arXiv:2305.01753v1 [cs.DC] 2 May 2023

任意图上具有检测功能的快速确定性聚集：多机器人之力

Anisur Rahaman Molla ！ 印
度统计学院，印度加尔各答

Kaushik Mondal ！ 印度理工学院罗帕尔分校，印度罗帕尔

William K. Moses Jr.[1] ！
英国达勒姆大学，达勒姆

摘要

多年来，人们围绕移动计算实体开展了大量研究。从对实际的微型（及更小型）机器人进行建模，到对网络上的软件进程进行建模，这一领域的许多重要问题都得到了研究。聚集是该领域的一个基础问题。聚集问题要求最初任意分布在一个含有 n 个节点的图的节点上的 k 机器人，以局部而非全局的方式进行协调和通信，在图中移动、相互寻找，并尽可能快速地汇聚到单个节点上。一个更难解决的问题是带检测的聚集：机器人聚集后，还必须识别出聚集已经完成，然后终止。

本文提出了一种确定性方法，用于在任何任意连通图上解决带检测的聚集问题，其速度快于现有的任意图上仅解决聚集问题（无需检测要求）的确定性解。与早期聚集研究不同，该方法利用系统中存在更多机器人的事实，从而比那些只关注聚集问题的论文更快地实现带检测的聚集。确定性聚集的当前最佳解 [Ta-Shma 和 Zwick, TALG, 2014] 需要 $\tilde{O}(n^5 \log \ell)$ 轮次，其中 ℓ 是机器人中最小的标签，$\tilde{O}$ 隐藏了一个多对数因子。我们设计了一种带检测聚集的确定性算法，其性能会根据机器人数量呈现以下权衡：（i）当 $k \geq \lfloor n/2 \rfloor + 1$ 时，算法需要 $O(n^3)$ 轮次；（ii）当 $k \geq \lfloor n/3 \rfloor + 1$ 时，算法需要 $O(n^4 \log n)$ 轮次；（iii）其他情况下，算法需要 $\tilde{O}(n^5)$ 轮次。算法无需知道 k，只需知道 n。

2012 ACM 主题分类 计算理论 → 分布式算法；计算理论 → 算法设计与分析

关键词和短语聚集，移动代理，移动机器人，分布式算法，任意图

资助 *Anisur Rahaman Molla*：A. R. Molla 部分获得了印度政府 DST INSPIRE 教师研究基金 DST/INSPIRE/04/2015/002801 以及 ISI DCSW/TAC 项目（档案编号 E5412）的资助。

Kaushik Mondal：K. Mondal 的部分研究经费由印度政府科学技术部的 FIST 项目资助，项目编号为 SR/FST/MS-I/2018/22(C)。*William K. Moses Jr.*：W. K. Moses Jr. 的部分研究经费由 NSF 的 CCF-1540512、IIS-1633720 和 CCF-1717075 项目资助，另有部分经费由 BSF 的 2016419 项目资助。

[1] 部分工作是在 William K. Moses Jr. 于美国休斯敦的休斯敦大学担任博士后研究员期间完成的。

1 引言

分布式计算的一个重要研究领域，关注计算实体何时能够在某个固定空间中移动并彼此交互。这类研究具有广泛的应用前景，涉及设计能够响应外部刺激的可编程物质 [25] 、群体机器人学 [39] ，乃至现代自动驾驶汽车技术 [4]；在网络任务中，通过移动机器人对流程建模，可为分布式信息检索、电子商务、信息传播、工作流应用和群件等诸多应用带来益处 [29]。当该空间被离散化时，图上的移动机器人这一广受欢迎的框架便可用于研究该领域的重要问题。该框架特别适用于研究现实世界中必须在包含房间和走廊的结构中移动的机器人相关问题。它也适用于研究软件代理在互联网中从一台计算机迁移到另一台计算机的行为。

在这一设定下实际研究的问题，通常有两种形式：要么让机器人协同在图中寻找某个目标（例如，探索 [5, 10, 13,16, 24, 30], 寻宝 [31]），要么形成某种特定构型（例如，聚集 [8, 9, 15, 37],分散 [2, 28, 32], 散布 [6, 20, 36, 40], 模式形成 [41], 收敛 [11]）。本文研究带检测的聚集问题（这是聚集问题的一个更难求解的变体）：多个机器人最初被任意放置在图上，必须在同一节点相遇，意识到聚集已经完成，随后终止。[2] 虽然这是一个抽象问题，但其解决方案可以用于更具体的场景，例如多个行人或机器人试图在离散空间中相遇，如由房间和走廊组成的迷宫，或由道路和交叉口组成的城市。此外，带检测的聚集也可作为子程序，用于解决探索、散布、分散等问题；当所有机器人已经聚集在一起时，解决这些问题会比机器人在图中任意分布时容易得多。

在尝试解决这个问题时，设计解决方案的一个根本区别在于机器人是否能够使用随机源。随机性或许能带来更快的解决方案，但我们希望聚焦于机器人无法使用这种额外力量的情形。在这种情形下，我们提出了一种算法来解决聚集问题；假设所有机器人最初都处于唤醒状态，该算法在时间效率上优于当前最佳水平。我们算法的一个关键洞见是，可以将图中的机器人数量视为算法设计者可利用的一种额外力量。可能需要使用多个机器人的任务包括探索某个区域 [16, 23, 34], 将机器人分散到给定区域 [2, 28, 33],、寻找宝藏等。任务完成后，我们可能希望将机器人聚集起来。

这与我们所考虑场景中的以往研究形成对比：以往研究给出了仅有两个机器人时解决聚集问题的方法，并假设该方法可以推广到多个机器人；通常采用这种方法，是因为已有研究表明，两个机器人的聚集解决方案可以轻松扩展到多个机器人 [26]。我们相信，提供一个此类问题求解方法的示例，将为更多更快速的解决方案铺平道路。为凸显这一点，请注意，聚集问题在 60 多年前就已提出 [38]，而此前最优的彩票娱乐

[2] 过去，当只有两个机器人时，聚集也被称为会合问题；为保持一致，本文通篇使用“聚集”这一术语。

已知的确定性解 [42, 43], 本身已有14年以上历史，需要 $\tilde{O}(n^5 \log \ell)$时间，其中 ℓ 是所有机器人中的最小标签。对于更困难的带检测的聚集问题，当机器人数量足够多时，我们的解最短只需 $O(n^3)$时间。

1.1 模型

考虑一个任意的连通无向图，该图有 n 个节点和 m 条边，且节点没有标签。每个节点的每条边都对应一个端口编号。节点的端口具有唯一标签，取值范围为 $[0, \delta - 1]$，其中 δ 是该节点的度。请注意，相邻节点之间的一条边，在两端节点处可能被赋予不同的端口编号。

系统中共有 k 个机器人。[3] 每个机器人都有一个唯一标识符（ID），也称为其标签，其取值范围为 $[1, n^b]$，其中 $b > 1$ 是一个常数。请注意，机器人的 ID 比特串长度不必相同。位于同一节点的两个机器人可以通过相互发送消息进行通信。通常，在聚集问题中会假设机器人相遇时能够获知这一事实，并认为这已足够。然而，正如 [21], 所指出的，检测与机器人之间的通信方式密切相关。在 [21], 中，他们考虑机器人通过蜂鸣模型 [12], 进行通信；这种模型可被认为是最弱的通信模型之一。而在 [7], 中，他们采用了更弱的概念，即机器人只能检测与自己位于同一节点的其他机器人的数量。我们采用消息传递框架：在同一轮次中位于同一节点的多个机器人可以彼此交换消息。这一框架也称为面对面模型，曾用于研究与移动机器人相关的其他问题，包括探索 [14] 和分散 [3, 33]。如果机器人从一个节点移动到相邻节点，它就能获知其经过的边在两端对应的端口编号。我们假设每个机器人都知道 n 的值以及自己的唯一 ID。特别需要指出的是，机器人不知道 k、b 的值，也不知道除 n 以外的任何图参数。

我们考虑一种按轮次推进的同步系统。在每一轮中，机器人按以下顺序执行操作：(i) 位于同一节点的机器人可以通过传递消息彼此通信；机器人按需进行本地计算；(ii) 每个机器人如果在步骤 (i) 中决定移动，就沿一条边移动到相邻节点。在图上的移动机器人框架中，重点是限制机器人所需的移动量，因为与本地计算和通信相比，移动在时间成本上昂贵得多。因此，在此框架下研究的问题（例如探索、分散）通常假设本地计算和通信不产生开销，且每轮的步骤 (i) 中可以进行任意数量的通信和计算。尽管这一假设很有利，为确保算法的实用性，我们设计的算法使得任何本地计算都至多为关于 n 的多项式复杂度。我们假设所有机器人最初都处于唤醒状态，并同时启动任何指定的算法。

现在，我们正式给出聚集问题和带检测的聚集问题的定义。

聚集： 初始时，k 个机器人被放置在一个含有 n 个节点的图的节点上。设计一种

[3] 在文献中，这些计算实体也被称为“移动代理”。为保持一致，全文统一使用“移动机器人”。

由每个机器人运行的算法，使得最终所有 k 个机器人都聚集到同一个节点。

带检测的聚集： 最初， k 个机器人被放置在一个具有 n个节点的图上。设计一种由每个机器人运行的算法，使得最终所有 k 个机器人在同一个节点相遇，并随后终止。

1.2 我们的贡献

据我们所知，我们的主要贡献是展示了第一种利用多个机器人，在图中实现带检测的移动机器人快速确定性聚集的方法。需要明确的是，在聚集中可以出于其他目的使用多个机器人，例如当机器人没有标签时打破对称性；但据我们所知，尚未有人利用多个机器人来实现更快的运行时间。此外，我们具体讨论的是图上的移动机器人聚集问题。图上移动机器人的探索问题已经采用了这种方法，并由此产生了非常快速的算法 [16]。

在本文中，我们提出了一种<style id='1'>确定性方法，用于解决任意<style id='3'>图上的带检测聚集问题；即使仅考虑任意图上的聚集问题（不要求检测），该方法也比现有的确定性解<style id='5'>更快。与早期关于聚集的研究相比，该方法利用了系统中存在更多机器人的事实，从而比那些仅关注聚集的先前论文更快地实现带检测的聚集。任意图上确定性聚集的当前最佳水平解 [43]需要 $\tilde{O}(n^5 \log \ell)$，其中 ℓ 是机器人中最小的标签， $\tilde{O}$ 隐藏了 n的多对数因子。我们设计了一种带检测聚集的确定性算法，并根据存在的机器人数量提供以下权衡：(i) 当 $k \geq \lfloor n/2 \rfloor + 1$时，该算法需要 $O(n^3)$轮次；(ii) 当 $k \geq \lfloor n/3 \rfloor + 1$时，该算法需要 $O(n^4 \log n)$轮次；(iii) 否则（对于任意 $k < \lfloor n/3 \rfloor + 1$），该算法需要 $\tilde{O}(n^5)$轮次。每个机器人仅需 $O(m \log\ n)$位内存。[4]

显然，当存在足够多的机器人时，即$k \geq \lfloor n/3 \rfloor + 1$，我们的算法解决带检测的聚集问题，其渐近速度快于任何先前用于聚集问题的算法。此外，值得注意的是，要求存在多个机器人的目的，是确保初始配置中存在彼此足够接近的机器人。因此，无论可用机器人的数量是多少，只要存在两个机器人，它们的跳数距离为 i，以下结论均成立。若 $i = 0,1,2$，则该算法可在 $O(n^3)$轮次内实现带检测的聚集。若 $i = 3,4$，则该算法可在 $O(n^4 \log n)$轮次内实现带检测的聚集。对于 $i \geq 5$，该算法可在 $\tilde{O}(n^5)$轮次内实现带检测的聚集。

1.3 技术难点与高层构想

聚集最困难的部分，是确保机器人拥有有效的图探索策略，使机器人最终能够相遇。通常的做法是为每个机器人提供一系列要执行的移动，以及相应的执行计划，使得最终每个机器人都能探索整个图，并且存在一段轮次序列，其中对于任意一对机器人，一个保持静止，另一个则移动。当前最佳水平的探索策略是使用通用探索

[4] 请注意，我们不 要求机器人知道 m 的值，只要求为它们提供足够的内存。

序列（UXS），并将其与静止轮次和移动轮次相结合，这些轮次对应于机器人 ID 字符串中的位。虽然这种方法有效，但使用 UXS 探索图所需的时间很长。

我们利用的一个关键思想是：当图上的机器人数量更多时，我们可以推断初始配置中距离最近的两个机器人之间的距离。更具体地说，我们证明，当 $k \geq \lfloor n/c \rfloor + 1$ 时，存在两个机器人，它们之间的跳数距离至多为 $2c - 2$。

这一洞见至关重要，但仅凭它还不足以实现快速聚集。事实上，Dessmark等人 [17] 提到，当两个机器人同时出发且它们之间的距离有界于 D 时，可以在 $O(D\Delta^D \log \ell)$ 轮次内实现聚集，其中 Δ 是图的最大度。然而，他们的结果针对的是两个机器人；若考虑将其推广到多个机器人，似乎 D 将反映初始配置中任意两个机器人之间的最大最短跳数距离。由于机器人被认为是由对手放置的，即使图上有很多机器人，这一运行时间也可能相当长。[5]

因此，我们必须设法巧妙地利用存在两台彼此距离很近的机器人的事实。首先，我们让这两台机器人以类似 Dessmark等人的方式相遇。 [17], 也就是说，我们让它们执行邻居搜索过程，以找到彼此。两台机器人相遇后，我们便利用它们来召集其余机器人。[6]具体而言，我们先让这两台机器人使用一种简单的令牌探索式算法构建图的地图：其中一台机器人充当可移动令牌，另一台则逐步了解该图中以它们最初相遇的节点为中心、半径不断增大的球内的信息。随后，这两台机器人便在图中四处移动，收集其他机器人。

然而，我们必须考虑可能存在多对这样的机器人的情况。我们希望最终能将所有机器人聚集到一起。当一些机器人移动而另一些机器人保持不动时，我们必须设计一种策略，确保所有机器人都被“捕获”并聚集到一起。为此，我们让地图构建步骤中的每一对机器人都按以下方式行动：一台留在原地（以捕获其他移动的机器人），另一台则以系统的方式在整个图中移动（以捕获其他静止的机器人）。具体由谁捕获谁，取决于每对机器人中最小的标签，它相当于这对机器人的标识符。如果来自较低ID的机器人对中的机器人遇到来自较高ID的机器人对中的机器人，那么较高ID的机器人就会被较低ID的机器人捕获；也就是说，较高ID的机器人会停止当前行动，之后“跟随”较低ID的机器人。这样一来，ID最低的机器人对就能捕获所有其他机器人，并最终将它们聚集到一起。

最后，还需要讨论兜底情况，即机器人数量不足，或者不存在足够接近、能够按上述方法实现更快的带检测聚集的机器人。在这种情况下，我们默认使用通用探索序列（UXS）帮助机器人彼此寻找，这与 Ta-Shma 和 Zwick [43] 的思路类似。不过，我们引入了自己的改进，以确保聚集带检测 能够实现。由于已知 n（图中的节点数），

[5] 一种可能的对抗性部署，是使运行时间最大化的部署，例如尽可能让机器人彼此远离。举例来说，假设所有机器人分成两组，两组之间相距 $D-1$。如果 $\Delta = \Omega(n)$ 且 $D = \Omega(n)$，那么 Dessmark 等人的算法的运行时间将呈指数级。对于随机部署，我们以高概率不会在每次部署中都遇到如此糟糕的配置。[6] 请注意，可能会有两个或更多机器人最终相遇。从功能上说，这与只有两个机器人相遇相同，因此在描述高层思路时，我们忽略这种情况；在算法描述中则会明确处理这种情况。

所有机器人都知道这一数值，因此它们可以确定性地计算出相同的 UXS，以确保能够探索整张图，例如长度为 T 的 UXS。接下来，我们让机器人按每 $2T$ 轮次为一个阶段执行以下一系列操作。每个机器人从最低有效位到最高有效位读取自己的标签。如果该位为 0，机器人就在当前节点等待 T 轮次，然后在接下来的 T 轮次中使用 UXS 探索整张图。如果该位为 1，机器人则按相反顺序行动。如果机器人遇到 ID 更高的机器人，就开始跟随该机器人，也就是说，它按照 ID 更高的机器人的 ID 位做出选择。如果机器人读完了所有位，就在所在节点停留 $2T$ 轮次。在此期间，如果没有 ID 更高的机器人（或比它正在跟随的机器人 ID 更高的机器人）出现，该机器人就判定聚集已经完成，随后终止。

1.4 相关工作

聚集是一个历史悠久的问题，早在 60 多年前，Schelling 就对其进行了研究 [38]。此后，关于这一问题的研究大致分为两类，取决于移动所依托的介质是连续空间（例如欧几里得平面）还是离散空间（例如图）。我们关注图；Flocchini [22] 对平面上的聚集研究进行了很好的综述。另一个划分维度是，解决方案是否要求机器人能够使用随机性。在当前工作中，我们关注图上的聚集问题的确定性解；Alpern [1] 对随机化解（不局限于图上的解）进行了很好的综述。本文重点研究机器人具有唯一标签这一特定情形；Pelc [35] 的综述中有一节专门讨论机器人匿名的情况。

对于目前所考虑的带标签机器人确定性聚集问题，针对任意图已有少量研究 [17, 26, 43]；此外，也有关于任意图上两台机器人带检测的聚集 [21] 以及任意图上多台机器人带检测的聚集 [7]的研究。Dessmark等人 [17] 研究了多种场景下两台机器人的聚集问题，其中包括任意图上的情形，并考虑两台机器人的启动时间之间存在 τ 的启动延迟。在这种场景下，时间从最后一台机器人唤醒时开始计算。他们提出了首个确定性算法，能够在任意连通图中解决聚集问题，且运行时间关于 τ、 n和 ℓ为多项式级，其中 ℓ 是两台机器人标签中较小的一个。具体而言，他们的算法运行时间为 $O(n^5\sqrt{\tau}\log\ell\log\ n + n^{10}\log^2 n\log\ \ell)$轮次。他们还证明，若假设机器人同时启动，即$\tau = 0$，则可在 $O(D\Delta^{D}\log\ell)$轮次内完成聚集，其中 D 是两台机器人之间的初始距离， Δ 是图的最大度。由于机器人被认为是由对手放置在图上的，而且存在直径为$\Omega(n)$量级的图，因此该算法在某些图上的运行时间会关于 n 呈指数级。Kowalski 和 Malinowski [26] 展示了如何以确定性方式实现聚集，即使机器人之间存在非零启动延迟，也只需 $O(\log^3\ell + n^{15}\log^{12} n)$轮次。此外，他们还证明，任何针对两台机器人聚集问题的解决方案，都可以调整为以相同的运行时间解决任意数量机器人（> 2）的聚集问题。当前最佳水平方面，Ta-Shma 和 Zwick [43] 将聚集问题的运行时间进一步降低至 $\tilde{O}(n^5\log\ell)$轮次，即使机器人之间存在非零启动延迟也不例外，其中 $\tilde{O}$ 记号隐藏了n 的多对数因子。以上所有研究针对的都是聚集问题，而 Elouasbi 和 Pelc [21]研究了这一场景下两台机器人的带检测聚集问题。他们证明，当允许机器人通过蜂鸣模型进行通信时，可以用确定性算法在 $\tilde{O}(n^5\log\ell)$轮次内解决带检测的聚集问题。Bouchard等人 [7] 考虑了更弱的模型：机器人只能检测与自己位于同一节点上的其他机器人数量；他们以确定性方式解决了带检测的聚集问题，运行时间关于 N 和$\log\ell$为多项式级，其中 N 是 n 的上界且所有机器人都知道；若所有机器人都不知道上界，运行时间则关于 n 呈指数级

若所有机器人都不知道 n 的上界。值得注意的是，除 [7], 之外，上述论文中的机器人起初并不知道 n 的值，而本文作出了这一假设。然而，与聚集问题的相关论文 [17, 26, 43], 不同，我们解决的是更难的带检测聚集问题。与蜂鸣模型下带检测聚集问题的论文 [21], 不同，本文中的机器人起初并不知道 k 的值（可以认为，由于 [21] 专门研究两台机器人的聚集问题，因此其协议设计中隐含地包含了这一知识）。此外，与 [7], 不同，我们的算法在机器人数量更多时更快，因为他们使用的是 [43]中的算法。

关于带检测的聚集问题的确定性解的下界，我们尚未发现任何非平凡的下界。一个平凡的下界是 $\Omega(n)$（考虑位于一条直线两端的两组机器人，它们至少需要 $n/2$ 轮次才能相遇）。

此外，也有研究探讨了其他设定，例如机器人可能遭遇崩溃故障或拜占庭故障；还有研究考察了其他性能指标，例如在优化时间（如本文所研究）之外，还要优化成本，即所有机器人进行的边遍历总数。Pelc 的综述 [35] 提到了这些及其他设定和指标。除了研究任意图之外，另一种研究该问题的方法是将输入图限制为某个特定类别，例如环。Kranakis 等人所著的书 [27] 是了解环（以及其他一些图类别）上相关解法的优质资源。

利用多个移动机器人可用性来实现更快解决方案的方法，也已用于图上的移动机器人场景中的其他问题，例如探索 [16, 19]。

2 在任意图上的聚集

本节介绍我们用于带检测的聚集的主算法。我们首先在第 2.1 节介绍一种使用通用探索序列（UXS）的聚集算法，该算法适用于任意数量的机器人，并能检测聚集是否完成。然而，基于 UXS 的算法需要较长时间，即 $\tilde{O}(n^5)$ 轮次。之后，我们将在主算法中将该算法作为子程序使用。

随后，我们介绍一种更快的带检测的聚集算法。对于该更快的算法，我们首先展示如何在一组略有限制的机器人节点输入配置上解决问题，然后展示如何扩展该算法，使其适用于所有输入配置。我们交替使用“分布”和“配置”这两个术语来指代机器人的分布。

2.1 使用通用探索序列（UXS）进行带检测的聚集

本节将证明，对于任意数量的机器人，在任意 n 节点图上，通过检测即可实现聚集（即，机器人能够检测到聚集已完成，并随后终止），其中机器人知道 n，但不知道 k。为此，我们使用以下关于通用探索序列（UXS）的结果，该结果可由 [43] 隐含推出。

一个机器人只要知道任意匿名图中的 n，就能计算出一个长度有界的 *UXS*，并利用该 *UXS* 在时间 $\tilde{O}(n^5)$ 内探索该图。由于序列长度有界，机器人会终止。令 T 为探索时间的一个上界，即 $T = \tilde{O}(n^5)$，其中 M 为实现该 UXS 所需内存的一个上界。

8 任意图上具有检测功能的快速确定性聚集：多机器人之力

利用上述结果和机器人的 ID 位，我们可以确定性地聚集任意数量的机器人，并进行检测。机器人的 ID 位串长度可能各不相同。此外，机器人只知道 n ，而不知道任何其他参数。聚集算法如下所示。

最初，从同一节点出发的机器人组成一个群组。如果某个节点上只有一个机器人，它就组成一个单体群组。群组中的每个机器人在图探索过程中都跟随 ID 最大的机器人。每个不跟随任何其他机器人的机器人（即群组中 ID 最大的机器人或单体机器人）都会从最低有效位到最高有效位逐位读取自己的 ID 位，并并行执行以下操作。

(i) 如果 **该位为** 1，机器人使用 UXS 在图上进行探索，持续 T 轮次，然后在完成探索时所在的节点等待接下来的 T 轮次。因此，总共需要 $2T$轮次。回想一下，该组中的其他机器人（如果有）只需跟随它。如果两个或多个组在这些 $2T$ 轮次中的任意一轮相遇，则相应的组将合并。此后，机器人开始跟随合并组中标识符最大的机器人。**(ii) 如果该位为** 0，机器人在当前节点等待前 T 轮次，然后在接下来的 T 轮次内使用 UXS 探索图。总共需要 $2T$ 轮次。同样，如果两个或多个组在这些 $2T$ 轮次中的任意一轮相遇，则相应的组将合并。此后，机器人开始跟随合并组中标识符最大的机器人。**(iii)** 如果（标识符最大的）机器人完成了对其所有 ID 位的扫描，则它再等待 $2T$ 轮次。如果在此期间它没有遇到任何组，则终止。否则，相应的组将合并，该机器人随后开始跟随合并组中标识符最大的机器人。

这种探索方式以及逐位等待的核心思想，是确保机器人能够终止。考虑一个等待 $2T$ 轮次的机器人。在此期间，如果没有其他机器人遇到这个正在等待的机器人，就意味着没有机器人仍在处理自己的位；否则，无论该机器人处理的是 0 位还是 1 位，都必定会遇到这个正在等待的机器人。

我们证明，上述过程能够在 $O(T \log L)$ 轮次内正确地聚集任意数量的机器人，其中 L 是这些机器人的最大标识符。此外，机器人能够检测到聚集已完成，并随后终止。该算法的正确性由以下引理得出。

▶**引理 1.** 在步骤（*iii*）中，一个不跟随任何机器人的机器人（例如 r）会等待 $2T$轮次。当机器人 r 等待时，当且仅当所遇到的探索机器人群体中最大标识符机器人的标识符长度（例如 r'）大于 r的标识符长度时，它才会遇到该群体。

证明。 首先证明“仅当”部分。

假设 r' 的 ID 长度等于 r的 ID 长度。那么r 和 r' 会同时完成 ID 位的扫描，并且两者都必须同时等待。也就是说，当 r 在等待时， r' 也必定在等待。

现在假设 r' 的 ID 长度小于 r的 ID 长度。由于 r' 是其所在组中 ID 最大的机器人，它必定比 r 更早完成 ID 扫描。因此，不可能出现 r' 前来与 r 相遇，而此时 r 正在等待的情况。

现在来论证“当”部分。设 r' 是 ID 长度大于 r 的机器人，且 r 正在等待 $2T$ 轮次。也就是说， r' 仍有一些位（至少一位）尚未扫描。因此，无论对应的位是 0 还是 1， r' 都必须探索该图。所以， r' 会在其等待的 $2T$ 轮次内与 r 相遇。◀

▶**引理 2.** 在步骤（*iii*）中，设 r 是一个没有跟随任何其他机器人的机器人。如果 r 等待 $2T$ 轮次，且在此期间没有机器人与 r 相遇，则聚集完成。

证明。 如果没有机器人在 r 等待期间与其相遇，根据引理 1，图中不存在标识符比它更大的机器人仍在处理其某些位。这意味着，其他没有跟随任何机器人的机器人，其标识符长度要么与 r相同，要么短于 r。

首先考虑这样的情况：机器人 r' 不跟随任何机器人，且其 ID 长度与 r相同。这意味着 r 和 r' 同时完成了各自位的处理。由于 r 和 r' 的 ID 必定在某一位上不同，因此它们必定在处理该特定位时已经会合。

现在考虑这样的情况：机器人 r' 不跟随任何机器人，且其 ID 长度小于 r的 ID 长度。由于 r' 不跟随任何机器人，因此它必定在 r 之前完成对其所有位的扫描。所以，当 r' 等待了 $2T$ 轮次时，它必定已经与 r 会合（见引理 1）。

证明完毕。 ◀

▶**引理 3。** 在步骤（*iii*）中，设 r 为一个不跟随任何其他机器人的机器人。如果 r 在 $2T$ 轮次后终止，则该终止是正确的。

证明。 根据引理 2，这一点显而易见，因为 r 仅当聚集完成后才会终止。 ◀

▶ **引理 4.** 设 r 是正在跟随其他机器人的机器人。机器人 r 始终 t正确终止。

证明。 引理 3 表明，每个未跟随任何机器人的机器人都会正确终止。现在 r 正在跟随某个机器人，设为 r'，这意味着 r' 没有跟随任何其他机器人。因此 r 只有在 r' 终止时才会终止。由于 r' 正确终止， r 也会正确终止。 ◀

▶ **引理 5.** 该算法运行 $O(T \log L)$ 轮次。

证明。 令 r 为标识符长度为 $O(\log L)$ 的最大标识符机器人， L 为标识符。根据我们的算法， r 必须完成处理其所有位。每个位需要 $2T$轮次。处理完所有位后， r 再等待 $2T$ 轮次，然后终止。因此时间复杂度为 $O(T \log L)$ 轮次。 ◀

回顾一下， $T = \tilde{O}(n^5)$。因此，该基于 UXS 的算法的时间复杂度变为$\tilde{O}(n^5)$轮次，因为我们假设机器人的标识符位于范围 $[1, n^b]$ 内，其中 b为某个常数。引理 3–5 得出以下结果。

▶**定理 6.** 考虑一个含有 n 个节点的连通无向匿名图，若干机器人（数量任意）以任意方式分布在各节点上。机器人只知道 n 的值。则存在一种确定性算法，可以在 $\tilde{O}(n^5)$ 轮次内通过检测将所有机器人聚集到某个节点，其中机器人的标识符位于 $[1, n^b]$ 范围内（某个常数 b）。该算法要求机器人仅知道 n 的值。每个机器人需要 $O(M + \log n)$ 位内存，其中 M 是实现 *UXS* 所需的内存。

在接下来的两个部分中，我们将介绍一种更快的聚集算法。首先考虑机器人的初始分布为<自身 id='1'>非分散</自身>的情形，即在各节点上的机器人分布中，至少有一个节点拥有两个或更多机器人。在第 2.2 节描述如何解决这一情形下的聚集问题后，我们将在第 2.3 节展示如何扩展我们的结果，使其不仅能够处理机器人的非分散初始分布，也能处理<自身 id='3'>分散</自身>分布，即每个节点初始时至多拥有一个机器人的情形。请注意，当机器人数量

至多等于节点数量 n，即 $k \leq n$ 时， k 。当 $k > n$ 时，很容易看出始终会存在一个节点拥有多于一个机器人（这也称为<鸽巢原理 id='5'>鸽巢原理</鸽巢原理>），因此任何满足 $k > n$ 的初始设置都是非分散分布。由于非分散初始配置和分散初始配置的并集构成所有输入配置的集合，因此第 2.3 节提出的算法适用于所有输入配置。我们证明，若位于两个相邻节点或相距若干跳数的节点上的至少两个机器人，或者机器人数量“很多”，则分散分布中的聚集算法会<更快 id='8'>更快</更快>地聚集机器人。这验证了一个直观事实：分散配置是聚集问题中最糟糕的配置。

2.2 从未分散配置出发进行带检测的聚集

在未分散配置中，初始时至少存在一个包含多个机器人的节点。机器人可以处于以下三种状态之一：(i) $finder$ (ii) $helper$ (iii) $waiter$。如果某个机器人在初始配置中并非独处，且其标识符是共址机器人中最小的，则将其状态设置为 $finder$。如果某个机器人在初始配置中并非独处，且其标识符不是共址机器人中最小的，则将其状态设置为 $helper$。如果某个机器人在初始配置中独处于某个节点，则将其状态设置为 $waiter$。每个机器人都维护一个变量 $group_{id}$。初始时，每个 $finder$ 机器人的 $group_{id}$ 均为其自身的标识符，而每个$helper$ 机器人在其 $group_{id}$中存储共址 $finder$ 机器人的标识符。每个 $waiter$ 机器人都将 -1 存入其 $group_{id}$。该算法分阶段运行。第一阶段专用于地图构建，第二阶段则用于聚集。每个机器人都能够检测聚集何时完成。在解释该算法之前，我们先给出一个高层次的基本思路。

由于每个 $finder$机器人都有一队 $helper$机器人陪伴， $finder$机器人在阶段 1 使用现有算法寻找图的地图。随后在阶段 2，每个 $finder$ 机器人按照它拥有的地图探索图。 $helper$ 和 $waiter$ 机器人留在初始位置，直到某个 $finder$ 机器人到达并接走它们。在探索过程中， $finder$ 机器人收集 $helper$ 和 $waiter$ 机器人。尽管此时看起来机器人可能会分散成多个群组，但我们稍后会说明，某个特定的 $finder$ 机器人如何在仅进行一次图探索后，将所有其他机器人聚集到自己的初始位置。下面给出完整算法。

算法 **Undispersed-Gathering:**

阶段 1（地图构建）： 状态为 $finder$ 和 $helper$ 的机器人参与此阶段。 $waiter$ 机器人留在原地。每个状态为 $finder$ 的机器人充当代理，其余共址机器人（状态为 $helper$）充当可移动令牌。它们协作运行 [18] 中介绍的使用可移动令牌的探索算法（用于地图构建）。每个 $finder$ 都可以在 $O(n^3)$ 轮次内计算出底层图的同构地图。由于每个 $finder$ 机器人都需要存储图的地图，因此内存需求变为 $O(m \log n)$，其中 m 是图的边数。构建地图后，每个 $finder$ 会与本阶段中充当其令牌的 $helper$ 机器人会合，并将 n 的值提供给辅助机器人。请注意，即使多个查找者-辅助者组合并行执行此算法，每个查找者仍能跟踪自己的辅助者（反之亦然），因为其辅助者的 $group_{id}$ 必须等于其自身的标识符。

将阶段 1 的轮次数记为 $R_1 = O(n^3)$。阶段 1 完成后，每个 $finder$机器人都知道一份图的地图的同构副本。 $finder$和$helper$ 机器人会等待 R_1 轮次结束。然后，从第 $R_1 + 1$轮开始，它们执行阶段 2。

阶段 2（聚集）：所有机器人都参与这一阶段。下面我们介绍处于不同状态的机器人在一轮次中的任务。

- **机器人算法：** $finder$**机器人：**每个 $finder$机器人根据其拥有的图（地图）计算一棵生成树，并沿着该生成树的边进行探索。设一台机器人$finder$ 在某轮次开始时位于某个节点。它与共址机器人通信，以了解彼此的状态以及 $group_{id}$。
  - 如果该节点中只有 $waiter$ 机器人，则 $finder$ 提供n 的值，并移动到另一个节点，根据其生成树继续探索，除非它已返回开始探索时所在的节点。
  - 如果该节点中存在其他 $finder$ 和/或 $helper$ 机器人，则当其 $group_{id}$ 在共址机器人 $group_{id}$中最小时， f 提供移动到另一个节点，根据其生成树继续探索，除非它已返回开始探索时所在的节点；否则，如果存在具有最小 $group_{id}$的 $finder$ 机器人，f 将其状态更改为 $helper$，把其 $group_{id}$ 更新为该 $finder$ 机器人的 $group_{id}$，并开始跟随该机器人；否则，如果存在具有最小 $group_{id}$的 $helper$ 机器人，f 将其状态更改为 $helper$，把其 $group_{id}$ 更新为该 $helper$ 机器人的 $group_{id}$ ，并停留在此节点。
  - 如果该节点上只有 $helper$ 机器人，且不存在 $helper$ 机器人的 $group_{id}$ 小于 f的 $group_{id}$ ，那么 f 会按照其生成树前往另一个节点继续探索，除非它已返回开始探索的节点；否则， f 会将其状态更改为 $helper$，将其 $group_{id}$ 更新为 $helper$ 机器人的 $group_{id}$ 中的最小值$group_{id}$ ，并停留在该处。
- **算法** $helper$ 机器人：</style}如果上一轮没有 $finder$ 机器人到达该节点， $helper$ 机器人就停留在该节点。如果有一个或多个 $finder$ 机器人到达，并且至少有一个 $finder$ 机器人的 $group_{id}$ 小于 $helper$ 机器人的 $group_{id}$，那么$helper$ 机器人会将其 $group_{id}$ 更改为所有同处一节点的 $finder$ 机器人中 $group_{id}$ 最小的 $finder$ 机器人的 $group_{id}$ ，并开始跟随它。
- **算法** $waiter$ 机器人：</style}如果上一轮没有 $finder$ 机器人到达， $waiter$ 机器人就停留在该节点。否则，如果有一个或多个 $finder$ 机器人到达， $waiter$ 就开始跟随 $group_{id}$ $finder$ 最小的机器人。它还会将其状态更改为 $helper$ ，并将其 $group_{id}$ 更新为 $group_{id}$ $finder$ 最小的机器人的 $group_{id}$ 。
- **终止：**每个机器人都会记录算法开始以来经过的轮次数。当计数器等于 $R_1 + 2n$ 时，每个机器人都会终止。

正确性：
阶段 1 的正确性由 [18] 得出。这里我们研究阶段 2 的正确性。设 f 是 $finder$ 个机器人中，其 $group_{id}$（记作 l）在所有 $finder$ 个机器人中最小的那个。设 S 为阶段 2 开始时 $group_{id}$ 等于 l 的 $helper$ 个机器人的集合。容易看出， S 非空，否则 f 就不会是一个 $finder$ 机器人。设阶段 2 开始时， f 和 S 中的 $helper$ 个机器人所在的节点为 v 。我们提出以下结论。

▶**引理 7**。当具有最小 $group_{id}$ 的寻找机器人 f 完成图探索并返回到它开始阶段 *2* 时所在的节点 v 时，所有其他机器人都已聚集在 v。

证明。在阶段 2 的第一轮中， f 一定会发现其 $group_{id}$ l 是共址机器人的标识符中的最小值，并开始图探索。注意， $group_{id}$ 机器人 S 的 l 也 f 会将其 $group_{id}$ 视为最小值

在共址机器人中。很容易看出，根据我们的算法， f 的状态从不改变，并且在完成图遍历后回到 v ，因为其 $group_{id}$ 始终是所有标识符中最小的。

考虑任意一个 $helper$ 机器人，其 h 大于 $group_{id}$ 。该机器人 l会停留在原地，直到某个 h 较小的机器人 $finder$到达。如果第一个到达 $group_{id}$所在节点的 $finder$恰好是 h，那么 f会与 h一起到达 v，因为 f会继续跟随 h。如果另一个 f较小的机器人 $finder$ 先遇到 f'，那么 $group_{id}$就会开始跟随 h。可以确定， h尚未探索节点 f'，否则它就会被困在那里。除非 f' 遇到另一个查找者 v 并成为 f' ，否则这一过程会持续下去。同样可以确定， f'' 尚未访问 $helper$。这一过程会持续到 f'' 到达 v 。如果 h 在某一轮次尚未到达 v，这意味着 $helper$ h 当前跟随的 v 机器人尚未完成探索。由于我们的算法按同步轮次进行，因此到这一轮时， $finder$ 也尚未完成探索。这表明，每个 h 机器人最终都会在 f 完成探索并返回 $helper$ 时到达 v 。

考虑任意一个 $waiter$ 机器人 w。根据我们的算法，每当 w 遇到某个$finder$机器人时，它就将状态改为 $helper$，更新 $group_{id}$并开始跟随该 $finder$机器人。我们已经证明，任何 $helper$ 机器人最终都会到达 v，因此这表明，当 f 完成探索并返回 v时，每个$waiter$ 机器人最终都会到达 v 。

考虑任意一个除 f之外的 $finder$机器人 f'。根据我们的算法，可能出现以下三种情况。

- f'以 $finder$机器人的身份探索图时到达 v。在这种情况下， f'会将其$group_{id}$ 改为 l，将状态改为 $helper$ ，并停留在 v。这是因为，在所有可用的 $group_{id}$中， l 是可能的最小 $group_{id}$ 。f'不会以 $finder$的身份到达 v。这会在以下两种情况下发生。探索过程中，
- f'遇到某个 $group_{id}$较小的 $finder$机器人 f''，于是将状态改为 $helper$后开始跟随 f'' 。我们已经证明，任何 $helper$机器人最终都会到达 v，因此这种情况得证。探索过程中，f' 遇到某个 $group_{id}$较小的 $helper$ 机器人，将状态改为 $helper$ 并停留在该节点。我们已经证明，任何 $helper$ 机器人最终都会到达 v，因此这种情况得证。

显然，这一阶段会在 $2n$ 轮次内完成机器人聚集，因为 ID 最小的寻找机器人需要恰好 $2n$ 轮次，沿着 $finder$ 根据其持有的图的地图副本计算出的某棵生成树，探索图中的所有节点。此外，到阶段 1 结束时，所有 $finder$ 和 $helper$ 机器人都知道 n ；在阶段 2 中，每个等待机器人至少会遇到一个 $finder$ 机器人，因此每个$waiter$ 机器人都会从某个 $finder$ 机器人那里收到 n 的值。因此，当计数器等于 $R_1 + 2n$时，每个机器人都可以终止。◀

现在给出本节的主要定理。

▶**定理 8.** 给定一个具有 n 个节点的匿名图（无向且连通），有 k 个机器人任意分布在各节点上，且至少一个节点上有多个机器人，则确定性算法 UNDISPERSED-GATHERING 可在 $O(n^3)$ 时间内将所有机器人聚集到单个节点，并且每个机器人使用 $O(m\log n)$ 内存，其中 m 是图中边的数量。此外，每个机器人都能检测到聚集何时完成。

证明。 我们的算法阶段 1 的正确性可由 [18] 得出。我们算法阶段 2 的正确性（即汇集所有机器人）已在引理 7 中给出。阶段 1 的时间复杂度为 $O(n^3)$ ，并由 [18]得出。由于除最小的机器人之外的任何机器人

ID 机器人可以是 $finder$ 机器人，由于存储图的地图而产生的内存需求为每个机器人 $O(m \log n)$ 位。阶段 2 的时间复杂度取决于具有最小 $group_{id}$ 的 $finder$ 机器人完成图探索并返回阶段 2 开始时初始位置所需的时间。由于 $finder$ 按照图的生成树进行探索，因此它恰好需要 $2n$ 轮次。因此，总体时间复杂度为 $O(n^3)$。

◀

设 Undispersed-Gathering 的运行时间上界为 R，其中 $R = R_1 + 2n \in O(n^3)$。稍后介绍适用于任意初始分布的主算法时，我们会用到这一点。

2.3 从分散配置中聚集

本节首先介绍一种使机器人从分散配置中聚集的算法。具体而言，给定初始时处于分散配置的 $k \leq n$ 个机器人，我们提出一种高效的方法，使其到达未分散配置。然后，应用第 2.2 节中的上述算法，即可解决聚集问题。为便于说明，我们先考虑以下较简单的情况。

假设在分散配置中，两个机器人分别位于两个相邻节点（即相距 1 跳）。我们证明，可以在 $O(n \log n)$ 时间内将此配置转换为未分散配置。为此，我们利用机器人的 ID 位串将它们汇集到单个节点，从而到达未分散配置。设两个机器人 r_u 和 r_v 分别位于两个相邻节点 u 和 v。由于机器人的标识符各不相同，r_u 和 r_v 的 ID 位串也不同（长度也可能不同）。机器人 r_u 和 r_v 执行以下过程来汇集。

1-跳跃会合： 该方法以周期为单位运行，每个周期由 $2(n-1)$ 轮次组成。机器人按照从“右”到“左”（即反向顺序）读取自身标识符的各个位，并执行以下操作。如果该位为 0，机器人不移动，在其节点停留 $2(n-1)$ 轮次。如果该位为 1，机器人就从端口编号 1 开始，依次访问所有邻居。由于访问一个邻居需要 2 轮次（前往该邻居再返回），机器人必须在 $2(n-1)$ 轮次内完成对所有邻居的访问。[7] 如果节点的度小于 $n-1$，机器人会在其节点等待，直到本周期剩余轮次结束。此外，由于标识符长度各不相同，机器人可能会比其他机器人更早扫描完自身标识符的所有位。机器人扫描完自身标识符的所有位后，会等待该过程结束——也就是 $an \log n$ 轮次，其中 a[8] 是某个较大的常数。无论是哪种情况，机器人都会等待，以便同步各周期的开始和结束时间。因此，每处理一位需要一个周期。由于我们假设标识符长度为 $O(\log n)$ 位，该过程会在 $O(n \log n)$ 轮次后停止。

上述过程保证两个相邻机器人的会合。由于机器人的 ID 位串不同，串中至少存在一个索引，使它们在该位置的位不同，即一个为 0，另一个为 1。对于第一个这样的索引，其中一个机器人不会

[7] 如果机器人已知图 Δ 的最大度，那么访问邻居可能需要 2Δ 轮次。不过，在某些情况下，假设 $\Delta = n-1$ 不会影响主要结果的渐近界。[8] 严格来说，a 的值应大于 b 在标识符范围 $[1, n^b]$ 内的值。如果不知道 a 的此类取值，则可将 a 设为 $\log \log n$。在这种情况下，以下所有运行时间界中的 $\log n$ 因子都将替换为 $(\log n \log \log n)$。为简单起见，我们假设 a 是一个较大的常数。

移动，而另一个机器人访问所有邻居。因此，它们会在那里相遇并集结。请注意，该过程并不保证任意一对相邻机器人都能集结，但能确保至少一对机器人集结，从而达到未分散配置。因此，我们得到以下引理。

▶**引理 9**。假设在分散配置中，两个机器人位于两个相邻节点上。那么经过 $O(n\log\ n)$ 轮次后即可达到未分散配置。

1 跳会合过程可以扩展到两个相距 i 跳数距离的机器人之间的会合。我们将该过程称为 i跳会合。

i跳会合： 同样地，该方法按周期运行，每个周期由$T(i) = \sum_{j=1}^{i} 2(n-1)^j$ 轮次组成。机器人通过从“右”向“左”（即逆序）查看其标识符的位来执行以下操作。如果该位为 0，机器人不移动，在其节点上停留 $T(i)$ 轮次。如果该位为 1，机器人将按照端口编号，通过深度优先搜索遍历访问距其 i 跳数距离以内的所有节点。同样地，如果某些节点的度小于 $n-1$，机器人会在其节点上等待该周期剩余的轮次。如果机器人扫描完其所有标识符位，还会在其节点上等待 $an\log\ n$ 轮次，其中 a 为某个较大的常数。

该过程需要 $(\sum_{j=1}^{i} 2(n-1)^j \log n)$ 轮次，即 $O(n^i \log\ n)$ 轮次。因此，我们得到以下一般性结果。

▶**引理 10.** 假设在分散配置中，两个机器人分别位于相距 i 跳的两个节点上。那么，需要 $T(i) = \sum_{j=1}^{i} 2(n-1)^j \log n = O(n^i \log n)$ 轮次才能到达未分散配置，其中$i \leq D$表示图的直径。

因此，由上述引理可知，当 $i > 5$ 时，i-Hop-Meeting 过程本身所需的时间比现有的聚集结果—— $\tilde{O}(n^5 \log n)$ 轮次——更长。对于 $i \leq 5$，我们的算法优于现有的最佳算法。

因此，对于任意数量的机器人以及机器人的任意初始分布，完整的算法如下。

算法 **Faster-Gathering：**

步骤 1： 每个机器人都运行未分散聚集算法。如果机器人的初始分布确实是未分散配置，那么聚集成功；如果机器人不是单独一人，则机器人终止（见引理 11）。否则，机器人不会移动，所有机器人都会在原来的位置（节点）停留 R 轮次。

(步骤 2 至 步骤 6)：对于 $i = 2, \ldots, 6$，执行以下步骤。
步骤 i： 经过 $(i-1)R$ 轮次后，如果尚未实现聚集（即初始分布是分散配置），每个机器人先运行 $(i-1)$ 跳会合过程，然后运行未分散聚集算法。如果至少有两个机器人位于相邻节点，则 1 跳会合会将初始分布转化为未分散配置。随后运行未分散聚集算法即可解决聚集问题。如果机器人不是单独一人，则每个机器人终止（见引理 11）。否则，机器人不会移动，所有机器人都会在各自的位置停留 R 轮次。

步骤 7： $6R$ 轮次后，如果尚未实现聚集，则使用第 2.1 节中介绍的 UXS 运行聚集算法。由于该算法适用于任意数量的机器人以及机器人的任意初始分布，因此聚集必定会通过检测实现。

现在我们证明，在算法的每个步骤结束时，所有机器人都会检测到聚集已经实现。

▶**引理 11.** 在 *Faster-Gathering* 算法前 *6* 步中任意一步结束时，如果某个节点上只有一台机器人，那么每台机器人都单独位于一个节点上。

证明。 前 6 步中的每一步都会在结束时调用 Undispersed-Gathering 算法。可能出现以下两种情况。

- 在某一特定阶段开始运行未分散聚集算法时，若机器人配置为分散配置，则所有机器人都是 $waiter$ 机器人，且不会执行任何操作。因此，由于分散配置始终不变，该步骤结束时每台机器人都单独位于一个节点上。在某一特定阶段开始运行未分散聚集算法时，若机器人配置为未分散配置，则根据定理 8，该步骤结束时没有机器人会单独位于一个节点上。◀

因此，根据上述引理可知，机器人可以通过检测是否有机器人单独位于一个节点上来判断是否已聚集。最后，第 7 步也保证了带检测的聚集。

Faster-Gathering 算法能够将机器人正确地聚集到单个节点，检测聚集是否完成并终止。利用 Undispersed-Gathering 和 i-Hop-Meeting 的时间界，可以轻松同步每个步骤。因此，该算法的时间复杂度可写为 $O(\min\{R + T(i)\ \tilde{O}(n^5)\}$ 轮次，对于$i = 0$，1，2，.....，5，其中 $T(0) = 0$ 表示非分散分布，$T(i) = O(n^i \log\ n)$，对于 $i = 1$，2，.....，5，详见引理 10；$\tilde{O}(n^5)$ 是使用 UXS 的聚集算法的时间复杂度，详见定理 6。下面给出主要结果。

▶**定理 12** (Faster-Gathering)。给定一个 n-节点、m-边的匿名图（无向且连通），并且 k 机器人任意分布在各节点上，则存在一种确定性算法，能够在以下时间内将所有机器人聚集到单个节点并检测聚集完成：

(i) $O(n^3)$ 轮次，如果机器人的初始分布是未分散的，或者已分散的，且至少有两个机器人之间的距离为2 。**(ii)** $O(n^i \log n)$ 轮次，如果机器人的初始分布是已分散的，且至少有两个机器人之间的距离为 i，其中 $i = 3$，4 和5。**(iii)** 否则，$\tilde{O}(n^5)$轮次。

该算法要求机器人只需知道 n 的值。每个机器人需要 $O(M + m\log\ n)$ 位内存，其中 M 是实现 *UXS* 所需的内存。

▶ 备注 13. 我们指出，如果已知初始配置中的跳数距离信息，那么通过直接运行算法中的相应步骤，Faster-Gathering 算法可以更快地完成。

▶ 备注 14. 进一步指出，如果机器人知道图的最大度 Δ，那么 Faster-Gathering 的时间复杂度将变为 $O(\min\{R+\Delta^i \log n, \tilde{O}(n^5)\})$ 轮次，其中 $i \leq 5$。这可由 i-Hop-Meeting 过程得出；在该过程中，每个周期包含 $T(i) = \sum_{j=1}^{i} 2\Delta^j$ 轮次，用于访问一个节点的所有邻居。

现在，我们展示一个关键结果：当存在多个机器人时，无法在分散配置中将它们全部彼此远离。换句话说，如果机器人数量足够多，则其中至少有两个必须彼此相近（例如，彼此之间最多相距 5 跳）。这一结论源自下面陈述的一个更一般的结果。

► **引理 15.** 假设 $\lfloor \frac{n}{c} \rfloor + 1$ 个机器人任意分布在一个有 n个节点的图上，其中 c为任意常数。那么至少存在两个机器人，它们之间的跳数距离至多为 $2c-2$ 。

证明。机器人数量为 $\lfloor \frac{n}{c} \rfloor + 1 \geq 2$，因为 $c \leq n$。通过矛盾法，假设任意一对机器人之间的距离至少为 $2c-1$。我们证明这会导致矛盾。考虑一个机器人，比如 r，它位于某个节点 v，并令 v' 为另一个机器人的所在节点。由于图是连通的，v 必须与 v' 连通。令 v_1、v_2、$\cdots$、v_{c-1} 为从 v 到 v' 的一条最短路径上的前 $c-1$ 个节点。那么，v_1、v_2、$\ldots$、v_{c-1} 到 v 的距离分别为 1、2、$\ldots$、$c-1$。因此，根据假设，任何机器人 r' 都不能位于距离节点 v_1、v_2、$\ldots$、v_{c-1} 小于 c 的位置；否则，r 和 r' 之间的距离就会小于 $2c-1$。这意味着，对于任何机器人 r，都必定存在一条至少包含 $c-1$ 个节点的空闲路径，即这些节点上没有机器人（因为图是连通的）。容易看出，v_i 不可能位于其他任何机器人的空闲路径上，比如 r'。如果确实如此，那么 r' 就是一个距离 v_i 不超过 c 的机器人，这就产生了矛盾，因为 v_i 是 r 的一条空闲路径的一部分。

由于 r 是任意一个机器人，因此上述结论对所有机器人都成立。这意味着，对于每个处于某个节点上的机器人，至少存在一组这样的 $c-1$ 指定节点。

由于我们从 $\lfloor \frac{n}{c} \rfloor + 1$ 个机器人开始，因此有 $(c-1)\lfloor \frac{n}{c} \rfloor + (c-1)$ 个不同的节点与这些机器人相对应。此外，$\lfloor \frac{n}{c} \rfloor + 1$ 个机器人位于 $\lfloor \frac{n}{c} \rfloor + 1$ 个节点上。因此，在假设条件下放置 $\lfloor \frac{n}{c} \rfloor + 1$ 个机器人所需的图中节点总数为：$(c-1)\lfloor \frac{n}{c} \rfloor + (c-1) + \lfloor \frac{n}{c} \rfloor + 1 = c\lfloor \frac{n}{c} \rfloor + c \geq n+1$。这与图中的节点数为 n 矛盾。因此，我们关于任意一对机器人之间的距离至少为 $2c-1$ 的假设是错误的。证毕。◄

因此，根据上述结果，即定理 12、引理 10、引理11 和引理15，我们得到以下关于带检测的聚集的结果。

► **定理 16** (带检测的聚集)。给定一个含 n 个节点的匿名图（无向且连通），以及任意分布在各节点上的 k 个机器人，则这些机器人可以确定性地聚集到单个节点，并且每个机器人都能在以下时间内检测到聚集：

(i) $O(n^3)$ 轮次，如果 $k \geq \lfloor \frac{n}{2} \rfloor + 1$。**(ii)** $O(n^4 \log n)$ 轮次，如果 $\lfloor \frac{n}{3} \rfloor + 1 \leq k < \lfloor \frac{n}{2} \rfloor + 1$。
(iii) $\tilde{O}(n^5)$ 轮次，如果 $k < \lfloor \frac{n}{3} \rfloor + 1$。
该算法要求机器人只需知道 n 的值。每个机器人需要 $O(M + m \log n)$位内存，其中 M 是实现 *UXS* 所需的内存。

证明。我们逐一展示所有情况下的界限。
(i) 如果有 $k \geq \lfloor \frac{n}{2} \rfloor + 1$ 个机器人，那么引理 15 确保存在一对跳数距离不超过 2 的机器人。根据定理 12，聚集可在 $O(n^3)$轮次内实现，并且根据引理 11，每个机器人都会检测到它。**(ii)** 类似地，对于 $\lfloor \frac{n}{3} \rfloor + 1 \leq k < \lfloor \frac{n}{2} \rfloor + 1$，引理 15 确保存在一对跳数距离不超过 4 的机器人。根据定理 12 和引理11，带检测的聚集可在 $O(n^4 \log n)$ 轮次内实现。
(iii) 对于 $k < \lfloor \frac{n}{3} \rfloor + 1$，根据定理 12，聚集可在 $\tilde{O}(n^5)$ 轮次内实现。◄

因此可以得出结论：当系统中有许多机器人时，算法 Faster-Gathering 能够比现有结果更快地解决带检测的聚集问题。具体而言，如果至少有 $\lfloor \frac{n}{3} \rfloor + 1$ 个机器人，那么我们的算法所需轮次不超过 $O(n^4 \log n)$，便能完成聚集，显著快于任何现有算法。

机器人（带检测）。如果机器人数量超过 $\lfloor \frac{n}{2} \rfloor + 1$，那么我们的算法表现会更好——只需 $O(n^3)$ 轮次。请注意，尽管拥有许多机器人是实现更快的带检测聚集的充分条件，但并非必要条件。如果初始配置中图上有一对机器人之间的距离不超过 5，那么无论有多少个 k 机器人（$k \geq 2$），都能实现同样的效果。这可由定理 12 得出。

3 结论与未来工作

本文研究了图中机器人带检测的聚集问题。我们提出了一种确定性算法，在满足某些条件的情况下，其运行速度快于所有已有的该问题确定性算法，甚至快于更简单的无检测聚集问题算法。

不过，要使该算法能够运行，我们假设所有机器人同时唤醒。一个有趣的未来研究方向是，探究即使机器人在任意时刻唤醒，我们能否利用这种借助多个机器人加速聚集的方法。此外，如果降低机器人的通信能力，例如仅允许它们发出蜂鸣或保持静默，是否仍能设计出快速算法，也值得研究。最后，可以尝试为少量机器人设计更快的确定性聚集算法，例如，探究对于常数个（或多对数个）机器人，是否存在一个 $o(n^5)$ 时间算法。

一个值得探讨的未来研究方向如下：我们不限制机器人在节点处交换的消息大小。考虑限制消息大小的模型，并研究由此对运行时间产生的影响，将会很有意思。

参考文献

1 Steve Alpern。会合搜索：个人视角。*Operations Research*，50(5):772–795，2002。2 John Augustine 和 William K. Moses Jr. 移动机器人分散：内存—时间权衡研究。*CoRR*，abs/1707.05629，2018。3 John Augustine 和 William K. Moses Jr. 移动机器人分散：内存—时间权衡研究。载于第*19*届分布式计算与网络国际会议论文集，第1–10页，2018。4 Claudine Badue、Rânik Guidolini、Raphael Vivacqua Carneiro、Pedro Azevedo、Vinicius B. Cardoso、Avelino Forechi、Luan F. R. Jesus、Rodrigo Ferreira Berriel、Thiago M. Paixão、Filipe Wall Mutz、Lucas de Paula Veronese、Thiago Oliveira-Santos 和 Alberto F. De Souza。自动驾驶汽车：一项综述。*Expert Syst. Appl.*，165:113816，2021。doi:10.1016/j.eswa.2020.113816。5 Evangelos Bampas、Leszek Gasieniec、Nicolas Hanusse、David Ilcinkas、Ralf Klasing 和 Adrian Kosowski。转子路由器模型中的欧拉回路锁定问题：我选择指针，你选择端口编号。载于*DISC*，第423–435页，2009。6 L. Barriere、P. Flocchini、E. Mesa-Barrameda 和 N. Santoro。在网格中均匀散布自主移动机器人。载于*IPDPS*，第1–8页，2009。7 Sébastien Bouchard、Yoann Dieudonné 和 Andrzej Pelc。想聚集？不必喋喋不休！载于第*39*届分布式计算原理研讨会论文集，第253–262页，2020。8 Mark Cieliebak、Paola Flocchini、Giuseppe Prencipe 和 Nicola Santoro。移动机器人的分布式计算：聚集。*SIAM*计算机杂志，41(4):829–879，2012。

9 Mark Cieliebak 和 Giuseppe Prencipe。自主移动机器人的聚集。载于第9届结构信息与通信复杂性国际研讨会（SIROCCO）论文集，第57–72页，2002。10 Reuven Cohen、Pierre Fraigniaud、David Ilcinkas、Amos Korman 和 David Peleg。有限自动机引导的带标签图探索。ACM算法汇刊，4(4):42:1–42:18，2008年8月。11 Reuven Cohen 和 David Peleg。通过重心算法实现机器人收敛。载于第11届结构信息与通信复杂性国际研讨会（SIROCCO）论文集，第79–88页，2004。12 Alejandro Cornejo 和 Fabian Kuhn。利用蜂鸣部署无线网络。载于分布式计算国际研讨会论文集，第148–162页。施普林格，2010。13 Shantanu Das。分布式计算中的移动代理：网络探索。EATCS公报，109:54–69，2013。14 Shantanu Das。移动代理的图探索。载于《移动实体的分布式计算》，第403–422页。施普林格，2019。15 Bastian Degener、Barbara Kempkes、Tobias Langner、Friedhelm Meyer auf der Heide、Peter Pietrzyk 和 Roger Wattenhofer。具有有限可视范围的自主机器人同步聚集的紧确运行时间界。载于第23届ACM并行算法与体系结构年度研讨会（SPAA）论文集，第139–148页，2011。16 Dariusz Dereniowski、Yann Disser、Adrian Kosowski、Dominik Pajak 和 Przemyslaw Uznański。快速协同图探索。Inf. Comput.，243(C):37–49，2015年8月。17 Anders Dessmark、Pierre Fraigniaud、Dariusz R. Kowalski 和 Andrzej Pelc。图上的确定性会合。Algorithmica，46(1):69–96，2006。18 Yoann Dieudonné、Andrzej Pelc 和 David Peleg。历经捣乱仍实现聚集。ACM算法汇刊，11(1):1:1–1:28，2014。19 Yann Disser、Frank Mousset、Andreas Noever、Nemanja Škorić 和 Angelika Steger。协同树探索的一般下界。载于结构信息与通信复杂性国际研讨会论文集，第125–139页。施普林格，2017。20 Yotam Elor 和 Alfred M. Bruckstein。环上的均匀多代理部署。理论计算机科学，412(8-10):783–795，2011。21 Samir Elouasbi 和 Andrzej Pelc。利用蜂鸣实现带检测的确定性会合。国际计算机科学基础期刊，28(01):77–97，2017。22 Paola Flocchini。聚集。载于《移动实体的分布式计算》，第63–82页。施普林格，2019。23 Pierre Fraigniaud、Leszek Gasieniec、Dariusz R. Kowalski 和 Andrzej Pelc。集体树探索。Networks，48(3):166–177，2006。24 Pierre Fraigniaud、David Ilcinkas、Guy Peer、Andrzej Pelc 和 David Peleg。有限自动机的图探索。理论计算机科学，345(2-3):331–344，2005年11月。25 Seth Copen Goldstein、Jason D. Campbell 和 Todd C. Mowry。可编程物质。Computer，38(6):99–101，2005。26 Dariusz R. Kowalski 和 Adam Malinowski。如何在匿名网络中会合。Theoretical Computer Science，399(1-2):141–156，2008。27 Evangelos Kranakis、Danny Krizanc 和 Euripides Markou。环上的移动代理会合问题。分布式计算理论讲义，1(1):1–122，2010。28 Ajay D. Kshemkalyani、Anisur Rahaman Molla 和 Gokarna Sharma。任意图上的快速移动机器人分散。载于 ALGOSENSORS，2019。29 Danny B. Lange 和 Mitsuru Oshima。使用移动代理的七个好理由。ACM通讯，42(3):88–89，1999。30 Artur Menc、Dominik Pajak 和 Przemyslaw Uznanski。转子路由器图探索的时间与空间最优性。Inf. Process. Lett.，127:17–20，2017。31 Avery Miller 和 Andrzej Pelc。会合与寻宝中成本和信息之间的权衡。J. Parallel Distributed Comput.，83:159–167，2015。

32 Anisur Rahaman Molla、Kaushik Mondal 和 William K. Moses Jr. 图上的拜占庭分散。载于第35届 IEEE 国际并行与分布式处理研讨会（IPDPS 2021），第942–951页。IEEE，2021。33 Anisur Rahaman Molla 和 William K. Moses Jr. 移动机器人的分散。载于第23届国际分布式计算与网络会议，第217–220页，2022。34 Christian Ortolf 和 Christian Schindelhauer。树的多机器人探索递归方法。载于结构信息与通信复杂性国际研讨会，第343–354页。施普林格，2014。35 Andrzej Pelc。确定性会合算法。载于《移动实体的分布式计算》，第423–454页。施普林格，2019。36 Pavan Poudel 和 Gokarna Sharma。网格中的时间最优均匀散布。载于 ICDCN，第228–237页，2019。37 Giuseppe Prencipe。一组自主移动机器人无法实现聚集。理论计算机科学，384(2-3):222–231，2007。38 Thomas C. Schelling。《冲突的策略》。牛津，1960。39 Zhiguo Shi、Jun Tu、Qiao Zhang、Lei Liu 和 Junming Wei。群体机器人系统综述。载于国际群体智能会议，第564–572页。施普林格，2012。40 Masahiro Shibata、Toshiya Mega、Fukuhito Ooshita、Hirotsugu Kakugawa 和 Toshimitsu Masuzawa。异步环中移动代理的均匀部署。载于 PODC，第415–424页，2016。41 Ichiro Suzuki 和 Masafumi Yamashita。分布式匿名移动机器人：几何图案形成。SIAM计算机杂志，28(4):1347–1363，1999。42 Amnon Ta-Shma 和 Uri Zwick。确定性会合、寻宝与强通用探索序列。载于《离散算法研讨会：第十八届 ACM-SIAM 离散算法年度研讨会论文集》，第7卷，第599–608页，2007。43 Amnon Ta-Shma 和 Uri Zwick。确定性会合、寻宝与强通用探索序列。ACM算法汇刊，10(3):12:1–12:15，2014。