arXiv:2302.03466v1 [cs.DC] 7 Feb 2023

Stand Up Indulgent Gathering*

Quentin Bramas[1],Anissa Lamani[1], 和 S´ébastien Tixeuil[2]

[1]斯特拉斯堡大学，法国国家科学研究中心，ICUBE，
法国[2]索邦大学，法国国家科学研究中心，LIP6，法国

2023年2月8日

摘要

我们研究在二维欧几里得空间中运行的移动机器人群。我们研究容崩聚集问题的一种变体：若没有机器人崩溃，机器人必须在有限时间内于同一任意位置相遇，而该位置事先未知；若一个或多个机器人在同一位置崩溃，其余正常机器人则聚集到崩溃位置以营救它们。受半同步设置下不可能性结果的启发，我们针对全同步设置提出了该问题的首个解法。该解法在标准的观察-计算-移动模型中运行，且不需要任何额外假设：机器人无记忆、无共同方向、没有重数检测能力，并且可以从任意位置（包括存在多重性点的位置）出发。此外，我们还证明，机器人聚集所需的时间与机器人之间的初始最大距离成正比。

关键词：移动机器人、聚集、容错。

1 引言

从算法角度研究移动机器人群，始于 Suzuki 和 Yamashita [16], ；他们模型的简洁性推动了分布式计算领域的许多研究成果 [14]。这类研究尤其关注一组自主机器人在二维欧几里得空间中演化时的计算能力。更确切地说，其目的是确定机器人能够完成哪些任务，以及完成这些任务需要付出多少代价。

通常，机器人被建模为二维平面上的点，并假定它们各自拥有坐标系和单位距离。此外，它们具有以下特性：(i) 匿名（无法彼此区分），(ii) 同质（执行相同的算法），(iii) 无记忆（无法记住过去的动作）。机器人无法直接交互，但配备了视觉传感器，能够感知周围环境并观察其他机器人的位置。机器人按周期运行，每个周期包含三个阶段：观察、计算和移动（LCM）。在第一个阶段（观察）中，机器人获取快照，以观察其他机器人的位置。在第二个阶段（计算）中，它们要么计算一个目的地，要么决定保持空闲。在最后一个阶段（移动）中，它们朝着计算出的目的地移动（如果有）。文献中采用了三种执行模型来表示机器人之间的同步程度。全同步模型（FSYNC）假定所有机器人同时激活，并同步执行其 LCM 周期。半同步模型（SSYNC）允许只有一部分机器人同时激活。异步模型（ASYNC）除了要求限定 LCM 周期中每个阶段的持续时间外，不作任何假设。

尽管在这一设定下已研究过不少问题（例如图案形成、分散、聚集、探索、巡逻等），聚集问题仍然是人们可以要求移动机器人执行的

*本研究部分由法国国家研究署项目 SAPPORO（编号：2019-CE25-0005-1）资助。本文中的初步成果曾在 SSS 2020 [5] 和 AlgoSensors 2021 [6]上报告。

最基本的任务之一，例如收集采集的数据或更新机器人算法。为解决聚集问题，要求机器人在有限时间内到达同一个事先未知的地理位置。尽管聚集问题看似简单，却因初始对称性和／或对抗性调度而产生了许多不可能性结果。文献中将两台机器人聚集的特殊情况称为会合。

随着机器人群中的机器人数量增加，如何应对机器人可能发生不可预测的崩溃，变得至关重要。以往关于容错聚集的研究考虑了该问题的两种变体。在弱变体中，只要求正常机器人（即未发生故障的机器人）聚集；而在强变体中，所有机器人（故障机器人和正常机器人）都必须聚集到同一点。显然，只有当所有故障机器人都在同一位置崩溃时，强变体才可行。本文研究容崩聚集问题的强版本，并将其称为站立式宽容聚集（SUIG）。具体而言，若满足以下两个条件，算法便可解决 SUIG 问题：（i）如果没有机器人崩溃，那么所有机器人都能在有限时间内聚集到同一位置，该位置事先未知；（ii）如果一个或多个机器人在某个位置 l 崩溃，那么所有机器人都能在有限时间内聚集到 l 。SUIG 问题可以看作聚集问题的一个更严格版本：一旦发生任何意外事件，就要求正常机器人救助故障机器人。当然，机器人无法直接通信，因此并不知道任何机器人是否处于崩溃状态。

相关工作 一项基础性结果 [16] 表明，当机器人以完全同步的方式运行时，可以确定性地解决会合问题；而如果允许某些机器人等待一段时间（例如 SSYNC 模型中就是这种情况），那么若没有额外假设，问题便无法解决。只要允许双价初始配置[1] [9], ，或者机器人不具备多重性检测能力 [15]，SSYNC 会合的不可能性结果就会扩展到 SSYNC 聚集。另一方面，即使机器人无法进行重数检测 [2]，FSYNC 聚集仍然可行 [8],。

对于至少由三个机器人组成的群体，早期在 SSYNC 模型下解决弱崩溃容错聚集的方法依赖于额外假设：（i）从不同配置开始（即每个位置至多有一个机器人），至多允许一个机器人崩溃 [1], ；（ii）机器人逐个激活 [11], ；（iii）机器人可能表现出概率行为 [12], ；（iv）机器人具有共同手性（即左右手性的概念相同） [4], ；（v）机器人对共同方向达成一致 [3]。事实证明，在 SSYNC 模型下解决确定性弱崩溃容错聚集并不需要这些假设，即使最多有 $n-1$ 个机器人可能崩溃 [7]。

SUIG 的情况主要得出了一些不可能性结果：当至多发生一次崩溃时，如果机器人没有持久坐标系，那么在 SSYNC 中，即使机器人逐个执行，也无法确定性地实现强崩溃容错聚集 $n \geq 3$ 机器人；此外，在公平调度器下，也无法在 SSYNC 中实现 $n \geq 3$ 机器人的概率强崩溃容错聚集 [11, 12]，[11, 12]。然而，如果机器人的相对速度有一个常数上界 [11, 12]，那么在 SSYNC 中，$n \geq 3$ 机器人的概率强崩溃容错聚集就可以实现。

这引出了以下开放问题：对于 $n \geq 2$ 机器人，能否确定性地将所有机器人聚集到同一处事先未知的位置（如果它们都没有崩溃），或者在其中一些机器人意外地在同一位置崩溃时，确定性地聚集到故障机器人的位置？要解决这个问题，是否需要额外的假设（例如具备重数检测能力，或从非双价配置乃至不同配置开始）？

贡献 我们对移动机器人群中 SUIG 的可行性给出了肯定回答。我们的方法具有构造性：我们给出了一个显式算法，可在 FSYNC 执行模型中解决 SUIG 问题；该算法仅使用无记忆机器人，且机器人的局部坐标系可以任意设定（它们无需对任何方向、手性或单位距离达成一致），也不依赖重数检测（机器人无法获知任何已占据位置上的机器人数量，或其近似值）。此外，我们算法的一个优良性质是，对初始

[1] 如果所有机器人均匀分布在恰好两个不同的位置上，则该配置为 $bivalent$。

机器人位置不作任何假设（尤其是，我们允许从具有多重性点的配置开始，甚至从双价配置开始），因此该算法是自稳定的 [13]。我们的 SUIG 算法的时间复杂度为 $O\left(\frac{D}{\delta}+\log\left(\frac{\delta}{u_{min}}\right)+\log^2\left(\frac{u_{max}}{u_{min}}\right)\right)$ 轮，其中 u_{max}（分别是 u_{min}）表示机器人之间的最大（分别是最小）单位距离， D 表示初始配置中两台机器人之间的最大距离， δ 表示保证机器人每轮至少移动的距离。

为实现这一结果，我们提出了一种新的分层切片技术，它可以根据层级，将在某一配置类中执行算法特定步骤的机器人划分开来。即使机器人对共同的单位距离没有共识，这一层级也能帮助它们协调行动。我们认为，这项技术或许也适用于移动机器人群的其他相关问题。

为说明层级的概念，我们首先给出一种在存在单次崩溃时解决会合问题的方法，称为 SUIR 问题（容错唤醒会合）。当 $k=2$ 时，会合问题是聚集问题的一个特例。为说明采用全同步设置的合理性，我们给出一个不可能性结果：即使机器人拥有共同的坐标系、具有包含无限多种颜色的全灯[2] ，且其移动是刚性的，在 SSYNC 下，SUIR 问题仍无解。

内容安排 我们首先在第2节定义模型。接着，在第3节给出不可能性结果。然后，我们在第4节通过考察容错唤醒会合（SUIR）问题进行铺垫，随后在第5节讨论容错唤醒聚集（SUIG）问题。最后，我们在第6节总结全文并提出一些开放问题。

2 模型

设 $\mathcal{R}=\{r_1,r_2,\dots r_k\}$ 为由 $k\geq 2$ 个机器人组成的集合，并将其建模为欧几里得二维空间中的点。假设机器人是匿名的（彼此无法区分）、同质的（执行相同的算法）且无记忆的（无法记住过去的操作）。

设 Z 为全局坐标系。设 $p_i(t)\in\mathbb{R}^2$ 为机器人 r_i 或一组在时间 t 与 r_i 位于同一位置的机器人的坐标，该坐标相对于 Z 给出。给定时间 t，时间 t 的配置定义为集合 $\mathcal{C}_t=\{p_1(t)\ p_2(t)\ \dots p_m(t)\}$ $(m\leq k)$，其中 m 是平面中占据点的数量，并且对于任意 i、$j\in\{1,\dots,m\}$、$i\neq j$， $p_i(t)\neq p_j(t)$。机器人不知道 Z。相反，每个机器人 r_i 都有自己的坐标系 Z_{r_i}，其原点位于 r_i 的当前位置。我们假设机器人方向不一致，也就是说，它们既不认同同一坐标轴，也没有共同的单位距离。注意，最小包围圆（SEC）与坐标系无关。

机器人按照由三个阶段组成的周期运行：观察、计算和移动。在观察阶段，机器人 r_i 观察所有其他机器人的位置，并构建集合 $V_{r_i}(t)=\{p_1(t)\dots,p_m(t)\}(m\leq k)$，其中 m 是平面上被占据点的数量， $p_i(t)$表示机器人或一组位于同一位置的机器人在 r_i的局部坐标系 Z_{r_i} 中的坐标（该坐标系经平移 $-r_i$ ，使得 r_i 始终位于中心）。我们将 $V_{r_i}(t)$称为机器人 r_i 在时间 t的局部视图。请注意，机器人无法区分由单个机器人占据的位置与由多个机器人占据的位置，也就是说，它们不具备重数检测能力。在计算阶段， r_i根据其局部坐标系 Z_{r_i}计算目标位置。最后，在移动阶段， r_i 朝计算出的目标位置移动。我们假设移动是非刚性的，也就是说，当机器人朝着移动阶段计算出的目标位置移动时，对手可以通过在通往目标位置的直线路径上的任意位置将机器人停下来，阻止它到达目标位置。不过，我们假设机器人至少会移动固定正距离 δ。所有机器人所共有的 δ 值是未知的。它可以任意小，并且在算法执行期间保持不变。

[2] *full-lights* 模型对应于配备了灯的机器人，这些灯有多种颜色，可由主控机器人更改，并能被其他机器人看到。因此，机器人可以与其他机器人进行显式通信。

在全同步模型（FSYNC）中，每个称为轮次的时刻都会激活所有正常机器人，且所有机器人同步执行其周期。在半同步模型（SSYNC）中，每轮可能只激活正常机器人的一个非空子集；与 FSYNC 模型类似，被激活机器人同步执行其周期。在后一种情况下，我们只考虑公平调度，即每个正常机器人都会被无限次激活的调度。

算法 A 是一个将局部视图映射到目的地的函数。当 r_i 在时间 t被激活时，算法 A 会输出 r_i 在其局部坐标系 Z_{r_i}中的目的地 d 。 A的一次执行$\mathcal{E} = (C_0, C_1, \ldots)$是一个配置序列，其中 C_0 是初始配置，且每个配置 C_{t+1} 都是通过应用 A从 C_t 得到的。

如果机器人在时间 t 之后的任何时间 $t' \geq t$都未被激活，则称其在时间 t 崩溃；也就是说，故障机器人停止执行其算法，并无限期地停留在同一位置。

Stand Up Indulgent Gathering 问题 如果对于任意初始配置 C_0 以及至多有一个崩溃位置（即机器人崩溃的位置）的任意执行$\mathcal{E} = (C_0, C_1, \ldots)$，都存在一个轮次 t 和一个点 p ，使得对于所有 $t' \geq t$，$C_{t'} = \{p\}$ 成立，则称算法解决了 Stand Up Indulgent Gathering（SUIG）问题。换言之，如果一台机器人崩溃，所有未崩溃的机器人都会前往并加入崩溃的机器人；如果没有机器人崩溃，所有机器人都会在有限轮次内聚集。

由于我们考虑的是无记忆机器人和任意初始配置，不失一般性，可以认为崩溃（如果发生）是在执行开始时发生的。

如前所述，为了阐明我们所提出方案的核心思想，我们先考虑 SUIG 问题的一个特殊情形，称为 Stand-Up Indulgent Rendezvous（SUIR），它研究在 $k = 2$ 时至多有一个机器人崩溃的情况下的聚集问题。

3 不可能性结果

在本节中，我们首先证明，即使机器人共享一个完整的坐标系，并且能够访问其他机器人可读取的无限持久化内存，SUIR 问题在 SSYNC 中仍不可解。在文献中， [10]，这种持久化内存机制称为具有无限种颜色的全灯模型。随后我们将扩展这一结果，证明在没有额外假设的情况下，SUIG 问题在 SSYNC 中同样不可解。

我们先介绍证明不可能性结果的一个关键要点。

引理 1 考虑具有刚性移动、配备拥有无限多种颜色的全灯的机器人，以及公共坐标系的 SSYNC模型。假设算法 A解决了 SUIR问题，那么，在从不同配置开始、且每次执行中仅激活机器人r 的任一执行后缀中（例如，因为另一台机器人已经崩溃），必定存在一个配置，使得算法 A 命令 r 移动到另一台机器人的位置。

证明。机器人 r 的任何未移动到另一台机器人所在位置的移动都无法实现聚集。如果这种情况无限次重复，就无法实现会合。□

定理 1 即使在具有刚性移动、配备拥有无限多种颜色的全灯的机器人，并且共享公共坐标系的 SSYNC模型中，SUIR问题也不可解。

证明。为反证起见，假设存在这样的算法。令 r 为其中一台机器人， r' 为另一台机器人。我们构造一个永远无法实现会合的公平无限执行。在某一轮次 t，根据算法在当前配置下的（确定性）输出，我们只激活 r、只激活 r'，或同时激活两者：

- 如果规定 r 保持静止：则只激活 r。在这种情况下，今后的轮次中只会激活 r ，配置始终不会改变，也永远无法实现会合。

- 如果规定 r 移动到 $p \neq r'$：则只激活 r
- 如果 r 被指示移动到 r'，且 r' 被指示移动：则激活两台机器人。
- 否则（r' 被指示保持静止）：仅激活 r'

现在我们证明该执行是公平的。为导出矛盾，假设该执行不公平，因此存在一个轮次 t，在此之后仅执行 r，或仅执行 r'。在第一种情况下，这意味着存在一个执行后缀，其中 r 从未被指示移动到另一台机器人处，这与引理 1 矛盾。现在，若在某个轮次 t之后仅激活 r'，则存在一个后缀，其中 r' 始终被指示保持静止，这同样与引理 1 矛盾。

我们选择的调度保证以下性质。仅激活 r 时，不会实现会合，因为 r 没有移动到 r'。仅激活 r' 时，不会实现会合，因为 r'处于静止状态。如果两台机器人都被激活，也不会实现会合，因为 r 正移动到 r'，而 r' 正在移动。

总之，存在一种无限公平执行，在该执行中机器人永远不会相遇，这与算法能够解决 SUIR 的初始假设相矛盾。□

推论 1 在没有关于初始配置的额外假设时，即使采用刚性移动、为机器人配备具有无限多种颜色的全灯，并共享一个公共坐标系，SUIG 问题在 SSYNC 中也不可解。

证明。从一个双价配置开始，即一个 $2n$ 机器人配置，其中 n 个机器人位于点 p_1，而 n 个机器人位于点 $p_2 \neq p_1$。p_1 中的所有机器人，以及 p_2 中的机器人，都可以被视为一个机器人（假设它们具有相同的坐标系和相同的初始内存，并且对手始终同时激活它们）。因此，根据定理 1，SUIR问题不可解，这意味着 SUIG 问题也不可解。□

4 Stand-Up Indulgent Rendezvous

本节重点讨论 Stand-Up Indulgent Rendezvous（SUIR）问题。SUIR问题是 SUIG 问题的一个特例，因为它只涉及两个机器人。我们提出一种算法，引入层级的概念，从而使无记忆无定向机器人能够在 FSYNC 模式下解决该问题。

我们首先证明，在一维空间中存在能够解决 SUIR问题的算法，意味着在二维空间中也存在能够解决同一问题的算法。这一点很重要，因为我们的算法是在一维空间中定义的。不过，请注意，反过来并不成立。事实上，我们还给出了一个问题的例子（具有一条公共完整坐标轴的无故障会合），该问题在二维空间中可解，却无法归约到一维空间。尽管这些结果直观易懂，但形式化证明并不简单。

4.1 向一维空间归约

定理 2 假设存在一种算法，可解决机器人受限于一维空间时的任意站立会合问题，或无故障会合问题。那么，存在一种算法，可解决二维空间中的任意站立会合问题，或无故障会合问题。

证明。设 A_1 是一种算法，可解决机器人受限于一维空间时的 SUIR问题。我们通过给出一种由机器人在二维空间中执行的新算法 A_2，对该定理给出构造性证明（同样的过程也适用于无故障会合算法）。

首先，对于时间t时机器人在二维平面上的配置 $C = \{r_{min}, r_{max}\}$，其中 $r_{min} < r_{max}$（按其坐标的字典序排列），我们将函数 v 定义如下：

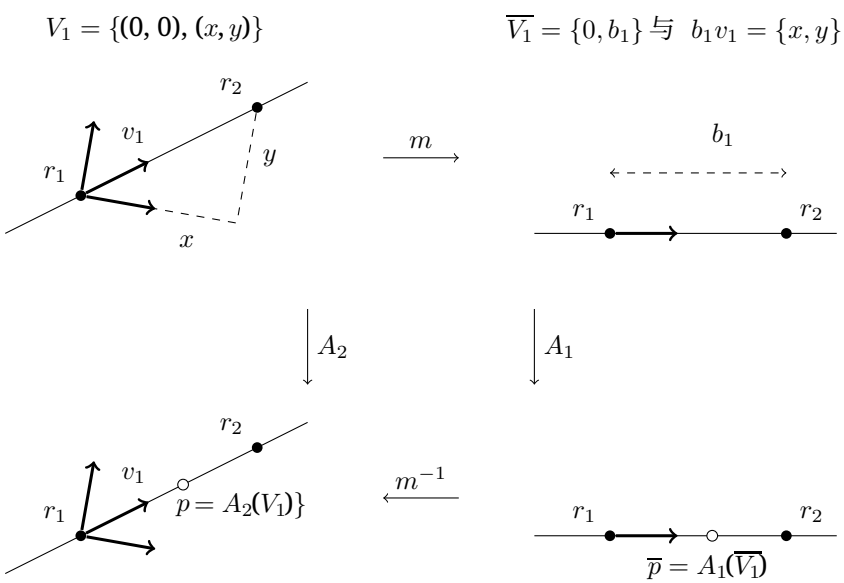

图1： r_1 在二维平面中的视图与在一维直线中的视图之间的关系。平面中的目标位置是通过对直线上计算得到的目标位置应用逆映射获得的。

$$\mathbf{v}(\{r_{min}, r_{max}\}) = \frac{r_{\max} - r_{\min}}{\|r_{\max} - r_{\min}\|}$$

其中，两个点 a 和 b 的差 $b - a$ 是向量 $\vec{ab}$，而 $\|.\|$ 表示欧几里得范数（因此， $\|r_{\max} - r_{\min}\|$ 是 $r_{\max}$ 和 $r_{\min}$之间的距离）。

设 r_1 和 r_2 分别表示两台机器人，其变换函数分别为 h_1 和 h_2。变换函数取自一个变换集合，该集合包含旋转、缩放、反射及其复合变换。我们假设，机器人 r 的变换函数由对手选择，但不会随时间改变。对于每个 $i = 1, 2$，令 $V_i = \{(0, 0)\ h_i(r_{3-i} - r_i)\}$ 表示机器人 r_i 在当前时间 t 的局部视图（即在机器人 r_i 的局部坐标系中看到的机器人位置集合）。每台机器人 r_i 都可以计算连接两台机器人的直线的方向向量$v_i = \mathbf{v}(V_i)$。请注意，如果机器人始终位于同一直线上，那么只要机器人尚未聚集，v_i 在整个执行过程中就保持不变。图1左上部分展示了机器人 r_1 的视图及其向量 v_1。

我们定义如下由机器人在二维空间中执行的算法 A_2。首先，如果机器人 r_i 的局部视图为 $\{(0, 0)\}$，则 A_2 输出 $(0, 0)$。

否则， r_i 可以将其局部视图 V_i 映射到一维空间中，得到 $\overline{V_i} = \{0, b_i\}$ withb_i ，使得 $h_i(r_{3-i} - r_i) = b_i v_i$，并在 $\overline{V_i}$上执行 A_1 。然后，将得到的目的地 $\overline{p} = A_1(\overline{V_i})$转换回二维空间，得到目的地 $p = \overline{p} v_i$，如图1所示。这样，机器人始终位于同一条直线上，并且只要机器人尚未聚集，v_i 就保持不变（聚集后，两个算法都会停止）。

设 $E = (C_0, C_1, \ldots)$ 为 A_2的一次任意执行。我们希望从 E构造出 A_1 的一次执行 $\overline{E}$ ，使得当且仅当在 E中实现会合时，在 $\overline{E}$ 中也实现会合。

回顾一下，我们使用用于分析的全局坐标系 Z 来分析每个配置 $C_i, i \in \mathbb{N}$。令 $v = \mathbf{v}(C_0)$。同样，由于机器人始终位于同一条直线上，因此只要机器人尚未聚集，对于任意 C_i ，都有 $v = \mathbf{v}(C_i)$。

令 O 为连接两台机器人的直线 L 上的任意一点。我们如下定义双射 m

将 L （在 Z 中）的映射点映射到全局一维坐标系（O， v）：

$$\forall a \in \mathbb{R}, m(O + av) = a$$

我们可以将函数 m 扩展到配置上，如下所示：
$m(C) = \{m(r) \mid r \in C\}$。

现在，设 $E = (C_0, C_1, \ldots)$ 为两个机器人的执行过程，其中一个机器人的变换函数为 $\overline{h}_1$，另一个为 $\overline{h}_2$，且在 (O, v) 中执行 A_1，当且仅当 $\overline{h}_i(a) = b$ 时， $h_i(av) = bv_i$。

现在我们证明，若 $C \xrightarrow{A_2} C'$，则 $m(C) \xrightarrow{A_1} m(C')$。为此，我们证明在 $m(C)$ 上执行 A_1 所得的结果与 $m(C')$一致。设 $C = \{O + a_1 v, O + a_2 v\}$, i 为一个被激活机器人， j 为另一个机器人。一方面，我们有 $m(C) = \{a_1, a_2\}$，并且根据构造，机器人 i 在 $m(C)$中的视图 $\overline{V_i}$为 $\{0, b_i\}$，其中 $b_i = \overline{h}_i(a_j - a_i)$，因此 r_i 在 $m(C)$中的全局目的地为 $a_i + \overline{h}_i^{-1}(\overline{p})$（其中 $\overline{p} = A_1(\overline{V_i})$）。另一方面，机器人 i的视图 V_i为 $\{(0, 0)\ h_i((a_j - a_i)v)\} = \{(0, 0)\ b_i v_i\}$，因此 r_i 在 Z中的全局目的地为 $O + a_i v + h_i^{-1}(\overline{p} v_i) = O + a_i v + \overline{h}_i^{-1}(\overline{p}) v$。由于 $m(O + a_i v + \overline{h}_i^{-1}(\overline{p}) v) = a_i + \overline{h}_i^{-1}(\overline{p})$，我们得到 $m(C) \xrightarrow{A_1} m(C')$（假设在 C 和 $m(C)$中激活的是相同的机器人）。

因此，对于所有 $i \in \mathbb{N}$， $\overline{C_i} = m(C_i)$。由于 A_1 解决了 SUIR问题，因此存在一个点 $p \in \mathbb{R}$和一个轮次 t，使得对于所有 $t' \geq t$， $m(C_{t'}) = \{p\}$。这意味着 $C_{t'} = \{O + pv\}$，因此A_2 解决了 SUIR问题。□

注意，定理 2 的逆命题并不成立。为证明这一点，我们给出一种算法，用于解决二维空间中的（无故障）会合问题，并假设机器人对齐于同一条完整坐标轴（即它们对该坐标轴的方向和定向达成一致）。在这一假设下，机器人可能对连接两者的直线的定向并不一致，因此，若假设定理 2 的逆命题成立，就意味着存在一种算法，能够在方向不一致的机器人组成的一维空间中解决该问题（而这样的算法并不存在；这一点可用与 [16] 中证明类似的方法证明）。

其思路是：如果配置对称（机器人可能具有相同的视图），那么机器人就移动到这样一个点：该点与这两个机器人构成一个等边三角形。由于这样的点有两个，机器人选择最北端的那个（机器人对齐于 y 轴，因此拥有共同北方）。否则，配置不对称，且存在唯一最北端机器人 r。该机器人不移动，另一机器人移动到 r。

算法1：无故障会合。机器人对齐于一条完整坐标轴，但可能没有共同的单位距离，且移动是非刚性的

数据： r： 执行该算法的机器人
令 $\{(0, 0)、(x, y)\}$ 为 r 的局部视图。
如果 $y = 0$ 则
1 移动到点 $\left(x/2, \left|\frac{x\sqrt{3}}{2}\right|\right)$
否则，如果 $y > 0$，则
2 移动到另一台机器人所在的位置 (x, y)。

定理3 算法1能够在非刚性移动条件下，解决无故障机器人仅具有一条公共完整坐标轴且单位距离不同的会合问题。

证明。若配置不对称，则执行第2行后，要么实现会合，要么正在移动的机器人仍位于连接两台机器人的同一直线上，因此得到的配置仍然不对称，并且指示同一台机器人朝同一目的地移动，因此它会在有限轮次内到达该目的地。

现在考虑初始配置 C 为对称配置的情况。如果两个机器人都到达目的地，会合将在一轮内完成。如果机器人在到达目的地之前停止，可能出现两种情况：它们停止时行进的距离不同，或者停止在相同的 y 坐标处。前一种情况下，得到的配置是非对称的，因此回到第一种情况。后一种情况下，配置仍然对称，但

两个机器人之间的距离至少缩短 $\frac{2\delta}{\sqrt{2}}$。因此，在每一轮中，机器人要么完成会合，要么到达非对称构型，要么以固定距离彼此靠近。由于后一种情况不可能无限次发生，其他两种情况至少会发生一次，因此会合将在有限轮次内完成。□

4.2 SUIR算法。

在下文中，我们考虑方向不一致的机器人（即它们对坐标轴和单位距离都没有共同认知），并假设移动操作是刚性的。在这些假设下，我们给出算法 2 来解决 SUIR问题。该算法定义在直线上。每个机器人都以某种方式为直线定向，但机器人之间可能对直线方向不一致。不过，由于直线方向是根据机器人自身的坐标系推导出来的，因此不会随时间变化。机器人 r 的不同移动取决于 r 是将自己视为位于另一个机器人的左侧还是右侧，以及它所在的层级。根据机器人自身的坐标系（因此也依据其自身的单位距离），距离另一个机器人 d 处的机器人的层级，是满足 $d \in [2^{-i}, 2^{1-i})$ 的整数 $i \in \mathbb{Z}$ 。在伪代码中，我们用

$$left \to \texttt{Move1}; right \to \texttt{Move2}$$

来表示：将自己视为位于左侧的机器人应执行 Move1，而将自己视为位于右侧的机器人则应执行 Move2。

算法 2：方向不一致的机器人所用的SUIR算法
令 d 为与另一台机器人之间的距离 令 $i \in \mathbb{Z}$ 使得 $d \in [2^{-i}, 2^{1-i})$ 如果 $i \equiv 0$ 模 2，则移动到中点 如果 $i \equiv 1$ 模 4，则左侧 $\to$ 移动到中点；右侧 $\to$ 移动到另一侧 如果 $i \equiv 3$ 模 4，则左侧 $\to$ 移动到另一侧；右侧 $\to$ 移动到中点

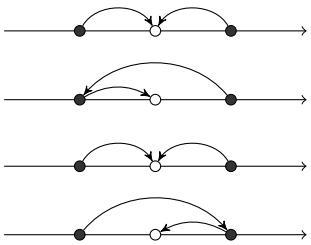

图2：取决于两个机器人的距离，共有四种可能的配置。我们将 $i \equiv 0$ 模 2 的情形拆分为两行，以便读者阅读。

图2总结了机器人 r 的八种可能视图及相应的移动。每一行表示机器人层级模四的同余关系；在每一行中，我们可以看到机器人无论将自己视为位于另一个机器人的右侧还是左侧时所执行的移动。请注意，图2中描述的情形并不一定意味着两个机器人实际上会同时执行相应的移动（因为它们可能有不同的视图）。

例如，如果机器人 r_1 的层级 i_1 模 4 同余于 1，且看到自己位于右侧，而另一台机器人 r_2 的层级 i_2 模 4 同余于 2，也看到自己位于右侧，那么 r_1 会移动到另一台机器人的位置，而 r_2 会移动到中点。假设两台机器人都到达了各自的目的地，那么它们之间的距离就会减半（无论坐标系如何），因此它们的层级都会增加 1；并且它们都会看到另一台机器人位于另一侧，所以现在每台机器人都看到另一台机器人位于自己的左侧。此时， i_1 模 4 同余于 2，而 i_2 模 4 同余于 3。由于两台机器人都观察到自己位于左侧， r_1 会移动到中点，而 r_2会移动到另一台机器人的位置。同样，由于它们之间的距离减半，它们的

层级都会增加 1；由于它们交换了位置，因此又观察到自己位于右侧。在新的配置中，i_1 模 4 同余于 3，而 i_2 模 4 同余于 0。通过执行算法 2，两台机器人都移动到中点，并实现会合。图 3 展示了这一执行过程。

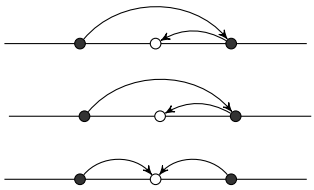

图 3：算法 2 的一种可能执行

4.3 正确性证明。

下面证明算法 2 的正确性。设 C 为任意配置，d 为两台机器人之间的距离（在全局坐标系 Z 中）。设 x 和 y 分别为左侧机器人和右侧机器人在 Z 中行进的距离。由于机器人彼此相向移动，执行一轮后，机器人之间的距离变为 $f(d,x,y) = |d-x-y|$。

引理2 若至少有一台机器人被指示移动到中点，则有 $f(d,x,y) \leq d - \min(\delta, d/2)$。

证明。对任意固定的 d，利用 f 关于第二个和第三个参数的对称性，可得 $f(d,x,y) = g_d(x+y)$，其中 g_d：$w \mapsto |d-w|$。我们知道，机器人行进的距离要么为 0（若一台机器人崩溃），要么至少为 $\min(d/2,\delta)$，但不可能出现 $x = y = 0$。此外，由于至少有一台机器人移动到中点，因此要么 (i) $x \leq d/2$且 $y \leq d$，要么 (ii) $x \leq d$且 $y \leq d/2$。因此，总和 $x+y$ 位于区间 $[\min(d/2,\delta)\ 3d/2]$内。

作为凸函数，g_d 的最大值在其定义域的边界处取得

$$\begin{aligned} f(d,x,y) = g_d(x+y) &\leq \max(g_d(3d/2), g_d(\min(d/2,\delta))) \\ &\leq \max(d/2, d-\min(d/2,\delta)) = d - \min(d/2,\delta) \end{aligned}$$

□

下一个引理是引理2的直接推论。

引理 3 在任意机器人之间的距离为 d（在 Z 中）的配置之后两轮，机器人之间的距离至少减少 $\min(\delta, d/2)$。

证明。根据引理2，我们知道，如果至少有一个机器人被指示移动到中点，那么一轮之后，两个机器人之间的距离至少减少 $\min(\delta, d/2)$。否则，我们知道两个机器人都被指示移动到另一个机器人所在的位置。因此，机器人之间的距离要么至多为 $d - \min(\delta, d/2)$，要么大于 $d - \min(\delta, d/2)$（但仍至多为 d）。前一种情况下，引理得证。后一种情况下，机器人的顺序会改变（左侧机器人变成右侧机器人，反之亦然）。无论它们的坐标系如何，都会发生这种情况（也许两个机器人都在视图中将自己看作位于右侧，那么它们都会将自己看作位于左侧）。此外，每个机器人的层级要么不变，要么增加一级。在所有情况下，两个机器人都被指示移动到中点。具体而言，层级为 $i \equiv 1 \bmod 4$ 的右侧机器人会变成左侧机器人，其层级要么不变，要么变为 $i+1 \equiv 2 \bmod 4$。无论哪种情况，下一轮中，该机器人都会被指示移动到中点。层级为 $i \equiv 3 \bmod 4$ 的左侧机器人会变成右侧机器人，其层级要么不变，要么变为 $i+1 \equiv 0 \bmod 4$。无论哪种情况，下一轮中，该机器人都会被指示移动到中点。

这样再经过一轮，距离至少会减少 $\min(\delta, d/2)$。 □

根据前面的引理，我们知道存在某一轮次，在此轮次之后机器人之间的距离始终至多为 δ，因此，只需在假设移动是刚性的情况下证明我们的算法正确即可（机器人始终能够到达其目标位置）。

如果在第 t 轮次，一个机器人认为自己位于右侧，而另一个机器人认为自己位于左侧，那么在时间 t，它们对直线方向达成一致。由于对每个机器人而言，直线方向不会改变，因此在整个执行过程中它们都对此保持一致（聚集时除外，因为此时直线未定义）。

类似地，如果在某一轮次，两个机器人都认为自己位于右侧（相应地，位于左侧），那么它们的直线定向相反，并且在整个执行过程中始终相反（同样，直到它们聚集为止）。因此，我们有如下备注。

备注 1 设两个方向不一致的机器人沿着连接它们的直线 L 移动，并执行算法 2。那么，在整个执行过程中（只要它们尚未聚集），它们要么始终具有共同的 L 定向，要么始终具有相反的 L 定向。

引理 4 假设移动是刚性的、没有崩溃，并且机器人对连接两者的直线具有共同定向，则算法 2 可以解决 SUIR问题。

证明。由于机器人具有共同定向，我们知道有一个机器人看到自己位于右侧，另一个机器人看到自己位于左侧。当然，机器人并不知道这一点，但我们在前面的注记中看到，只要机器人尚未聚集，整个执行过程中共同定向都会保持不变。

令 $(i, j) \in \mathbb{Z}^2$ 表示这样一个配置：左侧的机器人位于层级 i，右侧的机器人位于层级 j；当且仅当 $i \equiv k \bmod 4$ 且 $j \equiv l \bmod 4$ 时，我们记作 $(i, j) \equiv (k, l) \bmod 4$。

为了证明该引理，我们要证明对于任意配置 $(i, j) \in \mathbb{Z}^2$，机器人都能实现会合。任取一个配置 $(i, j) \in \mathbb{Z}^2$。我们考虑全部 16 种情形：

1. 若 $(i, j) \equiv (0, 0) \bmod 4$：一轮内即可会合。2. 若 $(i, j) \equiv (0, 1) \bmod 4$：我们到达配置 $(j + 1, i + 1) \equiv (2, 1) \bmod 4$。3. 若 $(i, j) \equiv (0, 2) \bmod 4$：一轮内即可会合。4. 若 $(i, j) \equiv (0, 3) \bmod 4$：一轮内即可会合。5. 若 $(i, j) \equiv (1, 0) \bmod 4$：一轮内即可会合。6. 若 $(i, j) \equiv (1, 1) \bmod 4$：我们到达配置 $(j + 1, i + 1) \equiv (2, 2) \bmod 4$。7. 若 $(i, j) \equiv (1, 2) \bmod 4$：一轮内即可会合。8. 若 $(i, j) \equiv (1, 3) \bmod 4$：一轮内即可会合。9. 若 $(i, j) \equiv (2, 0) \bmod 4$：一轮内即可会合。10. 若 $(i, j) \equiv (2, 1) \bmod 4$：我们到达配置 $(j + 1, i + 1) \equiv (2, 3) \bmod 4$。11. 若 $(i, j) \equiv (2, 2) \bmod 4$：一轮内即可会合。12. 若 $(i, j) \equiv (2, 3) \bmod 4$：一轮内即可会合。13. 若 $(i, j) \equiv (3, 0) \bmod 4$：我们到达配置 $(j + 1, i + 1) \equiv (1, 0) \bmod 4$。14. 若 $(i, j) \equiv (3, 1) \bmod 4$：我们到达配置 $(j, i) \equiv (1, 3) \bmod 4$。15. 若 $(i, j) \equiv (3, 2) \bmod 4$：我们到达配置 $(j + 1, i + 1) \equiv (3, 0) \bmod 4$

16. 若 $(i, j) \equiv (3, 3) \bmod 4$：我们到达配置 $(j+1, i+1) \equiv (0, 0) \bmod 4$

无论如何，会合至多经过三轮即可实现。 □

引理 5 假设采用刚性移动、不发生崩溃，且机器人对连接两者的直线具有相反定向，则算法 2 可以解决 SUIR问题。

证明。由于机器人具有相反定向，我们知道它们要么都看到自己位于右侧，要么都看到自己位于左侧。当然，机器人并不知道这一点，但我们在前面的注释中看到，只要机器人尚未聚集，相反定向就会在整个执行过程中保持不变。

在本证明中， $R\{i, j\}$ 表示这样一种配置：两个机器人都看到自己位于右侧，其中一个的层级为 i ，另一个的层级为 j。此处， i 与 j 之间的顺序无关紧要（因此采用集合记号）。类似地， $L\{i, j\}$ 表示这样一种配置：两个机器人都看到自己位于左侧，其中一个的层级为 i ，另一个的层级为 j。

这里，a 不失一般性地假设 $i \leq j$ 模 4，我们写作 $R\{i, j\} \equiv (k, l)$ 模 4 ，分别 $L\{i$,当且仅当 $j\} \equiv (k, l) \bmod 4$、 $i \equiv k \bmod 4$ 和 $j \equiv l \bmod 4$。

为了证明该引理，我们要说明，对于任意配置 $R\{i, j\}$ 或 $L\{i, j\}$，机器人都能实现会合。取任意配置 $(i, j) \in \mathbb{Z}^2$。我们考虑全部 20 种情况：

1. 若 $L\{i, j\} \equiv (0, 0) \bmod 4$：一轮后即可实现会合。2. 若 $L\{i, j\} \equiv (0, 1) \bmod 4$：一轮后即可实现会合。3. 若 $L\{i, j\} \equiv (0, 2) \bmod 4$：一轮后即可实现会合。4. 若 $L\{i, j\} \equiv (0, 3) \bmod 4$：我们到达配置 $R\{i+1, j+1\} \equiv (0, 1) \bmod 4$。5. 若 $L\{i, j\} \equiv (1, 1) \bmod 4$：一轮后即可实现会合。6. 若 $L\{i, j\} \equiv (1, 2) \bmod 4$：一轮后即可实现会合。7. 若 $L\{i, j\} \equiv (1, 3) \bmod 4$：我们到达配置 $R\{i+1, j+1\} \equiv (0, 2) \bmod 4$。8. 若 $L\{i, j\} \equiv (2, 2) \bmod 4$：一轮后即可实现会合。9. 若 $L\{i, j\} \equiv (2, 3) \bmod 4$：我们到达配置 $R\{i+1, j+1\} \equiv (0, 3) \bmod 4$。10. 若 $L\{i, j\} \equiv (3, 3) \bmod 4$：我们到达配置 $R\{i, j\} \equiv (3, 3) \bmod 4$。11. 若 $R\{i, j\} \equiv (0, 0) \bmod 4$：一轮后即可实现会合。12. 若 $R\{i, j\} \equiv (0, 1) \bmod 4$：我们到达配置 $L\{i+1, j+1\} \equiv (1, 2) \bmod 4$。13. 若 $R\{i, j\} \equiv (0, 2) \bmod 4$：一轮后即可实现会合。14. 若 $R\{i, j\} \equiv (0, 3) \bmod 4$：一轮后即可实现会合。15. 若 $R\{i, j\} \equiv (1, 1) \bmod 4$：我们到达配置 $L\{i, j\} \equiv (1, 1) \bmod 4$。16. 若 $R\{i, j\} \equiv (1, 2) \bmod 4$：我们到达配置 $L\{i+1, j+1\} \equiv (2, 3) \bmod 4$。17. 若 $R\{i, j\} \equiv (1, 3) \bmod 4$：我们到达配置 $L\{i+1, j+1\} \equiv (0, 2) \bmod 4$。18. 若 $R\{i, j\} \equiv (2, 2) \bmod 4$：一轮后即可实现会合。19. 若 $R\{i, j\} \equiv (2, 3) \bmod 4$：一轮后即可实现会合。20. 若 $R\{i, j\} \equiv (3, 3) \bmod 4$：一轮后即可实现会合。

无论如何，至多经过三轮次即可实现会合。 □

引理 6 假设采用刚性移动且有一个机器人崩溃，算法 2 可解决 SUIR问题。

证明。令 i 为正确机器人的层级 r。假设另一台机器人崩溃。机器人 r 会看到自己位于另一台机器人的右侧或左侧。

如果 r 看到自己位于右侧，那么根据其层级， r 要么移动到中点，要么移动到另一台机器人所在的位置。在前一种情况下， r 的层级增加一，且 r 继续看到自己位于右侧。在后一种情况下，一轮内即可实现会合。最多经过三轮， r 的层级就会与 1 模 4 同余，因此最多再经过四轮即可实现会合。

类似地，如果 r 看到自己位于左侧，那么最多经过四轮， r 的层级就会与 3 模 4 同余，并实现会合。□

定理 4 算法 2 在 FSYNC 下为方向不一致的机器人解决 SUIR问题。

证明。根据引理 3，两台机器人之间的距离每两轮至少减少 $\min(\delta, d/2)$。因此，最终两台机器人的距离会小于 δ，从这一时刻起，移动都是刚性的。现在假设移动是刚性的。如果一台机器人崩溃，则根据引理 6 即可实现会合。否则，根据机器人是否对连接两者的直线具有共同定向或相反定向（见备注 1），定理可分别由引理 4 或引理 5 得证。□

5 站立式宽容聚集

现在，我们已经通过 SUIR 问题的情形引入了层级的概念，接下来聚焦于站立式宽容聚集问题，其中 $k \geq 2$。有人可能认为，只需让正常机器人聚集到一个位置，然后应用算法 2 执行聚集即可。但这还不够。事实上，由于正常机器人可能具有不同的层级和定向，执行算法 2 后，它们可能会分散到多个位置。因此，不能直接应用算法 2。

5.1 算法描述

下面我们提出一种解决 SUIG 问题的算法。该算法的移动取决于机器人看到多少个被占据的位置（下文也称为点），以及这些位置是否共线。如果看到两个点，我们还会根据这两点之间的距离进一步划分配置集合。具体而言，我们将构型集合 Conf 划分如下：

$$Conf = \mathcal{T}_{SEC} \cup \mathcal{T}_{ext} \cup \bigcup_{k \geq 0} \left(\mathcal{A}^k \cup \mathcal{B}^k \cup \mathcal{C}_1^k \cup \mathcal{C}_2^k \cup \mathcal{C}_3^k \right)$$

- $\mathcal{T}_{ext}(p)$（或者当 p 任意时，简称 $\mathcal{T}_{ext}$）表示所有机器人共线，且只有一个极端机器人 p 与最近机器人之间的距离为 e 的配置集合，其中 $\frac{e}{d} \in \left\{ \frac{1}{9}, \frac{1}{8}, \frac{9}{80}, \frac{10}{81}, \frac{10}{18}, \frac{9}{16}, \frac{80}{81} \right\}$，这里 d 表示两个极端机器人之间的距离[3]。如果存在两个这样的点，则该配置不属于 $\mathcal{T}_{ext}(p)$。

- $\mathcal{T}_{SEC}$ 表示不只由两个点构成且不属于 $\mathcal{T}_{ext}$ 的配置集合。

[3]这些分数是算法中定义的两种移动（Move $\left(\frac{1}{9}\right)$ 和 Move $\left(\frac{1}{8}\right)$）所产生的结果。也可以使用其他分数（用于这些移动，因此也用于 $\mathcal{T}_{ext}$），只要能够在得到的配置中区分不同的移动组合即可。

• 其他配置集合构成了仅由两个点构成的配置集合 Conf_2 的一个划分。当一个配置仅由两个点构成时，我们将机器人 r 的层级定义为满足 $d_r \in [2^{-l_r}, 2^{-l_r+1})$ 的数 l_r，其中 d_r 是机器人 r所看到的两个点之间的距离。接着，我们对所有 $k \geq 0$ 定义

$$\begin{aligned}
\mathcal{A}^k &= \{C \in Conf_2 \mid S_k \leq l_r < S_k + k\} \\
\mathcal{B}^k &= \{C \in Conf_2 \mid S_k + k \leq l_r < S_k + 2k\} \\
\mathcal{C}_1^k &= \{C \in Conf_2 \mid S_k + 2k \leq l_r < S_k + 2k + 1\} \\
\mathcal{C}_2^k &= \{C \in Conf_2 \mid S_k + 2k + 1 \leq l_r < S_k + 2k + 2\} \\
\mathcal{C}_3^k &= \{C \in Conf_2 \mid S_k + 2k + 2 \leq l_r < S_k + 2k + 3\}
\end{aligned}$$

其中 $S_k = k(k+2)$，因此会考虑 $S_{k+1} = S_k + 2k + 3$ 以及所有层级 ≥ 0。此外，为简单起见，我们约定固定 $\mathcal{A}^0 = \{C \in \mathrm{Conf}_2 \mid l_r < 1\}$，而不将其设为空集。这样，无限序列 $\left(\mathcal{A}^k \cup \mathcal{B}^k \cup \mathcal{C}_1^k \cup \mathcal{C}_2^k \cup \mathcal{C}_3^k\right)_{k \geq 0}$ 便是 Conf_2 的一个划分。可以注意到，推导出的划分取决于观测机器人 r；也就是说，两个不同的机器人可能会在同一时刻将 Conf_2 中的某个配置视为属于不同的子集，因为它们使用的距离单位可能不同。

当配置由两个相距 d 的点组成时，如果机器人被指令朝另一个点移动距离 $e \times d$，我们就称它执行移动 Move (e)。现在我们可以介绍名为 SUIG（站立式宽容聚集）的算法。伪代码见算法 3，示意图见图4。当机器人仅占据两个点时（即配置属于 Conf_2），机器人的动作取决于其自身的坐标系。更确切地说，机器人 r 使用自己的坐标系来确定穿过两个占据点的直线方向。如果另一个点（机器人所在点以外的点）位于东侧，或者在没有点位于另一个点东侧的情况下位于北侧，那么 r 便认为该点位于另一个点的左侧；否则，r 会认为自己位于另一个点的右侧。在图示中，我们用带箭头的直线表示方向，并根据机器人认为自己位于右侧还是左侧，用弯曲箭头表示移动。在伪代码中，我们写道

$$left \rightarrow \texttt{Move1}; right \rightarrow \texttt{Move2}$$

通知机器人：如果它看到自己位于左侧，则执行 Move1；如果位于右侧，则执行 Move2。

算法3：SUIG算法

如果 $C \in \mathcal{A}^k$，则移动$\left(\frac{1}{2}\right)$
如果 $C \in \mathcal{B}^k$，则左侧 $\rightarrow$移动$\left(\frac{1}{9}\right)$；右侧 $\rightarrow$移动$\left(\frac{1}{10}\right)$

如果 $C \in \mathcal{C}_1^k$，则移动$\left(\frac{1}{2}\right)$
如果 $C \in \mathcal{C}_2^k$，则左侧 $\rightarrow$ Move(1)；右侧 $\rightarrow$ 移动$\left(\frac{1}{2}\right)$
如果 $C \in \mathcal{C}_3^k$那么向左 $\rightarrow$ 移动$\left(\frac{1}{2}\right)$；向右 $\rightarrow$ Move(1)

如果 $C \in \mathcal{T}_{ext}(p)$，那么移动到 p
else 移动到最小包围圆的中心。

5.2 正确性证明

下面我们证明该解的正确性。

引理 7 如果机器人没有聚集，那么它们最终会彼此保持在距离不超过 δ 的位置。

证明。若机器人在轮次 t 结束时没有聚集，则至少存在两个占据位置。设 $D(t)$ 为时间 t 时最小包围圆（SEC）的直径。为导出矛盾，假设对所有 $t \geq 0$, $D(t) > \delta$。由于机器人从不移动

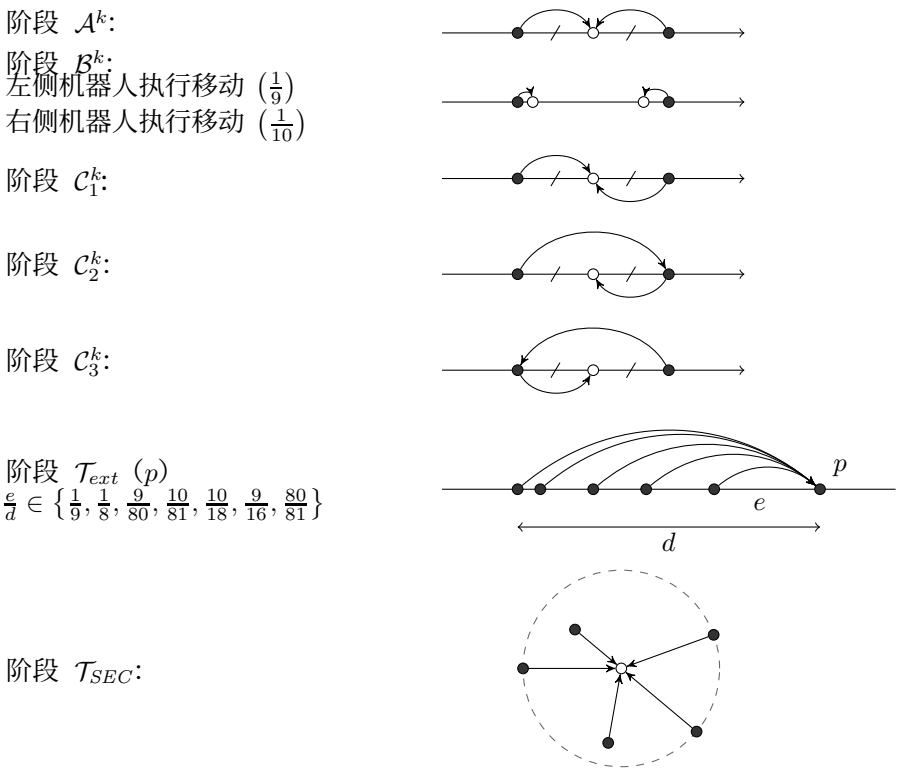

图4：算法3。

在它们形成的凸包之外，函数 $t \mapsto D(t)$ 非递增，因此收敛到某个值 $B \geq \delta$，并且存在一个时间 T 使得，

$$\forall t > T, \qquad D(t) \in [B, B + \delta/10). \tag{1}$$

考虑任意这样的时间 $t > T$。根据我们的算法，机器人要么移动到最小包围圆（SEC）的中心，要么移动到一个端点（处于阶段 $\mathcal{T}_{ext}$时），要么执行移动（e），$e \in \{\frac{1}{10}, \frac{1}{9}, \frac{1}{2}, 1\}$。我们根据机器人在配置 C（t）中执行的移动，考虑不同的情况。

- 情况A：如果机器人处于阶段 $\mathcal{T}_{ext}$（p），则该配置不是对称的，并且所有机器人具有相同的目标，即点 p。设 $q \neq p$ 为承载所有机器人的线段的另一个端点。如果在 q处没有故障机器人，则最小包围圆的直径至少减少 $\delta/10$ （最坏情况下，故障机器人距离 p为 $9D$（t）$/10$ ），这与关系 1矛盾。如果在 q处存在故障机器人，则在配置 C（$t+1$）中，没有正确机器人与位于 q 处的故障机器人重合，并且情况 B、 C 或 D 适用，从而得到矛盾。

- 情况 B：如果机器人处于阶段 $\mathcal{T}_{SEC}$，那么所有机器人（可能有一个故障机器人除外）都会朝 SEC 的中心移动。如果 SEC 的直径至多为 2δ，每个机器人都会到达其目标位置，并且时间 $t+1$ 时的 SEC 至少缩小一半，因此至少减少 $\delta/2$（因为时间 t 时的半径为 D（t） $\geq \delta$）。如果 SEC 的直径大于 2δ，如图 5 所示，那么 SEC 的直径至少减少 δ （如果没有发生崩溃，则会减少 2δ）。这两种情况都与关系 1 矛盾。

  在其余情况下， C（t）由两个点组成。

- 情况C：如果机器人不在同一阶段 $\mathcal{T}_{ext} \cup \mathcal{T}_{SEC}$，且没有机器人执行 Move(1)，那么位于两个端点的机器人执行 Move $\left(\frac{1}{10}\right)$、Move $\left(\frac{1}{8}\right)$ 或 Move $\left(\frac{1}{2}\right)$，最小包围圆直径

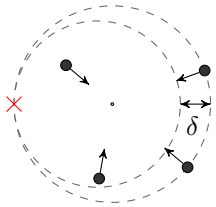

图 5：情况 B：当每个机器人（故障机器人除外）都向 SEC 的中心移动时，δ 使 SEC 的直径缩小

（即两个端点之间的距离）至少减少 $\min(\delta, d/10) \geq \delta/10$，这与关系 1 矛盾。

- 情况 D：如果机器人不处于阶段 $\mathcal{T}_{ext} \cup \mathcal{T}_{SEC}$，则一些机器人执行 Move(1)；并且如果存在一个没有正常机器人与其共点的故障机器人。在这种情况下，所有正常机器人都向故障机器人移动至少 $\delta/10$ 的距离（因为一些机器人可能处于不同的阶段），因此 SEC 的直径至少缩小 $\delta/10$，这与关系 1 矛盾。

- 情况 E：如果机器人不处于阶段 $\mathcal{T}_{ext} \cup \mathcal{T}_{SEC}$，则一些机器人执行 Move (1)；并且如果不存在故障机器人，或故障机器人与正常机器人共点。如果有机器人执行的移动不同于 Move (1)，那么下一个配置 C（$t+1$）不只包含两个点，因此可应用前一种情况得出矛盾。现在假设所有机器人都执行 Move (1)，且配置 C（$t+1$）只包含两个点。由于D（$t+1$）仍在区间 $[B, B+\frac{\delta}{10})$ 内，因此移动的机器人必定改变了其在线段上的一侧。所以，如果机器人的层级没有改变，它会在时间$t+1$执行 Move $\left(\frac{1}{2}\right)$。在时间 $t+1$，要么可应用前一种情况得出矛盾（如果机器人执行的移动不同于 Move (1)），要么仍适用同一种情况。当同一种情况适用时，如图 6 所示，所有机器人的层级都增加，并在时间 $t+1$再次执行 Move (1)。如果在时间 $t+2$仍适用同一种情况，这意味着它们的层级增加了两次，而这只有在两个点之间的距离至少减半一次时才会发生，这与关系 1 矛盾。因此，同样可以应用前一种情况并得出矛盾。

□

根据引理 7，我们不失一般性地假设机器人运动是刚性的。接下来的引理及其推论将说明如何利用阶段所包含的层级数逐渐增加这一事实（$\mathcal{A}^{k+1}$ 比 $\mathcal{A}^k$多 1 个层级）。引理 8指出，机器人向上“跳跃”不能超过 7 个层级；推论 2则利用这一事实指出，当 k 足够大时，存在某个时间，使得所有机器人都处于同一阶段 $\mathcal{A}^k$，分别为 $\mathcal{B}^k$。

下文中，层级最低的机器人指层级最低的那个机器人。 Δ_{level} 是任意两个机器人层级之间的最大差值，即 $\Delta_{level} = \max_{r,\ r'} |l_r - l_{r'}|$。该差值可能取决于配置，但有一个上界。事实上，如果层级最低的机器人比层级最高的机器人少一个单位距离的 2^c 倍，那么层级差至多为 c。

引理 8 在任意执行中，要么所有机器人聚集，要么存在一个层级 l，使得对于任意层级 $l' \geq l$，机器人都会到达一个由两个点组成的配置，并且层级最低的机器人所属层级处于区间 $[l', l'+7]$ 内。

证明。假设机器人没有聚集，根据引理 7，我们可以假设从某个时间 t_0起机器人进行刚性移动。如果 $C(t_0)$ 由两个点组成，令 l 为 $C(t_0)$ 中层级最低的机器人的层级。否则，所有机器人都被指示朝同一个目的地移动（阶段为 $\mathcal{T}_{SEC}$ 或$\mathcal{T}_{ext}$），因此，要么所有机器人在 $C(t_0+1)$ 中聚集，要么 $C(t_0+1)$ 由两个点组成（如果存在故障机器人）。在后一种情况下，令 l 为 $C(t_0+1)$ 中层级最低的机器人的层级。

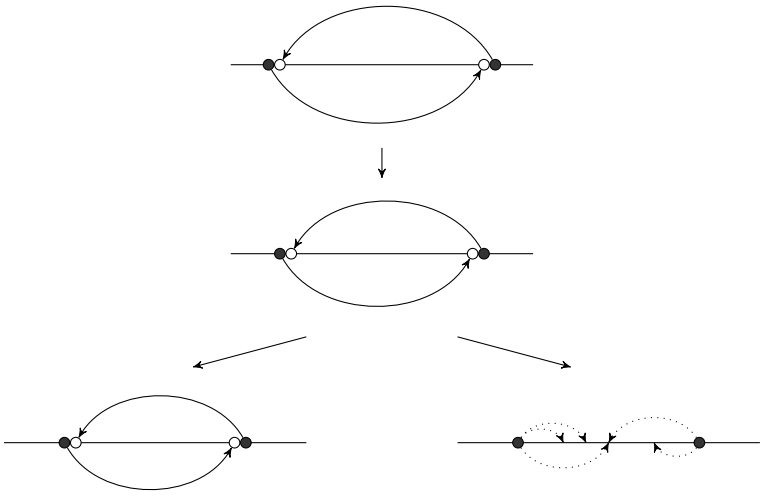

图 6：情况 E：如果机器人几乎交换了它们的位置并提升了层级，则它们可能再次全部执行 Move(1)，但这种情况不可能再发生一次，因为两点之间的距离必须至少减半，机器人才能看到其层级增加了 2。

为证明该引理，只需证明以下断言：如果时间 t 的配置 $C(t)$ 由两个距离为 d的点组成，那么 $C(t+1)$ 或 $C(t+2)$ 由两个距离为 $d' \geq \frac{d}{2^7}$的点组成，或者所有机器人在 $C(t+1)$ 或 $C(t+2)$ 中聚集。事实上，这意味着，形成两个点的相邻两个配置之间的层级增幅不会超过 7。因此，对于任意 $l' \geq l$ ，我们最终会到达一个配置，其中层级最低的机器人的层级位于区间 $[l', l'+7]$ 内，或者所有机器人聚集。

现在我们证明该断言。执行 $C(t)$ 中的算法后，要么所有机器人聚集，要么 $C(t+1)$ 由两个点组成，要么由三个或更多点组成。如果 $C(t+1)$ 由两个点组成，则任意两点之间的距离至少为 $(\frac{1}{9} - \frac{1}{10})d = \frac{1}{90}d \geq \frac{1}{2^7}d$。因此，从 $C(t)$ 到 $C(t+1)$，层级至多增加 7。

如果 $C(t+1)$ 由 3 个或更多点组成，则包含三个点的最小区间至少为$\frac{1}{10}d$，因此端点（回忆一下，机器人是共线的）之间的距离为 $d' \geq \frac{1}{10}d$。在 $C(t+1)$ 中，所有机器人都向同一目的地移动。如果没有故障机器人，或者故障机器人已经位于目的地，则下一个配置 $C(t+2)$ 是聚集的。否则，由于故障机器人位于端点，而目的地要么是端点，要么是中点，因此 $C(t+2)$ 由距离为 $d'' \geq d'/2$ 的两个点组成。因此，我们得到 $d'' \geq \frac{1}{20}d \geq \frac{1}{2^7}d$。

因此，断言得证，引理随之成立。 □

推论 2 对于任意 $k \geq 0$，最终要么所有机器人聚集，要么所有机器人都处于阶段 $\mathcal{A}^{k'}$，且满足 $k' \geq k$。对于阶段 $\mathcal{B}$，结论同样成立。

证明。设 l 由引理 8 定义，并令 $k > \Delta_{level} + 7$ 满足 $S_k \geq l$。现在可以对 $l' = S_{k'}$应用引理 8，并取 $k' \geq k$ （显然，我们有 $l' > l$ ，因此该引理适用）。由此可知，机器人最终要么聚集，要么到达一种配置：较低的机器人的层级位于区间 $[S_{k'}, S_{k'}+7]$内。由于 $k' > \Delta_{level} + 7$，所有机器人的层级都位于区间$[S_{k'}, S_{k'} + k' - 1]$内，并且它们都处于阶段 $\mathcal{A}^{k'}$。

将 $\mathcal{A}^{k'}$ 替换为 $\mathcal{B}^{k'}$，即可进行类似的证明。 □

现在我们可以证明算法的正确性。我们首先证明，在没有机器人崩溃的情况下，机器人会聚集（引理 9）；然后证明机器人可能在同一位置崩溃时的情况（引理 10）。为简单起见，在下文中我们不妨假设移动是刚性的（使用引理 7）。

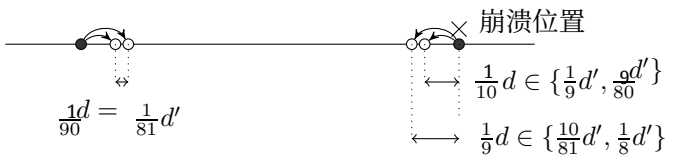

图 7：假设发生单次崩溃（本例中位于右侧），机器人在阶段 B 执行一次移动后的可能目的地。所有白色节点都表示正常机器人的可能目的地。我们可以看到，无论机器人最终位于何处，它们都能利用端点到最近机器人的距离之比，检测出崩溃位置。

引理 9 如果没有机器人崩溃，则所有机器人最终会聚集在同一个点。

证明。根据推论 2，机器人要么聚集，要么全部进入阶段 $\mathcal{A}^k$，其中 $k > 0$。因此，再经过一轮，所有机器人都会移动到中点并聚集。□

Lemma 10 如果机器人在同一位置崩溃，所有机器人最终都会聚集到同一个点 t.

证明。假设有些机器人在同一位置崩溃。根据推论 2，机器人要么聚集，要么全部进入阶段 $\mathcal{B}^k$，其中 $k > 0$。令 d 表示这两个点之间的距离。

如果有些正常机器人位于崩溃位置，那么一轮之后，至少会形成三个点。在所得配置中，所有机器人仍位于同一条直线上，最远的两台机器人之间的新距离 d' 取决于机器人执行了哪种移动（移动到 $\frac{1}{10}$，或移动到 $\frac{1}{9}$）。在情况 1 中，如果位于崩溃位置对面的机器人中至少有一台执行了 $\frac{1}{10}$ 移动，那么 $d' = \frac{9}{10}d$；否则，在情况 2 中，$d' = \frac{8}{9}d$。此外，原先位于崩溃位置的所有正常机器人现在与崩溃位置的距离为 $\frac{1}{10}d$ 或 $\frac{1}{9}d$（同样取决于它们执行了哪种移动）。因此，视 d' 的新值而定，崩溃位置位于直线的端点，其与最近的正常机器人之间的距离在情况 1 中属于 $\{\frac{1}{9}d', \frac{10}{81}d'\}$（例如，$\frac{1}{9}d'$ 是由执行 $\frac{1}{10}$ 移动的机器人得到的，即 $\frac{1}{10}d = \frac{1}{10}\frac{10}{9}d' = \frac{1}{9}d'$），或在情况 2 中属于 $\{\frac{9}{80}d', \frac{1}{8}d'\}$。

注意，直线另一端的机器人与最近机器人的距离要么为 $\frac{1}{81}d'$（如果至少有两个机器人以不同方式移动），要么大于 $\frac{1}{2}d'$（否则）。所有可能的情况都在图 7 中表示。

无论如何，所有机器人都能检测到崩溃位置所在之处。崩溃位置位于直线的某个端点，使得最近机器人与其的距离为 $\{\frac{1}{9}d', \frac{9}{80}d', \frac{10}{81}d', \frac{1}{8}d'\}$，其中 d' 是直线的长度（请回想，该比率不取决于 d'的值）。因此，正常机器人执行移动 $\mathcal{T}_{ext}$，再经过一轮后便会聚集在崩溃位置。

如果崩溃位置上没有正常机器人，且部分机器人的定向不同，也会发生同样的情况。所得配置包含三个点，其中两个点之间的距离为 $\frac{1}{81}d'$，而崩溃位置与最近正常机器人的距离为 $\frac{80}{81}d'$，因此正常机器人执行移动 $\mathcal{T}_{ext}$，再经过一轮后便会聚集在崩溃位置。

如果崩溃位置没有机器人，且所有机器人具有相同的定向，那么正常机器人会在每一轮逐渐靠近崩溃位置，直到有一台机器人进入阶段 $\mathcal{C}_1^k$。如果仍有一些正常机器人处于阶段 $\mathcal{B}^k$，那么，与前面的情况类似，得到的配置包含 3 个或 4 个点，并且所有正常机器人都能检测出崩溃位置，因为它是距离最近机器人 $\{\frac{10}{18}d', \frac{9}{16}d'\}$ 的端点。因此，正常机器人再进行一轮移动 $\mathcal{T}_{ext}$ 后，便会聚集在崩溃位置。

如果所有正常机器人同时进入阶段 $\mathcal{C}_1^k$，那么再经过一轮后，它们都会进入阶段 $\mathcal{C}_2^k$。如果它们位于崩溃位置的左侧，所有正常机器人都会聚集在崩溃位置；否则，它们都会移动到中点，进入阶段 $\mathcal{C}_3^k$，然后移动并聚集在崩溃位置。□

由引理 9 和引理 10，可得主定理。

定理 5 算法3在 FSYNC 下，对于任意初始配置，使用$n \geq 1$ 方向不一致的机器人，在没有重数检测且采用非刚性移动的情况下，可以解决 SUIG 问题。

只有在没有崩溃且移动均为刚性移动时，才能界定该算法的轮数复杂度。它取决于 $l_{\min}$，即层级最低的机器人的层级，以及 Δ_{level}，即层级最高的机器人与层级最低的机器人之间的层级差。

定理 6 如果移动均为刚性移动且没有发生崩溃，算法3的轮数复杂度为 s

$$O\left(\Delta_{level}^2 + \sqrt{l_{\min}}\right).$$

证明。
如果占据点多于两个，则一轮即可完成聚集。否则，聚集所需轮数取决于机器人的层级。

情形（i）：层级最低的机器人的层级小于 $S_{\Delta_{level}+7}$。我们知道，层级最低的机器人达到层级 $S_{\Delta_{level}+7}$，即达到阶段 $\mathcal{A}^{\Delta_{level}+7}$，需要 $O\left(\Delta_{level}^2\right)$轮。达到该层级后，机器人会在一轮内聚集（它们都会执行移动 $\left(\frac{1}{2}\right)$）。

情形（ii）：层级最低的机器人的层级 $l_{\min}$ 介于 S_k 和 S_{k+1}之间，且 $k \geq \Delta_{level} + 7$。在最坏情况下，层级最低的机器人达到层级 S_{k+1} 后再过一轮即可完成聚集（即所有机器人执行 $\mathcal{A}^{k+1}$时）。因此，这种情况下的轮数复杂度为 O（k） $\subset O\left(\sqrt{l_{\min}}\right)$ （因为k 的定义使得 $S_k = k$（$k + 2$） $\leq l_{\min}$）。□

事实上，当机器人可能发生崩溃时，对手可以让一个机器人在距离其他机器人任意近的位置崩溃，从而使其他机器人的层级大于 S_k ，且 k 可以任意大。要到达阶段 $\mathcal{B}^{k+1}$，轮次数可能为 Ω(k)，其中 k 可以任意大。类似地，如果没有崩溃，但移动不是刚性的，那么对手可以让一个机器人停在距离其他机器人任意近的位置，从而造成相同的情况。在这两种情况下，都可以从这一点开始应用前一个定理（可以看出，如果发生崩溃，但初始时有两个占据节点，前一个定理仍然成立）。

6 结论

我们提出了首个站立式宽容聚集（文献中也称为强容错聚集）解决方案。此外，我们的解决方案具有自稳定性（初始配置可以包含多重性点，甚至可以是双价的），不依赖重数检测能力、共同方向、定向或手性等额外假设，并且其运行时间与机器人之间的最大初始距离成正比。

我们所考虑的机器人的能力非常有限，因此在 SSYNC 等更宽松的执行模型中，该问题无法解决。然而，在满足一些额外假设的情况下，也许可以在 SSYNC 中解决 SUIG：初始配置不是双价的（否则，推论 1 中的不可能性结论仍然适用）、具备重数检测能力（否则，经典聚集的不可能性结论仍然适用 [15]），以及具备持久坐标系（否则，Defago 等人关于 SUIG 的不可能性结论仍然适用 [12] ）。这一开放问题留待未来研究。

参考文献

[1] Noa Agmon 和 David Peleg。自主移动机器人的容错聚集算法。SIAM 计算杂志，36(1):56–82，2006年。[2] Thibaut Balabonski、Am´elie Delga、Lionel Rieg、S´ebastien Tixeuil 和 Xavier Urbain。无需重数检测的同步聚集：经验证的算法。理论计算机系统，63(2):200–218，2019年。

[3] Subhash Bhagat、Sruti Gan Chaudhuri 和 Krishnendu Mukhopadhyaya。在单轴一致性条件下异步无记忆移动机器人的容错聚集。载于 M. Sohel Rahman 和 Etsuji Tomita（编），WALCOM：算法与计算——第9届国际研讨会，WALCOM 2015，孟加拉国达卡，2015年2月26日至28日。论文集，《计算机科学讲义》第8973卷，第149–160页。施普林格，2015年。[4] Zohir Bouzid、Shantanu Das 和 S´ebastien Tixeuil。可容忍多重崩溃故障的移动机器人聚集。载于第33届 IEEE 分布式计算系统国际会议（ICDCS）论文集，第337–346页，2013年7月。[5] Quentin Bramas、Anissa Lamani 和 S´ebastien Tixeuil。容错唤醒会合。载于 St´ephane Devismes 和 Neeraj Mittal（编），分布式系统的稳定化、安全性与保障——第22届国际研讨会，SSS 2020，美国德克萨斯州奥斯汀，2020年11月18日至21日。论文集，《计算机科学讲义》第12514卷，第45–59页。施普林格，2020年。[6] Quentin Bramas、Anissa Lamani 和 S´ebastien Tixeuil。Stand Up Indulgent Gathering。载于 Leszek Gasieniec、Ralf Klasing 和 Tomasz Radzik（编），传感器系统算法——第17届无线传感器网络算法与实验国际研讨会，ALGOSENSORS 2021，葡萄牙里斯本，2021年9月9日至10日。论文集，《计算机科学讲义》第12961卷，第17–28页。施普林格，2021年。[7] Quentin Bramas 和 S´ebastien Tixeuil。无手性的无等待聚集。载于结构信息与通信复杂性国际研讨会论文集，第313–327页。施普林格，2015年。[8] Reuven Cohen 和 David Peleg。异步机器人系统中引力算法的收敛性质。SIAM 计算杂志，34(6):1516–1528，2005年。[9] Pierre Courtieu、Lionel Rieg、S´ebastien Tixeuil 和 Xavier Urbain。聚集不可能性：一项证明。信息处理快报，115(3):447–452，2015年。[10] Shantanu Das、Paola Flocchini、Giuseppe Prencipe、Nicola Santoro 和 Masafumi Yamashita。带灯的自主移动机器人。理论计算机科学，609:171–184，2016年。[11] Xavier D´efago、Maria Gradinariu、St´ephane Messika 和 Philippe Raipin Parv´edy。容错且自稳定的移动机器人聚集。载于 Shlomi Dolev（编），《分布式计算》，第20届国际研讨会，DISC 2006，瑞典斯德哥尔摩，2006年9月18日至20日。论文集，《计算机科学讲义》第4167卷，第46–60页。施普林格，2006年。[12] Xavier D´efago、Maria Potop-Butucaru 和 Philippe Raipin Parv´edy。崩溃或拜占庭故障下移动机器人的自稳定聚集。《分布式计算》，33(5):393–421，2020年。[13] Shlomi Dolev。《自稳定化》。麻省理工学院出版社，2000年。[14] Paola Flocchini、Giuseppe Prencipe 和 Nicola Santoro（编）。《移动实体的分布式计算——移动与计算研究现状》。《计算机科学讲义》第11340卷。施普林格，2019年。[15] Giuseppe Prencipe。一组自主移动机器人聚集的不可能性。理论计算机科学，384(2-3):222–231，2007年。[16] Ichiro Suzuki 和 Masafumi Yamashita。分布式匿名移动机器人：几何图案形成。SIAM 计算杂志，28(4):1347–1363，1999年。