利用这一点来陈述算法执行过程中被破坏的分数总量，我们需要确保满足引理 4.5 中的条件。为此，我们给出以下引理。

引理 4.6。在阶段 $i=1$ 之后，由所有活跃节点诱导的图中的度至多为 $(\log n)^5$，概率为 $1-n^{-8}$。

最后，我们现在能够界定整个算法中被浪费的分数值总量。

引理 4.7. 被损坏的总分数值，即与取值为 $c_v > 1$ 的顶点关联的总分数值，至多为 $\varepsilon \max\{(\log n)^{18}, |M|\}$，其发生概率为 $1-n^{-7}$；其中 M 是算法在最终移除重节点之前计算出的分数赋值。

现在我们转而关注轻顶点，也就是被算法过早冻结的顶点。我们的第一个结果表明，顶点不太可能是轻的；事实上，我们甚至证明了一个比重顶点情形更强的结论。我们证明，取值较小的节点 c_v 被过早冻结的概率与 c_v 成正比，也就是说，取值越小，节点被冻结的可能性就越低。

引理 4.8. 值为 $c_v < 1-20\varepsilon$ 的顶点 v 在阶段 i 的第 j 轮被冻结的概率至多为 $c_v \cdot 10^{\frac{1}{(\log^{(i-1)} n)^{10}}}$。

证明。令 X 为表示 $\tilde{c}_v$ 在第 j 轮取值的随机变量。由于 $\tilde{c}_v$ 是 c_v的无偏估计量，因此有 $\mathbb{E}[X] = c_v$。我们希望界定 $\Pr[X > 1-10\varepsilon]$，这恰好是节点 v 被（错误地）冻结的概率。首先，请注意，与 引理 4.4 中的推导相同，我们有以下结论：

$$\begin{aligned}\Pr[X > 1-10\varepsilon] &\le \Pr[|X-\mathbb{E}[X]| \ge 9\varepsilon] \\ &\le \frac{1}{1000(\log^{(i-1)} n)^{10}}\end{aligned}$$

因此，对于 $c_v \ge 1/100$，该结论成立，以下我们假设 $c_v \le 1/100$。注意，我们可以将 X 写成如下形式：

$$X = \sum_{h \le j} \frac{w_h}{p_{i(h)}} \sum_{\substack{u \in N(v) \\ u \text{ active in round } h}} X_{u,h},$$

其中 $X_{u,\,h}$ 是事件“节点 u 在第 h 轮被采样”对应的指示变量。我们可以将节点过早被冻结的概率改写为：

$$\Pr[X > 1-10\varepsilon] = \Pr[X > (1+\frac{1-10\varepsilon-c_v}{c_v})c_v].$$

在下文中，令 $\delta = \frac{1-10\varepsilon-c_v}{c_v}$。注意，由于 $\varepsilon \le 1/1000$和 $c_v \le 1/100$，我们有 $\delta \ge 980$，并且显然 $\delta \le \frac{1}{c_v}$。

为了从定理 2.2 中获得更强的界，我们旨在最大化所分析随机变量 X 的期望值，同时确保随机二元变量的每个系数都小于 1。我们知道，这些系数的形式为 $\frac{w_h}{p_{i(h)}}$，其中 $h \le j$。正如前文所论证的，这至多为 $C^{\frac{1}{(\log^{(i)} n)}}$。

如前所述。因此，我们可以应用 定理 2.2 于 $C(\log^{(i)} n)X$ 并得到

$$\begin{aligned}
&\Pr[C(\log^{(i)} n)X > (1+\delta)\,\mathbb{E}[C(\log^{(i)} n)^5 X]] \\
&\le \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{C(\log^{(i)} n)\,\mathbb{E}[X]} \\
&\le \exp\Big((\delta-(1+\delta)\ln(1+\delta))C(\log^{(i)} n)c_v\Big) \\
&\le \exp\Big((1-\ln(1+\delta))\delta C(\log^{(i)} n)c_v\Big) \\
&\le \exp\Big((-\ln(1+\delta)/2-1)(1/c_v)C(\log^{(i)} n)c_v\Big) \\
&\le c_v\cdot\frac{1}{10(\log^{(i-1)} n)^{10}}.
\end{aligned}$$

□

在下一个引理中，我们将利用这样一个事实：轻节点被冻结的概率与其值成正比，从而证明被冻结的轻节点总数与当前匹配的规模成正比。

引理 4.9。在第 i 阶段的第 j 轮中，尽管具有 $c_v < 1-20\varepsilon$，但被冻结的顶点 v 数量至多为 $^{100}{}_{(\log^{(i)} n)^2}\cdot \max\{(\log n)^{18}, |M_j|\}$ 其成立的概率至少为 $1-n^{-9}$，其中 M_j 是第 j 轮开始时分数配赋的值。如果第 j 轮处于第 $i>1$ 阶段，我们假设剩余活跃顶点之间的度至多为 $(\log n)^5$。

证明。令 X_v 为示性随机变量：当节点 v 尽管具有 $c_v < 1-20\varepsilon$ 仍估计其值 $\tilde{c}_v > 1-10\varepsilon$ 时，该变量取 1。根据引理 4.8，我们知道 $\mathbb{E}[X_v] \le c_v \cdot {}_{10}{}^{1}{}_{(\log^{(i-1)} n)^{10}}$。此外，由于 $\sum_v c_v = 2|M_j|$，我们有：对于 $X=\sum_v X_v$，成立 $\mathbb{E}[X] \le 2\,|M_j|\,{}_{10}{}^{1}{}_{(\log^{(i-1)} n)^{10}}$。根据定义，$X$ 是在第 j 轮中被过早冻结的节点数量。

首先来看阶段 $i=1$中的轮次。在该阶段，每个节点 v 过早冻结的概率至多为 $\frac{1}{10n^{10}}$。因此，根据并集界，图中的任意一个 n 节点过早冻结的概率至多为 n^{-9}。

对于第 $i>1$ 轮中的节点，我们必须更加谨慎地进行分析，因为单个节点的失败概率太小，无法简单地对所有节点应用联合界。相反，我们将尝试使用定理 2.3 并对依赖度进行界定。注意，节点被提前冻结这一事件仅取决于其邻居中哪些节点被采样。因此，任何两个没有共同邻居的节点都是相互独立的。根据我们的假设，活跃顶点之间的度至多为 $(\log n)^5$,，这意味着依赖度为 $\Delta' \le (\log n)^{10}$。

最后一步，我们考虑两种情况：(1) $|M_j| \le (\log n)^{18}$ 和 (2) $|M_j| > (\log n)^{18}$。在第一种情况下，我们将分析 $\Pr[X \ge {}^{100}{}_{(\log^{(i)} n)^2}\cdot(\log n)^{18}]$,，同时我们已知 $\mathbb{E}[X] \le {}_{5}{}^{1}{}_{(\log^{(i-1)} n)^{10}}(\log n)^{18}$。因此，我们有

$$\begin{aligned}
&\Pr[X \ge \frac{100}{(\log^{(i)} n)^2}\cdot(\log n)^{18}] \\
&\le \Pr[|X-\mathbb{E}[X]| \ge \frac{99}{(\log^{(i)} n)^2}(\log n)^{18}] \\
&\le 8((\log n)^{10}+1)\cdot\exp\left(-\frac{99(\log n)^{18}}{3((\log n)^{10}+1)(\log^{(i)} n)^2}\right) \\
&\le 8((\log n)^{10}+1)\cdot\exp\left(-10(\log n)^2\right) \le n^{-9},
\end{aligned}$$

对于情况（2），其中 $|M_j| > (\log n)^{18}$,，论证相同，只不过我们用 $|M_j|$ 代替 $(\log n)^{18}$，并且（最坏情况下）得到相同的失败概率，因为我们有 $|M_j| > (\log n)^{18}$。 □

最后，现在我们能够界定算法结束时轻节点的总数。

引理 4.10. 过早被冻结的顶点数量，即取值为 $c_v < 1 - 20\varepsilon$ 的顶点数量，至多为 $\varepsilon \max\{(\log n)^{18}, |M|\}$，其中 M 是算法计算出的分数赋值，其概率为 $1 - n^{-7}$。

证明。 我们的目标是反复应用 引理 4.9，不过，在阶段 1 之后，我们还需要满足一个条件。我们可以使用 引理 4.6，它指出该条件以至少 $1 - n^{-8}$ 的概率得到满足，因此我们可以假设活跃顶点之间的度至多为 $(\log n)^5$。

如果某一轮满足引理 4.9的陈述，我们就称该轮成功。由于总共至多有 $\log_{1+\varepsilon} n \ll n$ 轮次，所有轮次均成功的概率至少为 $1 - n^{-8}$。

为了了解需要应用引理 4.9的频率，我们还需要知道阶段 i 中的轮次数量。根据以下条件：轮次 j 处于阶段 i，其中 i 是满足 $w_j \leq \frac{1}{(\log^{(i)} n)^3}$，且 $w_j = (1+\varepsilon)_{\Delta}^{\ j}$ 的最小值，我们可知阶段 i 包含少于 $5\log^{(i)} n \log(1+\varepsilon)$ 轮次。

这使我们能够界定过早被冻结的节点总数：

$$\begin{aligned}
&\sum_{j=1}^{\log_{1+\varepsilon}\Delta} \frac{100}{(\log^{(i(j))} n)^2} \cdot \max\{(\log n)^{18}, |M_j|\} \\
&\leq \sum_i \frac{5\log^{(i)} n}{\log(1+\varepsilon)} \cdot \frac{100}{(\log^{(i)} n)^2} \cdot \max\{(\log n)^{18}, |M|\} \\
&= \frac{500}{\log(1+\varepsilon)} \cdot \max\{(\log n)^{18}, |M|\} \sum_i \frac{1}{(\log^{(i)} n)}.
\end{aligned}$$

由于当 $\frac{1}{\log^{(i)} n} > \varepsilon \cdot \log(1+\varepsilon)\, 1000$ 时我们就会停止算法，因此同样可以用以下方式界定该数量：

$$\varepsilon \cdot \max\{(\log n)^{18}, |M|\}.$$

总体而言，失败概率至多为 $n^{-8} + n^{-8} \leq n^{-7}$，由此证明了我们的结论。 □

现在我们可以证明本节的主要结果了。结合引理 4.7 和引理 4.10，我们可以证明，仍然可以使用命题 4.1 中的收费论证，而由于部分节点的行为与基础算法中的不同，只会损失少量分数值。需要指出的是，我们并未尝试优化常数，尤其是匹配大小下界中的指数。

定理 4.11. 设 $\bar{M}$ 为算法计算出的匹配。以概率 $1 - n^6$，该匹配$\bar{M}$是最大（分数）匹配 M^*的一个 $(2 + 100\varepsilon)$-近似，即 $(2 + 100\varepsilon)\left|\bar{M}\right| \geq |M^*|$，适用于满足 $|M^*| \geq \Omega((\log n)^{18})$ 的图。

证明。 首先，注意到 M 确实是一个有效的匹配，因为每个节点的值都是 $c_v \leq 1$，与算法的执行过程无关。令 M 表示算法在删除所有值为 v 的节点 $c_v > 1$之前计算出的分数分配。首先，我们将给出 M 的值关于 M^*的上界。 引理 4.7 指出，与满足 $c_v > 1$ 的节点关联的分数值至多为$|M| - \left|\bar{M}\right| \leq \varepsilon \max\{(\log n)^{18}, |M|\}$，其发生概率为 $1 - n^{-7}$。由于 $\bar{M}$是一个有效的匹配，$\left|\bar{M}\right| \leq |M^*|$。结合以上结果可得

$$|M| - \varepsilon \max\{(\log n)^{18}, |M|\} \leq |M^*|,$$

由此可得，使用 $|M^*| \geq (\log n)^{18}$，

$$|M| \leq (1 + 2\varepsilon)\,|M^*|.$$

根据这一观察，我们现在也可以利用 $|M|$ 和引理 4.10得到其下界，该引理以 $1 - n^{-7}$的概率成立。在算法冻结的所有顶点中，除了 $\varepsilon \max\{(\log n)^{18}$个以外，$|M|\}$ 其余顶点的值都至少为 $1 - 20\varepsilon$。为了使用与基础算法类似的论证，我们可以设想给所有过早被冻结的顶点都增加 1 的分数值，这意味着我们最多增加

$\varepsilon \max\{(\log n)^{18}, |M|\}$ 总共。现在，每条边都至少与一个值不小于$1-20\varepsilon$的端点相接。因此，我们有

$$|M^*| \le \frac{2}{1-20\varepsilon}(|M| + \varepsilon \max\{(\log n)^{18}, |M|\}).$$

对于 $|M^*| = \Omega((\log n)^{18})$ 足够大时，这意味着 $|M| \ge (\log n)^{18},$，从而可以简化

$$|M^*| \le \frac{2(1+\varepsilon)}{1-20\varepsilon}|M| \le (2+50\varepsilon)|M|.$$

最后，一个如前所见，我们有 $|M| \le \frac{1}{1-\varepsilon}\left|\bar{M}\right|$。结合上述结果可得

$$|M^*| \le \frac{(2+50\varepsilon)}{1-\varepsilon}|M| \le (2+100\varepsilon)\left|\bar{M}\right|.$$

失败概率为 $n^{-7} + n^{-7} \le n^{-6}$。 □

备注 4.12。使用类似的论证，也可以证明能够计算出一个期望意义下为 $(2+100\varepsilon)$-近似的分数匹配，且对最大匹配的大小没有下界要求。

在证明该算法能够计算出最大匹配的良好近似解后，我们现在还将证明，它可以在睡眠模型中以较低的唤醒复杂度实现。

推论 4.13. 对于任意图G，若其最大分数匹配的大小至少为 $\Omega((\log n)^{18})$，则我们可以在分布式休眠模型中以 $O(\varepsilon^{-3})$ 的唤醒复杂度和 $O(\log \Delta)$ 的轮复杂度（始终如此），以 $1-n^{-5}$ 的概率计算最大分数匹配的 $(2+100\varepsilon)$-近似。

证明。 注意，近似保证由 定理 4.11 以概率 $1-n^{-6}$成立，因此我们只需证明该陈述的第二部分，即休眠模型中的复杂度。对于轮复杂度，由于当边权 w_j 为 1 时，算法会立即停止，因此轮复杂度上界为 $\log_{1+\varepsilon}\Delta = O(\log \Delta)$。唤醒复杂度分为两部分进行分析：首先，我们考虑节点在采样概率为 $p_i < 1$ 的阶段中保持唤醒所产生的唤醒复杂度；然后还需对仅模拟基础算法的轮次数量给出上界。

对于第一部分，我们知道，对于每个集合 S_h^j，其中 $h \le j$中的节点在第 $h-1$轮变为活跃状态，并在算法的剩余过程中始终保持活跃。如果 h 处于阶段 i，则集合 S_h^j 的期望大小为 $p_i \cdot n$，因此我们可以利用定理 2.1 得出结论：以至少 n^{-10} 的概率，有 $\left|S_h^j\right| \le 2 \cdot p_i \cdot n$。此外，以至少 n^{-9} 的概率，这一结论对所有轮次和所有集合 S_h^j 均成立。注意，从阶段 i 开始到算法结束，最多有 $^5\log^{(i)} n \log_{(1+\varepsilon)}$ 轮。因此，对于阶段 $i = i(h)$ 中固定的 h 值，集合 S_h^j ~~的数量~~最多为 $^5\log^{(i)} n \log_{(1+\varepsilon)}$。对于这些集合中的每一个，所有节点最多活跃 $10\log^{(i)} n \log_{(1+\varepsilon)}$ 轮。最后还需注意，阶段 i 中最多也有 $^5\log^{(i)} n \log_{(1+\varepsilon)}$ 轮。综合以~~上所有结~~果，我们得到的总唤醒复杂度为

$$\sum_i \frac{250(\log^{(i)} n)^3}{(\log(1+\varepsilon))^3} \cdot \frac{2C \cdot n}{(\log^{(i)} n)^4} = \frac{500 \cdot C \cdot n}{(\log(1+\varepsilon))^3} \sum_i \frac{1}{\log^{(i)} n}$$
$$\le \frac{500 \cdot C \cdot n}{(\log(1+\varepsilon))^3} \cdot 2 \cdot \varepsilon \cdot \frac{\log(1+\varepsilon)}{1000} = O(n/\varepsilon^3).$$

因此，该部分导致的平均唤醒复杂度为 $O(\varepsilon^{-3})$。

对于第二部分，当我们即将开始阶段 i 时就停止，其中 $^1\log^{(i)} n > \frac{\varepsilon \cdot \log(1+\varepsilon)}{1000}$。~~与之~~前一样，注意从阶段 i 开始，直到我们得到权重 $w_j \ge 1$ 为止，最多有 $^5\log^{(i)} n \log_{(1+\varepsilon)}$ 轮次。我~~们可以将~~其改写为

$$\frac{5000}{\varepsilon(\log(1+\varepsilon))^2} = O(\varepsilon^{-3}).$$

因此，整个算法的平均唤醒复杂度为 $O(\varepsilon^{-3}) + O(\varepsilon^{-3}) = O(\varepsilon^{-3})$，总体失败概率为 $n^{-6} + n^{-9} \le n^{-5}$。

□

4.5 顶点覆盖

在本节中，我们考虑顶点覆盖问题，它是前面讨论过的匹配问题的对偶问题。我们首先证明，使用基础算法也可以获得最小顶点覆盖的 $(2+\varepsilon)$-近似解，然后给出低唤醒复杂度算法的相应证明。

命题 4.14. 设 C^* 是 G中的一个最小规模顶点覆盖，并设 C 为基础算法中所有被冻结的顶点的集合。那么，C 是一个有效的顶点覆盖，并且我们有 $|C| \leq (2+4\varepsilon)\,|C^*|$。

证明。首先，我们来论证 C 确实是一个顶点覆盖。由于算法结束时，每条边都至少有一个冻结端点，因此所有冻结顶点构成的集合是一个顶点覆盖。接下来，为了证明近似保证，我们采用与匹配情形类似的收费论证。我们在 C中的每个顶点 v 上放置一美元。然后，按照各条关联边在算法结束时的分数值 x_e 所占比例，将这笔钱分配给所有关联边。因此，每条这样的边从 v处获得的金额至多为 $\frac{1}{1-\varepsilon}x_e$ 美元，因为 v 是一个满足 $c_v \geq 1-\varepsilon$的顶点。由于一条边可以从其每个端点获得这笔金额，因此一条边总共获得的金额至多为 $\frac{2}{1-\varepsilon}x_e$ 美元。现在，我们将所有边上的金额转移到其属于 C^*的端点上，并任意处理平局。由于我们知道每个顶点都有 $c_v \leq 1$，因此 C^* 中的每个顶点通过这种方式获得的金额至多为 $\frac{2}{1-\varepsilon}$ 美元。因此，我们有$|C| \leq (2+4\varepsilon)\,|C^*|$。

□

定理 4.15。令 C 表示算法在 第 4.2 节中冻结的所有顶点组成的集合。那么， C 是一个顶点覆盖，并且以概率 $1-n^{-6}$ 它是最小顶点覆盖的一个 $(2+100\varepsilon)$-近似C^*，适用于满足 $C^* \geq \Omega((\log n)^{18})$ 的图。

证明。首先注意， C 是一个有效的顶点覆盖，因为每条边都与至少一个已冻结的顶点关联。由于 C^*是一个最小顶点覆盖，因此对于最大匹配 M^*，有 $|C^*| \geq |M^*|$和 $|C^*| \leq 2\,|M^*|$。注意，这也意味着 $|M^*| \geq \Omega((\log\ n)^{18})$。令 M表示算法在最后一步移除所有满足 $c_v > 1$ 的重节点 v 之前计算出的分数赋值。与定理 4.11 的证明中一样，我们有 $|M| \leq (1+2\varepsilon)\ |M^*| \leq (1+2\varepsilon)\ |C^*|$，其成立概率为 $1-n^{-7}$。利用引理 4.10 可知，以相同的概率，最终取值为 $c_v < 1-20\varepsilon$ 的节点数至多为 $\varepsilon\,|M| \leq 2\varepsilon\,|C^*|$。此外，根据引理 4.7，以概率 $1-n^{-7}$，关联到取值为 $c_v > 1$ 的节点的分数值总量至多为 $\varepsilon\,|M| \leq 2\varepsilon\,|C^*|$。因此，总失败概率为 $3n^{-7} \leq n^{-6}$。

现在我们可以像基本情形中那样，进行同样的收费论证。我们在 C 中的每个顶点上放置 1 美元。对于所有具有 $c_v \leq 1-20\varepsilon$ 的轻顶点，我们各移除这 1 美元。正如之前所论证的，这至多会移除 $2\varepsilon\,|C^*|$ 美元。然后，我们按照关联边 x_e 的权重比例，分配所有剩余的美元。由于对于所有这样做的顶点都有 $c_v \geq 1-20\varepsilon$，因此每条边获得的值至多为$\frac{2}{1-20\varepsilon}x_e$，因为它可以从两个端点各获得这一数值。现在，我们想再次将这些美元重新分配给位于 C^* 中的顶点，但我们可能会遇到具有 $c_v \gg 1$ 的节点。和一开始一样，我们移除关联于这些节点且带有 $c_v > 1$ 的所有部分数值。这意味着我们至多移除 $\frac{2}{1-20\varepsilon}2\varepsilon\,|C^*|$ 美元。移除这些顶点后，我们现在可以将剩余的美元转移给 C^* 的顶点，使得 C^* 中的每个顶点至多获得 $\frac{2}{1-20\varepsilon}$ 美元。总而言之，这得出以下不等式：

$$|C| - 2\varepsilon\,|C^*| - \frac{4}{1-20\varepsilon}\varepsilon\,|C^*| \leq \frac{2}{1-20\varepsilon}\,|C^*|\,,$$

这意味着

$$|C| \leq \frac{2+6\varepsilon}{1-20\varepsilon}\,|C^*| \leq (2+100\varepsilon)\,|C^*|\,.$$

□

与匹配类似，该算法可以在分布式休眠模型中实现，从而得到以下结果：

推论 4.16。对于任意图 G，若其最小顶点覆盖大小为 $\Omega((\log n)^{18})$，则我们可以在分布式休眠模型中以唤醒复杂度 $O(\varepsilon^{-3})$，以 $1-n^{-5}$ 的概率计算最小顶点覆盖的 $(2+100\varepsilon)$-近似，并达到确定性轮复杂度 $O(\log \Delta)$。

4.6 最大匹配的改进近似算法

在本节中，我们提出一种通用的黑盒方法，通过常数次调用前一种算法，将任何能够为最大匹配问题计算常数近似的算法，转化为一种能够计算（$1+\varepsilon$）-近似的算法，其中 $\varepsilon > 0$ 为任意常数。就方法而言，我们的归约受到 Goldberg 等人的经典工作的启发 [GPV93] ，该工作提出了一种用于计算最大匹配的次线性时间并行算法。

此外，McGregor [McG05], 提出了一种算法，它能够以某种类似的方式，将最大匹配的（$1+\varepsilon$）近似归约为极大匹配问题（该算法主要针对半流式模型提出，但也可以适配到分布式计算模型中）。这种归约是随机化的（即使在二分图中也是如此），并且需要（$1/\varepsilon$）$^{O(1/\varepsilon)}$ 次调用。然而，这种归约并不能直接满足我们的设定，因为我们没有平均唤醒复杂度为 $O(1)$ 的极大匹配算法。我们认为，可以扩展 McGregor [McG05] 的归约，从而得到一种将（$1+\varepsilon$）近似归约为最大匹配的常数近似的替代方案。此外，在我们论文的会议版本发表之后，Fischer、Mitrovi´c 和 Uitto [FMU21] 在其已发表工作的更新版本中也提出了这样一种归约 [FMU22]。他们的归约具有更优的 ε多项式$(1/\varepsilon)$ 依赖关系，并且在一般图中也能确定性地运行。

不过，我们发现，与上述两项结果相比，我们的归约要简单得多。在二分图中，我们的归约是确定性的，并且只需要调用常数近似子程序 poly$(1/\varepsilon)$ 次。在一般图中，归约变为随机化过程，需要调用 $2^{O(1/\varepsilon)}$ 次。

在以完全一般的形式陈述算法之前，我们先作两个简化假设：黑盒算法计算一个极大匹配，且输入图是二分图。稍后我们将讨论如何去除这些假设。

定义 4.17. 给定一个图 G 和一个匹配 M，若路径 $P=(v_0,v_1,\ldots,v_{2\ell+1})$ 满足以下条件，则称其为相对于 M 的增广路径：对每个 $1\le i\le \ell$，边 $(v_{2i-1},v_{2i})\in M$，并且 v_0 和 $v_{2\ell+1}$ 都是自由顶点，即不存在顶点 u,w 使得 $(v_0,u)\in M$ 或 $(v_{2\ell+1},w)\in M$。

增广路径是最大匹配算法中的标准工具之一，其历史可以追溯到 Berge 定理 [Ber57], ，自 Hopcroft 和 Karp 的开创性工作以来，它便得到了广泛应用 [HK73]。这种路径之所以得名，是因为可以利用它增大M的规模，或称增广 M。具体而言，如果 P上的边依次为 e_1，e_2，……，$e_{2\ell+1}$，其中 $e_{2i}\in M$对于 $1\le i\le \ell$，那么我们可以从 e_{2i} 中移除所有边 M，并添加所有边 e_{2j+1}，对于 $0\le j\le \ell$，从而得到一个匹配 M'，其$|M'|=|M|+1$。

Hopcroft–Karp 算法 [HK73] 中使用的增广路径还有另一项性质：若考虑一个图 G 和一个匹配 M，使得不存在长度至多为 $2\ell-1$的增广路径。设 S 是一组互不相交、长度为 $2\ell+1$的增广路径，且 S 是极大的，即无法在不破坏任意两条路径互不相交这一条件的情况下，再向其中加入长度为 $2\ell+1$ 的增广路径。随后，沿 S中的所有路径进行增广，可得到一个匹配，使得不存在长度至多为 $2\ell+1$ 的增广路径。因此，任意增广路径的最小长度都增加了，这将作为我们衡量进展的标准。接下来，我们将结合以下结果，从而界定所求匹配的质量。

引理 4.18 (Hopcroft, Karp [HK73])。设 M^* 是 G 中的一个最大匹配，且 M 是一个匹配。如果有 $|M^*|>(1+\varepsilon)\,|M|$则存在一条长度至多为 $4\varepsilon^{-1}+1$的增广路径。

证明。考察图 $G'=(V(G),\ M^*\oplus M')$，它是 M^*和 M的对称差。注意，G' 包含孤立顶点、偶长度的环，以及奇长度和偶长度的路径。此外，注意任何奇长度的路径都必定是增广路径，而其他所有结构中，M 和 M^*的边数相同。还要注意，沿着所有这些路径进行增广会使 M 变成一个与 M^*大小相同的匹配，而沿每条这样的路径进行增广都会使 M 的大小增加一。因此，这样的增广路径至少有 $|M^*|-|M|>\varepsilon/2\,|M^*|$条。反设所有增广路径的长度都大于 $4\varepsilon^{-1}+1$。那么，每条增广路径都至少包含$2\varepsilon^{-1}$条来自 $|M|$的边。由于这样的路径至少有 $\varepsilon/2\,|M^*|$条，这将意味着 $|M|>|M^*|$，矛盾。因此，必定存在一条长度为 $4\varepsilon^{-1}+1$ 的增广路径。

□

二分图 我们首先给出适用于二分图的算法。总体思路是从一个匹配开始，并逐步改进它。第一步，我们尝试使其成为极大匹配，即沿着所有长度为 1 的路径进行增广。对于后续步骤，我们将从一个不存在长度至多为 $2i-1$ 的增广路径的图，推进到一个不存在长度至多为 $2i+1$ 的增广路径的图。我们首先假设可以使用一种计算极大匹配的算法，因为这样可以简化部分描述和论证，同时保留主要思路。

引理 4.19. 假设存在一个算法 $\mathcal{A}$，能够在二分图中计算极大匹配。再假设给定一个二分图 $H=(L\cup R,E)$，以及其中的一个匹配 M，使得不存在长度为 $\leq 2i-1$ 的增广路径，其中 $i\leq O(1/\varepsilon)$。那么，通过调用 $\mathcal{A}$ 至多 $\Theta(\varepsilon^{-3})$ 次，我们可以计算出 H 的一个诱导子图 H'，以及一个由长度为 $2i+1$ 的增广路径组成的集合 S；该集合在 H' 中是极大的，并且满足 $|V(H)|-|V(H')|\leq\varepsilon^2|M|$。

证明。我们将分 3 步进行证明：首先说明如何找到 H 的一个子图，以捕获所有相关信息；然后讨论算法如何在该子图上运行；最后分析算法为何能够达到所述效果。

层图 首先，我们按如下方式构建一个层图：Layer L_0 包含 L中的所有未匹配顶点。Layer L_1 包含 L_1中顶点的所有邻居。然后，层 L_2 包含所有通过一条边从 M 连接到 L_1中某个顶点的顶点。接着，我们以相同的方式创建层 L_3，但只考虑尚未包含在任何层中的顶点，依此类推。我们在层 L_{2i+1}处停止，在该层中，我们仅包含来自 R 且不与 M中的任何边关联的顶点。

定义同一个图的另一种方式如下：首先，将所有不属于M 的边从 L 定向到 R，并将 M 中的所有边从 R 定向到 L。然后，根据这一方向，同时从 L 中的所有未匹配顶点出发，执行 $2i+1$ 步广度优先搜索（BFS）。接着，删除所有不位于相邻层之间的边、在这 $2i+1$ 步过程中未被添加到任何层的顶点，以及所有被添加到 L_{2i+1} 但已匹配的顶点。

算法的其余部分仅在此分层图上执行。注意，任意长度为 $2i+1$ 的增广路径 $P=(v_0,\ldots,v_{2i+1})$ 都包含在这一分层结构中，并满足 $v_j\in L_j$。这是因为一方面，按照构造，P 必须包含在分层图中；另一方面，若假设任意 $v_j\notin L_j$，那么我们也有 v_{2i+1} 位于 L_{2i+1} 之前的某一层，这意味着存在一条长度至多为 $2i-1$ 的增广路径。因此，我们知道只需考虑符合分层图结构的增广路径。

分层图上的增广算法 增广算法的工作过程如下：我们从路径集合 S_0 开始，它包含所有从 L_0 的顶点出发、长度为零的路径。第一步，我们计算 S_0 与其所有邻接顶点之间的匹配，这恰好是 L_1。我们将所有已匹配顶点对应的路径延伸两跳：先到其匹配的邻居，再沿着 M 中连接该邻居与 L_2 的边继续前进。所有未匹配的顶点都会从算法的后续处理中移除。在之后的步骤中，过程几乎相同：每一步 j，我们都通过计算 S_j 的端点与其所有出邻居之间的匹配，尝试延伸这些路径。不过，如果长度为 >1 的路径未能匹配，我们不会将其删除，而是“回溯”并停用其最后两个顶点，也就是说，这两个顶点在算法余下的过程中不再参与。随后，我们将该路径加入下一次迭代的路径集合 S_{j+1}。所有匹配成功的路径都会像第一步中那样延伸，并加入 S_{j+1}。如果一条路径到达 L_{2i+1}，即最后一层，它就会停止参与算法，也就是说，它变为非活动状态。总的来说，我们执行此过程 $\Theta(\varepsilon^{-3})$ 次迭代。最后，我们将所有到达 L_{2i+1} 的路径组成集合并作为 S 返回，同时移除仍处于活动状态的路径上的所有顶点，从而得到子图 H'。不过，算法运行期间因回溯而停用的所有顶点仍然属于 H'。

分析 我们声称，经过 $\Theta(\varepsilon^{-3})$ 次迭代后，最多剩下 $\varepsilon^3|M|$ 条活跃路径。我们将证明，最多会发生 $6|M|$ 次更新（在第一次迭代之后），且每条活跃路径在每次迭代中至少会触发一次更新。注意，在第 L_1 至第 L_{2i} 层中，最多有 $2|M|$ 个顶点，因为这些层中的所有顶点都与来自 M 的边相 incident。此外，这些顶点每个最多更新两次：一次是在其被加入路径时，另一次是在其可能从路径中删除时。剩余的 $2|M|$ 是由于

从 L_0 中的顶点开始，第一次迭代后最多剩下 $|M|$ 个，之后每个顶点最多只会被删除一次；以及从 L_{2i+1} 中的顶点开始，只有当路径变为非活跃时才会更新，而每条路径最多只会发生一次这种情况，因此最多有 $|M|$ 条路径会触发更新。假设经过 $10\varepsilon^{-3}$ 次迭代后，活跃路径超过 $\varepsilon^3|M|$ 条。每条路径在每次被更新的迭代中至少触发了一次更新，因此总共会有 $10|M|$ 次更新，正如我们之前论证的那样，这是不可能的。由于每条活跃路径最多包含 $2i-1 \le 1/\varepsilon$ 个顶点，移除所有活跃路径最多会移除 $\varepsilon^2|M|$ 个顶点。

最后，我们需要证明 S 是 H' 中增广路径的极大集合。这个极大性条件的另一种表述是：证明移除 S 中路径上的所有顶点后，会使 H' 中的第一层和最后一层（即 L_0 和 L_{2i+1}）断开。我们将通过证明算法的每一步（除第一步外）都满足以下两个性质来证明这一点：(1) 从 S_j 中移除这些路径后，会使 H 中的 L_0 层和 L_{2i+1} 层断开；(2) 对于截至第 j 步（包括该步）已删除的所有边 $e = (v_k, v_{k+1}) \in M$，在 $H \setminus S_j$ 中不存在从 v_{k+1} 到 L_{2i+1} 的路径。先来看算法的第一步。由于我们计算了 L_0 与 L_1 之间的极大匹配，移除该匹配的端点后，L_0 与 L_1 将断开。第二个性质显然成立，因为此时还没有这样的边被删除。

因此，假设算法执行步骤 j 后，性质 (1) 和 (2) 成立，我们要证明执行步骤 $j+1$ 后它们仍然成立。我们先证明性质 (2)，由此即可推出性质 (1)。考虑算法在步骤 $j+1$ 中删除的一条边 $e = (v_k, v_{k+1}) \in M$。对于层 L_{k+2} 中 v_{k+1} 的每个邻居 u，v_{k+1} 未与 u 匹配有几种原因。如果 u 已经属于一条活动路径，或者在本轮中已与另一条活动路径匹配，那么性质 (2) 显然成立。如果 u（连同其来自 M 的关联边）已在先前的一次迭代中被删除，那么我们可以直接使用归纳假设，也能立即得到性质 (2)。因此，剩下的情况是 u 也因回溯而在迭代 $j+1$ 中被删除。不过，此时我们可以将边 $(u, w) \in M$ 归咎于 v_{k+1} 未在步骤 $j+1$ 中匹配。注意，e 可以归咎于多条边。如果在解释 w 为何未被延伸时出现类似情况，我们也将这条新边归咎于它。注意，在每一步归咎中，我们都会向前推进一层；最终，在层 L_{2i+1} 中，我们无法再将任何边归咎于 M，因为 L_{2i+1} 中的所有顶点都未匹配。因此，这个过程必定会终止，否则我们就会得到一条可以延伸到 L_{2i+1} 的活动路径，这与我们所求得匹配的极大性矛盾。接下来证明性质 (1)：对于所有已延伸的路径，该性质显然仍然成立；对于所有其他路径，我们刚刚已经证明了性质 (2)，因此集合 S_{j+1} 仍然将 L_0 与 L_{2i+1} 分隔开。 □

现在我们可以将这一论证推广到这样一种情形：我们只有一个能够计算最大匹配的常数近似、而无法计算极大匹配的算法。首先，我们将介绍算法可以具有的另一种近似保证概念。

定义 4.20。如果匹配 M 在 G 中满足 δ-极大，则称其为 δ-极大匹配，即 $G \setminus V(M)$ 的最大匹配大小至多为 $\delta \cdot |M|$。

换言之，这意味着从图 G 中移除匹配 M 后，得到的图不包含较大的匹配。给定一个用于计算最大匹配的任意常数近似算法，我们也可以轻松地得到这样一个 δ-极大匹配。

引理 4.21. 给定一个计算最大匹配的 c-近似的算法 $\mathcal{A}$，我们可以使用至多 $O(1/c \log 1/\delta)$ 次调用 $\mathcal{A}$，找到一个 δ- 极大匹配 M。

证明。我们进行了 $I = \Theta(1/c \log 1/\delta)$ 次迭代。在每次迭代中，我们在由剩余节点诱导的子图上应用算法 $\mathcal{A}$，将其计算出的匹配加入输出匹配，并从图中移除所有已匹配的顶点，然后继续下一次迭代。设 $|M^*|$ 为原图中最大匹配的大小。在第一次迭代中，算法 $\mathcal{A}$ 找到一个大小为 $|M^*|/c$ 的匹配，因此，由剩余节点诱导的子图中的最大匹配大小至多为 $(1-1/c)|M^*|$。在第二次迭代中，算法 $\mathcal{A}$ 找到一个匹配，其大小至少为第一次迭代后剩余图中最大匹配的 $1/c$ 倍。因此，第二次迭代后剩余图中的最大匹配大小至多为 $(1-1/c)^2|M^*|$。类似地，通过简单的归纳可知，经过 i 次迭代后，第 i^{th} 次迭代后剩余图中的最大匹配大小至多为 $(1-1/c)^i|M^*|$。设 M 为在 I 次迭代中计算出的输出匹配。于是我们知道 $|M| + (1-1/c)^I|M^*| \ge |M^*|/2$，因为输出匹配与剩余图中的一个最大匹配构成整个图的一个极大匹配，因此其大小

至少为 $|M^*|/2$。我们得出 $|M| \geq (1/2 - (1-1/c)^I)|M^*| \geq |M^*|/3$。因此， M 是一个 δ-极大匹配，因为 $((1-1/c)^I \cdot |M^*|)/(|M^*|/3) = 3(1-1/c)^I \leq 3e^{-\Theta(\log 1/\delta)} \leq \delta$。 □

利用这一 δ-极大性概念，我们现在可以将结果扩展到计算最大匹配 c-近似的算法。

引理 4.22. 假设存在一个算法 $\mathcal{A}$，可在二分图中计算最大匹配的 c-近似。另给定一个二分图 $H = (L \cup R, E)$，以及 H 中的一个匹配 M，且不存在长度为 $\leq 2i-1$ 的增广路径。那么，通过调用 $\mathcal{A}$ 至多 $O(1/c \cdot \varepsilon^{-3} \cdot \log 1/\varepsilon)$ 次，我们可以计算出 H 的一个子图 H'，以及一组长度为 $2i+1$ 的增广路径 S，且该路径组在 H' 中是极大的。此外，我们有 $|V(H)| - |V(H')| \leq \varepsilon^2 |M|$，并且图 $H'' = (V(H') \; E(H) \setminus E(H'))$ 不包含大小为 $\varepsilon^2 |M|$ 的匹配。

证明。 首先，我们将使用 引理 4.21 ，并假设我们使用的不是算法 $\mathcal{A}$ ，而是算法 $\mathcal{A}'$，该算法计算一个 δ-极大匹配。这只会使调用算法 $O(1/c \cdot \log 1/\delta)$ 的次数增加 $\mathcal{A}$的开销。寻找增广路径的算法与 引理 4.19中的相同，但我们计算的是一个 δ-极大匹配，而不是一个（完全）极大匹配。在计算出一个匹配后，我们移除所有连接未匹配顶点的边（注意，在极大匹配的情况下，不会有边被移除）。在算法结束时，我们执行与 引理 4.19中相同的操作，但 H' 不再是诱导子图，因为我们还移除了一些连接未匹配节点的边。

注意， H'' 包含所有因其端点未匹配而被移除的边。在算法的每一步中，该边集中的最大匹配至多为 $\delta |M|$，因此在所有迭代中， H'' 中的匹配大小最多为 $\Theta(\varepsilon^{-3})\delta |M|$，而对于 $\delta = \Theta(\varepsilon^5)$，该值至多为 $\varepsilon^2 |M|$。

我们有 $|V(H)| - |V(H')| \leq \varepsilon^2 |M|$ 仍然与极大匹配的情况相同，因此我们唯一需要论证的是，所计算出的集合 S 在 H'中是极大的。请注意，这一点仍然成立，因为所有可用于扩展所计算匹配、使其成为极大匹配的边都已被移除，因此 L_0 和 L_{2i+1} 仍然不连通。

我们在每次 $\Theta(\varepsilon^{-3})$迭代中计算一个 δ 极大匹配，每次都会调用 $\mathcal{A}$ $O(1/c \cdot \log 1/\varepsilon)$ 次。因此，该算法总共会被调用 $O(1/c \cdot \varepsilon^{-3} \cdot \log 1/\varepsilon)$ 次。 □

最后，我们现在可以反复利用这一点，寻找长度为 $2i+1$ 的增广路径，对其进行增广，然后继续处理不包含任何长度至多为 $2i+1$ 的增广路径的图。我们不断重复这一过程，直到不存在长度为 $5\varepsilon^{-1}$ 的增广路径；此时根据 引理 4.18 ，我们知道所计算出的匹配必定是一个 $(1+\varepsilon)$-近似。

定理 4.23。假设给定一个算法 $\mathcal{A}$ ，该算法计算二分图中最大匹配的 c近似。又假设给定一个二分图 H。那么，我们可以在 H 中找到最大匹配的 $(1+3\varepsilon)$近似，且最多调用 $\mathcal{A}$ $O(1/c \cdot \varepsilon^{-4} \cdot \log 1/\varepsilon)$ 次。

证明。我们从一个空匹配 M_0 开始，并在 $H_0 = H$ 上使用引理 4.18 中的算法（该算法又使用$\mathcal{A}$），以便在子图 H_1中找到一组长度为 1 的增广路径。我们沿着所有这些路径进行增广，即将这些边添加到 M_0 中，从而得到 M_1。根据引理 4.18 的保证，增广之后， H_1中不再存在长度为 1 的增广路径，因此我们继续进行下一次迭代。我们再次调用引理 4.18 中的算法，这次在图 H_1上，并设置 $i = 1$。这将得到另一个子图 H_2 以及一组增广路径。我们再次沿着所有这些路径进行增广，根据这些边相对于 M_1的归属翻转其边，从而得到 M_2。这样， H_2中将不再存在长度为 3 的增广路径，我们继续执行这一过程，总共进行 $\ell = 2\varepsilon^{-1} + 1$ 次迭代。此时，我们在子图 H_ℓ 中得到一个匹配 M_ℓ ，并且相对于 M_ℓ， H_ℓ 中不存在长度为$2\ell - 1 = 4\varepsilon + 1$ 的增广路径。根据引理 4.18，我们知道 M_ℓ 是 H_ℓ中最大匹配的 $(1+\varepsilon)$近似；但是，我们能对 H中 M_ℓ 的近似质量作何判断呢？我们设置 $M = M_\ell$，并将证明 M同样是 H中最大匹配的 $(1+7\varepsilon)$近似。

每一步中，我们移除了 $\varepsilon^2 |M|$ 个顶点，因此在全部 $\ell = 2\varepsilon^{-1} + 1$ 次迭代中，我们从 H 中总共最多移除 $3\varepsilon |M|$ 个顶点，从而得到 H_ℓ。不过，我们也移除了一些边。每次迭代中，我们都会移除一组边，其中不包含大小超过 $\varepsilon^2 |M|$ 的匹配。因此，所有这些边的并集中的匹配大小至多为 $\ell \cdot \varepsilon^2 |M| \leq 3\varepsilon |M|$。所以，$H$ 中的最大匹配 M^* 的大小至多为 $|M^*| \leq (1+7\varepsilon) |M|$。

对于算法 $\mathcal{A}$ 的调用次数，每次迭代会产生 $O(1/c \cdot \varepsilon^{-3} \cdot \log 1/\varepsilon)$ 次调用，因此所有迭代总共会产生 $O(1/c \cdot \varepsilon^{-4} \cdot \log 1/\varepsilon)$ 次调用。 □

一般图 我们现在将展示如何使用二分图算法来计算一般图中的改进匹配。核心思路是随机将图划分为两个部分，然后仅使用跨越该划分的边来尝试增广匹配。

定理 4.24. 假设我们有一个算法 $\mathcal{A}$，该算法给定一个二分图 H 后，可以在 H中计算最大匹配的 c-近似。那么，对于任意图 G，我们可以在期望意义下找到一个最大匹配的 $(1+\varepsilon)$-近似匹配，且最多使用 $2^{O(1/\varepsilon)}$ 次 $\mathcal{A}$调用。

证明。为了计算初始匹配 M_0，我们随机将 G 的顶点划分为两部分，删除同一划分内的所有边，得到所得图 H，并使用 $\mathcal{A}$ 在 H 上计算所得图中最大匹配的 c-近似。注意，考察一个最大匹配 M^*时，M^*中的每条边以 $1/2$的概率保留在 H中，因此 H中最大匹配的期望大小至少为 $|M^*|/2$。因此，在期望意义下，M_0至少是 G中最大匹配的 $2c$-近似。

现在我们按照迭代 $i = 1, 2, \ldots$ 的顺序进行，其中在第 i 次迭代中，我们从一个匹配 M_i 开始。假设 M_i 尚未达到 $(1+\varepsilon)$-近似，即我们有 $|M^*| - |M_i| > \varepsilon/2\,|M^*|$。我们将 G 随机划分为两个集合 L 和 R，并删除所有位于同一划分内的边。此外，我们还删除 M_i 中所有位于同一划分内的边的端点。令 H 为所得图。首先，注意，无论我们如何改变 H 中的匹配，我们始终可以将其添加回 G。这是因为我们删除了所有位于同一划分内的匹配边的端点，因此这些顶点无法在 H 中进行匹配。

我们现在将论证 H 具有足够大的改进潜力：由于 M_i 不是一个 $(1+\varepsilon)$-近似解，根据与引理 4.18 类似的论证，我们可知至少存在 $\varepsilon/4\,|M^*|$条互不相交的增广路径，其长度至多为 $\ell = 8\varepsilon^{-1} + 1$。这些路径中的每一条都以至少 $2^{-\ell}$的概率得以保留，即其所有边都跨越划分 $L \cup R$。因此，从期望上看，我们至少保留了 $\varepsilon/4 \cdot 2^{-\ell}\,|M^*|$ 条长度至多为 ℓ的增广路径。令 $M_H \subseteq M_i$为 M_i中在 H中得以保留的子集，并令 M_H^*为 H中的一个最大匹配。注意，从期望上看，$M_H^* \geq M^*/2$。由于对于每一条被保留的增广路径，我们都可以将 M_H 的大小增加 1，因此我们有（从期望上看）$|M_H| + \varepsilon/4 \cdot 2^{-\ell}\,|M^*| \leq |M_H^*|$。现在，我们可以利用定理 4.23 在 H中计算 M_H^*的一个 $(1+\varepsilon/8 \cdot 2^{-\ell})$- 近似解，并将其记为 M'_H。将其与 M_H 进行比较，我们从期望上得到

$$\begin{aligned} |M'_H| - |M_H| &\geq (1 - \varepsilon/8 \cdot 2^{-\ell})\,|M_H^*| - (1 - \varepsilon/4 \cdot 2^{-\ell})\,|M_H^*| \\ &\geq \varepsilon/8 \cdot 2^{-\ell}\,|M_H^*| \geq \varepsilon/16 \cdot 2^{-\ell}\,|M^*|\,. \end{aligned}$$

这意味着，假设匹配 M_i 不是一个 $(1+\varepsilon)$-近似算法，我们就能计算出一个匹配M_{i+1}，其规模比 M_i大 $\varepsilon/16 \cdot 2^{-\ell}\,|M^*|$。

这种情况最多只能发生 $2^{O(1/\varepsilon)}$ 次，因为此后我们就会得到一个规模大于M^*的匹配。因此，在 $2^{O(1/\varepsilon)}$ 次迭代内，我们一定能得到一个匹配 M，其期望近似比为 $(1+\varepsilon)$，对应于 M^*。每次迭代中，我们执行该算法 $\mathcal{A}$ $2^{O(1/\varepsilon)}$ 次；总共有 $2^{O(1/\varepsilon)}$ 次迭代，因此调用 $\mathcal{A}$ 的总次数为 $2^{O(1/\varepsilon)}$。 □

备注 4.25。如果我们使用的不是确定性算法来计算最大匹配的 c-近似，而是使用一个能以期望意义或高概率输出 c-近似的算法，那么这种增广方法同样适用。

4.7 总结

定理 1.2 分为两部分，一部分关于匹配，另一部分关于顶点覆盖。关于顶点覆盖的证明见 定理 4.15。关于匹配，还需要两个步骤。首先，该算法只能计算分数匹配。要得到整数匹配，我们可以使用 引理 4.2，从而得到一个常数近似的整数匹配（在期望意义下）。最后，我们可以在 定理 4.24 中将这一组合用作黑盒，将其改进为一个（$1+\varepsilon$）近似。对于节点平均唤醒复杂度，如 定理 4.13所示，求常数近似的最大匹配时，其唤醒复杂度为常数；舍入过程总共只需常数轮，然后在 定理 4.24的框架中，我们只需重复执行常数次。

5 结论

我们证明了，在保持标准 Luby 算法的轮复杂度为 $O(\log n)$ 的同时，可以以 $O(1)$ 的平均唤醒复杂度计算一个极大独立集。此外，我们还证明了，在期望意义下，可以达到相同的界限，分别用于计算最大匹配的 $(1+\varepsilon)$-近似解以及最小顶点覆盖的 $(2+\varepsilon)$-近似解。

在未来的研究中，一个值得探讨的问题是：能否获得优于轮复杂度的最坏情况下唤醒复杂度保证，以及能否以较低的唤醒复杂度（无论采用何种度量方式）计算一个极大匹配。

致谢

第一作者感谢 Yuval Emek 就此前一次关于极大独立集（MIS）问题的尝试所进行的讨论。第二作者获得了瑞士国家科学基金会的资助，项目编号为 200021 184735。

参考文献

[ABI86] Noga Alon、Lazlo Babai 和 Alon Itai。针对极大独立集问题的一种快速且简单的随机并行算法。《算法杂志》，7(4):567–583，1986。[BBH+19] Alkida Balliu、Sebastian Brandt、Juho Hirvonen、Dennis Olivetti、Mika¨el Rabie 和 Jukka Suomela。极大匹配和极大独立集的下界。发表于《计算机科学基础研讨会（FOCS）论文集》，2019。[BE10] Leonid Barenboim 和 Michael Elkin。利用 Nash-Williams 分解实现稀疏图的亚对数级分布式极大独立集算法。《分布式计算》，22(5-6):363–379，2010。[BE13] Leonid Barenboim 和 Michael Elkin。《分布式图着色：基础与最新进展》。Morgan & Claypool 出版社，2013。[BEPS16] Leonid Barenboim、Michael Elkin、Seth Pettie 和 Johannes Schneider。分布式对称性破缺的局部性。《ACM 杂志》，63:20:1–20:45，2016。[Ber57] Claude Berge。图论中的两个定理。《美国国家科学院院刊》，43(9):842，1957。[BFS12] Guy E Blelloch、Jeremy T Fineman 和 Julian Shun。贪心顺序极大独立集和匹配算法在平均意义下具有并行性。发表于第二十四届 ACM 算法与体系结构并行性研讨会论文集，第 308–317 页，2012。[BGKO22] Alkida Balliu、Mohsen Ghaffari、Fabian Kuhn 和 Dennis Olivetti。局部图问题的节点平均和边平均复杂度。手稿，2022。[BT19] Leonid Barenboim 和 Yaniv Tzur。具有改进顶点平均复杂度的分布式对称性破缺。发表于第 20 届国际分布式计算与网络会议论文集，第 31–40 页，2019。[BYCHGS17] Reuven Bar-Yehuda、Keren Censor-Hillel、Mohsen Ghaffari 和 Gregory Schwartzman。最大独立集和最大匹配的分布式近似。发表于国际分布式计算原理研讨会（PODC）论文集，第 165–174 页，2017。[CGP20] Sou myottam Chatterjee、Robert Gmyr 和 Gopal Pandurangan。休眠是高效的：在 O(1) 轮内实现节点平均唤醒复杂度的极大独立集算法。发表于分布式计算原理研讨会（PODC）论文集，2020。[Feu20] Laurent Feuilloley。一个具有普通 ID 的普通节点需要多长时间才能输出结果？《理论计算机科学》，811:42–55，2020。[Fis20] Manuela Fischer。通过舍入改进确定性分布式匹配算法。《分布式计算》，33(3):279–291，2020。[FMU21] Manuela Fischer、Slobodan Mitrovi´c 和 Jara Uitto。在半流式模型中，通过 poly(1/ε) 次遍历实现确定性的 (1+ε)-近似最大匹配。arXiv 预印本 arXiv:2106.04179，2021。[FMU22] Manuela Fischer、Slobodan Mitrovi´c 和 Jara Uitto。在半流式模型及更广泛的场景中，通过 poly(1/ε) 次遍历实现确定性的 (1+ε)-近似最大匹配。发表于第 54 届 ACM SIGACT 年度计算理论研讨会论文集，第 248–260 页，2022。[FN18] Manuela Fischer 和 Andreas Noever。对并行随机贪心极大独立集算法的紧约束分析。发表于第二十九届 ACM-SIAM 离散算法年度研讨会论文集，第 2152–2160 页。SIAM，2018。

[GGK+18] Mohsen Ghaffari、Themis Gouleakis、Christian Konrad、Slobodan Mitrovi´c 和 Ronitt Rubinfeld。改进的用于极大独立集、匹配和顶点覆盖的大规模并行计算算法。载于《2018 年 ACM 分布式计算原理研讨会论文集》，第 129–138 页，2018 年。[GGR21] Mohsen Ghaffari、Christoph Grunau 和 V´aclav Rozho˘n。改进的确定性网络分解。载于《ACM-SIAM 离散算法研讨会（SODA）论文集》，第 2904–2923 页，2021 年。[Gha16] Mohsen Ghaffari。改进的极大独立集分布式算法。载于《第 27 届 ACM-SIAM 离散算法研讨会（SODA）论文集》，第 270–277 页，2016 年。[Gha19] Mohsen Ghaffari。使用小消息的分布式极大独立集算法。载于《第三十届 ACM-SIAM 离散算法年度研讨会论文集》，第 805–820 页。SIAM，2019 年。[GP19] Mohsen Ghaffari 和 Julian Portmann。使用小消息改进网络分解及其在极大独立集、邻域覆盖等问题中的应用。载于《第 33 届国际分布式计算研讨会（DISC 2019）》。Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik，2019 年。[GPV93] Andrew V Goldberg、Serge A Plotkin 和 Pravin M Vaidya。用于匹配及相关问题的次线性时间并行算法。《算法杂志》，14(2)：180–213，1993 年。[HK73] J. E. Hopcroft 和 R. M. Karp。二分图最大匹配的 $n^{5/2}$ 算法。《SIAM 计算杂志》，2(4)：225–231，1973 年。[HKP01] M. Ha´n´ckowiak、M. Karo´nski 和 A. Panconesi。计算极大匹配的分布式复杂度。《SIAM 离散数学杂志》，15(1)：41–57，2001 年。.[Joh99] Ojvind Johansson。图的简单分布式 $\Delta+1$-着色。《信息处理快报》，70(5)：229–232，1999 年。[KMW04] Fabian Kuhn、Thomas Moscibroda 和 Roger Wattenhofer。哪些问题无法在本地计算！载于《第 23 届 ACM 分布式计算原理研讨会（PODC）论文集》，第 300–309 页，2004 年。[KMW16] Fabian Kuhn、Thomas Moscibroda 和 Roger Wattenhofer。本地计算：上下界。《ACM 杂志》，63(2)，2016 年。[Lin87] Nathan Linial。分布式图算法——从局部数据得到全局解。载于《第 28 届 IEEE 计算机科学基础研讨会（FOCS）论文集》，第 331–335 页，1987 年。[LPSP15] Z. Lotker、B. Patt-Shamir 和 S. Pettie。改进的分布式近似匹配。《ACM 杂志》，62(5)，2015 年。[Lub86] Michael Luby。求解极大独立集问题的简单并行算法。《SIAM 计算杂志》，15：1036–1053，1986 年。[McG05] Andrew McGregor。在数据流中寻找图匹配。载于《近似、随机化与组合优化：算法与技术》，第 170–181 页。Springer，2005 年。[Nao91] Moni Naor。分布式环着色概率算法的下界。《SIAM 离散数学杂志》，4(3)：409–412，1991 年。[Pel00] David Peleg。《分布式计算：一种对局部性敏感的方法》。SIAM，2000 年。[Pem01] Sriram V Pemmaraju。公平着色推广了 Chernoff-Hoeffding 界。载于《近似、随机化与组合优化：算法与技术》，第 285–296 页。Springer，2001 年。

[PPP+17] Shreyas Pai、Gopal Pandurangan、Sriram V Pemmaraju、Talal Riaz 和 Peter Robinson。CONGEST 模型中的对称性破缺：用于支配集的时间和消息高效算法。载于第31届国际分布式计算研讨会（DISC 2017）。Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik，2017 年。[PR01] Alessandro Panconesi 和 Romeo Rizzi。稀疏网络的一些简单分布式算法。《分布式计算》，14(2)：97–100，2001 年。[PS92] Alessandro Panconesi 和 Aravind Srinivasan。着色和网络分解问题的改进型分布式算法。载于第24届 ACM 计算理论研讨会（STOC）论文集，第581–592页，1992 年。[RG20] V´aclav Rozhoˇn 和 Mohsen Ghaffari。多对数时间确定性网络分解与分布式去随机化。载于 ACM 计算理论研讨会（ STOC ）论文集，页码待定，arXiv:1907.10937，2020 年。[SW08] Johannes Schneider 和 Roger Wattenhofer。适用于增长有界图的对数星分布式极大独立集算法。载于第二十七届 ACM 分布式计算原理研讨会论文集，第35–44页，2008 年。