# 对抗性缺陷视图模型下自主移动机器人的聚集

Prakhar Shukla[1], Seshunadh Tanuj Peddinti[2] and Subhash Bhagat[1]

[1]*Indian Institute of Technology Jodhpur, India*
[2]*Sardar VallabhBhai National Institute of Technology Surat, India*
*Email: p23ma0011@iitj.ac.in[1], sbhagat@iitj.ac.in[1], i22ma045@amhd.svnit.ac.in[2]*

arXiv:2603.05788v1 [cs.DC] 6 Mar 2026

**本文研究在分布式 *Look–Compute–Move*模型下，于欧几里得平面中运行的一组 $N \geq 2$ 自主移动机器人的聚集问题。我们考虑在对抗式 *defected view*模型下运行的无记忆机器人；在该模型中，由于对抗性可见性故障，被激活的机器人可能只能观察到部分机器人。因此，每个观察阶段获得的信息都可能不完整且动态变化。我们的目标是在此类感知限制下，保证机器人能在有限时间内确定性地聚集到一个事先未知的位置。我们在两种不同的调度假设下提出了两种分布式算法。在完全同步（$\mathcal{FSYNC}$）模型下，我们证明了在对抗性$(4,\ 2)$ 缺陷视图环境中可以实现有限时间聚集，从而解决了一个先前悬而未决的情形，且无需额外能力或坐标一致性。在异步（$\mathcal{ASYNC}$）模型下，我们证明了在一般的对抗性$(N,\ K)$ 缺陷视图模型中可以实现有限时间聚集；在该模型中，对于任意 $1 \leq K < N-1$，被激活的机器人至多能观察到其余 $N-1$ 机器人中的 $K$。两项结果在非刚性运动条件下均成立。针对 $\mathcal{ASYNC}$模型提出的算法假设各机器人对一个坐标轴的方向和定向达成一致。**

*Keywords: Swarm robotics, Gathering Problem, Distributed algorithms, Autonomous and asynchronous robots, Oblivious robots, Defected View Model, One-axis agreement*

## 1. 引言

群体机器人学研究大量简单的自主机器人如何在不依赖集中式控制或全局监督的情况下，协同完成任务。受蚁群、鸟群和鱼群等自然系统的启发，群体机器人系统强调去中心化、可扩展性、鲁棒性和容错性。群体中的每个机器人通常都只有有限的感知、计算和通信能力；然而，通过局部交互，整个机器人群能够表现出复杂且可靠的集体行为。

凭借这些特点，机器人群系统尤其适合部署在人类难以进入、存在危险或难以预测的环境中。具有代表性的应用领域包括在放射性区域或污染区域开展作业、农业监测、安全与监控、搜救任务，以及探索未知或非结构化环境。此外，使用简单且相同的机器人使此类系统具有成本效益且易于扩展，从而能够在需要时进行大规模部署。

在群体机器人学中，机器人通常被假定为**匿名**的，也就是说，它们没有唯一标识符，无法根据外观或行为彼此区分。它们通常被建模为**无记忆**，不会保留过去动作或观测的记忆，并且只根据对环境的当前感知行动。机器人不会通过交换消息进行显式通信；相反，它们通过观测其他机器人的位置来实现隐式协调。这些假设显著限制了单个机器人的能力，使协调问题既具有挑战性，也在理论上颇具研究价值。

一种广泛研究的机器人行为抽象是**观察–计算–移动**周期。在该模型中，活动机器人首先观察其他机器人的位置（**观察**），然后根据观测构型计算目的地（**计算**），最后朝计算出的目的地移动（**移动**）。机器人激活和移动通常由对抗性调度器控制。在**异步(**$\mathcal{ASYNC}$**)**模型中，激活彼此独立地发生，且每个阶段的持续时间有限但不可预测。

该模型捕捉了现实中的时间不确定性，被认为是最一般且最具挑战性的执行模型之一。

机器人协调的另一个重要方面是运动假设。在**刚性**运动模型下，机器人总能通过一次移动到达预定目的地。相比之下，**非刚性**运动模型允许对抗性调度器在机器人到达目标之前中断其移动，但保证机器人至少移动最小距离 $\delta > 0$，从而取得进展。非刚性模型更贴近现实，也显著增加了算法设计和正确性证明的难度。

在这一框架下，聚集问题 是群体机器人学和分布式计算中最基本的协调问题之一。简单来说，聚集要求系统中的所有机器人移动到一个事先未知的共同位置。聚集是完成图案构造、领导者选举和协作探索等更复杂任务的基础操作。因此，人们已在可见性、同步性、记忆、通信、重数和坐标一致性等各种假设下，对这一问题进行了广泛研究。

尽管已有大量相关研究，聚集问题的许多变体仍颇具挑战，尤其是在机器人处于对抗式感知条件下运行时。理解聚集问题可解所需的最小假设，对于理论认识和实际系统设计都至关重要。因此，有必要研究在更弱模型下的聚集问题：在这些模型中，尽管可见性、一致性和运动能力受到严重限制，机器人仍必须实现协调。

本文研究可见性故障模型下的聚集问题，并探讨自主机器人群协调能力的基本界限。此类可见性故障可能由多种因素引起，例如内存故障、有限感知能力、硬件故障或暂时性的环境干扰。

### 1.1. 相关工作

群体机器人学中存在许多几何协调问题。在过去二十年里，研究人员研究了若干基本的队形问题，包括聚集、收敛、圆形构造、图案构造、群集和平面构造 [2, 13, 14, 18, 20, 24, 25, 35]。其中，聚集问题是分布式移动机器人学中最基本且研究最广泛的问题之一。

人们已在许多不同的计算模型和故障假设下研究聚集问题 [23]。这项研究的一个关键目标是确定机器人成功聚集所需的最低能力，并考察同步性、可见性、记忆、通信和容错性的不同水平。下面，我们根据调度模型。

- **无限可见性模型：**大多数现有研究都假设机器人具有无限可见性。我们首先回顾这一假设下的研究成果。
  - 二维欧几里得平面

    **完全同步模型：** 在完全同步（FSYNC）模型中，无需任何额外假设即可解决聚集问题[22]。Agmon 和 Peleg 提出了一种容错聚集算法，能够容忍多达 $\lceil \frac{n}{3} \rceil - 1$ 个拜占庭故障 [2]。Flocchini等人研究了部署在 $\mathbb{R}^2$闭曲线上的机器人的聚集和领导者选举问题，并在记忆、通信、可见性、定向和速度等方面采用不同假设，提出了若干随机化算法 [21]。

    **SSYNC模型：** 对于 $n = 2$，聚集问题被称为会合问题。Suzuki和Yamashita证明，在局部坐标系之间不存在任何一致性时，即使具备强多重性检测，会合也是不可能的 [31]。后续研究表明，在半同步模型中，当机器人配备仅使用两种颜色的灯时，会合问题是可解的[34]。对于 $n > 2$，Prencipe证明，在没有多重性检测且局部坐标系之间不存在任何一致性的情况下，聚集是不可能的 [30]。

    聚集问题也在崩溃故障 模型下得到了研究。Agmon和Pe- leg提出了首个容错聚集算法，该算法在弱多重性检测下能够容忍一次崩溃故障 [2]。Bramas和Tixeuil随后提出了一种能够容忍任意数量崩溃故障的算法，其前提是具备强多重性检测 [8]。Bha- gat和Mukhopadhyaya提出了两种基于弱多重性检测的容错算法 [5]。Pattanayak 等通过允许引导轨迹扩展了这些结果，使系统在弱多重性检测下能够容忍任意数量的故障[29]。Izumi等提出了一种面向具备强局部多重性检测的无记忆机器人的常数轮随机聚集算法 [26]。D´efaGo等在各种故障和调度器模型下建立了不可能性结果并提出了随机化解法 [16]。

    **ASYNC模型：** Cieliebak等人在假设机器人具备弱多重性检测能力的情况下，解决了异步机器人的聚集问题 $n > 2$ [13]。Bhagat等人提出了一种在可见性受阻条件下的容错聚集算法，该算法假设机器人能够达成单轴一致，并可容忍任意数量的

崩溃故障 [6]。Bhagat 和 Mukhopadhyaya 研究了存在多边形障碍物的环境中的聚集问题，并针对非旋转对称的初始配置提出了解决方案[7]。Pattanayak等人在 ASYNC$_{IC}$模型下提出了一种随机化会合算法，假设已知机器人之间的速度比。Cicerone等人研究了一种涉及预定义会合点的聚集问题变体，并提出了多种算法，其中一种优化了极小极大距离准则 [12]。

聚集问题也针对胖机器人（具有物理尺寸的机器人）进行了研究。Czyzowicz等人解决了 $n = 4$ 胖机器人 [15], 的这一问题，而 Agathangelou等人则在假设具有共同手性的前提下，将这一结果推广到了任意数量的机器人 [1]。

– 三维欧几里得空间
在不同的计算模型下，研究者也在三维欧几里得空间中研究了聚集问题和平面形成问题。Bhagat等人研究了 $\mathbb{R}^3$ 中不透明机器人的聚集问题，并提出了两种假设单轴一致性的算法：一种适用于无故障机器人的 ASYNC 模型，另一种适用于能够容忍任意数量崩溃故障的 SSYNC 模型 [4]。针对不同模型，也有人提出了若干平面形成问题的算法 [33, 32, 35]。

• **有限可见性模型：** 关于受限可见性条件下的聚集问题，目前仅有少数研究成果。

– 二维欧几里得平面
Flocchini 等人研究了有限可见性条件下异步机器人的聚集问题，并假设它们在两个坐标轴上达成一致[19]。Chatterjee 等人为具有非均匀可见范围的机器人提出了一种解决方案 [10]。随后，S. Chaudhuri等人在假设单轴一致性和刚性移动的条件下，提出了一种有限轮次的分布式算法 [11]。– 三维欧几里得空间 Braun 等人研究了有限可见性条件下同步机器人的聚集问题，并在假设刚性移动的前提下，给出了所需轮次数的上下界 [9]。Di Luna 等人提出了一种分布式算法，机器人利用角度可见性在圆的边界上聚集 [17]。Yamauchi 等人研究了三维欧几里得空间中有限可见性机器人的图案构造 [36]。

• **缺陷视图模型：** 与有限可见性模型和无限可见性模型不同，缺陷视图模型允许机器人因感知中的对抗性缺陷而忽略对某些机器人的观测，无论它们相距多远。Kim等人最先提出了这一模型 [27] ，并针对在对抗式距离型（$N$，$K$）缺陷视图模型下完全同步运行的机器人，提出了一种聚集算法。他们还证明了（3，1）模型以及对抗式（$N$，$K$）模型的放宽版本存在不可能性结果。然而，在对抗式（4，2）缺陷视图模型下，聚集问题的可解性仍未解决 [28]。最近，Aliberti等人重新研究了在 SSYNC 调度器下运行的机器人所面临的这一问题，并得出了两个新的不可能性结果 [3]。

### 1.2. 动机与贡献

当自主移动机器人的感知能力不可靠时，协调它们是一项具有挑战性的任务。在许多现实场景中，由于障碍物、干扰、感知故障或记忆故障，机器人可能无法观测到所有其他机器人。为描述这类情况，我们考虑近期引入的缺陷视图模型 [28], ，在该模型中，活动机器人可见的机器人集合由对抗方选择。这显著增加了协调的难度，因为机器人必须依据不完整且动态变化的信息做出决策。一个自然且基本的问题是：在如此严重的可见性缺陷下，使用最低限度的机器人能力，是否仍能实现聚集?

我们的第一项贡献涉及 [28]中提出的对抗式（4,2）缺陷视图模型下聚集问题的一个基本且此前尚未解决的情形。在对抗式（4,2）模型下，每个活动机器人只能观测其他三个机器人中的两个。我们分析了由部分且不断变化的观测所带来的困难，并提出一种分布式聚集算法，保证在有限时间内完成聚集。所提出的算法运行于**完全同步**系统中，且不假设任何额外能力，例如记忆、多重性检测或坐标系上的一致性。该结果确立了缺陷视图模型最小非平凡实例上聚集的可解性。

本文的另一项贡献是研究对抗式（$N, K$）缺陷视图模型下包含 $N$ 个机器人的一般情形。在同一次执行中，不同机器人对应的参数 $K$ 可以在 1 到 $N - 2$ 之间变化；对于同一机器人，该参数在算法的不同次执行中也可能变化。我们为**异步**（$\mathcal{ASYNC}$）机器人设计了一种有限时间分布式算法，该算法适用于所有 $1 \leq K \leq N - 2$。因此，即使机器人每次激活时只能观察到其他机器人中的一个，聚集仍然是可能的。该算法仅假设机器人在局部坐标轴的方向和定向上达成一致。

| 调度器 | 观测 | 轴一致性 | 移动 | N | K | 聚集问题的结果 |
|---|---|---|---|---|---|---|
| FSYNC | 对抗式 | No | *Rigid* | $\geq 5$ | N-2 | 可在 $\mathcal{O}(1)$ 轮内解决。[28] |
| FSYNC | 基于距离 | No | *Rigid* | $\geq 4$ | N-2 | 可在 $\mathcal{O}(1)$ 轮内解决。[28] |
| Any | 宽松 | No | *Rigid* | Any | $\leq \left\lfloor \frac{N-1}{2} \right\rfloor$ | 不可解 [28] |
| SSYNC | Any | No | *Rigid* | Any | $\leq N-2$ | 不可解[3] |
| SSYNC | Any | No | *Rigid* | 4 | 2 | 甚至不同位置聚集也不可解 [3] |
| **FSYNC** | **对抗式** | **NO** | **非刚性** | **4** | **2** | **本文已解决** |
| **异步模型** | **对抗式** | **单轴** | **非刚性** | $\geq 3$ | **Any** | **本文已解决** |

表 1：缺陷视野模式下聚集问题的已有结果与新结果 l.

两种算法均采用非刚性移动模型，在该模型下，机器人的移动可能会受到对抗性干扰而中断，但能够保证机器人至少朝计算出的目的地前进一段最小距离。总体而言，这些结果表明，尽管存在对抗性可见性和移动约束，确定性聚集仍然可以实现，从而显著拓展了分布式机器人协调领域的现有成果。

## 2. 一般模型与问题定义

令 $\mathcal{R} = \{r_1, r_2, \ldots, r_N\}$ 表示由 $N \geq 2$ 个相同、匿名且无记忆的机器人组成的集合。机器人被建模为**点状**实体，初始时位于二维欧几里得平面 $\mathbb{R}^2$ 中的任意位置。每个机器人都具备基本的运动能力，可以在平面上自由移动。

- **执行模型**
  在任意时刻，机器人可以处于激活或未激活状态。处于激活状态时，机器人执行标准的**观察–计算–移动**周期。系统由对抗性调度器控制，该调度器决定机器人的激活时间以及各阶段的持续时间。

  机器人按照**非刚性移动模型**移动。具体来说，机器人计算出目标点后，调度器可能会在机器人到达目标点之前将其停止。不过，可以保证，在每个激活周期中，机器人至少会朝计算出的目标移动最小距离 $\delta > 0$，除非它更早到达目标点。

- **坐标系与可见性**
  机器人不共享全局坐标系。相反，每个机器人 $r_i$ 都配有自己的局部坐标系，其原点与自身当前位置重合。除非另有说明，不同机器人的局部坐标系在定向和缩放方面可能有所不同。

  机器人具有**无限可视范围**，这意味着在没有故障的情况下，一个机器人有可能观测到系统中的所有其他机器人。然而，在本研究中，机器人的观测会受到**可见性缺陷**的制约，从而限制在观察阶段中能够被感知的机器人集合。

- **构型与重数**
  多个机器人可能占据平面上的同一点。机器人不具备任何多重性检测能力。因此，如果多个机器人位于同一位置，观察到该位置的机器人会将其视为只有一个机器人占据。

  系统在时间 $t$ 的**构型**定义为当前由机器人占据的 $\mathbb{R}^2$ 中不同点的集合。我们将时间 $t$ 的构型记为

  $$\mathbb{P}(t) = \{p_1(t), p_2(t), \ldots, p_m(t)\},$$

  其中 $1 \leq m \leq N$，且每个点 $p_j(t) \in \mathbb{R}^2$ 都代表至少一个机器人占据的位置。为方便记号，有时我们用 $p_i$ 代替 $p_i(t)$，用 $p_i'$ 代替 $p_i(t')$。这种简化不会造成关于机器人位置的歧义。

- **(N, K)缺陷视图模型：**考虑一个由 $N$ 个小型机器人组成的系统。在活动机器人 $r_i$ 的观察阶段，该机器人无法观察到完整的构型 $\mathbb{P}(t) \setminus \{r_i\}$。相反，对手会从其他 $N-1$ 个机器人占据的位置集合中，任意选取至多 $K$ 个机器人位置，其中

  $$1 \leq K \leq N-2.$$

  如果不同的机器人位置数量（不包括 $r_i$）小于或等于 $K$，那么 $r_i$ 会观察到所有这些位置。否则，对手会选取恰好 $K$ 个位置组成的子集。

观测位置的子集可能在不同激活周期中发生变化，并且不对连续观测之间的一致性作出任何假设。该模型刻画了严重的对抗性感知限制：活动机器人无法控制其观测哪些机器人，并且可能在每个激活周期中感知到系统的不同局部视图。

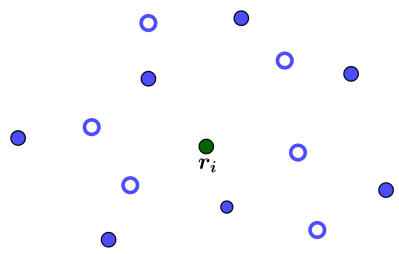



图1： 实心点表示活动机器人 $r_i$ 观测到的机器人，而 (o)表示 $r_i$ 遗漏的机器人。

### 3. 对抗性（4,2）缺陷模型

对抗式（4,2）-缺陷视图模型是在Kim 等人 [27], 的开创性工作中提出的，并被认定为一项开放问题。尽管基于距离的 （4,2）-缺陷视图下的聚集问题最近已被解决 [28], ，但对抗式变体仍未得到解决，即便在完全同步设置下亦然。在本节中，我们通过研究四个机器人在完全同步环境中于对抗式（4,2）-缺陷模型下运行的聚集任务来应对这一开放问题，并提出一种分布式算法来解决该问题。

令 $\mathcal{R} = \{r_1, r_2, r_3, r_4\}$ 表示部署在二维欧几里得平面上的四个匿名且相同的机器人组成的集合。在任意时刻 $t$，机器人 $r_i$ 的位置记为 $p_i(t)$，时刻 $t$ 所有机器人位置组成的集合记为

$$\mathbb{P}(t) = \{p_1(t), p_2(t), p_3(t), p_4(t)\}.$$

$\mathbb{P}(t)$ 的凸包记为 $CH(t)$。

我们将时刻 $t$ 构型的几何跨度 定义为凸包上任意两点之间的最大欧几里得距离:

$$\mathcal{G}(t) = \max\{\|a - b\| \mid a, b \in CH(t)\}.$$

量 $\mathcal{G}(t)$ 表示机器人构型的整体空间范围，并将在我们的分析中用作进度度量。

机器人在完全同步(FSYNC)模型下运行。也就是说，所有机器人在离散轮次中同时且瞬时地执行观察–计算–移动周期。每个机器人都是无记忆的，没有持久记忆，也不具备任何唯一标识符。

根据对抗式 (4,2)-缺陷视野模型，在每个观察阶段，活动机器人至多能观测到其余三台机器人中的两台的位置。对手控制观测哪些机器人，并且可以在不同机器人之间及不同轮次之间任意改变选择。观测机器人的视野中有哪些机器人被隐藏，它并不知情。

对于机器人 $r_i$，令 $\mathcal{O}_i(t)$ 表示其在时间 $t$ 观测到的位置集合，其中 $r_i$ 。由于机器人始终能观测到自己的位置，因此有

$$\mathcal{O}_i(t) = \{p_i(t)\} \cup \{p_j(t), p_k(t)\},$$

其中 $j$， $k \in \{1, 2, 3, 4\} \setminus \{i\}$ 和 $j \neq k$。例如，如果机器人 $r_1$ 观测到机器人 $r_2$ 和 $r_3$，那么

$$\mathcal{O}_1(t) = \{p_1(t), p_2(t), p_3(t)\}.$$

机器人无法检测重数。因此，如果两个或更多机器人占据同一位置，进行观测的机器人会将该位置感知为恰好被一个机器人占据。此外，当两个机器人（例如 $r_1$ 和 $r_2$）位于同一位置时，进行观测的机器人可能感知到 $r_1$ 或 $r_2$，但绝不会同时感知到两者。

为了说明对抗性观测所导致的不对称性，请看以下示例。假设机器人 $r_3$ 观测到机器人 $r_1$ 和 $r_4$，从而得到

$$\mathcal{O}_3(t) = \{p_1(t), p_3(t), p_4(t)\},$$

当机器人 $r_4$ 观测到机器人 $r_2$ 和 $r_3$时，得到

$$\mathcal{O}_4(t) = \{p_2(t), p_3(t), p_4(t)\}.$$

在这种情形下，每个机器人对同一个全局构型都有不同的局部视图，并且没有任何机器人仅凭其局部观测就能推断出完整集合 $\mathbb{P}(t)$。

图2展示了此类对抗性观测模式的一个示例。

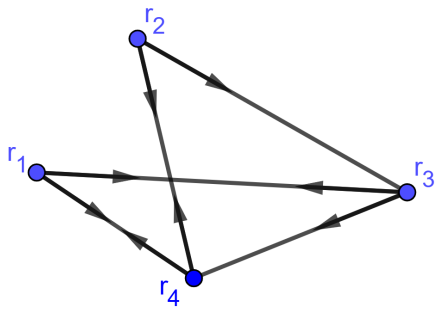



图2： 两个位置之间存在一条边，并不意味着它们可以相互观测。箭头表示机器人之间的观测关系。

如果机器人 $r_i$ 观测到共线的位置，并确定自身位置严格位于两个可见位置之间，那么 $p_i$ 被归类为内部位置。否则， $p_i$ 被归类为外部 位置。

在该模型下，我们提出一种有限时间分布式算法，用以解决聚集问题，且无需任何额外能力、事先一致性或机器人之间的显式协调。假设机器人按照非刚性运动模型移动，即对手可能

在机器人抵达其计算出的目的地之前中断其移动，同时仍确保最小的正向进展。

为了在算法执行过程中计算目的地，我们现在介绍若干定义和辅助函数，活动机器人仅依据其局部观测即可使用这些定义和函数。

• **等腰三角形的顶点和底边：**等腰三角形的**顶点**定义为两条等长线段的交点。与顶点相对的边被视为三角形的底边，如图3所示。

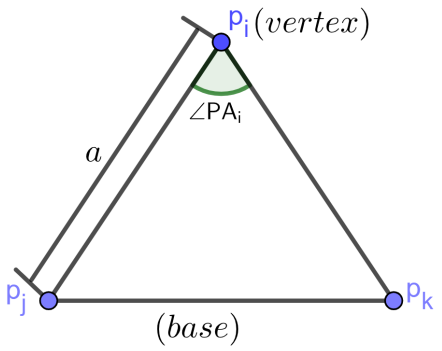



图3： 等腰三角形示例。

• **ComputeMidpoint()：**此函数以局部观测集 $\mathcal{O}_i(t)$ 以及相对于观测位置的机器人 $r_i$ 当前所在位置为输入，并返回 $\mathcal{O}_i(t)$ 中选定的一对位置所连接的线段的中点。

• **ComputeTriangle()：**此函数以观测集 $\mathcal{O}_i(t)$ 和机器人 $r_i$ 为输入，并构造由 $\mathcal{O}_i(t)$ 中三个位置形成的三角形。

• **计算最长线段()：**给定观测集 $\mathcal{O}_i(t)$，此函数返回由 $\mathcal{O}_i(t)$ 中所有位置形成的三角形的最长边。

• **ComputeVertexAngle()：**该函数以两条等长线段为输入，并返回它们在公共端点处的夹角（顶角）。对于机器人 $r_i$，此角度记为 $\angle PA_i$ ，如图3所示。

## 4. 算法1

### 4.1. 高层思路

高层思路。 在对抗式 (4,2)-缺陷视野模型中，每个机器人只能观测到经过对抗式选择的机器人位置子集，且无法就坐标达成一致，无法进行多重性检测，并且运动是非刚性的。因此，该算法设计为仅依赖根据每个机器人的观测集计算得出的局部几何信息。

在每个完全同步轮次中，机器人会分析由其观测集确定的几何构型，并将其归类为若干简单情况，例如单点、共线构型，或形成三角形的非共线构型。根据分类结果，机器人会确定性地选择一个目的地，该目的地由观测构型的特定几何特征定义，包括中点、质心或所选边的中点。

所有计算得到的目的地都位于观测位置的凸包内，机器人要么朝这些目的地移动，要么保持静止。因此，机器人绝不会远离观测构型移动，系统的整体空间跨度也不会增大。通过精心设计的等待规则来处理可能阻碍进展的对称构型，从而确保即使在对抗性缺陷观测下，对称性最终也会被打破。

尽管视野不完整且具有对抗性，这些局部规则的协同执行仍能保证机器人在可能的情况下反复缩小彼此间距。一旦所有机器人都处于这样一种构型：它们观测不到任何位于不同位置的其他机器人，就能正确推断聚集已经完成并终止。因此，对于对抗式(4，2)-缺陷模型下完全同步、无记忆的机器人，算法1确保它们在有限时间内聚集到一个单一且初始时未知的位置。

### 4.2. 算法1的详细描述

根据观测到的机器人位置数量，以及这些位置与自身当前位置共同形成的几何结构，每个活动机器人$r_i$ 都会在计算阶段确定其目的地，并在移动阶段朝该目的地移动。该算法将观测构型划分为有限种简单几何情形，例如单点、共线构型或形成三角形的非共线构型。对于每种情形，目的地都仅依据局部可观测的几何特征确定。这些移动规则能够确保持续朝聚集目标前进，同时在观测不完整且具有对抗性的情况下保持正确性。

• **情形-1 $|\mathcal{O}_i(t)| = 1$**：假设机器人 $r_i$ 在时间 $t$的观测阶段没有观测到任何其他机器人。在对抗性缺陷视图模型下，$r_i$ 推断不存在位于与 $p_i(t)$ 不同位置的机器人，因此认为 $p_i(t)$ 是期望的聚集点。因此， $r_i$停止后续执行。

• **情形-2 $|\mathcal{O}_i(t)| = 2$**：假设机器人 $r_i$ 位于点 $p_i(t)$，并且仅观测到一个位置，记为 $p_j(t)$，且该位置不同于 $p_i(t)$。利用函数ComputeMidpoint()，机器人 $r_i$ 计算线段 $p_i(t)p_j(t)$的中点 $m$ ，该线段连接其当前 $p_i(t)$与 $p_j(t)$所观测到的

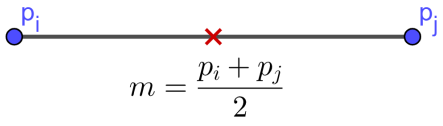



图4：（情况*1*示意图）。处于两个位置的机器人都可以计算出 $m$ 作为中点。

位置。在计算周期的移动阶段， $r_i$ 朝计算得到的中点移动，如图4所示。

• **情况3 $|\mathcal{O}_i(t)| = 3$**：假设机器人 $r_i$ 位于 $p_i(t)$，并观察到两个不同的机器人位置，即 $p_j(t)$ 和 $p_k(t)$，且二者均不同于 $p_i(t)$。令 $\mathcal{O}_i(t)$ 表示由这三个位置组成的集合。 $\mathcal{O}_i(t)$ 中的点可能共线，也可能不共线。我们的算法根据 $\mathcal{O}_i(t)$ 所形成的几何构型，确定 $r_i$ 的运动。相应的子情况如下所述。

– **情况3.1：**假设集合 $\mathcal{O}_i(t)$ 形成**共线构型**。首先，考虑机器人 $r_i$ 位于位置 $p_i(t)$ 且满足 $p_i(t) \in p_j(t)\,p_k(t)$ 的情况。在这种情况下，机器人 $r_i$ 使用辅助函数 ComputeMidpoint() 计算线段 $p_j(t)\,p_k(t)$ 的中点。随后，机器人将该中点设为目的地，并朝其进行非刚性运动。

现在考虑机器人位置 $p_i(t)$ 是共线构型端点之一的情况，也就是说， $p_i(t)$ 是一个极点。不失一般性，假设 $p_k(t)$ 是另一个极点。在这种情况下，机器人 $r_i$ 使用函数 ComputeMidpoint() 计算线段$p_i(t)p_k(t)$的中点，并通过执行非刚性运动朝其移动。图5展示了这种情况的示意图。

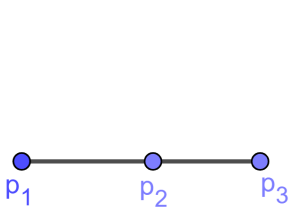



图5：（情况*3.1*的示意图）。位于位置 $p_1$、 $p_2$ 和 $p_3$ 的任一机器人都看不到位置 $p_4$ 处的机器人；此时 $\mathcal{O}_i$ 将共线于 $1 \le i \le 3$.

– **情况3.2：** 假设由集合 $\mathcal{O}_i(t)$所诱导的构型是**非共线**的。在这种情况下，三个位置构成一个三角形。这个三角形可能具有某些特殊几何性质。机器人 $r_i$ 使用辅助函数 ComputeTriangle()计算与 $\mathcal{O}_i(t)$相关联的三角形，该函数还会识别三角形的类型。根据这些几何性质，即使构型中存在对称性，该算法也能使机器人正确移动，同时不断减小构型的跨度。子情况的描述如下：

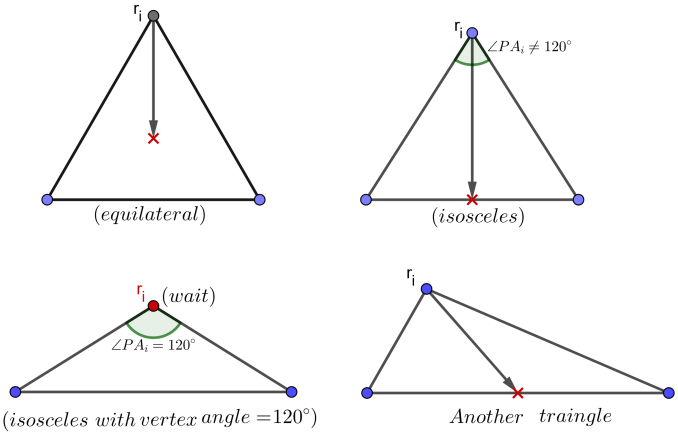



图6：（情况*3.2*的示意图）。如果 $\mathcal{O}_i(t)$ 不共线，则每个与 $\mathcal{O}_i(t)$ 相关联的活跃机器人 $r_i$ 都可能遇到以下构型类型之一。

∗ **情况3.2.1：**假设集合 $\mathcal{O}_i(t)$ 形成一个**等边三角形**。在这种情况下，机器人 $r_i$ 计算该等边三角形的质心作为其下一个目的地，并启动朝向该点的非刚性运动（见图6）。

∗ **情况3.2.2：**现在考虑集合 $\mathcal{O}_i(t)$ 构成一个**等腰三角形**的情况。假设机器人 $r_i$ 位于该三角形的顶点处，即 $p_i(t)$ 对应于顶点。在这种情况下，机器人 $r_i$ 使用辅助函数 ComputeVertexAngle() 计算顶角 $\angle PA_i$（如图3所示）。

如果顶角满足 $\angle PA_i$ =120°，那么机器人 $r_i$ 等待。当机器人的全局配置与图11所示的构型相符时，这一等待策略使算法能够打破对抗性缺陷观测所造成的对称性。如果$\angle PA_i \neq 120°$，那么机器人 $r_i$ 计算等腰三角形底边的中点，将其设为目的地，并在移动阶段朝着

计算出的中点进行非刚性运动。

如果机器人 $r_i$ 不位于等腰三角形的顶点处，也就是说， $p_i(t)$位于底边的一个顶点处，那么机器人 $r_i$会计算三角形底边的中点，并朝该点进行非刚性运动。

∗ **情况3.2.3：** 假设集合 $\mathcal{O}_i(t)$确定的三角形既不是等边三角形，也不是等腰三角形。在这种情况下，位于 $p_i(t)$的机器人 $r_i$首先使用辅助函数计算最长线段()计算三角形的最长边。设 $\mathcal{L}_i$为最长线段。随后，它计算该最长边的中点，并朝计算出的中点进行非刚性运动。

每个活动机器人都会独立执行算法1，所有机器人的协同执行可确保它们在有限时间内聚集到一个单一且预先未指定的位置。

### 4.3. 正确性

本文这一部分将证明主定理，从而确立算法1的正确性。具体而言，我们将证明，在对抗式（4、2）缺陷视图模型下，所提出的算法能够保证所有机器人在有限时间内聚集到单点。证明表明，尽管机器人存在缺陷视野且进行非刚性运动，同步机器人仍能实现聚集。

我们的分析通过确定执行过程中保持不变的关键性质，并证明系统会单调地朝着同一位置推进来展开。我们还证明，该算法不依赖任何额外能力，例如全局知识、显式通信、多重性检测，也不需要模型假设之外的额外一致性条件。因此，算法1在给定的对抗式设置下，为聚集问题提供了正确且独立完整的解决方案。

LEMMA 4.1. 如果机器人共线，那么至少有两个机器人会计算出相同的中点，所有共线的机器人都会在有限轮数内聚集于单点。

证明。 假设存在四个共线位置，分别为$p_1, p_2, p_3$ 和 $p_4$。如图7所示， $p_1$ 和 $p_4$ 位于角位置。在(4,2)-缺陷模型下，每个机器人都会以对抗式方式选择两个不同于自身所在位置的位置。根据它们的观测，

**情况1**。 如果所有机器人都处于外部位置，那么必然有 $\mathcal{O}_1 = \{p_1, p_2, p_3\}$或 $\{p_1, p_2, p_4\}$ 或 $\{p_1, p_3, p_4\}$，同样地， $\mathcal{O}_4 = \{p_1, p_2, p_4\}$或 $\{p_1, p_3, p_4\}$或 $\{p_2, p_3, p_4\}$，但在这种情况下，集合$\mathcal{O}_2 = \{p_2, p_3, p_4\}$，以及 $\mathcal{O}_3 = \{p_1, p_2, p_3\}$，

**算法1 ComputeDestination($r_i$) 要求：观测集 $\mathcal{O}_i$, 当前位置 $p_i$确保：目标点 $d_i$1：如果 $|\mathcal{O}_i| = 1$ 则 2：$d_i \leftarrow p_i$3：否则如果 $|\mathcal{O}_i| = 2$ 则 4：令 $p_j$ 为 $\mathcal{O}_i$中的另一个点 5：$d_i \leftarrow$ ComputeMidpoint($p_i, p_j$) 6：否则如果 $|\mathcal{O}_i| = 3$ 则 7：令 $p_j$、 $p_k$ 为 $\mathcal{O}_i$中的另外两个点 8：如果 $\{p_i, p_j, p_k\}$ 共线，则 9：令 $q_{\min}$、 $q_{\max}$ 为$\{p_i, p_j, p_k\}$的极值点 10：如果 $p_i \in \{q_{\min}, q_{\max}\}$ 则 11：令 $p_{\text{far}}$ 为 $p_i$ 中距离$\{p_j, p_k\}$最远的点 12：$d_i \leftarrow$ ComputeMidpoint($p_i, p_{\text{far}}$) 13：否则 14：$d_i \leftarrow$ ComputeMidpoint($q_{\min}, q_{\max}$) 15：结束如果 16：否则 17：如果 $\{p_i, p_j, p_k\}$ 构成等边三角形，则 18：$d_i \leftarrow$ ComputeCentroid($p_i, p_j, p_k$) 19：否则如果 $\{p_i, p_j, p_k\}$ 构成等腰三角形，则 20：令 $v$ 为该等腰三角形的顶点 21：令 $\{b_1, b_2\}$ 为与 $v$相对的底边端点 22：如果 $p_i = v$ 且 $\angle b_1 v b_2 = 120°$ 则 23：$d_i \leftarrow p_i$24：否则 25：$d_i \leftarrow$ ComputeMidpoint($b_1, b_2$) 26：结束如果 27：否则 28：令 $\{s_1, s_2\}$ 为 $\triangle p_i p_j p_k$最长边的端点 29：$d_i \leftarrow$ ComputeMidpoint($s_1, s_2$) 30：结束如果 31：结束如果 32：否则 33：$d_i \leftarrow p_i$34：结束如果 35：返回 $d_i$**

$r_1$ 要么计算线段 $p_1p_3$ 的中点，要么计算 $p_1p_4$的中点。$r_4$ 要么计算线段 $p_1p_4$ 的中点，要么计算 $p_2p_4$的中点。机器人 $r_2$ 和$r_3$ 分别计算线段 $p_2p_4$ 和 $p_1p_3$的中点。因此，如果 $r_1$ 和 $r_4$ 计算出相同的中点，证明即告完成。如果它们计算出不同的中点，那么 $r_1$ 或 $r_4$ 计算出的中点必定与 $r_2$ 或 $r_3$相同。

**情况2.** 如果只有一个非角点机器人（比如 $r_3$ ）发现自己位于 $p_3$ 处的内部 位置，那么 $r_3$ 计算

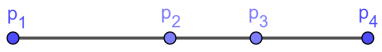



图7： $\mathbb{P}(t)$共线。

$p_1p_4$ 或 $p_2p_4$的中点，而 $r_1$、$r_2$、 $r_4$ 计算的中点与情况1相同。同样，对于所有可能的中点选择，我们可以得出结论：至少有两个机器人计算出相同的中点。

**情况3**。 如果两台机器人发现自己处于内部位置，那么 $r_2$ 会计算 $p_1p_3$的中点或 $p_1p_4$的中点， $r_3$ 会像情况2中那样计算中点，而 $r_1$ 和 $r_4$ 则像情况1中那样计算。根据所有可能计算出的中点，我们可以得出结论：至少有两台机器人计算出相同的中点。

由于没有机器人移动到线段 $p_1p_4$ 之外，且机器人 $r_1$、 $r_4$ 始终在线段内移动，因此线段长度单调递减，并会小于$2\delta$。在这种情况下，根据上述各情况，可以说每台机器人都会相应地计算中点，并在一轮内到达该点。现在只剩下3个位置。下一轮中，假设 $p_1'$、 $p_2'$和 $p_3'$ 是三个共线的位置，那么根据所提出的算法，位于这些位置的所有机器人都会计算出同一个中点并到达该点。这样，所有共线的机器人将在有限轮数内聚集到同一位置。

□

LEMMA 4.2.若4 个互异机器人位置的凸包不包含任何等腰三角形或等边三角形，则该构型至多有3条最长线。

证明。 设 $p_1, p_2, p_3$和 $p_4$ 为凸包 $CH()$中的四个机器人位置，由这些位置共可画出6条线段，如图8所示。每个观测集至多有3个互异元素。根据观测集，可形成 $4_{C_3}$ 个互异三角形。设这些三角形为 $\Delta_1 = \Delta_{123}$、$\Delta_2 = \Delta_{124}$、 $\Delta_3 = \Delta_{134}$、 $\Delta_4 = \Delta_{234}$。

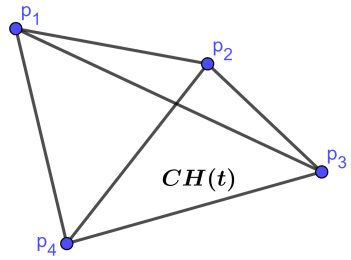



图8： （引理4.2的示意图）凸包 $CH(t)$ 既不含等边三角形，也不含等腰三角形。

为得出矛盾，假设存在4条最长线段，即每个三角形都有一条唯一的最长线。不失一般性，假设 $\mathcal{L}_1 = p_1p_3$、 $\mathcal{L}_2 = p_1p_4$、 $\mathcal{L}_3 = p_3p_4$、 $\mathcal{L}_4 = p_2p_4$ 是最长线段。 $\mathcal{L}_k$ 是三角形$\Delta_k$的最长线。设 $\mathcal{L} = \{\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_4\}$ 为最长线段的集合。根据一个关键的组合事实，每条边恰好出现在四个三角形中的两个里。如果 $\mathcal{L}_k$ 是三角形 $\Delta_k$的最长线。设$\Delta_{k'}$ 为包含边 $\mathcal{L}_k$的任意另一个三角形，则$\exists\mathcal{L}_{k'}$ 使得$|\mathcal{L}_{k'}| > |\mathcal{L}_k|$。由所有三角形，我们可以得出，

$$|\mathcal{L}_1| < |\mathcal{L}_2| < |\mathcal{L}_3| < |\mathcal{L}_4| < |\mathcal{L}_1| \tag{1}$$

由方程1可知， $|\mathcal{L}_1| < |\mathcal{L}_1|$ 这是不可能的。因此我们的假设是错误的；长度互异的最长线段至多有3条。

□

LEMMA 4.3. 如果所有机器人都是单个机器人且位于凸包的顶点上，那么至少有两个机器人会计算最长对角线的中点。

证明。 设4个机器人分别位于不同的位置，且它们的观测集既不构成等腰三角形，也不构成等边三角形。那么根据鸽巢原理，每条边恰好出现在四个三角形中的两个三角形里，因此必有一条最长对角线出现在两个三角形中，所以至少有两个机器人会计算最长对角线的中点。

□

LEMMA 4.4.如果所有机器人都是单个机器人且位于凸包上，那么至少有两个机器人会计算同一条最长线的中点。

证明。 通过类似于引理4.3的论证，我们可以证明这个引理。

□

LEMMA 4.5. 所有机器人位置的几何跨度 $\mathcal{G}(t)$ 单调递减，即对于任意 $t' > t$，跨度 $\mathcal{G}(t) \leq \mathcal{G}(t')$。

证明。在引理4.1中，我们已经看到线段的长度单调递减。现在，假设机器人不共线，那么根据所提出的算法，机器人要么朝凸包边上的中点移动，要么移至凸包内部。令$CH(t)$为时间$t$时所有四个位置的凸包。令$p_1(t)p_2(t)$为该凸包的一条边。首先，假设位于 $p_1(t)$的机器人移至凸包内部。设它停在某个位置，例如 $p_1(t')$，该位置位于 $CH(t)$内部。类似地，还有另一个顶点，例如$p_3(t)$，那么

$$||p_1(t') - p_3(t)|| \leq ||p_1(t) - p_3(t)|| \tag{2}$$

在另一种情形中，当机器人沿凸包的外边移动时，若 $p_1(t')$位于 $CH(t)$的边上，且处于顶点 $p_1(t)$和 $p_2(t)$之间，则根据凸性，我们有 $p_1(t') = \lambda p_1(t) + (1-\lambda)p_2(t)$，其中 $0 < \lambda < 1$。于是，对于任意顶点 $p_3(t)$

$$\begin{aligned} ||p_1(t') - p_3(t)|| &\leq (1-\lambda||p_1(t) - p_3(t')|| + \lambda||p_2(t) - p_3(t')|| \\ &< max(||p_1(t) - p_3(t')||, ||p_2(t) - p_3(t')||) \end{aligned} \tag{3}$$

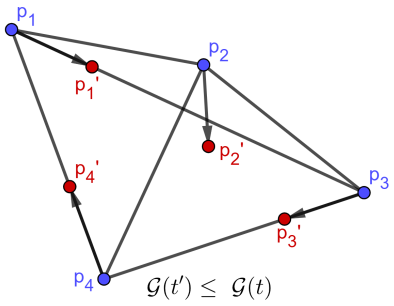



图9： 展示机器人以非刚性方式朝各自目标点移动的示例。

根据方程（2）、（3），我们可以得出：如果定义几何跨度的点对中包含两个机器人位置中的任意一个，那么移动后，我们有$\mathcal{G}(t') < \mathcal{G}(t)$。由于凸包中的机器人不会无限期等待，且至少存在两个位置定义几何跨度，因此我们可以得出，机器人位置的几何跨度单调递减，如图9所示。

□

LEMMA 4.6. 假设所有机器人均为单体。如果任意两个机器人位置之间的距离均小于或等于 $delta$，则最多经过三轮即可实现聚集。

证明。 根据引理4.5，机器人位置的任意构型都会单调递减并收敛到满足 $max||x - y|| \leq \delta$ 的构型，其中对所有 $x, y \in CH$ 均成立。这意味着每个机器人都会在同一周期到达计算出的目的地。现在，我们将讨论所有可能的情形。**如果有三条最长线段：** 如图所示，如果所有机器人都位于位置 $p_1$, $p_2$, $p_3$, $p_4$，并计算出相同的最长线段，那么它们会聚集到该最长线段的中点。在第二种情形中，假设 $r_1$ 和 $r_2$ 观察到三角形 $\Delta_{124}$，且假设 $r_3$ 和 $r_4$ 观察到三角形 $\Delta_{134}$，那么它们会分别到达线段 $p_2p_4$ 和 $p_1p_3$ 上的计算中点。下一轮中，只会有两个位置，每个位置上的机器人都会计算该线段的中点，并聚集到同一位置。现在，假设在对抗式(4,2)-缺陷视图模型下，每个机器人观察到的视野都不同，那么根据引理4.3，至少有两个机器人会针对同一条最长线段计算其中点。不失一般性，假设 $r_1$ 和 $r_2$ 针对同一条线段（例如 $p_2p_4$）计算其中点，而 $r_3$ 和 $r_4$ 针对例如 $p_1p_3$ 和 $p_3p_4$ 的直线计算中点并朝其移动；在下一轮中，所有机器人都会观察到同一个三角形，并计算出相同的目的地，从而实现聚集。在所有这些情形下，最多经过 2 轮即可实现聚集。

**如果有四条最长线段：** 根据引理4.2，必定存在一个等腰或等边三角形。现在，根据 [28]，中讨论的引理16，这种类型的构型可以在两轮内聚集。

**如果一个机器人位于等腰三角形或等边三角形内部：** 首先，假设位置 $p_4$位于图10中的等腰三角形 $\Delta p_1p_2p_3$ 内部（其顶角为 $\angle PA_1 \neq 120°$），那么如果 $r_1$ 观察到 $p_2$, $p_3$，则移动到线段 $p_2p_3$（底边）的中点。假设 $r_2$（分别是 $r_3$）观察到$p_1$, $p_4$，则移动到 $p_1p_2$（分别是$p_1p_3$）的中点；当 $r_4$ 移动时，它会伴随其中一个机器人，下一轮即可实现聚集。现在

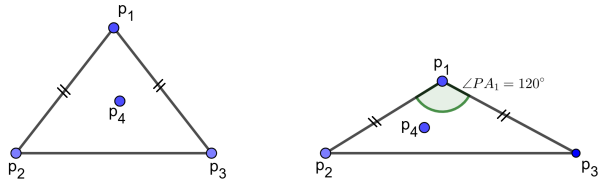



图10：左侧是一个普通的等腰三角形，右侧的三角形是一个顶角为 120°的等腰三角形。

假设位置 $p_4$ 位于等腰三角形$\Delta p_1p_2p_3$内部（其顶角为 $\angle PA_1 = 120°$），因此 $r_1$ 无法移动，那么如果 $r_4$ 观察到与 $r_2$ 或 $r_3$相同的情形，就会出现一个伴随位置；下一轮中，所有机器人都会看到这个等腰三角形，并移动到其底边的中点， $r_1$除外。第三轮中， $r_1$ 和所有其他机器人都会移动到剩余的单条线段的中点。类似地，对于 $p_4$ 不位于三角形中心的等边三角形，我们也可以得出相同结论。

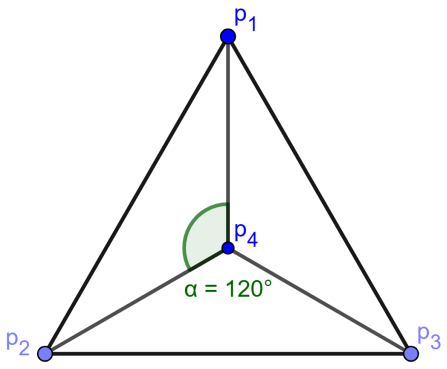



图11： （情况*3.2*示意图）一种配置，其中三个机器人位于等边三角形的顶点处，剩余的一个位于三角形的质心处。

考虑图11所示的情形： $p_4$ 位于等边三角形 $\Delta p_1p_2p_3$ 的中心。显然， $\angle p_1p_4p_3 = \angle p_2p_4p_3 = \angle p_1p_4p_2 = 120°$。假设 $r_1$ 观察到 $p_2$ 和 $p_3$，即等边三角形$\Delta_{123}$和 $r_2$, $r_3$ 观察到 $p_1$, $p_4$，而 $r_4$ 观察到 $p_2$, $p_3$。

显然， $\Delta_{124}$， $\Delta_{134}$ 和 $\Delta_{234}$ 构成一个等腰三角形，其顶点为 $p_4$ 和 $\angle PA_4 = 120°$。在这种情况下， $r_1$ 移动到 $p_4$ ，而 $r_2$ （分别是$r_3$）移动到底边$p_1p_2$（分别是$p_1p_3$）的中点， $r_4$ 则等待。假设 $r_2$ 和 $r_3$ 的新位置分别为 $p'_2$、 $p'_3$ 。 $\Delta_{12'3'}$ 又构成一个等腰三角形，其顶点为 $p_1$ 和 $\angle PA_1 = 120°$ ，因此机器人 $r_4$ 和 $r_1$ 等待，而 $r_2$， $r_3$ 在第二轮移动到底边$\Delta_{12'3'}$ 。第三轮中，位于两个位置的机器人都移动到中点，并在同一位置集合。□

根据引理4.1、4.5和4.6，可得以下定理。

THEOREM 4.1.算法*1*可在有限时间内，在对抗式(4,2)缺陷视野模型下聚集所有同步机器人。

## 5. 对抗式$(N, K)$缺陷视野模型下的聚集

本节研究一组 $\mathcal{R} = \{r_1, r_2, \ldots, r_N\}$ $N \geq 3$ 点状机器人在二维欧几里得平面中异步运行时的聚集问题。每个机器人都独立执行经典的观察–计算–移动周期，并采用**非刚性运动模型**；在该模型中，对手可以中断机器人向计算所得目的地的移动，但始终保证机器人取得最小的正向进展。

机器人在各自的局部坐标系中对 $Y$轴的方向具有共同的一致性，因此对北–南方向有一致的认知。然而，它们不具备**多重性检测** 能力，因此占据同一位置的多个机器人会被视为一个机器人。

该模型的一个关键特征是**对抗性缺陷视图**。在观察阶段，机器人可能无法观测到最多 $N - K$ 个其他机器人，这些机器人由对手选定。因此，在任何一次观测中，最多可见 $K$ 个机器人位置，其中包括机器人自身的位置。我们用 $\mathcal{O}$ 表示在观察阶段观测到的机器人位置集合。

令 $\mathbb{P}(t) = \{p_1(t)\ p_2(t) \ldots, p_m(t)\}$ 表示时间 $t$所有 $m \leq N$ 个不同机器人位置构成的集合，其中每个位置可以由一个或多个机器人占据。在该模型中，我们假设机器人对 $Y$轴的定向和方向有共同的认知。 $Y$轴的正方向视为北，负方向视为南。这种共同的一致性使我们能够根据机器人的 $y$坐标，按水平线对机器人位置进行排序。这些水平线的集合记作

$$\mathcal{L}(t) = \{L_1(t), L_2(t), \ldots, L_b(t)\},$$

其中 $|\mathcal{L}(t)| = b$ 是不同水平线的总数。该集合按从北到南的方向排列，即 $L_1(t)$ 是最北端直线，也称为最上水平线，而 $L_k(t)$ 表示从北数起的第 $k^{th}$ 条直线。 $L_k(t)$ 的基数记为 $|L_k(t)|$，表示任意时刻 $t$ 位于 $k^{th}$ 水平线上的机器人位置数量。图12(a)展示了这种排列的一个示例。

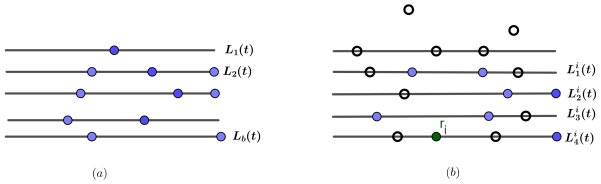



图12：*(a)* 显示了 $\mathbb{P}(t)$中机器人的排序，*(b)* 显示了$\mathcal{O}_i(t)$中机器人的排序（机器人 $r_i$ 在时间 $t$观测到的机器人）。

对于位于位置 $p_i(t)$ 的单个机器人 $r_i$ ，观测位置集合 $\mathcal{O}_i(t)$（包括其自身位置）同样可以排列成一组水平线，记为 $\mathcal{L}^i(t)$。这些直线按从北到南的顺序排列，其中 $L_k^i(t)$ 表示此顺序中的第 $k$ 条水平线，而 $|L_k^i(t)|$表示在图12(b)中所示的、 $L_k^i(t)$ 上对 $r_i$ 可见的机器人位置数量。

由于机器人视觉缺陷，观测到的结构可能与实际的全局配置不同。具体而言，如果 $r_i$ 没有观测到自身上方的任何位置，它就会得出结论：在时刻 $t$，自己位于最顶部的观测直线 $L_1^i(t)$ 上。类似地，如果某些角位置不可见，机器人可能会错误地认为自己位于观测直线的角位置。

如果观测集 $\mathcal{O}_i(t)$ 共线，那么 $r_i$ 会根据可见位置的相对顺序进一步对自身位置进行分类。如果 $r_i$ 发现自身位置严格位于另外两个可见位置之间，就将自身定义为处于内部 位置；否则，就将自身定义为处于外部 位置。

现在，我们介绍若干函数和符号，供设计算法2时使用。

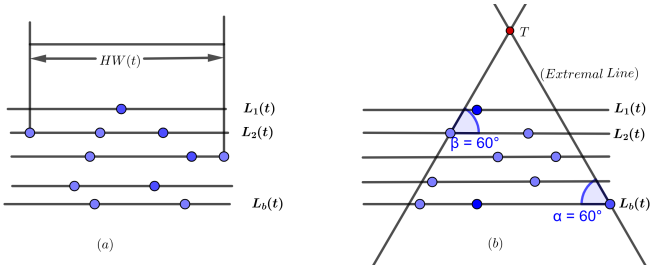



图13： *(a).* 展示了构型 $\mathbb{P}(t)$ 的水平宽度（$HW(t)$），*(b).* 展示了机器人位置（用于定义水平宽度和极值线）可能并不总是相同。

- **水平宽度** $HW(t)$**：** 对于时间 $t$ 时的任意配置 $\mathbb{P}(t)$，水平宽度定义为

图13(a)所示的， $\mathbb{P}(t)$中两个最边角位置之间的水平距离。

• **极值线：** 是最接近的 60°非平行直线，所有位于 $\mathbb{P}(t_0)$ 中的机器人都部署在这两条直线之间，而位于这些直线上的机器人的位置定义为**极值位置/点**，如图13(b)所示。

• **ComputeLevel()：** 对于任意机器人 $r_i$，此函数以 $\mathcal{O}_i(t)$作为输入，计算水平线集合 $\mathcal{L}^i(t)$，并按照从北到南的方向对这些直线排序。

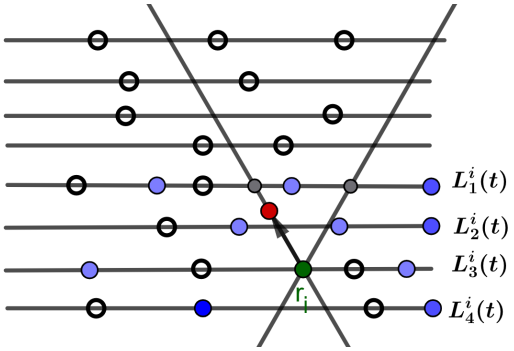



图14： 机器人 $r_i$ 在 $L_3^i(t)$ 中对自身进行排序，因为它在 $L_1(t)$ 中漏掉了一台机器人；随后计算 *Go* 线，并相应地朝直线$L_1^i(t) \simeq L_5(t)$上的交点移动。

• **ComputeGoLines()：** 如果机器人 $r_i$ 观察到水平线 $p_i(t)$上方有一个或多个机器人位置，它就会调用函数 ComputeGoLines()。该函数从 $p_i(t)$出发计算两条不同的 Go 线，使每条线与经过 $p_i(t)$的水平线形成一个按顺时针方向计算的 $60°$ 角。这些 Go 线即使在高度对抗性的情形下也能确保垂直进展，并防止 $r_i$ 因对抗性缺陷视图而无限振荡（如图14所示）。

• **计算角点：**对于任意机器人 $r_i$，此函数接收 $L_1^i(t)$，并返回最北水平线 $L_1^i(t)$ 上的角位置。

• **ComputePerpendicular()：**机器人 $r_i$ 使用函数 ComputePerpendicular()，从最北水平线 $L_1^i(t)$ 计算得到的角位置向其当前水平线作垂线。这样， $r_i$ 就能避免沿 Go-Line 不必要的移动。

•**计算交点：** $r_i$ 使用函数 ComputeIntSec()，计算垂线与 $Go-lines$的交点。

• **最近点（）：** 使用此函数， $r_i$ 比较两个点。它将这两个点作为输入，并返回距离 $p_i(t)$ （当前位置）最近的点。

## 6. 算法2

### 6.1. 算法的高层思路

在对抗式（$N$， $K$）缺陷视野模型中，每个机器人都受到严格限制。在任意一次激活期间，机器人可能无法观察到许多其他机器人；其视野可能在不同激活之间任意变化；其移动也可能被对手中断。此外，机器人是异步、无记忆的，且不具备多重性检测能力。这些限制使协调与聚集变得困难。

所提出的算法通过仅使用始终可在本地获取的简单几何信息来克服这些困难，并有助于在对抗式缺失的情况下保持垂直。机器人之间唯一共同的一致性是 $Y$ 轴的方向。借助这一共享方向，即使视野不完整，机器人也能一致地比较哪些观测位置在自身上方或下方。

从高层次来看，每个机器人都会根据自己能看到的位置的垂直排序来决定如何移动。该算法遵循简单的*Go-Line*策略，确保取得两种进展。首先，机器人会取得垂直进展：只要条件允许，活动机器人就会向北移动。这能确保机器人逐渐在更高的水平层上对齐。重要的是，即使机器人只能看到另一个机器人，也能取得垂直进展。

其次，该算法确保水平收缩。移动规则经过设计，使机器人群的水平宽度 $HW(t)$ 永远不会增大，并且通常会随着时间推移而缩小。机器人在移动时遵循所观察到的相对位置，因此自然会发生这种收缩。

该算法能够适应非刚性运动。即使机器人在到达目的地之前就停止，它也总会沿预定方向至少移动最小距离 $\delta > 0$。因此，每次激活都会确保沿着计算得出的最北水平线取得一定的垂直进展。所提出的 Go-Line策略可确保不会因视觉中的对抗性缺陷而发生振荡。

通过反复应用这些简单的局部规则，机器人群的构型会随时间逐渐收缩。一旦构型足够紧凑，聚集便可在有限时间内完成。因此，算法2仅利用在 $Y$-轴上的一致性，无需额外协调，即可保证异步无记忆机器人在对抗式（$N$， $K$）缺陷视野模式下实现聚集。

### 6.2. 算法2的描述

当机器人 $r_i$ 在时间 $t$ 变为活跃状态时，它会观测到一个局部构型 $\mathcal{O}_i(t)$，由于（$N$， $K$）缺陷视图模型，该构型可能不完整。利用对 $Y$ 轴方向的共同一致性， $r_i$

将包括自身在内的观测位置划分为若干条平行水平线，并按由南向北的顺序排列。根据自身在该排列中的位置以及观测构型所决定的几何结构，$r_i$ 选择合适的移动规则。根据由被观察机器人的数量及其在最北水平线上的分布情况所定义的不同子情况，$r_i$ 仅利用局部信息计算一个目标点，并朝其移动；尽管非刚性运动可能导致移动中断，它仍会取得至少为$\delta > 0$的最小进展。

• **情况 A** $|\mathcal{O}_i| = 1$**：** 机器人 $r_i$ 除了自己的位置外，没有观测到其他机器人位置。因此，$r_i$ 得出结论：所有机器人都位于 $p_i(t)$。因此，$p_i(t)$ 被确定为聚集点，且 $r_i$ 停止执行算法。

• **情况 B** $|\mathcal{O}_i| > 1$**：** 假设机器人 $r_i$ 观测到至少一个除自身位置以外的机器人位置。然后，机器人$r_i$ 使用函数 ComputeLevel() 将观测到的位置（包括自身位置）划分为水平层，并按由北向南的顺序排列。根据自身所在层相对于观测位置的情况，机器人 $r_i$ 按照算法中的相应规则开始移动。

– **情况 B(1)：机器人** $r_i$ **位于最上水平线上。** 设机器人 $r_i$ 位于位置 $p_i(t)$。如果机器人 $r_i$ 没有观测到任何严格位于经过 $p_i(t)$的水平线上方的机器人位置，则将该水平线确定为最上水平线，并记为 $L_1^i(t)$。

∗ **情况 B(1.1)：** 如果机器人 $r_i$ 观测到至少一个位于最上水平线 $L_1^i(t)$ 严格下方的水平线上的机器人位置，则 $r_i$ 等待。

∗ **情况 B（1.2）：** 假设机器人 $r_i$ 位于位置 $p_i(t)$，并观察到最上水平线 $L_1^i(t)$上的所有机器人位置。在这种情况下，机器人 $r_i$ 会进一步将其位置归为以下两类之一。

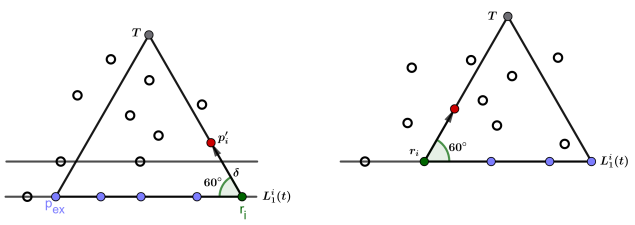



图15：左侧，$r_i$ 发现自己位于最上水平线上的外部位置，并计算等边三角形的顶点。右侧，$r_i$ 漏掉一个角点，发现自己位于外部位置，并计算目的地。

**（i）内部位置：** 如果机器人 $r_i$沿直线 $L_1^i(t)$在 $p_i(t)$的两侧都观察到至少一个机器人位置，那么 $p_i(t)$就被认定为内部位置。如果 $p_i(t)$是内部位置，那么 $r_i$ 保持静止。

**（ii）外部位置：** 若机器人 $r_i$ 沿直线 $L_1^i(t)$ 在 $p_i(t)$ 的至少一侧未观测到任何机器人位置，则将 $p_i(t)$ 识别为外部位置。设 $p_j(t)$ 为机器人 $r_i$观测到的另一个外部位置。机器人 $r_i$使用函数 ComputeTriangle()构造等边三角形 $\triangle p_i T p_j$，使得顶点 $T$ 严格位于直线 $L_1^i(t)$ 之上。机器人 $r_i$ 将 $T$设为目标点，并在移动阶段沿线段 $p_i(t)T$向点 $T$移动。

– **情况-B(2)：机器人** $r_i$ **不位于最上水平线上。** 设位于 $p_i(t)$ 的机器人 $r_i$在观测 阶段观测到一组位置。它将这些观测存储在 $\mathcal{O}_i(t)$ 中，并使用辅助函数ComputeLevel()对其进行排序。若在穿过 $p_i(t)$ 的直线$L_k^i(t)$ 严格上方的某条水平线上至少存在一个被观测到的位置，则 $r_i$ 将自身识别为位于第 $k^{\text{th}}$ 条水平线上。在此情况下，活动机器人 $r_i$ 计算**Go-Lines**$(60°)$，其由两条不同的射线组成，这些射线从 $p_i(t)$ 发出并向北，每条射线与 $L_k^i(t)$ 形成一个 $60°$ 顺时针角，如图 16所示。

随后，$r_i$ 根据相对于计算出的 Go 线的、位于最北侧（最上水平线）的水平线 $L_1^i(t)$ 上可见机器人位置的几何构型，确定其路径：

– **情况-B(2.1)：** 假设机器人 $r_i$ 仅观察到一个位置，即最上水平线上的 $p_j(t)$，也就是$|L_1^i(t)| = 1$。如果机器人位置 $p_j(t)$ 位于机器人 $r_i$的垂直轴上，则 $r_i$ 将$p_j(t)$ 设为目的地，并沿其垂直轴朝向 $p_j(t)$ 执行非刚性运动。

现在假设 $p_j(t)$ 位于 $r_i$ 的垂直轴上。然后，机器人 $r_i$ 使用辅助函数 ComputeGoLines() 计算 Go 线。如果 $p_j(t) \in L_1^i(t)$，使得 $p_j(t)$ 位于 Go 线之外，那么机器人 $r_i$ 计算 Go 线与最上方水平线 $L_1^i(t)$ 的交点。机器人 $r_i$ 将距离 $p_j(t)$ 最近的交点设为目的地，并沿相应的 Go-Line 向其移动。

现在考虑位置 $p_j(t) \in L_1^i(t)$ 位于 Go 线内部的情况。为避免不必要的移动并确保水平跨度不会增加，机器人 $r_i$ 使用 ComputePerpendicular()，从 $p_j(t)$ 向经过 $p_i(t)$ 的水平线作垂线。随后，它计算该垂线与相应 Go-Line 的交点，将此交点设为目的地，并沿该 Go-Line 向其移动，如图 16(c) 所示。

– **情形 B(2.2)：** 假设机器人 $r_i$ 在最上方水平线 $L_1^i(t)$ 上观察到多个机器人位置，即 $|L_1^i(t)| > 1$，那么根据它们相对于从 $p_i(t)$ 出发的 Go 线的部署情况，

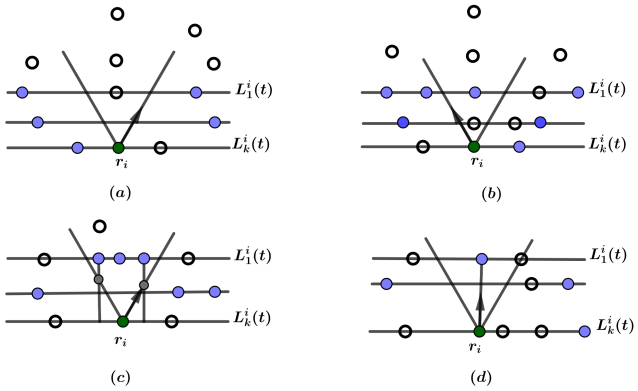



图 16：展示活动机器人沿 *Go-Line* 相应移动的情况。

∗ **情形 B(2.2.a)：** 如果两条 Go 线都与最上方水平线 $L_1^i(t)$ 相交，且交点位于该线上可见位置之间，则 $r_i$ 计算两条 Go 线与 $L_1^i(t)$ 之间的两个交点。将这两个交点分别记为 $d_i$ 和 $d_i'$。随后，机器人 $r_i$ 任意选择其中一点作为目的地，并沿相应的 Go 线朝目的地至少移动 $\delta > 0$ ，如图 16(a) 所示。

∗ **情形 B(2.2.b)：** 如果恰好一条 Go-Line 与最上方的水平线$L_1^i(t)$相交，且交点位于该线上可见位置之间，而另一条 Go-Line 与 $L_1^i(t)$ 相交于可见位置范围之外，那么 $r_i$ 将沿着交点位于可见位置之间的 Go-Line 朝交点移动，该交点记为$d_i$，如图 16(b)所示。

∗ **情形 B(2.2.c)：** 如果两条 Go 线出发——从 $r_i$ 出发的 Go-Line 与最上方的水平线 $L_1^i(t)$相交于可见位置范围之外的点时，$r_i$使用 ComputePerpendicular() 函数，从 $L_1^i(t)$的角位置向经过 $p_i(t)$的直线作垂线。然后计算这些垂线与对应 Go-Line 的交点，分别记为 $d_i$ 和 $d_i'$。最后， $r_i$ 使用 Nearest($d_i$, $d_i'$)选取两点中较近的一点，并沿着对应的 Go-Line 朝该点移动，如图 16(c)和(d)所示。

每个活动机器人都会重复执行所提出的算法，直到所有机器人聚集在同一位置。如果机器人不再观察除自身以外的任何机器人位置，则算法终止。

### 6.3. 正确性

在本节中，我们证明算法2在对抗式（$N, K$）缺陷视野模式下解决聚集问题的正确性，其中 $1 \leq K \leq N-2$。尽管观测条件受到严重限制——每个活动机器人最多可能无法观测到 $N-2$ 个其他机器人，且观测位置集可能在连续的 $Look$ 阶段之间任意变化——我们证明该算法能够保证所有机器人最终聚集到同一位置。

本节余下部分将这些观测结果形式化，并给出一系列引理，共同证明：对于任意 $N \geq 3$，算法2仅依靠垂直（$Y$）轴上的一致性，无需任何额外的机器人能力，便能在对抗式（$N, K$）缺陷视野模式下正确解决聚集问题。

LEMMA 6.1.无论对抗式缺失如何，*Go- Lines* 都能确保活动机器人沿向上方向取得有保证的垂直进展。

证明。 位于 $L_1()$ 下方的每个活动机器人始终沿其垂直轴或计算出的 Go 线移动。首先，设 $r_i \in L_k^i()$ 为任意一个活动机器人，且 $|L_1^i()| = 1$，记为 $p_j$；那么，如果 $p_j$ 与 $p_i$ 具有相同的 $X-$ 坐标，则 $r_i$ 沿其垂直轴朝 $p_j$ 移动的距离至少为 $\delta$ ，如图 16(d) 所示；否则，$r_i$ 始终沿计算出的 Go 线移动。每个能看到其水平线以上位置的活动机器人 $r_i$ 总是计算 Go 线(60°)。根据所提出的算法，如果机器人 $r_i \in L_k^i()$ 在可见的最顶端直线 $L_1^i()$ 上计算其下一个目的地，那么它沿该 $60°$ 线至少移动 $\delta$ ，因此在每次移动中， $r_i$ 位置的 $y-$ 坐标至少增加 $\delta\frac{\sqrt{3}}{2}$ 个单位。如果 $L_1^i()$ 上位于 Go 线与 $L_1^i()$ 的交点之间的所有位置均如图 16(c) 所示，那么 $r_i$ 计算

**算法2** 机器人 $r_i$的移动算法

**要求：** 观测集 $\mathcal{O}_i(t)$，直线 $L_1^i(t)$，$L_k^i(t)$
**确保：** 机器人 $r_i$的目的地 $d_i$

**如果** $|\mathcal{O}_i(t)| = 1$ **那么**
  $d_i \leftarrow p_i(t)$ ▷没有可见的机器人，等待
**否则**
  假设 $|\mathcal{O}_i(t)| > 1$
  **如果** $r_i \in L_1^i(t)$ **那么**
    **如果** $r_i$ 观察到 $L_1^i(t)$ 以下的位置**那么**
      $d_i \leftarrow p_i(t)$ ▷ 等待
    **否则如果** $r_i$ 看到 $L_1^i(t)$ 上的所有位置**则**
      **如果** $r_i$ 位于一个内部位置**则**
        $d_i \leftarrow p_i(t)$ ▷ 等待
      **else**
        令 $p_{ex}$ 为 上的另一个外部可见位置 $L_1^i(t)$
        $d_i \leftarrow$ `ComputeVertex`$(p_i(t)\ p_{ex})$
      **结束如果**
    **结束如果**
  **否则如果** $r_i \in L_k^i(t)$ **并且** $r_i$ 观察到上方的位置 $L_k^i(t)$ **then**
    角点 $\leftarrow$ `ComputeCorner`$(L_1^i(t))$
    围棋线 $\leftarrow$ `ComputeGoLines`$(L_1^i(t))$
    **如果** $|L_1^i\ (t)\ | = 1$ **则**
      令 $p_j \in L_1^i(t)$
      **如果** $p_j$ 位于 $r_i$ 的垂直轴上**则**
        $d_i \leftarrow p_j$
      **else**
        从 $p_j$ 向 $L_k^i(t)$计算垂线
        $d_i \leftarrow$ `ComputeIntSec`(垂线，Go 线)
      **结束如果**
    **else**
      **如果** $L_1^i(t)$ 的所有点都位于 Go 线之外**则**
        $d_i \leftarrow (Go\text{-}Line) \cap L_1^i(t)$
        $r_i$ 沿其中一条 Go 线向 $d_i$ 移动
      **else if** 恰好 one Go-Line 相交 $L_1^i(t)$ 位于可见位置之间**则**
        $d_i \leftarrow (Go\text{-}Line) \cap L_1^i(t)$
        $r_i$ 沿着该 Go-Line 朝 $d_i$ 移动
      **否则，如果**两条 Go 线位于可见位置之间 **then**
        $(d_i,\ d_i') \leftarrow (Go\text{-}Line) \cap L_1^i(t)$
        $r_i$ 沿着其中一条 $d_i$ 或 $d_i'$ 朝一个方向移动相应的 Go-Line
      **else**
        从.....的角点计算垂线 $L_1^i(t)$
        $(d_i, d_i') \leftarrow$ `ComputeIntSec`(垂线, Go 线)
        $r_i$ 朝 `Nearest`$(d_i, d_i')$ 移动，沿着相应的 Go-Line
      **结束 if**
    **结束 if**
  **结束 if**
**结束 if**
**返回** $d_i$

使用 Computeperpendicular()，根据位于 $L_1^i()$ 至 $L_k^i()$ 上的角位置计算垂线；当这些垂线与 Go 线相交时，$r_i$ 选择其中一条 Go 线上的最近交点并朝其移动。当机器人发现自己位于最上水平线上的外部位置，且看不到下方的任何位置时，它会利用另一个观测到的外部位置计算一个等边三角形，并沿该三角形的非水平边朝向向上方向移动，并继续前进沿向上方向至少移动 $\delta\frac{\sqrt{3}}{2}$ 个单位，然后下方的机器人再次追赶最高位置。该过程持续进行，直到所有机器人聚集在同一位置。

Go 线能够抵御对抗式隐藏，因为沿 Go 线移动并不取决于活动机器人在最上水平线上看到或漏看了多少机器人；所提出的策略只关注这些机器人相对于计算出的 Go 线的位置。由于对抗式隐藏，机器人可能在每一轮改变其 Go-Line，但它至少会朝着可见的最顶端直线移动 $\delta\frac{\sqrt{3}}{2}$ 个单位，以到达 $r_i$。 □

LEMMA 6.2. 机器人位置的水平宽度 $HW()$ 单调递减，$HW(t+1) \leq HW(t)$。

证明。 根据算法2，每个活动机器人都沿着 $60°$ Go 线朝向上方向移动。首先，考虑构型的水平宽度由极值机器人决定的情况：如果最底部的极值机器人沿极端线移动$\delta$ 距离，则总宽度至少缩小 $\frac{\delta}{2}$ 个单位。算法的设计确保非极值机器人不会移出极端线。假设$r_i \in L_k()$ 是一个非极端机器人，且从 $p_i()$ 出发的 Go 线与 $L_1^i()$ 的交点位于 $L_1^i()$ 的跨度之外，那么 $r_i$ 根据$L_1^i()$ 在 $L_k^i()$ 上的最边角位置计算垂线，并计算垂线与 Go 线的交点，然后移动到其中一个交点，如图所示。如果 $L_1^i()$ 的最边角位置位于 Go 线上，且它也是一个极值点，那么只有 $r_i$ 可以到达极端线，但在任何情况下都不能越过它。如果构型的水平宽度并非由极值机器人决定，那么这些机器人就像非极值机器人一样行动。该过程持续进行，直到所有机器人到达共同目的地。极值机器人向上移动，非极值机器人追赶它们。每一轮中，机器人位置的水平宽度都单调递减，即 $HW(t) \leq HW(t+1)$。 □

LEMMA 6.3。如果 $L_1(t_0)$ 上的极值点之间的距离为 $d$，那么所有机器人一定会聚集在两条极端线的交点 $T$ 处，该交点位于$Y_{new} = Y_{max}(0) + \frac{\sqrt{3}}{2}d$。

证明。 设在 $L_1(t_0)$处，极值点之间的距离为 $d$。两条极值线互不平行，且各自与 $L_1(t_0)$成 $60°$ 角，因此它们必定相交于直线 $L_1(t_0)$上方 $\frac{\sqrt{3}}{2}d$ 个单位处的某一点 $T$。由引理7和引理8可知，机器人沿向上方向的进展是有保证的，且不会越出极值线。考虑极值机器人的动作。若任一极值机器人与某一非极值位置计算等边三角形，则它总是将顶点计算在其正在沿其移动的极端线上。如果它

与极值机器人计算三角形顶点，则该顶点必定是两条极值线的交点 $T$。算法保证没有任何机器人越过$T$。若任一极值机器人到达 $T$，则它将处于最高位置，并等待其他机器人。位于同一条以及另一条极端线上的机器人必须在有限时间内到达 $T$，并等待非极值机器人。由于算法所施加的垂线规则，非极值机器人将在有限步之后位于同一垂直轴上，并在有限时间内到达 $T$。因此，聚集可在 $Y_{max}(0)$上方 $\frac{\sqrt{3}}{2}d$ 个单位处的 $T$ 实现，如图17所示。

□

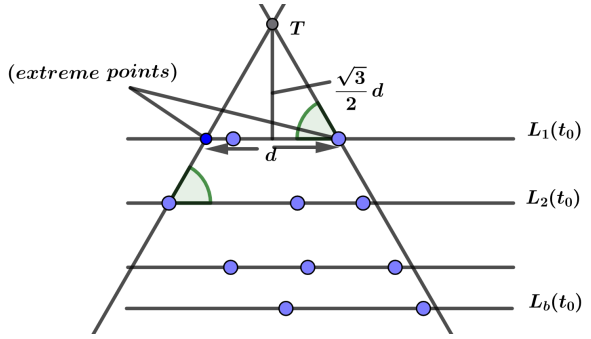



图17： 由于直线 $L_1(t_0)$ 上极值线之间的距离为 $d$，因此它们必定在 $L_1(t_0)$上方 $\frac{\sqrt{3}}{2}d$ 个单位处相交。

由引理2.1、2.2和2.3可知：

THEOREM 6.1. 在对抗性 $(N, K)$ 缺陷视图模型( $N \geq 3$)中，算法*2*能够为未配备多重性检测的异步机器人解决聚集问题，前提是它们在局部坐标系的垂直轴上具有一致性。

## 7. 仿真结果与实验验证

为通过实证验证所提出的聚集算法在缺陷可见性模型下的正确性和收敛性，我们使用 Python 实现了一个离散时间模拟器。这些构型是为模拟目的而均匀随机生成的。机器人按照标准的观察–计算–移动周期运行。在每一轮中，都会根据相应的对抗性可见性模型生成缺陷视野。除非另有说明，机器人均位于欧几里得平面上的任意初始位置。当所有机器人占据同一位置时，执行终止。为量化聚集过程，我们测量构型随时间变化的空间跨度。具体而言，对于第 $t$轮的构型，我们定义：

### 7.1. $(4,2)$ 对抗性缺陷视图模型

我们首先评估算法在 (4，2) 对抗性缺陷视图模型下的表现。

**共线初始构型：** 图18(a)显示，在FSYNC模型（$N = 4$、 $K = 2$）下，初始构型和观测到的机器人子集均以伪随机方式生成（使用固定随机种子以确保可复现性）。由于所有机器人最初均共线，因此无需进行垂直对齐，且整个过程中垂直跨度始终为零。水平跨度在同步轮次中单调且近似线性地减小，表明全局收缩均匀。即使在非刚性运动和有限可见性的情况下，每一轮也都能保证持续取得进展，展现出完全同步条件下稳定且可预测的收敛性。

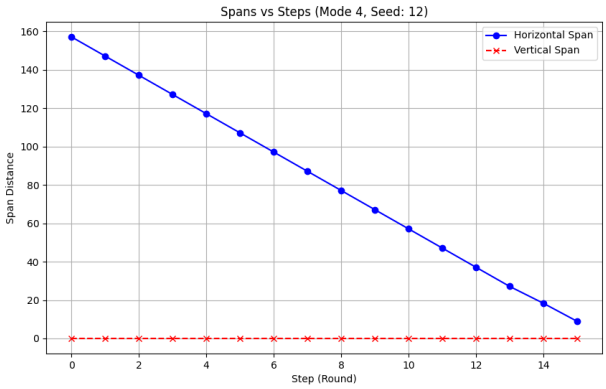
**非共线初始构型：** 图18(b)展示了在 FSYNC 模型下，$N$ =4 和 $K = 2$ 的凸包面积演变情况，其中机器人最初以伪随机方式放置在一个非共线构型中。每个机器人观测到的子集也以伪随机方式生成（使用固定随机种子以确保可复现性）。凸包面积在各同步轮次中单调递减，表明构型持续进行全局收缩（图19(a)、(b)、(c)展示了面积收缩的示例）。平滑且逐渐趋缓的衰减趋势表明，机器人正持续协同行动并朝聚集状态移动；早期轮次因初始分散程度较大而收缩更快，随着机器人接近聚集点，收缩速度则逐渐放缓。

### 7.2. $(N, K)$ 缺陷视图模型

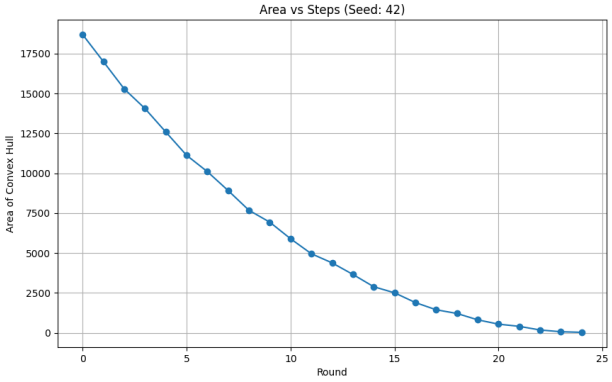
算法2的理论正确性通过严格的数学分析得以确立，该分析涵盖所有允许的异步执行和对抗性可见性缺陷。因此，仿真研究的目的并非重新验证正确性，而是为算法的几何行为、局部决策过程和经验尺度特性提供直观认识。

Python 实现围绕两个主要类组织：Robot 和 GatheringSystem。每个机器人都会维护其笛卡尔位置、轨迹历史、可见集合以及计算得出的目的地。运动严格遵循非刚性模型：机器人被激活时，会朝目的地移动，移动距离至多为固定距离 $\delta$，从而确保最小进展，同时允许部分移动，这与理论假设一致。

全局执行由 GatheringSystem 处理。在每个异步步骤中，使用带种子的伪随机调度器选取一个非空机器人子集，以可复现的方式模拟异步性。对于每个被激活的机器人，伪随机选择一个可见性上界 $K \in [1, N-2]$，并且其视野中最多暴露 $K$ 个机器人（包括其自身）




(a)




(b)

图18：(a) 展示了机器人共线时，水平宽度单调减小，且在整个执行过程中垂直跨度始终为零。(b) 展示了由非共线位置的凸包形成的面积会随着轮次数增加而减小。

。随后严格按照算法规定应用几何规则：将可见机器人划分为若干水平层；最顶层上的机器人要么等待，要么在位于内部时保持静止，要么在处于极端位置时计算等边三角形顶点；最顶层以下的机器人沿 $60°$ Go 线朝适当的交点移动（图 21 展示了此类移动的一个示例）。因此，该模拟直接编码了对抗性变化的可见性条件下的所有几何情形。

通过两个全局度量评估收敛：水平宽度（$\max x - \min x$）和垂直跨度（$\max y - \min y$）。当这些量低于固定阈值（此处为 $10^{-1}$）时，即判定为收敛。对于每个群体规模 $N \in \{5, 7, \ldots, 49\}$，使用不同的伪随机种子执行多次独立运行，并绘制收敛所需的平均异步步数。

所得图表（如图20所示）呈现出清晰且一致的规律。水平收敛曲线是对每种群体规模下5次独立伪随机运行取平均得到的，而垂直收敛曲线则是对17次独立运行取平均得到的。这种平均处理减轻了伪随机异步调度器和对抗性可见性选择所导致的波动，从而揭示算法内在的几何行为，而非特定运行中的变化。

水平收敛始终受到严格限制（约为 500–650 步），且随着 $N$ 增大仅呈现轻微增长。这一表现体现了极值线不变量的作用：只有极值机器人会主动缩小全局宽度，而内部机器人不会使其扩大。因此，水平收缩以可控且几乎不受规模影响的方式推进。不同运行结果之间的离散程度相对较小，这进一步表明水平聚集在结构上具有稳定性，且受.....的影响较弱

对抗性可见性变化。

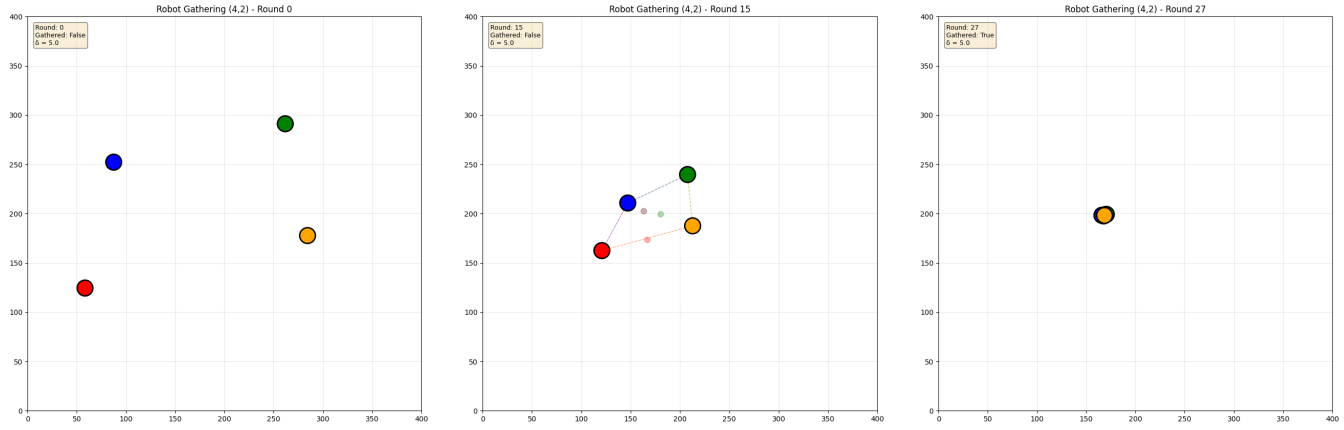
相比之下，垂直收敛对机器人群规模的依赖更为明显，在较大的 $N$ 取值下需要数千步。这种缩放行为与分层几何收缩机制相符：垂直方向的缩减是通过逐步消除上部水平层实现的。规模较大的机器人群通常初始垂直分散度更高、中间层也更多，因此需要更多收缩阶段。不过，平均曲线仍平滑且整体呈单调趋势，没有观察到振荡或发散。这证实，收敛时间增加是几何缩放导致的，而非异步执行中的不稳定性。

需要强调的是，仿真并不试图穷举所有异步执行；理论证明已经提供了此类保证。相反，仿真以可视化和定量方式说明，局部几何规则如何共同实现全局收缩。在所有测试的群体规模下均能在有限时间内收敛，这直观地印证了该算法在对抗式（$N$，$K$）缺陷可见性下的鲁棒性和可扩展性。

## 8. 结论

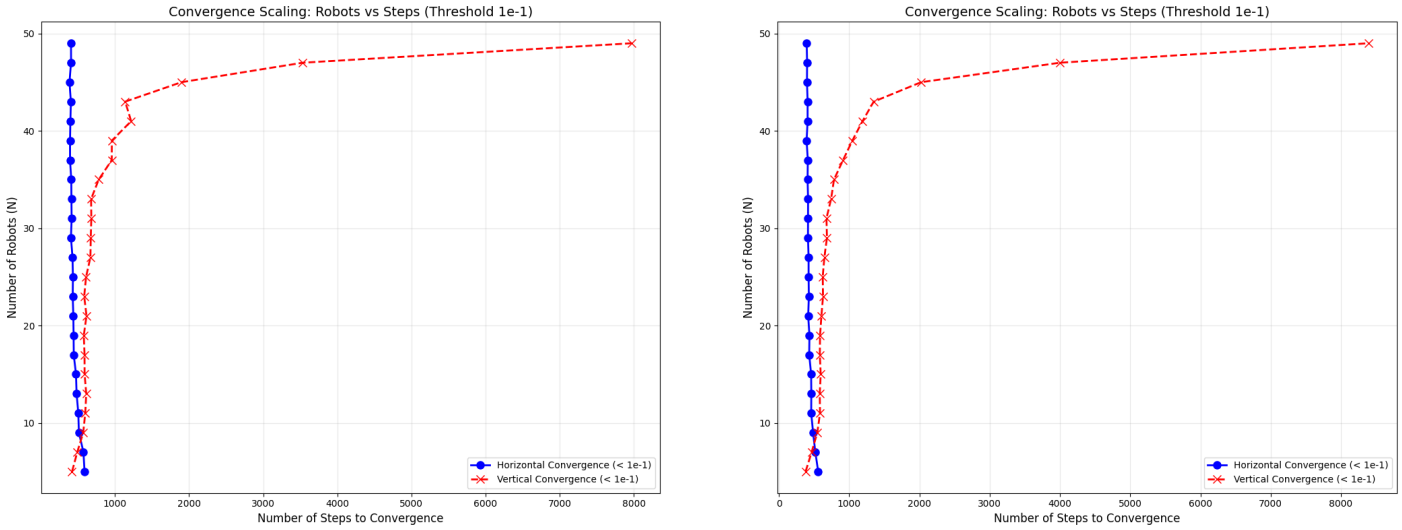
本文研究了自主机器人群在近期提出的缺陷视图模型下的聚集问题；在该模型中，机器人只能观测到其他机器人的一个有限子集，且该子集由对抗方选择。该模型刻画了严苛的感知限制，并扩展了文献中常用的经典可见性假设。

我们首先解决了一个开放问题：在




(a) (b) (c)

图19：非共线构型在对抗式（4，2）缺陷视图模型下的演化：(a) 第0轮，(b) 第15轮，(c) 第27轮。



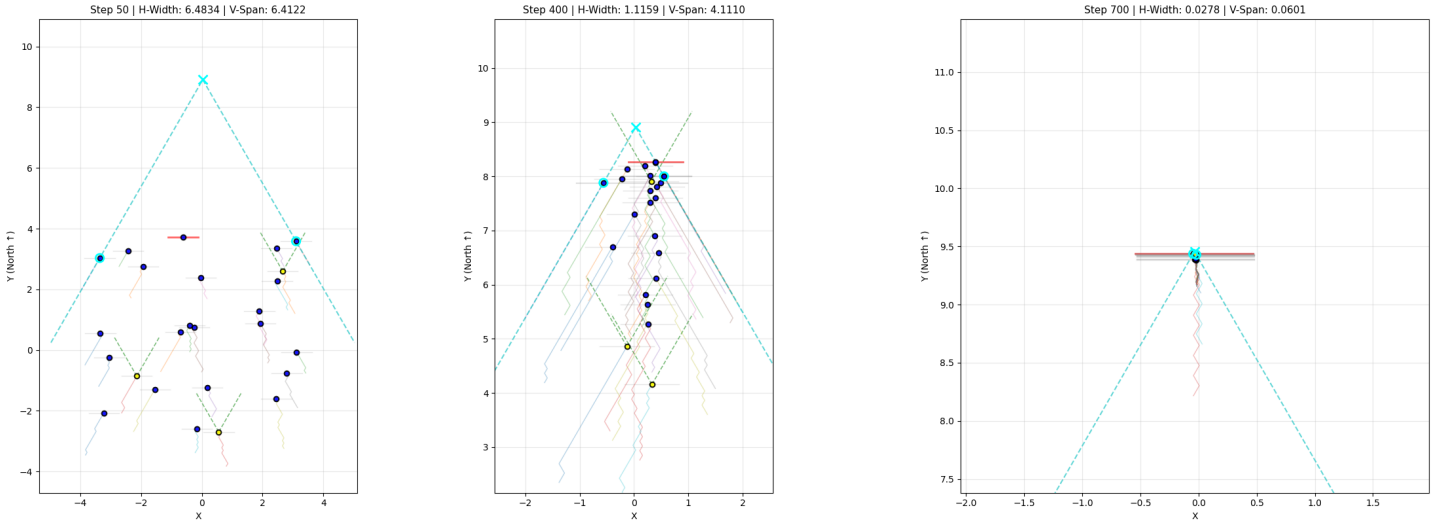

(a) 5次运行 (b) 17次运行

图20：17次运行的平均结果显示，在对抗性缺陷序列下，垂直跨度衰减得更加平滑且一致；两个图均显示出稳定的水平收敛。

文献[28]所述的对抗式 (4,2) 缺陷视图模型下，四个完全同步的机器人如何实现聚集。针对这一设定，我们提出了一种分布式算法，可保证在有限时间内实现聚集。该算法无需假设机器人具备多重性检测、记忆或局部坐标系一致性等额外能力，从而证明即使存在对抗性观测，在最小假设下仍可实现聚集。

随后，我们研究了对抗式（$N$，$K$）缺陷视图模型下**异步模型**机器人的聚集问题，其中所有值 $1 \leq K \leq N-$ 均满足 2。具体而言，我们提出并分析了一种有限时间分布式算法，即使在极端情况下——机器人可能无法看到其余 $N-1$ 个机器人中的 $N-2$ 个，且每次激活时只能观察到另一个机器人——也能正常工作。假设机器人对各自局部坐标系中的**单轴**有共同认知，所提出的算法即可实现有限时间聚集。这一结果显著拓展了现有的可解性界限，并展示了在异步执行和高度对抗性感知条件下，聚集问题令人惊讶的鲁棒性。与大多数关于缺陷视图模型的现有研究不同，我们的算法采用**非刚性移动模型**。我们增强了结果的鲁棒性和实际意义。

未来研究仍有若干富有挑战性的方向。一个自然的拓展是放宽轴一致性的假设，研究在缺陷视野模式下、局部坐标




(a) (b) (c)

图 21：蓝色直线表示极值线，其交点定义了聚集点。机器人向计算出的目的地移动，从而使水平跨度收缩。该实验对应于$N = 25$，且观测参数 $K \in [1, 23]$动态变化。

系之间完全不具备一致性的情况下实现聚集的可行性。另一个重要方向是在高维欧几里得空间中研究类似问题，此时几何障碍和可见性约束会变得更加复杂。此外，探索在更弱的机器人能力或替代性的障碍与可见性假设下实现聚集，或许能进一步加深我们对缺陷观测模型中协调基本极限的理解。

## 参考文献

[1] Chrysovalandis Agathangelou、Chryssis Georgiou 和 Marios Mavronicolas。《在平面中聚集多个有体积移动机器人的分布式算法》。载于 *Proc. ACM Symposium on Principles of Distributed Computing (PODC)* 250 259, 2013 ，页码 – 。

[2] Noa Agmon 和 David Peleg。《自主移动机器人的容错聚集算法》。*Journal on Computing* 36 1 56 82, 2006 ，( ): 页码 – 。[3]Andrea Aliberti、Yonghwan Kim 和 Yoshiaki Katayama。《容错观测的局限：在 (4,2)-缺陷视野模型中的聚集不可能性》。载于 *2025 Thirteenth International Symposium on Computing and Networking Workshops (CANDARW)*，第 121–127 页，2025年11月。[4] S. Bhagat、S. Gan Chaudhuri 和 K. Mukhopadhyaya。《三维空间中不透明机器人的聚集》。载于 *Proc. the 19th International Conference on Distributed Computing and Networking, (ICDCN-2018)*，第 2:1– 2:10 页，2018年。[5] S. Bhagat 和 K. Mukhopadhyaya。《半同步机器人的容错聚集》。载于 *Proc. the 18th International Conference on Distributed Computing and Networking (ICDCN-2017)* 6, 2017 ，页码 。

[6]Subhash Bhagat、Sruti Gan Chaudhuri 和 Krish- nendu Mukhopadhyaya。《容错聚集单轴一致性下异步无记忆移动机器人的容错聚集。*J. Discrete Algorithms*，36 62，2016年 50:页码 – 。

[7]Subhash Bhagat 和 Krishnendu Mukhopadhyaya。存在障碍物时异步机器人的聚集。载于 *11th International Conference and Workshop on Algorithms and Computation (WALCOM)*，第 279–291 页，2017年。[8] Quentin Bramas 和 S´ebastien Tixeuil。无需手性的无等待聚集。载于 *Proc. International Colloquium on Structural Information and Communication Complexity (SIROCCO)* 313 327, 2014 ，第 – 页。[9] Michael Braun、Jannik Castenow 和 Friedhelm Meyer auf der Heide。三维空间中移动机器人的局部聚集。载于 *International Colloquium on Structural Information and Communication Complexity*，第 63–79 页。Springer，2020年。[10] Avik Chatterjee、Sruti Gan Chaudhuri 和 Kr- ishnendu Mukhopadhyaya。非均匀有限可见性下异步机器人群的聚集。载于 *International Conference on Distributed Computing and Internet Technology* 174 180 ，第 – 页。Springer，2015年。[11] Sruti Gan Chauhuri。有限感知范围下自主移动机器人的聚集。载于 *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*，第 909–914 页，2018年。[12] Serafino Cicerone、Gabriele Di Stefano 和 Alfredo Navarra。机器人在会合点的聚集：可行性与最优分辨率算法。*Distributed Comput.* 31 1 1 50, 2018 ，( ): –。

[13] Mark Cieliebak、Paola Flocchini、Giuseppe Prencipe 和 Nicola Santoro。移动机器人分布式计算：聚集。*SIAM Journal on Computing*，41 4 829 879，2012年 ( ): 第 – 页。

[14] R. Cohen 和 D. Peleg。通过重心算法实现机器人汇聚。载于 *Structural Information and Communication Complexity* 3104 *Lecture* , 第 卷, 页 *Notes in Computer Science* 79 88 码 –。Springer Berlin Heidelberg, 2004年。[15] Jurek Czyzowicz、Leszek Gasieniec 和 Andrzej Pelc。平面上少量有体积移动机器人的聚集。*Theoretical Computer Science* 410 6 481 499, , (): 第 – 2009 页。[16] Xavier D´efago、Maria Potop-Butucaru 和 Philippe Raipin Parv´edy。崩溃故障或拜占庭故障下移动机器人的自稳定聚集。*Distributed Computing*, 33(5): 393–421, 2020年。[17] Giuseppe Antonio Di Luna、Ryuhei Uehara、Giovanni Viglietta 和 Yukiko Yamauchi。匿名无记忆机器人在有限可见性下的圆上聚集。*Theoretical Computer Science*, 1025: 114974, 2025年。[18] P. Flocchini、G. Prencipe、N. Santoro 和 G. Viglietta。移动机器人分布式计算：解决均匀圆形构造问题。载于 *International Conference on Principles of Distributed The 18thSystems (OPODIS 2014)* 217 232, 2014 , 第 – 页。[19] P. Flocchini、G. Prencipe、N. Santoro 和 P. Widmayer。有限可见性下异步机器人的聚集。*Theoretical Computer Science*, 337(1-3): 147–168, 2005年。[20] P. Flocchini、G. Prencipe、N. Santoro 和 P. Wid- mayer。异步匿名无记忆机器人的任意图案构造。*Theoretical Computer Science* 407 1 3 412 447, 2008 , (-): 第 – 页。[21] Paola Flocchini 、 Ryan Killick 、 Evangelos Kranakis 、 Nicola Santoro 和 Masafumi Yamashita。连续环上移动机器人的聚集与选举。载于 Pinyan Lu 和 Guochuan Zhang 编, *30th International Symposium on Algorithms and Computation ISAAC* 149, 8 1 8 19, , , 第 卷, 2019 页码: –: 。[22] Paola Flocchini、Giuseppe Prencipe 和 Nicola Santoro。*Distributed Computing by Oblivious Mobile Robots* 。分布式计算理论综合讲座。Morgan & Claypool出版社，2012年。[23] Paola Flocchini、Giuseppe Prencipe 和 Nicola Santoro（编）。*Distributed Computing by Mobile Entities* , *Current Research in Moving and Computing*。Springer，2019年。[24] N. Fujinaga、H. Ono、S. Kijima 和 M. Yamashita。无记忆 CORD机器人通过最优匹配进行图案构造。载于 *Principles of Distributed Systems* 6490 *Lecture Notes in Computer* , 第 卷, 页 *Science* 1 15 2010 码 –。Springer Berlin Heidelberg, 。[25] Vincenzo Gervasi 和 Giuseppe Prencipe。无需通信的协调：群集问题的情形。*Discret Applied Mathematics*, 144(3): 324–344, 2004年。[26]T. Izumi、T. Izumi、S. Kamei 和 F. Ooshita。无记忆移动机器人多项式时间随机聚集的可行性。 *IEEE Trans. Parallel Distrib.Syst.*, 24(4): 716–723, 2013年4月。[27]Yonghwan Kim、Masahiro Shibata、Yuichi Sudo、Junya Nakamura、Yoshiaki Kataya ma 和 Toshimitsu Masuzawa。存在缺陷视图时的聚集。*arXiv preprint arXiv:2208.08159* 2022 , 。[28]Yonghwan Kim、Masahiro Shibata、Yuichi Sudo、Junya Nakamura、Yoshiaki Katayama与Toshimitsu Masuzawa。具有缺陷视野的移动机器人聚集。载于 *26th International Conference on Principles of Distributed Systems (OPODIS 2022)*, 第14–1页。Schloss Dagstuhl–Leibniz-Zentrum f¨ur Informatik，2023年。[29]D. Pattanayak、K. Mondal、H. Ramesh与P. S. Mandal。存在崩溃故障时具有弱重数检测的移动机器人聚集。*Journal of Parallel Distributed Computing*, 123:145–155, 2019年。[30] Giuseppe Prencipe。一组自主移动机器人的聚集不可能性。 *Theoretical Computer Science* 384 2 3 222 231, 2007 , (-): 第–页。[31] Ichiro Suzuki与Masafumi Yamashita。分布式匿名移动机器人：几何图案构造。 *SIAM Journal on Computing*, 28:第1347–1363页，1999年。[32] Yusuke Tomita、Yoshiaki Yamauchi、Shin-ichi Kijima与Masafumi Yamashita。无手性的同步移动机器人平面构造。 *arXiv preprint arXiv:1705.06521* 2017 , 。[33] Tetsuya Uehara、Yoshiaki Yamauchi、Shin-ichi Kijima与Masafumi Yamashita。半同步机器人在三维欧几里得空间中的平面构造。载于 *Proceedings of the International Symposium on Stabilization Safety and Security of Distributed Systems (SSS)* , , , 383 398, 2016 第–页。[34]G. Viglietta。具有可见比特的两个机器人的会合。载于 *9th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS-2013)* , , 第– 291 306, 2013 页。[35] Yoshiaki Yamauchi、Tetsuya Uehara、Shin-ichi Kijima与Masafumi Yamashita。同步移动机器人在三维欧几里得空间中的平面构造。 *Journal of the ACM* 64 3 16 1 16 43, , (): : –: 。2017 [36]Yukiko Yamauchi与Masafumi Yamashita。视野受限的移动机器人图案构造。载于 *International Colloquium on Structural Information and Communication Complexity* 201 212 , 第–页。Springer，2013年。