Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Gathering robots in graphs: The central role of synchronicity ☆, ☆☆

Serafino Cicerone [a], Gabriele Di Stefano [a], Alfredo Navarra [b,*]

[a] Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Via Vetoio I-67100 L'Aquila, Italy
[b] Department of Mathematics and Computer Science, University of Perugia, Via Vanvitelli 1, I-06123 Perugia, Italy

ARTICLE INFO

Article history:
Received 24 April 2020
Received in revised form 2 September 2020
Accepted 10 October 2020
Available online xxxx
Communicated by L. Christoph

Keywords:
Distributed algorithms
Mobile robots
Synchrony
Gathering
Graph classes
Time complexity

ABSTRACT

The *Gathering* task for k robots disposed on the n vertices of a graph G requires robots to move toward a common vertex from where they do not move anymore.
When dealing with very weak robots in terms of capabilities, considering synchronous or asynchronous settings may heavily affect the feasibility of the problem. In fact, even though dealing with asynchronous robots in general requires more sophisticated strategies with respect to the synchronous counterpart, sometimes it comes out that asynchronous robots simply cannot solve the problem whereas synchronous robots can.
We study general properties of graphs that can be exploited in order to accomplish the gathering task in the synchronous setting, obtaining an interesting and innovative sufficient condition for the feasibility of the gathering task in graphs, regardless the topology.
Furthermore, we consider dense and symmetric graphs like complete and complete bipartite graphs where in general the topology does not allow to distinguish vertices where to finalize the gathering. In such topologies, we *fully characterize* the solvability of the gathering task in the synchronous setting by suitably combining some strategies arising from the general approach with specific techniques dictated by the considered topologies.
From the lower bound point of view in terms of number of synchronous time units required to accomplish the gathering task, in general nothing better than $\Omega(D_G)$, with D_G being the diameter of the input graph G, can be provided. For both complete and complete bipartite graphs, we prove a lower bound of $\Omega(\log_\phi k)$, with ϕ being the well-known golden ratio. Combined with the provided algorithms, this reveals to be asymptotically tight for complete graphs while for complete bipartite graphs an additive factor of $\delta(k)$ is achieved, with $\delta(k)$ being the function that returns the number of divisors of the integer k.

1. Introduction

In this paper, we consider the *Gathering* task by means of a swarm of very weak - in terms of capabilities - robots moving on graphs. Initially, each robot occupies a different vertex of the graph. The task requires robots to move toward a

☆ Preliminary results appeared in the Proceedings of the 26th Int.'l Colloquium on Structural Information and Communication Complexity (SIROCCO) 2019 [13].

☆☆ The work has been supported in part by the Italian National Group for Scientific Computation (GNCS-INdAM).

* Corresponding author.
E-mail addresses: serafino.cicerone@univaq.it (S. Cicerone), gabriele.distefano@univaq.it (G. Di Stefano), alfredo.navarra@unipg.it (A. Navarra).

https://doi.org/10.1016/j.tcs.2020.10.011
0304-3975/

common vertex from where they do not move anymore. The gathering task is one of the basic primitives extensively studied in the context of robot-based computing systems.

Whether or not a mobile robot system can solve a given problem typically depends on the capabilities one assumes for robots. A common approach in distributed computing is to detect the minimal capabilities that are necessary so as robots can perform basic tasks. The rationale behind this approach is twofold: it is theoretically interesting to answer the minimality question; the weaker the model assumed to solve a task, the wider its applicability, including more powerful robots prone to faults.

1.1. Robots' model

In this paper, robots are considered to be:

- *Anonymous*: no unique identifiers;
- *Autonomous*: no centralized control;
- *Dimensionless*: no occupancy constraints, no volume, modeled as entities located on vertices of a graph;
- *Oblivious*: no memory of past events;
- *Homogeneous*: they all execute the same *deterministic*[1] algorithm;
- *Silent*: no means of direct communication;
- *Disoriented*: no common coordinate system, no common left-right orientation;

The ability to solve a given task depends also on the magnitude of synchrony among the robots. Concerning synchrony, one of the most studied scenario assumes robots to alternate within two main states *active* or *inactive*. When active, a robot executes a `Look-Compute-Move` (`LCM`) cycle by performing the following three operations in sequence, each of them associated with a different state:

- `Look`: The robot observes the environment. The result of this phase is a snapshot of the positions of all robots with respect to its own perception.
- `Compute`: The robot executes the designed algorithm, using the data sensed in the `Look` phase as input. When robots are moving in graphs, the result of this phase either is the vertex where the robot currently resides (i.e., the robot does not move) or it is a vertex among those in the direct neighborhood (i.e., at most one edge per cycle can be traversed by a robot).
- `Move`: The robot moves toward the computed target. If the target is the current position, then the robot stays still, i.e., it performs what is called a *nil* movement.

Actually moves are always considered as *instantaneous*. This results in always perceiving robots on vertices and never on edges during `Look` phases. Hence, robots cannot be seen while moving, but only at the moment they may start moving or when they arrived. The rationale behind this assumption is that the graph may model a communication network, whereas robots model software agents.

In the literature, different characterizations of the environment have been considered according whether robots are fully-synchronous, semi-synchronous (cf. [45–47]), semi-asynchronous (cf. [9]) or asynchronous (cf. [3,8,10,12,24,31,33]). These synchronization models are illustrated in Fig. 1 and defined as follows:

- *Fully-Synchronous* (FSync): All robots are always active, continuously executing in a synchronized way their LCM cycles. Hence the time can be logically divided into global rounds. In each round all the robots, obtain a snapshot of the environment, compute on the basis of the obtained snapshot and perform their computed move.
- *Semi-Synchronous* (SSync): It coincides with the FSync model, with the only difference that not all robots are necessarily activated in each round.
- *Semi-Asynchronous* (SAsync): Robots are activated independently. Like in FSync or SSync, the duration of each phase is assumed to be always the same. Differently from FSync or SSync, two activated robots can be in different phases even though phases are synchronized.
- *Asynchronous* (Async): Robots are activated independently, and the duration of each phase is finite but unpredictable. As a result, robots do not have a common notion of time.

In Async, the amount of time to complete a full `LCM`-cycle is assumed to be finite but unpredictable. Moreover, in the SSync, SAsync, and Async cases it is usually assumed the existence of an *adversary* which determines the computational cycles timing. Such timing is assumed to be *fair*, that is, each robot performs its `LCM`-cycle within finite time and infinitely often. Without such an assumption the gathering would be unsolvable as the adversary could prevent some robots to ever move.

[1] No randomization features are allowed.

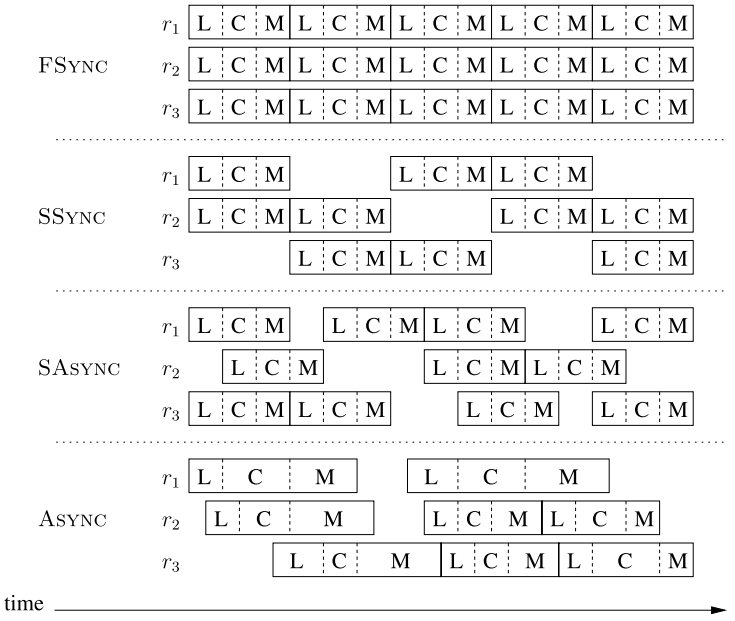

Fig. 1. The execution model of computational cycles for each of FSYNC, SSYNC, SASYNC, and ASYNC robots. The inactivity of robots is implicitly represented by empty time periods.

It is worth to remark that the four synchronization schedulers induce the following hierarchy (see [23,26]): FSYNC robots are more powerful (i.e., they can solve more tasks) than SSYNC robots, that in turn are more powerful than SASYNC robots, that in turn are more powerful that ASYNC robots. This simply follows by observing that the adversary can control more parameters in ASYNC than in SASYNC, and so forth. In other words, protocols designed for ASYNC robots also work for SASYNC, SSYNC and FSYNC robots. On the contrary, any impossibility result proved for FSYNC robots also holds for SSYNC, SASYNC and ASYNC robots.

Whatever the assumed scheduler is, the activations of the robots determine specific time instants $t = 0, 1, 2, \ldots$ during which at least one robot is activated. Apart for the ASYNC case where the notion of time is not shared by robots, for the other types of schedulers robots are synchronized. In the FSYNC case, each robot is active at each time unit. In the SSYNC and SASYNC cases we assume that at least one robot is active at each time t. Focusing on FSYNC and SSYNC robots, if $C(t)$ denotes the configuration observed by some robots at time t during their `Look` phase, then an *execution* of an algorithm $\mathcal{A}$ from an initial configuration C is a sequence of configurations $\mathbb{E} : C(0), C(1), \ldots$, where $C(0) = C$ and $C(t+1)$ is obtained from $C(t)$ by moving each robot (which is active at time t) according to the result of the `Compute` phase as implemented by $\mathcal{A}$.

Notice that, given an algorithm $\mathcal{A}$, in SSYNC, SASYNC, or ASYNC there exists more than one execution from $C(0)$ depending on the activation of the robots, whereas in FSYNC the execution is unique as it always involves all robots in all time instants.

For FSYNC robots, the *time complexity* of an algorithm $\mathcal{A}$ is the maximum amount of time units required by $\mathcal{A}$ for processing any gatherable initial configuration. For the other types of schedulers, the time complexity usually refers to the number of required *rounds*, where a round is the time (finite but unpredictable) within which all robots complete a `LCM`-cycle at least once. For the gathering problem, a natural lower bound for the time complexity of any algorithm is $\Omega(D_G)$, where D_G is the diameter of the graph underlying the configuration.

It is very common (as dictated by impossibility results) that in combination with the `LCM`-model, robots are endowed with the so-called *multiplicity detection* capability (see e.g. [15,38]). Basically, when more than one robot resides on the same vertex x, then x is said to be occupied by a *multiplicity*. During the `Look` phase, robots can perceive *multiplicities*. The multiplicity detection capability might be *local* or *global*, depending whether the multiplicity is detected only by robots composing the multiplicity itself or by any robot performing the `Look` phase, respectively. Moreover, the multiplicity detection can be *weak* or *strong*, depending whether a robot can detect only the presence of a multiplicity or if it perceives the exact number of robots composing the multiplicity, respectively.

1.2. Related work

The gathering problem has been deeply investigated and fully characterized for robots moving on the Euclidean plane [15] endowed with global weak multiplicity detection. The problem is in fact a special case of the more general

Table 1
Achievements about Gathering on graphs. Results marked by (*) are those contained in this paper.

Graph	Ref.	Model	Multiplicity	Achievement
Any	[29]	FSync	global strong	some impossibility
	[29]	Async	global strong	basic feasibility
	(*)	SSync	global strong	sufficient condition
Trees	[17,18]	Async	none	full characterization
	[29]	Async	global strong	traveled distance opt
Rings	[38]	Async	global weak	asymmetric conf.s
	[37]	Async	global weak	symmetric dense conf.s
	[20]	Async	global weak	conf.s with 6 robots
	[36]	Async	local weak	opt asymmetric
	[19]	Async	global weak	(almost) full characterization
	[21]	Async	local weak	asymmetric conf.s
	[22]	Async	local weak	full characterization
	[29]	Async	global strong	traveled distance opt
	[25]	SSync	global weak	separation with Async
Regular Bipartite	[34]	Async	local weak	full characterization
Finite Grid	[17]	Async	none	full characterization
Infinite Grid	[5,16]	FSync	none	time opt (limited visibility)
	[28]	Async	global strong	traveled distance opt
Oriented Hypercubes	[4]	Async	global weak	traveled distance opt
Complete	(*)	FSync	global strong	opt full characterization
	[14]	SSync	global strong	impossibility
Complete Bipartite	(*)	FSync	global strong	full characterization
	[14]	SSync	global strong	some conf.s

pattern formation problem [10,12,45]. A slightly different model imposing robots to gather at some visible and predetermined points (called meeting points) provided in the Euclidean plane has been also investigated and fully characterized, see [6–8].

In contrast, on graphs not much is known, see the recent surveys [32] and [42]. Apart for some preliminary results contained in [29], holding for general graphs, so far only a few of specific topologies have been investigated like trees [17,18,29], rings [19–22,25,29,36–38], regular bipartite graphs [34], finite [17] or infinite [5,16,28] grids, hypercubes [4], complete graphs and complete bipartite graphs [14], also from an optimization perspective. In particular, Table 1 summarizes the achieved results with respect to the different topologies. Most of such topologies are very symmetric, that is, vertices can be partitioned into a few of classes of equivalence. The choice of such topologies has been dictated by the requirement to make harder the design of a resolution algorithm as robots cannot exploit much topological properties. For instance, if a tree or a finite grid admits only one center, then all robots can detect it and move there, even asynchronously. Contrary, in rings, infinite grids, complete graphs or hypercubes, all vertices are equivalent and the synchronicity may heavily impact on feasibility. In rings, a full characterization has been provided for FSync and SSync robots [25] endowed with global weak multiplicity detection, whereas in the Async case a few of configurations are still not known whether they are solvable or not [19,27].

Concerning feasibility aspects, the aim has been usually that of finding the minimal set of assumptions under which robots are able to solve the problem. One of the main observations for gathering in the Euclidean plane has been to show that Async robots are as much powerful as FSync ones [15], except for the only case of exactly two robots. This case is unsolvable in the SSync context whereas it is solvable by two FSync robots, i.e. it represents a separator between SSync and FSync.

In general graphs, a full characterization for the gathering task is missing, even for FSync robots. Clearly, on graphs, movements are very constrained with respect to the Euclidean plane, and this heavily affects the possibility to design general strategies that are independent from the underlying graph topology. A main observation coming out from the literature about gathering in graphs is that feasibility is very dependent on the assumed level of synchronicity. On the one hand, dealing with Async robots is much harder than considering synchronized settings. On the other hand, it may happen that Async robots simply cannot solve some instances, hence sensibly reducing the scope of research for resolution algorithms. In other words, the graph context seems requiring deep investigation on the different results one may achieve when switching from the Async to the SAsync, SSync or FSync cases.

1.3. Our results

In this paper, we focus on FSync robots endowed with global strong multiplicity detection, see Table 1. First, we study general properties of graphs that can be exploited in order to accomplish the gathering task in the SSync (and hence also in the FSync) setting. The investigation leads to obtain an interesting and innovative sufficient condition for the feasibility, applicable to any topology. Intuitively, according to a measure d related to the symmetry of the input graph and to the number k of robots, the gathering can be achieved if some primality properties between d and k hold.

Although we provide some evidence about the wide range of applicability of our achievement with respect to many different topologies, still there are cases where the topology cannot be exploited to solve the gathering problem. To this respect, we then consider dense and symmetric graphs like complete and complete bipartite graphs where Async, SAsync or SSync robots cannot build their strategy on particular sets of vertices, due to the high symmetry of such graphs. In such topologies we fully characterize the solvability of the gathering task in the FSync setting for robots endowed with global strong multiplicity detection. We first show that with weaker assumptions on the multiplicity detection only very restricted cases can be solved. Then, by suitably combining some strategies arising by the general approach with specific techniques dictated by the considered topologies, we provide the corresponding resolution algorithms.

We also evaluate the time complexity of our algorithms to accomplish the gathering task in terms of number of synchronous time units required. From the lower bound point of view, a natural limit to the number of moves required by any algorithm is $\Omega(D_G)$, with D_G being the diameter of the input graph G. Interestingly, we are able to prove that the addressed topologies, namely complete and complete bipartite graphs - where $D(G) \leq 2$, present a lower bound of $\lfloor \frac{\log_\phi(\sqrt{5}\cdot k-1)-1}{2} \rfloor$ time units, where k represents the number of robots and ϕ is the well-know golden ratio. Concerning the provided resolution algorithms, by combining their complexity with the lower bound, we get an asymptotically optimal solution for complete graphs. For complete bipartite graphs, we provide a resolution algorithm that requires $O(\log k + \delta(k))$ time, where $\delta(k)$ is the function that returns the number of divisors of the integer k. Comparing this complexity with the lower bound, it is worth to remark that our algorithm is tight each time $\delta(k) = O(\log k)$. In fact, from [30] it is known that $\delta(k)$ behaves like a normal random variable with mean $\log\log k$ and standard deviation $\sqrt{\log\log k}$. However, concerning the growth rate of $\delta(k)$ it is known that $\delta(k) = o(n^\epsilon)$ for all $\epsilon > 0$; more precisely, $\delta(k) \leq k^{\frac{\log 2}{\log\log k}}$ [35].

1.4. Outline

In the next section, we provide some definitions, notation and preliminary results inherited from the literature concerning the feasibility of the gathering task in graphs. Section 3 provides a new general result about a sufficient condition for the resolution of the gathering task in graphs, regardless the underlying topology. Section 4 concerns the full characterization of the gathering in complete graphs by means of FSync robots, along with the analysis of the performance of the provided algorithm in terms of synchronous time units. Section 5 concerns the full characterization of the gathering in complete bipartite graphs by means of FSync robots, and the analysis of the performance of the proposed algorithm. Section 6 contains the lower bound to the complexity of the gathering problem in the addressed graph topologies. Finally, Section 7 provides useful discussions for future works.

2. Problem definition and general impossibility results

The topology where robots are placed on is represented by a simple and connected graph $G = (V, E)$, with vertex set V and edge set E. The cardinality of V is represented as $|V|$, n or $|G|$. A function $\lambda : V \to \mathbb{N}$, from the set of vertices to the set of integer numbers represents the number of robots on each vertex of G, and we call (G, λ) a *configuration* whenever $\sum_{v\in V} \lambda(v)$ is bounded and greater than zero. A vertex $v \in V$ such that $\lambda(v) > 0$ is said *occupied*, *unoccupied* otherwise. A subset $V' \subseteq V$ is said *occupied* if at least one of its elements is occupied, *unoccupied* otherwise. A configuration is *initial* if each robot lies on a different vertex (i.e., $\lambda(v) \leq 1$ for each $v \in V$). A configuration is *final* if all the robots are on a single vertex (i.e., $\exists u \in V : \lambda(u) > 0$ and $\lambda(v) = 0,\ \forall v \in V \setminus \{u\}$). The gathering problem can be formally defined as the problem of transforming an initial configuration into a final one. Throughout the paper we assume that each initial configuration is composed of at least two robots (otherwise the problem is trivially solved). A *gathering algorithm* for this problem is a deterministic distributed algorithm that brings the robots in the system to a final configuration in a finite number of LCM-cycles from any given initial configuration, regardless of the adversary.[2] Formally, an algorithm $\mathcal{A}$ solves the gathering problem for an initial configuration C if, for any execution $\mathbb{E} : C = C(0), C(1), C(2), \ldots$ of $\mathcal{A}$, there exists a time instant $t > 0$ such that $C(t)$ is final and no robots move after t, i.e., $C(t') = C(t)$ holds for all $t' \geq t$. Given an initial configuration $C = (G, \lambda)$, if there exists a gathering algorithm for C we say that C is *gatherable*, otherwise we say that C is *ungatherable*.

We now recall from [29] the notions of configuration automorphisms and symmetries to be applied to general graphs, and accordingly we also recall general impossibility results.

[2] In this work, we show that even in the case of FSync robots the adversary can control some variables that can impact on the solvability of the gathering task.

2.1. Configuration automorphisms and symmetries

Two undirected graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic* if there is a bijection φ from V to V' such that $\{u, v\} \in E$ if and only if $\{\varphi(u), \varphi(v)\} \in E'$. An *automorphism* on a graph G is an isomorphism from G to itself, that is a permutation of the vertices of G that maps edges to edges and non-edges to non-edges. The set of all automorphisms of G forms a group called *automorphism group* of G and denoted by $\mathrm{Aut}(G)$. Two vertices $u, v \in V$ are *equivalent* if there exists an automorphism $\varphi \in \mathrm{Aut}(G)$ such that $\varphi(u) = v$.

The concept of isomorphism can be extended to configurations in a natural way: two configurations (G, λ) and (G', λ') are isomorphic if G and G' are isomorphic via a bijection φ and for each vertex v in G, $\lambda(v) = \lambda'(\varphi(v))$. An *automorphism* on a configuration (G, λ) is an isomorphism from (G, λ) to itself and the set of all automorphisms of (G, λ) forms a group that we call *automorphism group* of (G, λ), denoted by $\mathrm{Aut}((G, \lambda))$.

If $|\mathrm{Aut}(G)| = 1$, that is G admits only the identity automorphism, then G is said *asymmetric*, otherwise it is said *symmetric*. Analogously, if $|\mathrm{Aut}((G, \lambda))| = 1$, we say that the configuration (G, λ) is *asymmetric*, otherwise it is *symmetric*. In a configuration (G, λ), two distinct robots r and r' on vertices v and v', respectively, are *equivalent* if there exists $\varphi \in \mathrm{Aut}((G, \lambda))$ such that $v' = \varphi(v)$. Note that $\lambda(v) = \lambda(v')$ whenever v and v' are equivalent. From an algorithmic point of view, it is important to remark that when there are equivalent vertices in a configuration, a robot r located at a vertex u cannot distinguish its position from that of a robot r' located at vertex $v = \varphi(u)$. As a consequence, no algorithm can distinguish between two equivalent robots, and then it cannot avoid that the two robots perform a same move simultaneously.

Let $G = (V, E)$ be a graph and $C = (G, \lambda)$ be a configuration defined on G. Given $\varphi \in \mathrm{Aut}(C)$ different from the identity, the *cyclic subgroup* of order p generated by φ is given by $\Gamma = \{\varphi^0, \varphi^1 = \varphi \circ \varphi^0, \varphi^2 = \varphi \circ \varphi^1, \ldots, \varphi^{p-1} = \varphi \circ \varphi^{p-2}\}$, where φ^0 is the identity automorphism, $\varphi^i \neq \varphi^0$ for each $0 < i < p$, and $\varphi^p = \varphi \circ \varphi^{p-1} = \varphi^0$. In C, the *orbit* of any vertex $v \in V$ generated by the cyclic subgroup Γ is defined as $\Gamma v = \{\gamma(v) \mid \gamma \in \Gamma\}$.

The next theorems exploit the concepts of cyclic subgroup and orbit to provide a sufficient condition for a configuration to be ungatherable. Before recalling such a result, we first need the following definition.

Definition 1. *[29]* Let $G = (V, E)$ be any graph and let $C = (G, \lambda)$ be any configuration. An automorphism $\varphi \in \mathrm{Aut}(C)$ is called *partitive* on $V' \subseteq V$ if the cyclic subgroup $\Gamma = \{\varphi^0, \varphi^1 = \varphi \circ \varphi^0, \varphi^2 = \varphi \circ \varphi^1, \ldots, \varphi^{p-1} = \varphi \circ \varphi^{p-2}\}$ generated by φ has order $p > 1$ and is such that $|\Gamma u| = p$ for each $u \in V'$.

Let $G = (V, E)$ be any graph and let $C = (G, \lambda)$ be any configuration that is partitive on $V' \subseteq V$ according to any cyclic subgroup Γ. In such a case, notice that the following properties hold:

1. the orbits Γv, for each $v \in V'$, form a partition of V';
2. vertices/robots belonging to a same orbit are pairwise equivalent.

For the sake of simplicity, from now on we say that a configuration *C is partitive* whenever there exists an automorphism $\varphi \in \mathrm{Aut}(C)$ which is partitive on the entire set V.

The following two theorems use the above definition and notation to provide impossibility results for the gathering problem on general graphs. Since they refer to FSync robots, they also hold for SSync, SAsync and Async robots.

Theorem 1. [29] *Let $G = (V, E)$ be any graph and let $C = (G, \lambda)$ be any non-final configuration. If there exists $\varphi \in Aut(C)$ partitive on V then C is ungatherable.*

It is worth to remark that the above theorem requires the existence of an automorphism φ, which in turn is based on the function λ defining the exact number of robots on each vertex. Hence, Theorem 1 holds when the robots are endowed with the global-strong multiplicity detection. Stating a negative result, it follows that such a theorem holds even when considering weaker robots (i.e., without global-strong multiplicity detection).

Fig. 2 shows four initial configurations. Among them, configurations C_1 and C_2 admit an automorphism which is partitive on the entire vertex set V. In both cases, these automorphisms generate a partition on V consisting of three orbits (in C_1 each orbit has size three while in C_2 each orbit has size two). By the above theorem we deduce that both C_1 and C_2 are ungatherable, since each move allowed by any algorithm can be executed synchronously, due to the adversary, by all the robots belonging to the same orbit. This would always produce a new partitive configuration (for each possible move planned by the algorithm, the robots belonging to the same orbit always remain separated and located in any of the orbits).

The following theorem shows that some configurations can be gathered only at some predetermined vertices.

Theorem 2. [29] *Let $G = (V, E)$ be any graph, $C = (G, \lambda)$ be any configuration, and $V' \subset V$ unoccupied. If there exists an automorphism $\varphi \in Aut(C)$ that is partitive on $V \setminus V'$, then any gathering algorithm can not assure the gathering on a vertex in $V \setminus V'$.*

In this work, we will use the above theorem in the following rephrased form:

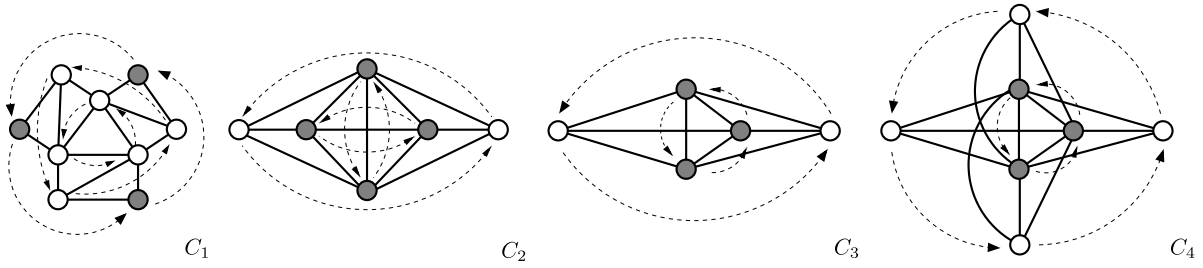

Fig. 2. Four configurations where gray vertices indicate the presence of one robot and arrows indicate mappings due to cyclic subgroups. In particular, C_1 and C_2 show three cyclic subgroups each, while C_3 and C_4 present two cyclic subgroups each.

Corollary 3. *Let $G = (V, E)$ be any graph, $C = (G, \lambda)$ be any configuration, and $V' \subset V$ unoccupied. If there exists an automorphism $\varphi \in Aut(C)$ that is partitive on $V \setminus V'$, then each gathering algorithm for C (if any) must move robots toward V'.*

The rationale of this corollary is the following: if the algorithm moves robots toward vertices in $V \setminus V'$ then the adversary can synchronously move all the equivalent robots belonging to a same orbit, each one to a different vertex. This produces a new configuration which is still partitive on $V \setminus V'$ and where the orbits in $V \setminus V'$ maintain the same size as in the initial configuration. As a consequence, as long as an algorithm moves robots toward vertices in $V \setminus V'$, the configuration remains partitive. Hence, in order to aim to gather, the move toward unoccupied vertices must be performed, eventually.

Fig. 2 shows configurations C_3 and C_4 that represent cases where Corollary 3 can be applied. In fact, both these configurations do not admit any automorphism which is partitive on the entire set V, but both admit an automorphism which is partitive on the subset defined by all the occupied vertices. According to the corollary above, any gathering algorithm for C_3 or C_4 (if any) must finalize the gathering on a vertex which was initially unoccupied. It is worth to remark that both C_3 and C_4 admit more than one automorphism which is partitive on proper subsets of V. In particular, C_3 admits at least two partitive automorphisms (one on the occupied and the other on the unoccupied vertices) inducing two orbits with two and three elements, respectively. The same kind of automorphisms induces two orbits with three and four elements, respectively, in C_4. In Section 3.3 these configurations will be considered again after providing a sufficient condition for gathering.

3. A sufficient condition for gathering in arbitrary graphs

In this section, we provide a sufficient condition for gathering FSync robots in arbitrary graphs. This result exploits new concepts we define in the following, like *recognizable subgraphs* and *d-primality*.

3.1. Recognizable subgraphs

Informally, an induced subgraph H of a graph G is said *recognizable* if any automorphism of G maps H on itself, that is, H cannot be confused with other subgraphs. Formally:

Definition 2. An induced subgraph $H = (V', E')$ of a graph $G = (V, E)$ is *recognizable* if $V' = \{\varphi(v) \mid v \in V'\}$ for each automorphism $\varphi \in \mathrm{Aut}(G)$.

In the following, we provide interesting properties of recognizable subgraphs. Let $G = (V, E)$ be a graph: if $V' \subseteq V$, then $G[V']$ denotes the subgraph induced by V'. By definition, the empty subgraph is recognizable. Then, the following properties hold:

1. G is recognizable;
2. if V_1 and V_2 are the vertex sets of recognizable subgraphs of G, then $G[V_1 \cup V_2]$ is recognizable;
3. if G is asymmetric then any subgraph composed of a single vertex is recognizable.

Definition 3. A recognizable subgraph H of a graph $G = (V, E)$ is *minimal* if there is no proper subgraph H' of H such that H' is recognizable.

Additional properties about recognizable subgraphs are provided by the following statements.

Lemma 4. *Let $H = (V', E')$ a recognizable subgraph of $G = (V, E)$. Then $G[V \setminus V']$ is recognizable.*

Proof. Let us assume $G[V \setminus V']$ is not recognizable. Then, there exists $\varphi \in \mathrm{Aut}(G)$ and a vertex $v \in V \setminus V'$ such that $\varphi(v) \notin V \setminus V'$. Then $\varphi(v) = v' \in V'$. This means that $\varphi^{-1}(v') \in V \setminus V'$, that is not possible being H recognizable. □

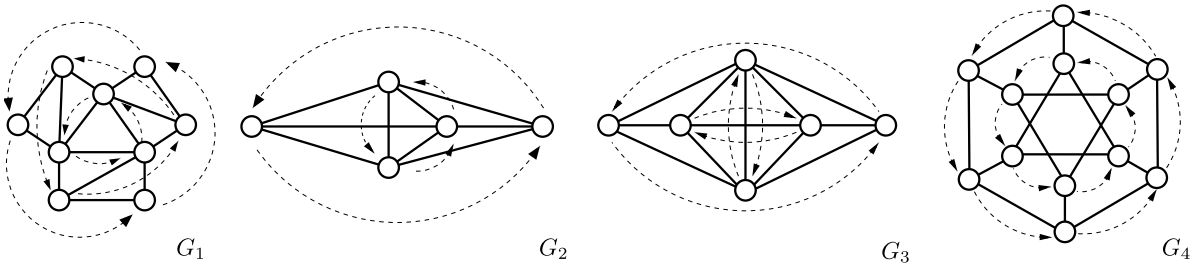

Fig. 3. Visualization of some graphs along with the unique vertex partition generated by the minimal recognizable subgraphs (cf. Theorem 6). The dotted arrows show the equivalence among vertices belonging to the same minimal recognizable subgraph (cf. Corollary 7). Notice that in G_4 a subgraph induced by a minimal recognizable subgraph is composed of two isomorphic connected components.

Lemma 5. *Let $G[V_1]$ and $G[V_2]$ be two distinct minimal recognizable subgraphs of $G = (V, E)$, with $\emptyset \neq V_1 \subseteq V$ and $\emptyset \neq V_2 \subseteq V$. Then $V_1 \cap V_2$ is empty.*

Proof. By contradiction, let $V_1 \cap V_2 \neq \emptyset$. For each $\varphi \in \mathrm{Aut}(G)$, if $v \in V_1 \cap V_2$, then $\varphi(v) \in V_1$ and $\varphi(v) \in V_2$, and hence $\varphi(v) \in V_1 \cap V_2$. This means that $G[V_1 \cap V_2]$ is recognizable, a contradiction being $G[V_1]$ and $G[V_2]$ minimal. □

Theorem 6. *The set containing all the minimal recognizable subgraphs of $G = (V, E)$ forms a unique partition of V.*

Proof. The required partition can be found with the following procedure. If G is minimal recognizable, we are done. If G is not minimal recognizable, let $G[V_1]$ be a minimal recognizable subgraph of G and let $\{V_1, V \setminus V_1\}$ be a partition of V. By Lemma 4, $G[V \setminus V_1]$ is recognizable. If it is also minimal recognizable we are done, otherwise recursively apply the procedure to $G[V \setminus V_1]$. Regarding the uniqueness, by contradiction let us suppose that there exist $\mathcal{V} = \{V_1, V_2, \ldots, V_p\}$ and $\mathcal{V}' = \{V'_1, V'_2, \ldots, V'_{p'}\}$ distinct partitions of V such that $G[V_i]$, $G[V'_j]$ are minimal recognizable graphs, for each $i = 1, 2, \ldots, p$ and $j = 1, 2, \ldots, p'$. Now, since both $\mathcal{V}$ and $\mathcal{V}'$ are different partitions of V, then there exists a vertex $v \in V$ that belongs to distinct sets V_{i*} and V'_{j*}. This implies that $V_{i*} \cap V'_{j*} \neq \emptyset$, while Lemma 5 requires $V_{i*} \cap V'_{j*} = \emptyset$, a contradiction. □

The following corollary shows that each pair of vertices inside a minimal recognizable subgraph of G concerns two equivalent vertices.

Corollary 7. *Let $G = (V, E)$ be a graph and let $\{V_1, V_2, \ldots, V_p\}$ be the unique partition of V induced by the minimal recognizable subgraphs. For each pair of vertices $u, v \in V_i$, $i = 1, 2, \ldots, p$, u and v are equivalent.*

Proof. By contradiction, assume that there are two vertices u and v in a set V_i such that for each $\varphi \in \mathrm{Aut}(G)$, $\varphi(u) \neq v$. Consider the set of vertices $U = \{\gamma(u) \mid \gamma \in \mathrm{Aut}(G)\}$. Let us show that $G[U]$ is recognizable. To this end, consider an element $w \in U$. Then, there exists $\gamma' \in \mathrm{Aut}(G)$ such that $w = \gamma'(u)$. For each $\gamma'' \in \mathrm{Aut}(G)$, it must be that $\gamma''(w) \in U$ because $\gamma'' \circ \gamma'$ is in $\mathrm{Aut}(G)$, then $\gamma''(w) = \gamma''(\gamma'(u))$ is in U. By the generality of w, this implies that $G[U]$ is recognizable. However, v is not in U, hence $G[U]$ is a proper recognizable subgraph of $G[V_i]$. This is a contradiction being $G[V_i]$ a minimal recognizable subgraph. □

Fig. 3 visualizes some graphs where Theorem 6 and Corollary 7 apply. Let $G = (V, E)$ be a graph and H be a minimal recognizable subgraph of G. Notice that if H is disconnected, all the connected components are pairwise isomorphic (cf. graph G_4 in Fig. 3). In the remainder we use $c(H)$ and $s(H)$ to denote the number of connected components of H and the size of each connected component of H, respectively.

3.2. d-primality and batches

In order to make use of a recognizable subgraph H for gathering purposes, we need to relate the number k of robots of a given configuration to the topology of H.

Consider the following scenario: (1) take a configuration C composed of k FSync robots that must move according to an algorithm $\mathcal{A}$, (2) in C all the robots are equivalent (i.e., $\mathcal{A}$ cannot distinguish among them, so all of them will move), (3) the target of the move is represented by d vertices, and (4) $\mathcal{A}$ cannot distinguish among the d possible destinations (so, an adversary can decide on the actual target for each moving robot). In such a situation we want to determine when the adversary cannot produce a configuration C' composed of $d' \leq d$ multiplicities with k/d' robots per multiplicity (which may result in a partitive configuration). To this end, in this section we define the concepts of *d–primality* and *batch*.

Definition 4. Let k and d be two positive integers both greater than zero. We say that k *is* d*–prime* if $lpf(k) > d$, where $lpf(k)$ denotes the *least prime factor of* k.

Notice that when k is d–prime the following properties hold:

- $k > d > 0$;
- if $d = 1$ then every $k > 1$ is d–prime;
- for each integer $2 \leq d' \leq d$, d' does not divide k.

Definition 5. Let $G = (V, E)$ be any graph, $C = (G, \lambda)$ be a configuration with k robots, $\mathcal{A}$ be a gathering algorithm for C, and $\mathbb{E} : C = C(0), C(1), C(2), \ldots$ be any execution of $\mathcal{A}$ that starts from C. Given an integer i, $0 < i \leq k$, the *batch* $B_i(t)$, $t \geq 0$, is the subset of vertices $\{u \in V \mid \lambda(u) = i\}$ in $C(t)$.

We simply use B_i when we are not interested to any specific time instant. Given a batch B_i, we use the following additional notions:

- the *order* of B_i is i, i.e., there are i robots in each vertex in B_i. $B_{\min}$ and $B_{\max}$ denote the non-empty batches with minimum and maximum order, respectively;
- the *size* of B_i is $|B_i|$, i.e., the number of vertices in B_i.

By the above definitions, in a configuration C with k robots:

1. $|B_1| = k$ if and only if C is initial;
2. $B_k \neq \emptyset$ (i.e., $B_{\max}$ has order k) if and only if C is final.

Lemma 8. *Let $G = (V, E)$ be a graph with n vertices, H be a subgraph of V, and $C = (G, \lambda)$ be a non-final configuration composed of k robots, $2 \leq k \leq n$, such that k is $\max\{c(H), s(H)\}$–prime. If all the k robots reside on vertices of H, then the following conditions hold:*

1. *if the robots are on different connected components of H, then there are connected components of H with different number of robots;*
2. *if the robots are all on the same connected component of H, then in H there are batches with different orders.*

Proof. Since k is $\max\{c(H), s(H)\}$–prime, then $lpf(k) > \max\{c(H), s(H)\}$. This implies that k is both $c(H)$–prime and $s(H)$–prime. These relationships imply $k \geq lpf(k) > s(H)$ and $k \geq lpf(k) > c(H)$; in other words, k is greater than both $c(H)$ and $s(H)$.

We now analyzed two cases according whether different components of H are occupied or not. Assume $c(H) > 1$ and the robots are on different components of H. Since k is $c(H)$–prime then the k robots cannot be equally distributed over any subset of the $c(H)$ connected components. This implies the first property.

Assume all the robots are on the same component of H. Since k is $s(H)$–prime, then the k robots cannot be equally distributed over any subset of the $s(H)$ vertices. Then, the second property holds. □

3.3. A sufficient condition for the resolution of the gathering

In this section, we use the notions of recognizable graphs, d-primality, and *graph canonization* to provide a sufficient condition to the solvability of the gathering problem by means of SSync robots. This condition is applicable to any graph topology.

In graph theory the *graph canonization* is the problem of finding a canonical form of a given graph G. A canonical form is a function that associates to G a labeled graph $Canonic(G)$ such that $Canonic(G)$ is isomorphic to G and every graph G' that is isomorphic to G is such that $Canonic(G') = Canonic(G)$ [2]. The labels are from a linearly ordered set (e.g., $\{1, 2, \ldots, n\}$). The graph canonization problem is at least as computationally hard as the graph isomorphism problem, which is not known to be solvable in polynomial time nor to be NP-complete. However in [1] a linear time algorithm is reported that with probability at least $1 - \exp(-O(n \log n / \log\log n))$ produces a canonical labeling. This justifies the good behavior in practice of the deterministic algorithm proposed by McKay [39] whose complexity is studied in [40].

Theorem 9. *Let $G = (V, E)$ be a graph with n vertices, and let $C = (G, \lambda)$ be an initial configuration composed of k* SSync *robots, $2 \leq k \leq n$. If there exists a minimal recognizable subgraph H of G such that k is $\max\{c(H), s(H)\}$–prime, then C is gatherable.*

Proof. Let us consider the set S containing all the minimal recognizable subgraphs H' of G that minimizes $\max\{c(H'), s(H')\}$. Consider a canonical labeling $Canonic(G)$ and let $\bar{H} \in S$ be the subgraph with the vertex having the minimum label.

Being the labeling unique up to vertex equivalence, and being the labels lexicographically ordered, all the robots can agree on $\bar{H}$.

By hypothesis there exists a minimal recognizable subgraph H of G such that k is $\max\{c(H), s(H)\}$–prime. Then k is also $\max\{c(\bar{H}), s(\bar{H})\}$–prime, being $\max\{c(\bar{H}), s(\bar{H})\} \leq \max\{c(H), s(H)\}$. Let $\bar{V}$ be the vertex set of $\bar{H}$. We show there exists an algorithm $\mathcal{A}_g$ able to gather all robots in C on a vertex of $\bar{V}$. If $\mathbb{E} : C(0), C(1), \ldots$ denotes any execution of $\mathcal{A}_g$, then $\mathcal{A}_g$ will apply the same move m to each active robot in any obtained configuration $C(t)$. Such a move m is defined as follows:

- (*target of m*) if there are no robots in $\bar{V}$, the target $T(t)$ for m coincides with $\bar{V}$. If there are robots on different components of $\bar{H}$, $T(t)$ corresponds to vertices of the connected components that contain the largest number of robots. If all the robots are on a single component of $\bar{H}$ then the target $T(t)$ is the set containing each vertex $v \in \bar{V}$ such that $\lambda(v)$ is maximum.
- (*robots moving according to m*) robots allowed to move are all those not on vertices of $T(t)$.
- (*trajectory*) if a moving robot r is not on a vertex in $T(t)$, it can move toward an adjacent vertex along a shortest path to a vertex in $T(t)$ having minimum distance from r. If all the robots are on the same connected component of $\bar{H}$, the shortest path is chosen among only those having all the vertices in that component.

Note that it might happen that during a LCM-cycle, only robots lying on the vertices of $T(t)$ are activated by the adversary (we recall we are assuming the SSync model), that is no one moves during that cycle. However, by the fairness assumption, all other robots will be activated within finite time.

We now prove that any execution of $\mathcal{A}_g$ starting from $C = C(0)$ guarantees the existence of a time $t^* > 0$ such that $C(t^*)$ is a final configuration.

According to $\mathcal{A}_g$, at starting time $t_0 = 0$, as long as there are no robots in $\bar{H}$, m allows all robots to move toward $\bar{V}$ along shortest paths. This implies that there exists a first time $t_1 \geq t_0$ when at least one robot is inside $\bar{H}$.

For each time $t \geq t_1$, subgraph $\bar{H}$ is not empty because, from time t_1 onward, $T(t) \subseteq \bar{V}$ and robots on $T(t)$ do not move.

Now, assume that at time t_1 the robots in $\bar{H}$ are on different components of $\bar{H}$. Then $T(t_1)$ is the set of vertices of the components of $\bar{H}$ with the largest number of robots. The set $T(t_1)$ is always distinct within the set of all the vertices v such that $\lambda(v) > 0$. In fact, if there are robots outside $\bar{H}$, the vertices in which they reside do not belong to $T(t_1) \subseteq \bar{V}$, and if all the robots are inside $\bar{H}$, by Lemma 8, there are at least two connected components that contain a different number of occupied vertices. In the latter case, notice that the trajectories to reach $T(t_1)$ contain vertices in $V \setminus \bar{V}$ (that is outside $\bar{H}$), being the components disconnected in $\bar{H}$. Moreover, note that the shortest paths from a component to another one could pass through intermediate components of $\bar{H}$. Then, it could be possible that the target components change after a move. From time t_1 onward, moves are repeated until a time t_2 is reached when all the robots are on a single component in $\bar{H}$.

To analyze the convergence toward a single component, let us consider for each time $t \geq t_1$ the pair $p(t) = (k - \eta_t, d(t))$, where η_t is the total number of robots on a component of $G[T(t)]$ and $d(t) = \sum_{v \notin T(t), \lambda(v) > 0} d(v, T(t))$ be the total sum of the distances of robots not in $T(t)$ to any closest vertex in $T(t)$. We show that $p(t+1) < p(t)$, where the operator '$<$' refers to the lexicographic order. In this way, we prove that there exists a time t_2 such that $p(t_2) = (0, 0)$, which is equivalent to say that all the robots are on a single component of $\bar{H}$.

For each time t such that $t_1 \leq t < t_2$, it is possible that some active robots outside $\bar{H}$ reach vertices of $\bar{H}$ at time $t+1$, not necessarily on a component of $G[T(t)]$. Then it could be possible that $\eta_{t+1} > \eta_t$ as some components can increase their number of robots and hence $k - \eta_{t+1} < k - \eta_t$, which in turn implies $p(t) < p(t+1)$. If η_t remains equal to η_{t+1} then, by the analysis above, it could be that $T(t+1) \supseteq T(t)$, but in this case $d(t+1) < d(t)$ and then $p(t+1) < p(t)$. Hence, in both cases we have $p(t+1) < p(t)$.

The moves are repeated for each $t_1 \leq t < t_2$ and then, being $k - \eta_t \geq 0$ and $d(t) \geq 0$, there will be a time t_2 such that $p(t_2) = (0, 0)$ and all the robots are on the same component H' of $\bar{H}$. According to the target, the move m, and the trajectories specified by $\mathcal{A}_g$, we are sure the robots will not leave H' for each $t \geq t_2$.

At time $t_2 \geq t_0$ either all the robots are on a single vertex, and the gathering is achieved, or, by Lemma 8, there are batches with different orders.

In the latter case, consider a time $t \geq t_2$ and let H' be the component of $\bar{H}$ where all the robots reside and let V' be its vertex set. Let us define the pair $q(t) = (k - \lambda_t, d(t))$, where λ_t is the maximum $\lambda(v)$ for $v \in V'$ at time t and $d(t)$ is defined as above. We show that $q(t+1) < q(t)$, where operator '$<$' still refers to the lexicographic order. In this way, we prove that there exists a time t^* such that $q(t^*) = (0, 0)$, which is equivalent to say that $B_k(t^*)$ is not empty and hence $C(t^*)$ is a final configuration. We analyze three different cases that may arise at time $t' > t$, during which the activated robots are not only those in $T(t)$:

- At time $t' > t$, a robot reaches a vertex in $T(t)$. In this case, $\lambda_{t'} > \lambda_t$ and hence $k - \lambda_t < k - \lambda_{t'}$, which in turn implies $q(t') < q(t)$;

- At time t', (1) no robots reach a vertex in $T(t)$, and (2) the moving robots form a new multiplicity in H' such that $\lambda_{t'} > \lambda_t$. This case may occur when the shortest paths of two or more moving robots share a common vertex in $V' \setminus T(t)$. As in the previous item, $\lambda_{t'} > \lambda_t$ implies $q(t') < q(t)$;
- At time t', (1) no robots reach a vertex in $T(t_j)$, and (2) the moving robots do not form a new multiplicity in H' such that $\lambda_{t'} > \lambda_t$. This case may occur when the moving robots must traverse many edges in H' before reaching $T(t)$. In this situation $\lambda_{t'} = \lambda_t$ holds. We now show that $d(t') < d(t)$, which in turn implies $q(t') < q(t)$. Two subcases must be analyzed:
  - at t', the greater new multiplicity formed in H' is smaller than λ_t. In this case, $T(t') = T(t)$. As all robots moved toward $T(t)$, then $d(t') < d_t$ trivially holds.
  - at t', the greater new multiplicity formed in H' is equal to λ_t. In this case, $T(t) \subset T(t')$. Notice that the robots forming such a new multiplicity do not contribute to $d(t')$, while all the others have reduced their distance to $T(t) \subset T(t')$. This implies $d(t') < d(t)$.

In any case we have $q(t') < q(t)$ and hence there will be a time instant t^* such that $q(t^*) = (0, 0)$. This guarantees that the gathering is eventually accomplished. □

Consider again configurations $C_1, \ldots, C_4$ shown in Fig. 2. We have already observed that both configurations C_1 and C_2 have an automorphism which is partitive on the entire vertex set V: according to Theorem 1 they are ungatherable. On the contrary, C_3 represents a case in which Theorem 9 applies: this configuration is gatherable by SSync robots since $k = 3$ and there exists a minimal recognizable subgraph H with $c(H) = 2$ and $s(H) = 1$ (the empty vertices form such a subgraph, cf. G_2 in Fig. 3).

Conversely, C_4 is a configuration where Theorem 9 cannot be applied since there are $k = 3$ robots and two minimal recognizable subgraphs: the first has one component with three vertices, the second has four components with one vertex each. In [14], the notion of *weak-partitive configuration* is introduced. In certain situations, it extends the definition of configurations admitting partitive automorphism recalled in Section 2.1 and the negative result stated in Theorem 1. According to this extended version of partitive configuration, we can say that C_4 is weak-partitive and hence ungatherable even by means of FSync robots.

Another example can be derived from graph G_4 in Fig. 3: if five robots are located anywhere on such a graph, a configuration gatherable by SSync robots is obtained (in fact, there is a minimal recognizable subgraph H with $c(H) = 2$ and $s(H) = 3$ and 5 is 3–prime).

3.4. Applications of Theorem 9

In this section, we show that Theorem 9 provides a powerful means for gathering purposes as long as the input graph admits some topological properties like a limited number of centers, medians, bounded degree vertices and so forth. It can also be applied in some cases when the initial configurations admit multiplicities.

Configurations on graphs with special properties. In a graph, a center is a vertex that minimizes the maximum distance (in terms of number of edges) from any other vertex. A graph may admit more than one center. A well-known result states that a tree T admits either one center or two neighboring centers [43]. Let $C = (T, \lambda)$ be any initial configuration on T, assume there are two centers in T (say c_1 and c_2), and let T_1 and T_2 be the two subtrees rooted at c_1 and c_2, respectively, when the edge connecting c_1 and c_2 is removed. In such a scenario we say that C is *balanced* when the sub-configurations (T_1, λ) and (T_2, λ) are isomorphic. It is known that C is gatherable by Async robots if and only if C is not balanced [11,17].

In the context of trees, Theorem 9 provides a simple gathering algorithm when C contains an odd number k of robots. In fact, the subgraph H containing the centers of T is recognizable, it has at most two elements and hence it easily follows that k is $|H|$–prime.

The same approach can be used for finite grids. Let H be the set containing all centers of a squared grid G: then H has one, two or four elements. Hence, in a configuration C with k robots, Theorem 9 provides a simple gathering algorithm for C when k is odd but not a multiple of three. In fact, in the case k is a multiple of three and the grid admits four centers, then k is not $|H|$–prime, i.e., Theorem 9 does not apply.

Another graph property that could be exploited concerns medians. In a graph a median is a vertex that minimizes the summation of the distances from any other vertex.

In 2-trees it is known that medians induce a complete graph of size at most three [44]. Hence, in a configuration C with k robots, Theorem 9 provides a simple gathering algorithm for C when k is odd but not a multiple of three.

Initial configurations with multiplicities. Most of the previous works concerning robots located on graphs make the assumption that the initial robot positions are unique, that is, in the initial configuration, no two robots share the same node. In Section 2 we have assumed the same model. Anyway, some authors have addressed the gathering and other problems with respect to initial configurations that admit multiplicities (see, e.g. [41]). In such cases, Theorem 9 can still be used to directly solve the problem in certain situations, regardless of the underlying graph topology. For instance, consider a configuration $C = (G, \lambda)$ composed of k robots and admitting multiplicities: it can be observed that if k is $|G|$–prime, then the algorithm

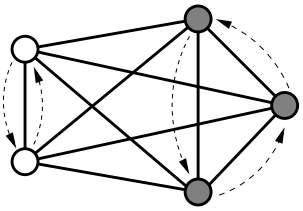

Fig. 4. A configuration defined on a complete graph. Visualization of the automorphisms σ and σ' acting on the occupied and unoccupied vertices, respectively.

given in the proof of Theorem 9 is able to solve the gathering problem for C. A special case of this condition applies when k is prime and $k > |G|$.

As we are going to see in the next sections, still some of the provided techniques can be exploited for FSync robots even in very symmetric graphs like complete and complete bipartite graphs where the topology does not provide any feature to distinguish among vertices.

4. Gathering in complete graphs

We start by defining specific automorphisms holding in any initial configuration $C = (G, \lambda)$ with $k \geq 2$ robots when $G = (V, E)$ is a complete graph. Let $V' \subseteq V$ be the set of unoccupied vertices and hence $V \setminus V'$ be the set of occupied vertices. In the remainder, we denote by σ and σ' the automorphisms defined as follows:

- σ (σ', respectively) considers the subgraph induced by vertices in $V \setminus V'$ (V', respectively) as a cycle and maps each vertex into the next vertex in that cycle.

As an example, these automorphisms are depicted in Fig. 4. Notice that σ is partitive on $V \setminus V'$ and σ' is partitive on V'. In particular, they generate one orbit coincident with $V \setminus V'$ and one orbit coincident with V', respectively.

Property 1. *Let $C = (G, \lambda)$ be an initial configuration with $k \geq 2$ robots, where $G = (V, E)$ is a complete graph, and $V' \subseteq V$ be the subset of unoccupied vertices. Let $\mathcal{A}$ be any gathering algorithm for C. The automorphism σ is partitive on $V \setminus V'$, and by using Corollary 3 we get that $\mathcal{A}$ must move robots toward vertices in V'. Hence, the following properties hold:*

- *Since the automorphism σ makes all robots equivalent, $\mathcal{A}$ cannot distinguish which robots to move. Then with* FSync *robots, all robots move synchronously;*
- *Since the automorphism σ' makes all vertices in V' equivalent, then the adversary may decide which are the real destinations within V' for the moving robots.*

4.1. Concerning the multiplicity detection

Before providing our full characterization about gathering in complete graphs by means of FSync robots endowed with global strong multiplicity detection, we show that considering weaker assumptions on the multiplicity detection gives rise to very restricted cases of solvability.

Consider any initial configuration $C = (G, \lambda)$ with $k \geq 2$ robots when $G = (V, E)$ is a complete graph with n vertices.

- If $n = k + 1$ holds, then the gathering is easily solvable by letting move all robots toward the only unoccupied vertex, regardless the multiplicity detection capability assumption.
- If $n = k + 2$, instead, we can distinguish between two cases according to the parity of k. If k is even, then any configuration is ungatherable because there exists a partitive automorphism on the entire vertex set and hence Theorem 1 applies. If k is odd, then the gathering can be finalized only if robots can detect the parity of a multiplicity (that is with the local- or global-strong multiplicity detection). By Property 1, after the first move toward unoccupied vertices, either the gathering is accomplished or one vertex is occupied by an even number of robots and the other one by an odd number of robots. As subsequent move then, for instance, we can move all robots from the multiplicity composed by an odd number of robots toward the other occupied vertex, hence finalizing the gathering. Clearly, the problem is unsolvable if the local- or global-weak multiplicity detection is assumed, as then the two multiplicities would look the same. The only exception is provided when $k = 3$, in which case there will be one single robot and a multiplicity (of two robots). Hence even with the local-weak multiplicity detection we may allow the single robot to move toward the other occupied vertex to finalize the gathering.
- If $n > k + 2$, it can be shown that the gathering is always unsolvable if the global strong multiplicity detection is not assumed. When k is even, the adversary may force to have exactly two multiplicities of the same size, and from

there it is impossible to finalize the gathering. For k odd, of course the configuration is ungatherable if $k = 3$ (the adversary always selects k distinct destinations). For $k \geq 5$, by moving robots toward unoccupied vertices as dictated by Property 1 may produce two multiplicities, and then again the configuration is ungatherable with the local- or global-weak multiplicity detection. If the local-strong multiplicity detection is assumed, we now show there are two possible scenarios that may occur where robots would not be able to accomplish the gathering. After the first move we focus on two possible configurations that can be obtained: (a) a configuration composed of a single robot and two multiplicities, each composed of $\frac{k-1}{2}$ robots; (b) a configuration made of two vertices occupied by single robots and a third vertex occupied by the remaining $k-2$ robots.
From (a), to finalize the gathering, the algorithm may move the single robot toward one of the other occupied vertices, and then, as above, the gathering can be finalized by moving the multiplicity with an odd number of robots toward the other occupied vertex. Instead, by letting move from (a) the robots composing the multiplicities (or even all robots) may lead to a configuration isomorphic to the considered one as robots cannot discriminate the destination. From (b), consistently with the move that the algorithm has to implement for case (a), the single robots should move. Again, since robots cannot discriminate among destinations, if the robots in the multiplicity are also moved, then the adversary may rotate indefinitely the occupancies of the three occupied vertices. If robots in the multiplicity are instead not allowed to move, then the two single robots may exchange their positions forever.

By the above discussion, from now on we consider FSYNC robots endowed with global-strong multiplicity detection as not much can be done otherwise.

4.2. The characterization result

The next theorem shows a full characterization of the gathering problem within the considered setting.

Theorem 10. *Let G be a complete graph with n vertices, and let $C = (G, \lambda)$ be an initial configuration with $k \geq 2$ FSYNC robots. C is gatherable if and only if k is $(n-k)$–prime.*

Proof. ($\Leftarrow$) We show there exists an algorithm $\mathcal{A}_{clique}$ able to gather all robots in C. This algorithm is defined by the following two moves:

- *Move m_1. It is applied when the configuration is initial (i.e., no multiplicity occurs): each robot moves toward an arbitrary unoccupied vertex;*
- *Move m_2. It is applied when there are multiplicities. Let $\tilde{B}$ be the batch of minimum size, of greater order in case of tie. Each robot not in $\tilde{B}$ moves toward an arbitrary vertex in $\tilde{B}$.*

We now prove that for any execution $\mathbb{E} : C = C(0), C(1), C(2), \ldots$ of $\mathcal{A}_{clique}$ there exists a finite time instant $t > 0$ such that $C(t)$ is final and $C(t') = C(t)$ for each $t' > t$.

In $C(0)$ each robot performs move m_1. Then, there are two possible cases for $C(1)$: either it is final (and the proof is concluded), or $C(1)$ contains batches with different orders (since k is $(n-k)$–prime). In the latter case, since there are multiplicities in $C(1)$, move m_2 is applied. Let $\tilde{B}(i)$ be the batch of minimum size, of greater order in case of tie, generated at the i-th time unit. As specified by move m_2, each robot not in $\tilde{B}(1)$ moves toward an arbitrary vertex in $\tilde{B}(1)$. Now, concerning the obtained configuration $C(2)$, again two cases may occur: either it is final, or the following analysis can be performed.

We have already observed that there are batches with different order in $C(1)$, and m_2 moved *all* robots not in $\tilde{B}(1)$ toward vertices in $\tilde{B}(1)$. It follows that in $C(2)$ all the k robots are in $\tilde{B}(1)$ and hence $\tilde{B}(2) \subseteq \tilde{B}(1)$. Moreover, by reminding that k is $(n-k)$–prime and that $|\tilde{B}(1)| < n-k$, then k is $|\tilde{B}(1)|$–prime as well. Hence, also in $C(2)$ there are batches with different orders. This implies the following property:

- $\tilde{B}(2) \subset \tilde{B}(1)$.

Hence, if $C(2)$ is not final again move m_2 can be applied. Since each time m_2 is applied the size of $\tilde{B}$ decreases, then it is clear that in a finite number of time units the size of $\tilde{B}$ will be 1. From there, either the configuration is final or it becomes as such in the subsequent time unit.

($\Rightarrow$) Assume k is not $(n-k)$–prime. If $n-k=0$ (the only possible value of $n-k$ for which the definition of d-primality does not apply) then the automorphism σ is partitive on the entire vertex set and hence, by Theorem 1, C is ungatherable. The relationship $n-k=1$ cannot hold otherwise k is $(n-k)$–prime, against hypothesis. Hence, in the remainder we assume $n-k \geq 2$.

According to Corollary 3, we can assume that the move planned by $\mathcal{A}$ is toward an arbitrary unoccupied vertex. Since k is not $(n-k)$–prime by hypothesis, then we get $n-k \geq lpf(k)$, that is there exist at least $lpf(k) \geq 2$ unoccupied vertices in $C(0)$. Hence, the adversary may select $lpf(k)$ unoccupied vertices as targets to create a configuration $C(1)$ consisting of $lpf(k)$ occupied vertices with $\frac{k}{lpf(k)}$ robots per vertex.

In $C(1)$ it is still possible to apply automorphism σ, which implies that Property 1 also holds in this configuration. We now show that from $C(1)$ any possible move creates a configuration $C(2)$ isomorphic to $C(1)$, thus preventing the possibility of gathering all robots.

Since in $C(1)$ all the moving robots are moved toward the unoccupied vertex (cf. Corollary 3), then the adversary selects different destinations per multiplicity; notice that this is possible because, being $n-k \geq lpf(k)$ in $C(0)$, then $n-lpf(k) \geq k \geq lpf(k)$ and hence in $C(1)$ the number of unoccupied vertices (i.e., $n-lpf(k)$) is greater than or equal to the number of occupied vertices (i.e., $lpf(k)$). It follows that in any case the configuration is ungatherable. □

The following theorem provides an upper bound to the complexity of the gathering problem on complete graphs.

Theorem 11. *In any gatherable configuration defined on complete graphs composed of k* FSync *robots, the gathering problem can be solved in $O(\log k)$ time.*

Proof. Let $C=(G,\lambda)$ be any gatherable initial configuration with k robots and defined on a complete graph G with n vertices. Being C gatherable, Theorem 10 implies that k is $(n-k)$–prime.

Consider the algorithm $\mathcal{A}_{clique}$ proposed in the proof of Theorem 10 to process C. In any possible execution, $\mathcal{A}_{clique}$ first applies move m_1 to bring all robots on the initially $n-k$ unoccupied vertices, and then it applies a sequence M composed of moves m_2 only. It follows that the complexity of $\mathcal{A}_{clique}$ depends on the length of the sequence M of consecutive moves m_2 it applies. In the following we provide an upper bound to such a length.

We recall that move m_2 leads each robot not in $\tilde{B}$ toward an arbitrary vertex in $\tilde{B}$. From the proof of Theorem 10 we know that each time m_2 is applied the size of $\tilde{B}$ is reduced. In order to maximize the length of M, at any time t the adversary will arrange all the moving robots among all the possible destinations so that $\tilde{B}(t)$ is as large as possible.

It can be observed that, at the time t, the greater the number of batches formed by the move m_2 performed at time $t-1$, the smaller the size of $\tilde{B}(t)$. According to this observation, the best disposal strategy for the adversary would be to create just one single batch, but this is not possible since k is $(n-k)$–prime. Hence, at each move, the adversary disposes all the moving robots into two distinct batches. Moreover, always to maximize the length of M, the sizes of these two batches must be comparable. This can be obtained when $|\tilde{B}|$ is roughly halved each time, thus producing an execution requiring at most $\log(n-k)$ time units. Since k is $(n-k)$–prime, then $k \geq lpf(k) > n-k$ and hence the final complexity is $O(\log k)$. □

5. Gathering in complete bipartite graphs

In this section, we study the gathering problem of FSync robots endowed with global strong multiplicity detection on complete bipartite graphs. Throughout the section, we use the following notation: if $G=(V_1 \cup V_2, E)$ is a complete bipartite graph and $C=(G,\lambda)$ is any initial configuration on G, then n_1 and n_2 denote the number of vertices of V_1 and V_2, respectively; k_1 and k_2 denote the number of robots on V_1 and V_2, respectively; if $k_1>0$ and $k_2>0$, then $lcpf(k_1,k_2)$ denotes the *least common prime factor* of k_1 and k_2; and we say that partition V_i, $i=1,2$, is *unoccupied* (*occupied*, resp.) if $k_i=0$ ($k_i>0$, resp.).

In the following we extend to bipartite graphs some concepts introduced in Section 4. Let $G=(V_1 \cup V_2, E)$ be a complete bipartite graph, $C=(G,\lambda)$ be a configuration with k_1+k_2 robots, $\mathcal{A}$ be a gathering algorithm for C, and $\mathbb{E}: C=C(0), C(1), C(2), \ldots$ be an execution of $\mathcal{A}$ that starts from C. Then:

- $B_i^1(t)$ and $B_i^2(t)$ denote the *projections* of a batch $B_i(t)$ onto V_1 and V_2, respectively; formally, $B_i^1(t)=B_i(t)\cap V_1$ and $B_i^2(t)=B_i(t)\cap V_2$;
- the projections $B_i^1(t)$ and $B_i^2(t)$ induced by any batch $B_i(t)$ are simply called *p-batches*;
- $B_{\min}^1(t)$ and $B_{\min}^2(t)$ are the p-batches induced by $B_{\min}(t)$;
- $B_{\max}^1(t)$ and $B_{\max}^2(t)$ are the p-batches induced by $B_{\max}(t)$;
- $\tilde{B}^1(t)$ and $\tilde{B}^2(t)$ are the p-batches of minimum size, and of greater order in case of tie, in the partitions V_1 and V_2, respectively;
- $\Lambda^1(t)$ and $\Lambda^2(t)$ denote the number of *occupied vertices* at time t in the partitions V_1 and V_2, respectively.

The next lemma provides a partial characterization of the gathering on complete bipartite graphs. As we will show in the proof, the gathering is realized by requiring that in any time unit, the robots swap the partition in which they reside.

It is worth to remark that such a technique is mandatory. In fact, it can be shown that without the simultaneous exchange of the position of the robots between the two partitions of the graph, it is not possible to perform the gathering. An explicative example to this respect is given by the configuration C with $n_1=n_2=8$, $k_1=5$ and $k_2=3$. Let $\mathcal{A}$ be any gathering algorithm that does not allow the swapping operation. Clearly, by moving the robots of one side toward the unoccupied vertices of the other side may generate a partitive configuration with 8 robots occupying the 8 vertices on one side and 8 unoccupied vertices on the other side. Hence, we can consider $\mathcal{A}$ to move the robots from one side toward the occupied vertices of the other side. It is possible to perform an exhaustive analysis showing that in each case the adversary is able to force $\mathcal{A}$ to produce an ungatherable configuration. In fact, if the k_1 robots are moved on the k_2 occupied vertices,

the adversary produces a configuration C' with one batch B_4^2 of size 1 and one batch B_2^2 of size 2. If B_4^2 is moved on V_1 a symmetric configuration is created with p-batches B_2^1 and B_2^2 both of size 2. On the contrary, if B_2^2 is moved on V_1, a symmetric configuration is created with p-batches B_4^1 and B_4^2 of size 1. By Theorem 1, both these configurations are ungatherable. Moves on unoccupied vertices does not help as, in this case, the adversary maintains the same batches, even if on different partitions.

If, as first move, the $k_2 = 3$ robots are moved on the $k_1 = 5$ occupied vertices, the adversary produces a batch B_2^1 of size 3 and a batch B_1^1 of size 2, and, with the next move, these two batches are on separate partitions. Without loss of generality assume that the two batches are B_2^2 of size 3 and a batch B_1^1 of size 2. If robots on B_2^2 are moved on B_1^2, then a batch B_4^1 of size 2 is created and the resulting configuration is ungatherable by Theorem 1. If robots on B_1^1 are moved on B_2^2, then again the ungatherable configuration C' with one batch B_4^2 of size 1 and one batch B_2^2 of size 2 is created.

On the other hand, if moves that swap the robots on the two initial batches are admitted, the configuration C can be gathered. In fact, if the k_1 robots are moved on the three occupied vertices of V_2 (when the robots on these vertices are moved on V_1), the adversary can move all the k_1 robots on a single vertex, and then the gathering is accomplished on the next move, or on more vertices. In the latter case a batch B of size 1 and order strictly less than 4 is always created: the gathering can be then easily accomplished by moving all the robots not in B on V_1 and then all the robots in V_1 on B.

Lemma 12. *Let $G = (V_1 \cup V_2, E)$ be a complete bipartite graph with $n_1 + n_2$ vertices, and let $C = (G, \lambda)$ be an initial configuration composed of $k_1 + k_2$* FSYNC *robots, with $k_1 > 0$ and $k_2 > 0$. Then, C is gatherable if one of the following conditions holds:*

1. *k_1 and k_2 are coprime;*
2. *k_1 and k_2 are not coprime, $0 < n_1 - k_1 < lcpf(k_1, k_2)$ or $0 < n_2 - k_2 < lcpf(k_1, k_2)$, $k_1 \neq k_2$ or $n_1 \neq n_2$.*

Proof. We show there exists an algorithm $\mathcal{A}_p$ (algorithm for *partial* cases) able to gather all robots in C when one of the conditions in the statement holds. This algorithm is defined by the following moves:

- *Move m_1. It is applied when $\Lambda^1(t) = 1$ and $\Lambda^2(t) = 1$. If $k_1 \neq k_2$ then each robot in $B_{\min}(t)$ moves toward the unique vertex in $B_{\max}(t)$ else each robot in the larger partition between V_1 and V_2 moves toward the unique occupied vertex in the other partition;*
- *Move m_{2a}. It is applied when $\Lambda^1(t) = 1$ and $\Lambda^2(t) > 1$: each robot moves toward the unique vertex in $\tilde{B}^1(t)$.*
- *Move m_{2b}. It is applied when $\Lambda^1(t) > 1$ and $\Lambda^2(t) = 1$: each robot moves toward the unique vertex in $\tilde{B}^2(t)$.*
- *Move m_3. It is applied when either C is initial and k_1 and k_2 are coprime or C is not initial and $\Lambda^1(t) > 1$, $\Lambda^2(t) > 1$: each robot in V_1 moves toward $\tilde{B}^2(t)$ and each robot in V_2 moves toward $\tilde{B}^1(t)$.*
- *Move m_4. It is applied when C is initial and the integers k_1 and k_2 are not coprime. If $n_1 - k_1 = 0$ then set T_1 corresponds to V_1, else T_1 contains all the unoccupied vertices in V_1. Symmetrically, if $n_2 - k_2 = 0$ then set T_2 corresponds to V_2, else T_2 contains all the unoccupied vertices in V_2. Then, each robot in V_1 moves toward an arbitrary vertex in T_2 and each robot in V_2 moves toward an arbitrary vertex in T_1.*

Let $\mathbb{E} : C = C(0), C(1), \ldots$ be any possible execution generated by $\mathcal{A}_p$. In the first part of this proof we assume that the initial configuration C fulfills the first condition of the statement, i.e. k_1 and k_2 are coprime. According to this assumption, $\mathcal{A}_p$ applies move m_3 to $C(0)$. Without loss of generality, we also assume that $k_1 < k_2$.

By applying m_3 to $C(0)$, it is obtained a configuration $C(1)$ where all robots in V_1 moved toward the k_2 occupied vertices in V_2 and, symmetrically, all robots in V_2 moved toward the k_1 occupied vertices in V_1. Notice that as long as the condition that generates the move m_3 is verified, robots will continue to swap the partition they reside. To analyze what happens to the configurations generated by the continuous application of m_3, we define the following predicates:

- $\mathcal{Q}'(t)$: $|\tilde{B}^1(t)| < |\tilde{B}^1(t-2)|$;
- $\mathcal{Q}''(t)$: $\Lambda^1(t) = 1$;
- $\mathcal{Q}'''(t)$: $C(t)$ is final.

We prove now there exists a finite time $t' > 0$ such that by applying algorithms $\mathcal{A}_p$ to $C(0)$, then $\mathcal{Q}'(t)$ holds for each time $2 \leq t \leq t'$, $\mathcal{Q}''$ holds at time t', and $\mathcal{Q}'''(t)$ holds for each time $t > t'$ (and hence $\mathcal{A}_p$ results to be a gathering algorithm). Notice that the defined predicates concern only partition V_1, but according to the symmetry of all the moves defining $\mathcal{A}_p$, they hold for V_2 as well.

Notice that $Q''(t)$ may hold whenever the adversary allows the robots to be moved toward the same destination - if this is not the case, there will be a sequence of applications of move m_3. So, assume the algorithm $\mathcal{A}_p$ uses a consecutive sequence of $t \geq 2$ moves m_3 and, by contradiction, assume that $\mathcal{Q}'(t)$ does not hold. This means that $|\tilde{B}^1(t)| = |\tilde{B}^1(t-2)|$ holds. In particular, it implies that:

- in $C(t-1)$, all the k_2 (or k_1) robots moved on V_1 have been equally distributed over the $|\tilde{B}^1(t-2)|$ possible destinations – hence $\tilde{B}^1(t-1) = \tilde{B}^1(t-2)$;

- in $C(t)$, all the k_1 (or k_2) robots moved on V_1 have been equally distributed over the $|\tilde{B}^1(t-1)|$ possible destinations – hence $\tilde{B}^1(t) = \tilde{B}^1(t-1)$.

It follows that $|\tilde{B}^1(t)| = |\tilde{B}^1(t-1)| = |\tilde{B}^1(t-2)|$ results to be a common divisor for both k_1 and k_2, against the hypothesis that k_1 and k_2 are coprime. This implies that $\mathcal{Q}'$ holds as long as move m_3 is applied, and in turn that the size of $\tilde{B}^1$ is reduced every two consecutive LCM-cycles. As a consequence, there exists the requested finite time $t' > 0$ such that $\mathcal{Q}''(t')$ holds.

When $\mathcal{Q}''(t')$ holds, in the obtained configuration $C(t')$ at least one partition between V_1 and V_2 contains just one occupied vertex. Then, at $C(t')$ one among moves m_1, m_{2a}, or m_{2b} will be performed. In any case it is easy to observe that $\mathcal{Q}'''(t'+1)$ holds, that is $C(t'+1)$ is a final configuration. This proves that $\mathcal{A}_p$ is able to gather each configuration C where the first condition of the statement holds.

In the second part of the prove we assume that the second condition of the statement holds. According to this assumption, from the hypothesis $0 < n_1 - k_1 < lcpf(k_1, k_2)$ or $0 < n_2 - k_2 < lcpf(k_1, k_2)$ we get that at least one set between V_1 and V_2 has unoccupied vertices. Without loss of generality, let us assume that $0 < n_1 - k_1 < lcpf(k_1, k_2)$ holds; in particular, this implies there are unoccupied vertices in V_1.

When $\mathcal{A}_p$ is applied to $C(0)$, since we assumed that the second condition of the statement holds, move m_4 is used. According to such a move, all robots in V_2 move toward the $n_1 - k_1 > 0$ unoccupied vertices in V_1. Concerning robots in V_1, they move toward V_2, in particular toward the unoccupied vertices if $n_2 - k_2 > 0$ otherwise toward the k_2 occupied vertices.

Then, at most $n_1 - k_1$ vertices become occupied in V_1 after the first move. Since k_1 and k_2 are not coprime and $0 < n_1 - k_1 < lcpf(k_1, k_2)$ by hypothesis, then $n_1 - k_1 < k_1$ and $n_1 - k_1 < k_2$. Thus, moving k_2 robots over $n_1 - k_1$ vertices leads to form multiplicities in V_1 – i.e., $C(1)$ is not initial. Notice that it is possible there is only one p-batch in V_1 after the move, and this happens when $n_1 - k_1$ is larger than or equal to a divisor of k_2 (in this case the adversary may equally distribute all the moving robots among all the $n_1 - k_1$ possible destinations).

If the adversary does not allow that only a single vertex is occupied in V_1 or in V_2 after the first move, then the algorithm $\mathcal{A}_p$ applies move m_3 to $C(1)$. The analysis of the configurations generated after $C(1)$ proceeds as in the first part of the proof, and hence still uses predicates $\mathcal{Q}'$, $\mathcal{Q}''$, and $\mathcal{Q}'''$.

Assume the algorithm, after generating $C(1)$ via m_4, uses a consecutive sequence of $t \geq 2$ moves m_3. By contradiction, let us assume that $\mathcal{Q}'(t)$ does not hold. This means that $|\tilde{B}^1(t)| = |\tilde{B}^1(t-2)|$ holds, and in particular that the following properties hold:

- $|\tilde{B}^1(t-2)| \leq n_1 - k_1$;
- in $C(t-1)$, all the k_2 (or k_1) robots moved on V_1 have been equally distributed over the $|\tilde{B}^1(t-2)|$ possible destinations – hence $\tilde{B}^1(t-1) = \tilde{B}^1(t-2)$;
- in $C(t)$, all the k_1 (or k_2) robots moved on V_1 have been equally distributed over the $|\tilde{B}^1(t-1)|$ possible destinations – hence $\tilde{B}^1(t) = \tilde{B}^1(t-1)$.

It follows that $|\tilde{B}^1(t)| = |\tilde{B}^1(t-1)| = |\tilde{B}^1(t-2)|$ results to be a common divisor for both k_1 and k_2, against the hypothesis that $|\tilde{B}^1(t-1)| \leq |\tilde{B}^1(t-2)| \leq n_1 - k_1 < lcpf(k_1, k_2)$. This implies that $Q'(t)$ holds. Hence $\mathcal{Q}'$ holds as long as move m_3 is applied, and in turns that the size of $\tilde{B}^1$ is reduced every two consecutive LCM-cycles. As a consequence, there exists the requested finite time $t' > 0$ such that $\mathcal{Q}''(t')$ holds.

When $\mathcal{Q}''(t')$ holds, in the obtained configuration $C(t')$ at least one partition between V_1 and V_2 contains just one occupied vertex. As described in the first part of this proof, one move among m_1, m_{2a}, or m_{2b} is performed on $C(t')$. Note that, due to condition (2) – i.e., $k_1 \neq k_2$ or $n_1 \neq n_2$ – move m_1 is well defined. This leads to accomplish the gathering just in one LCM cycle. It follows that $\mathcal{A}_p$ is able to gather each configuration C even when the second condition of the statement holds. □

The following theorem makes use of the previous lemma for providing a full characterization of the gathering problem on complete bipartite graphs by FSYNC robots.

Theorem 13. *Let $G = (V_1 \cup V_2, E)$ be a complete bipartite graph with $n_1 + n_2$ vertices, and let $C = (G, \lambda)$ be an initial configuration composed of $k_1 + k_2$ FSYNC robots. C is gatherable if and only if one of the following conditions hold:*

1. *$k_2 = 0$, k_1 is n_2–prime;*
2. *$k_1 = 0$, k_2 is n_1–prime;*
3. *$k_1 > 0$, $k_2 > 0$, k_1 and k_2 are coprime;*
4. *$k_1 > 0$, $k_2 > 0$, k_1 and k_2 are not coprime, $0 < n_1 - k_1 < lcpf(k_1, k_2)$ or $0 < n_2 - k_2 < lcpf(k_1, k_2)$, $k_1 \neq k_2$ or $n_1 \neq n_2$.*

Proof. (⇐) We show that Algorithm $\mathcal{A}_{bip}$ described if Fig. 5 is able to gather all robots in C when C fulfills one of the conditions expressed in the statement (notice that the conditions are mutually exclusive).

Algorithm: $\mathcal{A}_{bip}$
Input: Configuration $C = (G, \lambda)$ composed of $k_1 + k_2$ FSYNC robots and defined on a complete bipartite graph $G = (V_1 \cup V_2, E)$ with $n_1 + n_2$ vertices. It fulfills one condition in the statement of Theorem 13.

1 Let $k = k_1 + k_2$;
2 **if** ($n_1 > n_2$ *and* k *is* n_2*–prime*) *or* ($n_2 > n_1$ *and* k *is* n_1*–prime*) **then**
3 call $\mathcal{A}_g$ // it handles configurations fulfilling Conditions 1 or 2 of Theorem 13
4 **else**
5 call $\mathcal{A}_p$ // it handles configurations fulfilling Conditions 3 or 4 of Theorem 13

Fig. 5. Algorithm $\mathcal{A}_{bip}$ for gathering FSYNC robots in a complete bipartite graph G. It uses algorithms $\mathcal{A}_g$ (from proof of Theorem 9) and $\mathcal{A}_p$ (from proof of Lemma 12).

Assume that Condition 1 holds. In such a case it is interesting to observe that $n_1 \neq n_2$ holds. In fact, since k_1 is n_2–prime then $n_2 < lpf(k_1) \leq k_1 \leq n_1$. In particular, k_1 is n_2–prime implies $n_1 > n_2$, and hence $\mathcal{A}_{bip}$ calls $\mathcal{A}_g$ at Line 3 for moving robots in $C(0)$. Notice that each time $\mathcal{A}_{bip}$ restarts for handling configurations $C(1), C(2), \ldots$, always $\mathcal{A}_g$ is executed. Since $n_1 > n_2$, then V_2 induces a minimal recognizable subgraph of G and k_1 is n_2–prime by hypothesis. Hence, by Theorem 9 the gathering is eventually accomplished on a vertex of V_2. The case in which Condition 2 holds is symmetric to the previous one.

The remaining cases are straightforward. In fact, if one between Conditions 3 and 4 holds, then $\mathcal{A}_{bip}$ calls $\mathcal{A}_p$ at Line 5. According to Lemma 12, $\mathcal{A}_p$ is able to gather the given input configuration.

($\Rightarrow$) Assume that none of the conditions expressed in the statement applies, then we show that C is ungatherable.

If one between V_1 or V_2 is unoccupied, then neither k_1 is n_2–prime nor k_2 is n_1–prime (otherwise at least one among Conditions 1 and 2 holds). Let us analyze the case in which V_2 is unoccupied and consequently k_1 is not n_2–prime. The case in which V_1 is unoccupied is then symmetric. In this case, the proof is similar to that of Theorem 10. In fact, since G is a complete bipartite graph, there exists an automorphism in $\mathrm{Aut}(C)$ that makes all robots pairwise equivalent. In other words, if $\mathcal{A}$ is any gathering algorithm for C, then any move planned by $\mathcal{A}$ is performed by each robot. In particular, each move applied at $C = C(0)$ must move each robot in V_1 toward an arbitrary vertex in V_2. Since k_1 is not n_2–prime, then we get $n_2 \geq lpf(k_1)$, that is there exist at least $lpf(k_1) \geq 2$ unoccupied vertices in the partition V_2 in $C(0)$. Hence, the adversary may select $lpf(k_1)$ unoccupied vertices in V_2 as targets to create a configuration $C(1)$ consisting of $lpf(k_1)$ occupied vertices with $\frac{k_1}{lpf(k_1)}$ robots per vertex. It is easy to observe that in $C(1)$ there exists an automorphism that makes the $lpf(k)$ multiplicities pairwise equivalent. Then, from $C(1)$ the adversary allows only to generate $C(0)$ again, thus preventing the possibility of gathering all robots.

If both V_1 and V_2 are occupied, then k_1 and k_2 are not coprime otherwise Condition 3 holds. Assume now that Condition 4 is false. This implies different sub-cases to be analyzed.

The first sub-case is given when $k_1 > 0$, $k_2 > 0$, k_1 and k_2 are not coprime, and the following properties hold: $k_1 = k_2$ and $n_1 = n_2$. This means that C admits an automorphism which is partitive on the entire vertex set, and hence C results to be ungatherable according to Theorem 1.

The second sub-case is given when $k_1 > 0$, $k_2 > 0$, k_1 and k_2 are not coprime, and one of the following specific cases occur (where we denote $d = lcpf(k_1, k_2) \geq 2$ for the sake of simplicity):

(a) $n_1 - k_1 \geq d$ and $n_2 - k_2 \geq d$;
(b) $n_1 - k_1 \geq d$ and $n_2 - k_2 = 0$;
(c) $n_1 - k_1 = 0$ and $n_2 - k_2 \geq d$;
(d) $n_1 - k_1 = 0$ and $n_2 - k_2 = 0$.

Assume that (a) holds in $C(0)$. In this configuration, there exists only the p-batch $B_1^1(0)$ in V_1 and, symmetrically, only the p-batch $B_1^2(0)$ in V_2. Since k_1 and k_2 are not coprime, the size of each of such p-batches is a multiple of $d \geq 2$. Let $\mathcal{A}$ be any possible gathering algorithm for $C(0)$ and let $\mathbb{E} : C(0), C(1), \ldots$ be any possible execution performed by $\mathcal{A}$. We now prove by induction on t that in each configuration $C(t)$ both the following properties hold:

- there are at most two p-batches in $C(t)$ and the size of each p-batch is a multiple of d (notice this implies that $C(t)$ is not final since in each final configuration there exists only one p-batch with size one);
- either both V_1 and V_2 contain at least d unoccupied vertices, or one set between V_1 and V_2 is unoccupied.

It can be observed that such properties hold in $C(0)$ by hypothesis. By induction, let us assume that they are also valid in any configuration $C(t)$, $t \geq 0$. We prove that they are still valid in $C(t+1)$.

Assume that in $C(t)$ robots are moved from V_1 to V_2. Since all robots in a p-batch are pairwise equivalent, when the algorithm selects some robots in V_1 to be moved into V_2, it must select all robots belonging to one or more p-batches. Moreover, since the size of each p-batch in $C(t)$ is a multiple of d, always a multiple of d robots is moved. The strategy of the adversary is based on selecting d distinct destinations and arranging all the moved robots so that they

are equally distributed on the d destinations. Concerning the destinations, notice that the adversary may always determine them, regardless whether the algorithm selects occupied or unoccupied destinations:

- *The algorithm selects occupied destinations*. Since there are at most two p-batches in $C(t)$, this implies that there is one p-batch B' in V_1 and one p-batch B'' in V_2. By hypothesis, both B' and B'' contain a multiple of d vertices. In particular, let $|B''| = \ell \cdot d$, $\ell \geq 1$. According to the described strategy of the adversary, the moved robots are equally distributed over d vertices within B''.
  If $\ell = 1$, after the move we have that B'' maintains the same size but increases its order, there is only the p-batch B'' in $C(t+1)$, and V_1 is unoccupied. On the contrary, if $\ell > 1$, after the move we have that B'' is divided into two distinct p-batches with sizes d and $(\ell - 1) \times d$, respectively, $C(t+1)$ has exactly the two p-batches obtained by splitting B'', and V_1 is unoccupied.
  This proves that the inductive hypothesis holds also in $C(t+1)$ when the algorithm moves robots from V_1 toward occupied vertices in V_2.
- *The algorithm selects unoccupied destinations*. In this case, the adversary chooses d distinct vertices in V_2. This is possible according to the inductive hypothesis. Moreover, according to the described strategy of the adversary, the moved robots are equally distributed over d unoccupied vertices.
  If V_2 was occupied in $C(t)$, then $C(t+1)$ contains at most two p-batches (each with size multiple of d), and V_1 is unoccupied. On the contrary, if V_2 was unoccupied in $C(t)$ then one p-batch moved from V_1 to V_2. This leads to a configuration $C(t+1)$ with at most two p-batches (in case of two, one p-batch per side), both with size multiple of d. Concerning the unoccupied vertices, V_1 increased by at least d the number of unoccupied vertices (resulting from the vertices left from the moved robots), while V_2 has at least $n_2 - k_2 \geq d$ unoccupied vertices that correspond to the unoccupied vertices in the initial configuration $C(0)$.
  This proves that the inductive hypothesis holds also in $C(t+1)$ when the algorithm moves robots from V_1 toward unoccupied vertices in V_2.

By symmetry, the above analysis still holds when robots are moved from V_2 to V_1. Moreover, the above analysis holds even when the algorithm $\mathcal{A}$ swaps robots between V_1 and V_2 instead of simply moving robots from one side to the other at a time.

Assume now that one among cases (b), (c), or (d) holds. It can be observed that in such a case robots can perform only a subset of movements allowed in case (a). Since we have shown that in (a) each possible movement leads to an ungatherable configuration, the same holds for the other cases. □

The next corollary concerns the complexity of the algorithm described in the previous theorem for solving the gathering problem on complete bipartite graphs by means of FSync robots. In such a statement, we use the *divisor function* $\delta(n)$, that is the function that returns the number of divisors of any integer n, including 1 and the number n itself.

Corollary 14. *In any gatherable configuration defined on complete bipartite graphs and composed of k* FSync *robots, the gathering problem can be solved in $O(\log(k) + \delta(k))$ time.*

Proof. Let $C = (G, \lambda)$ be a gatherable initial configuration with $k = k_1 + k_2$ robots and defined on a complete bipartite graph G. Consider the algorithm $\mathcal{A}_{bip}$ proposed in the proof of Theorem 13 which calls exclusively one between algorithms $\mathcal{A}_g$ and $\mathcal{A}_p$ to process C.

If C is processed by $\mathcal{A}_g$, then a proof similar to that of Theorem 11 applies. The only difference now is that robots require two moves to 'emulate' what they do with one move in complete graphs, i.e., decreasing the size of $\tilde{B}$. Hence, the time required to accomplish the gathering is at most doubled with respect to that achieved in complete graphs. Hence, when C is processed by $\mathcal{A}_g$, the final time complexity is $O(\log k)$.

If C is processed by $\mathcal{A}_p$, then we get from the proof of Lemma 12 that such algorithm distinguishes between two cases, according whether k_1 and k_2 are coprime or not. In the first case it applies an initial sequence of moves in which only m_3 is used. After that, just applying once a move among m_1, m_{2a}, or m_{2b} leads to a final configuration. In the latter case the same sequence of moves is used except for the initial application of a single move m_4. Hence, the complexity of $\mathcal{A}_p$ depends on the length of the sequence M of consecutive moves m_3 it applies. In the following we provide an upper bound of such a length.

We know that, at each step, m_3 moves all robots from V_1 to $\tilde{B}^2$ and symmetrically all robots from V_2 to $\tilde{B}^1$. From the proof of Lemma 12 we also know that each pair of consecutive applications of move m_3 reduces the size of both $\tilde{B}^1$ and $\tilde{B}^2$. Without loss of generality, let us assume that $k_1 < k_2$ and the gathering is finalized on a vertex in V_1. Then, in order to maximize the length of M, at any time the adversary will arrange all the moving robots among all the possible destinations so that $\tilde{B}^1$ is as large as possible. It can be observed that, at any time t, the greater the number of batches formed by the move m_2 performed at time $t-1$, the smaller the size of $\tilde{B}(t)$. According to this observation, and by considering that it is not always possible to form only one batch, the adversary may select at each time t the *best disposal strategy* between the following two. By denoting as r (with $r = k_1$ or $r = k_2$) the number of robots to be moved toward $\tilde{B}^1(t-1)$, then:

- *Disposal strategy 1:* if $|\tilde{B}^1(t-1)| \geq lpf(r)$, then the r robots are arranged on the vertices of $\tilde{B}^1(t-1)$ so as to form a single batch with size p and order r/p, being p the largest divisor of r smaller or equal to $|\tilde{B}^1(t-1)|$;
- *Disposal strategy 2:* if $|\tilde{B}^1(t-1)| < lpf(r)$ or r is prime, then the r robots are arranged on the vertices of $\tilde{B}^1(t-1)$ so as to form two batches such that $|\tilde{B}^1(t)|$ is as large as possible.

According to these strategies, if the adversary applies always the first case, then the length of M is bounded by $O(\delta(k))$. If it applies always the second case, then the length of M is bounded by $O(\log k)$. If the two strategies are combined, the length of M is bounded by $O(\log(k) + \delta(k))$.

Summarizing, the complexity of the algorithm $\mathcal{A}_{bip}$ is $O(\log(k) + \delta(k))$. □

6. Lower bound

In this section, we provide a lower bound for the gathering problem that holds in both complete and complete bipartite graphs. This lower bound requires to quickly recall the Fibonacci sequence. The i-th element of such a sequence is denoted by F_i and is given by the sum of the two preceding ones. Formally:

$$F_0 = 0,$$
$$F_1 = 1,$$
$$\ldots$$
$$F_i = F_{i-1} + F_{i-2}.$$

The first elements of this series are 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, ... Notice that there exists a useful relationship between the Fibonacci number F_i and the corresponding index i. Consider the well known equality

$$F_i = \frac{\phi^i - \left(\frac{-1}{\phi}\right)^i}{\sqrt{5}},$$

where $\phi = \frac{\sqrt{5}+1}{2}$ is the golden ratio (see, e.g., [35]). By noting that $\frac{1}{\phi}$ is less than 1, then $F_i < \frac{\phi^i+1}{\sqrt{5}}$, which implies the following inequality:

$$i > \log_\phi \left(\sqrt{5} \cdot F_i - 1\right). \tag{1}$$

Lemma 15. *Let F_l be any Fibonacci number and let $1 \leq j \leq l$. F_l can be expressed as a linear combination of F_j and F_{j-1}, that is $F_l = a \cdot F_j + b \cdot F_{j-1}$ for some natural numbers a and b.*

Proof. We prove the statement by also showing that the values a and b are themselves two consecutive Fibonacci numbers, namely $a = F_{l-j+1}$ and $b = F_{l-j}$. Hence, in order to show that

$$F_l = a \cdot F_j + b \cdot F_{j-1} = F_{l-j+1} \cdot F_j + F_{l-j} \cdot F_{j-1} \tag{2}$$

we proceed by induction on j. The relation is true for $j = 1$ since $F_l = F_l \cdot F_1 + F_{l-1} \cdot F_0 = F_l \cdot 1 + 0$. Assuming that the relation is true for j, we have to show that it is true for $j+1$, that is $F_l = F_{l-j} \cdot F_{j+1} + F_{l-j-1} \cdot F_j$. By replacing F_l we obtain

$$F_{l-j+1} \cdot F_j + F_{l-j} \cdot F_{j-1} = F_{l-j} \cdot F_{j+1} + F_{l-j-1} \cdot F_j,$$

which is equivalent to

$$(F_{l-j+1} - F_{l-j-1}) \cdot F_j = (F_{j+1} - F_{j-1}) \cdot F_{l-j},$$

and in turn to the following identity $F_{l-j} \cdot F_j = F_j \cdot F_{l-j}$. □

Let G_ℓ be a graph with F_ℓ vertices and such that it is either a *complete graph* or an *independent set* (i.e., a set of vertices no two of which are adjacent). Let $G_{\ell_1 * \ell_2}$ be a graph with $F_{\ell_1} + F_{\ell_2}$ vertices obtained from G_{ℓ_1} and G_{ℓ_2} by connecting every vertex in G_{ℓ_1} to every vertex in G_{ℓ_2} with an edge. Given $G_{\ell_1 * \ell_2}$, G_{ℓ_1} and G_{ℓ_2} are called the *first* and the *second component* of $G_{\ell_1 * \ell_2}$, respectively.

Let C_ℓ, $\ell \geq 4$, be an initial configuration defined on $G_{\ell*(\ell-1)}$, and such that each vertex in its first component G_ℓ is occupied whereas each vertex in its second component $G_{\ell-1}$ is unoccupied. Notice that, by definition, C_ℓ contains F_ℓ robots. As an example, in Fig. 4 it is represented a configuration C_ℓ, with $\ell = 4$, built using two complete graphs as components.

Theorem 16. *If C_ℓ is composed of* FSync *robots, then each gathering algorithm for C_ℓ requires at least* $\left\lfloor \frac{\log_\phi\left(\sqrt{5}\cdot F_\ell - 1\right)-1}{2} \right\rfloor$ *time units, where ϕ is the golden ratio.*

Proof. Let us call *fibo* any non-empty subset S of vertices of G such that $|S|$ is a Fibonacci number, the vertices in S are pairwise equivalent and each vertex in S is occupied by the same number of robots. Moreover, according to its definition, the initial configuration C_ℓ is composed by two fibos.

Let $\mathcal{A}$ be any possible gathering algorithm for $C_\ell = C(0)$ and let $\mathbb{E} : C(0), C(1), \ldots$ be any possible execution performed by $\mathcal{A}$. We assume that the adversary will act in such a way to maintain any obtained configuration $C(t)$, $t > 0$, composed by a set of fibos, whatever the moves are. Obviously, if in $C(t)$ all the fibos have size greater than one, the configuration is not final.

We now show that the adversary's strategy can be always applied. This is equivalent to show by induction on t that the following property holds: *in any possible execution of $\mathcal{A}$, the configuration $C(t)$ produced at time $t \geq 0$ is composed by a set of fibos*. It is worth to remark that in this inductive proof we will also show an additional property: each time the algorithm produces a new configuration, the index of the size of the smallest fibo is reduced by at most two. Formally, let $F_{m(t)}$ be the size of the smallest fibo in $C(t)$, then such a size becomes at most $F_{m(t)-2}$ in $C(t+1)$.

The base of the induction holds for $t = 0$, since $C(0)$ coincides with the initial configuration C_ℓ which is composed by just two fibos. Now, assume by the inductive hypothesis that $C(t)$, $t > 0$, is composed by a set of fibos. Then, we show that the same property holds in $C(t+1)$ by analyzing any possible move performed by $\mathcal{A}$ from $C(t)$. Moreover we show that $F_{m(t+1)} \geq F_{m(t)-2}$.

Let $R(t)$ be the set containing all the robots that $\mathcal{A}$ moves in $C(t)$. Given $r \in R(t)$, in general $\mathcal{A}$ must define one or more vertices as target for r (the latter case occurs when the algorithm leaves r to arbitrarily select a vertex among many). If v_r is a target for r, then according to Property 1 and since each vertex in $C(t)$ belongs to a fibo by inductive hypothesis, then each vertex equivalent to v_r can be reached by r. Denoting by T_r all the possible targets of r, then it follows that T_r is made by a set of fibos (recall that the vertices of a fibo are pairwise equivalent). In particular, any fibo of $C(t)$ is either entirely included in T_r or completely excluded. In such a case, the adversary selects a fibo with smallest size in T_r as target for r. Denote such a fibo as τ_r and call it *effective target* of r. It can be observed that τ_r is the effective target not only for r but also for each robot r' which is equivalent to r in $C(t)$.

The above arguments can be applied to any robot in $R(t)$, and hence there exists a target τ_r for each robot $r \in R(t)$. Finally, notice that a generic fibo τ may result the effective target of non-equivalent robots in $R(t)$. Hence, in general there are many fibos in $C(t)$ occupied by robots that have τ as effective target: denote as S_τ the set containing all such fibos, call this set *effective source* for τ, and denote as Φ any element in S_τ.

We now analyze what the algorithm produces when all robots in an effective source S_τ are moved toward the corresponding effective target τ. To this end, denote as F_i the size of τ and let F_j be the size of the smallest fibo in S_τ. We analyze two cases, according whether $F_j \geq F_i$ or not.

- If $F_j \geq F_i$, consider the subdivision of τ into two subsets τ_1 and τ_2 of size F_{i-1} and F_{i-2}, respectively, which come by the definition of Fibonacci numbers as $F_i = F_{i-1} + F_{i-2}$. Hence, the adversary moves the robots in any fibo $\Phi \in S_\tau$ on τ_1 and τ_2, maintaining the two fibos. This is always possible because, if $|\Phi| = F_h$ for some h, then, as shown in Lemma 15, F_h can be expressed as a linear combination of F_{i-1} and F_{i-2}, that is $F_h = aF_{i-1} + bF_{i-2}$ for suitable integers a and b.
- If $F_j < F_i$, consider the subdivision of τ into two subsets τ_1 and τ_2 of size F_j and $F_i - F_j$, respectively. Then, the adversary moves all the robots of any fibo $\Phi \in S_\tau$ toward τ_1 by creating two further fibos of size F_{j-1} and F_{j-2} as shown in the previous case. Concerning τ_2, observe that it can be considered as composed by a set of fibos of sizes F_j and F_{j-1}, respectively. In fact, since $F_i = a'F_j + b'F_{j-1}$, for some integers a' and b', then $F_i - F_j = (a'-1)F_j + b'F_{j-1}$.

Summarizing, we have shown that the inductive hypothesis also holds in $C(t+1)$. Moreover, notice that in both the above cases the move performed by $\mathcal{A}$ transforms the size $F_{m(t)}$ of the smallest fibo in $C(t)$ into at most $F_{m(t)-2}$, and this is a lower bound to the size $F_{m(t+1)}$ of the smallest fibo in $C(t+1)$. This proves that each time the algorithm produces a new configuration, the index of the size of the smallest fibo is reduced by at most two.

We now analyze how the size of the smallest fibo evolves during any execution performed by $\mathcal{A}$. In $C(0)$ the smallest fibo has size $F_{m(0)} = F_{\ell-1}$. Since we have shown that the size of the smallest fibo is reduced at each time unit, then the algorithms terminates at a time t when $F_{m(t)} = 1$, which means $m(t) = 2$ or $m(t) = 1$. Since at each time unit the index of the size of the smallest fibo is reduced by at most two, then $\mathcal{A}$ performs at least $\left\lfloor \frac{\ell-1}{2} \right\rfloor$ moves. According to Equation (1), this implies that $\mathcal{A}$ requires at least $\left\lfloor \frac{\log_\phi\left(\sqrt{5}\cdot F_\ell - 1\right)-1}{2} \right\rfloor$ time units. □

Corollary 17. *There exists a gatherable configuration C composed of k* FSync *robots and defined on complete graphs or on complete bipartite graphs such that any gathering algorithm for C requires at least* $\left\lfloor \frac{\log_\phi\left(\sqrt{5}\cdot k - 1\right)-1}{2} \right\rfloor$ *time units.*

Proof. Consider the graph G_ℓ when it is built by using two complete graphs as components: by definition, G_ℓ is complete. Consider again G_ℓ but now assume that its components are two independent sets: by definition, G_ℓ is a complete bipartite graph. Consider also a configuration $C = C_\ell$ defined on G_ℓ when this graph is either complete or complete bipartite. The statement simply follows by considering this configuration along with Theorem 16. □

Corollary 18. *There exists an asymptotically optimal algorithm for any gatherable configuration defined on complete graphs and composed of* FSync *robots.*

Proof. The lower bound provided by Corollary 17 together with the complexity provided in Theorem 11 imply that algorithm $\mathcal{A}_{clique}$ defined within the proof of Theorem 10 is an asymptotically optimal algorithm for solving the gathering problem on complete graphs. □

Concerning complete bipartite graphs, the lower bound provided by Corollary 17 together with the complexity provided in Corollary 14 shows an additive factor of $\delta(k)$. As already pointed out in Section 1.3, whenever $\delta(k) = O(\log k)$ the algorithm is optimal. Note that, from [30] it is known that $\delta(k)$ behaves like a normal random variable with mean $\log\log k$ and standard deviation $\sqrt{\log\log k}$. However, concerning the growth rate of $\delta(k)$ it is known that $\delta(k) = o(n^\epsilon)$ for all $\epsilon > 0$; more precisely, $\delta(k) \leq k^{\frac{\log 2}{\log\log k}}$ [35].

7. Concluding remarks

We have considered the gathering problem of synchronous weak robots moving in graphs. First we have studied general properties that allow to solve the problem by means of SSync robots, regardless of the underlying topology. Then, we have focused on dense and symmetric graphs like complete and complete bipartite graphs, where we fully characterize when the gathering can be accomplished by means of FSync robots. Concerning the complexity of the proposed algorithms, $O(\log k)$ time units are required in complete graphs, whereas $O(\delta(k))$ time unites are necessary for complete bipartite graphs, where $\delta(n)$ is the function that returns the number of divisors of any integer n. Comparing this complexities with the provided lower bound of $O(\log_\phi k)$, with ϕ being the golden ratio, this leads to an asymptotically optimal algorithm for complete graphs and to a little gap in the case of complete bipartite graphs.

An interesting research direction concerns the case of SSync robots. In fact, while in complete graphs it is known SSync robots are not able to solve the gathering (see [14]), it remains open what they can do in complete bipartite graphs. An initial study on this respect can be found in [14]. However, a full characterization is still missing.

Our investigation highlights how the graph environment is very sensible to synchronization issues. This opens a wide area of research since FSync or SSync robots have not been considered much in graphs so far.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] László Babai, Ludek Kucera, Canonical labelling of graphs in linear average time, in: 20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29–31 October 1979, IEEE Computer Society, 1979, pp. 39–46.
[2] László Babai, Eugene M. Luks, Canonical labeling of graphs, in: Proceedings of the 15th Annual ACM Symposium on Theory of Computing, Boston, Massachusetts, USA, 25–27 April, 1983, ACM, 1983, pp. 171–183.
[3] Subhash Bhagat, Sruti Gan Chaudhuri, Krishnendu Mukhopadhyaya, Formation of general position by asynchronous mobile robots under one-axis agreement, in: Proc. 10th Int.'l WS on Algorithms and Computation (WALCOM), in: LNCS, vol. 9627, Springer, 2016, pp. 80–91.
[4] Kaustav Bose, Manash Kumar Kundu, Ranendu Adhikary, Buddhadeb Sau, Optimal gathering by asynchronous oblivious robots in hypercubes, in: Proc. 20th Int.'l Symp. on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (Algosensors), in: LNCS, vol. 11410, 2018, pp. 102–117.
[5] Jannik Castenow, Matthias Fischer, Jonas Harbig, Daniel Jung, Friedhelm Meyer auf der Heide, Gathering anonymous, oblivious robots on a grid, Theor. Comput. Sci. 815 (2020) 289–309.
[6] Serafino Cicerone, Gabriele Di Stefano, Alfredo Navarra, Minimum-traveled-distance gathering of oblivious robots over given meeting-points, in: Proc. 10th Int.'l Symp. on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (Algosensors), in: LNCS, vol. 8847, Springer, 2014, pp. 57–72.
[7] Serafino Cicerone, Gabriele Di Stefano, Alfredo Navarra, Minmax-distance gathering on given meeting-points, in: Proc. 9th Int.'l Conf. on Algorithms and Complexity (CIAC), in: LNCS, vol. 9079, Springer, 2015, pp. 127–139.
[8] Serafino Cicerone, Gabriele Di Stefano, Alfredo Navarra, Gathering of robots on meeting-points: feasibility and optimal resolution algorithms, Distrib. Comput. 31 (1) (2018) 1–50.
[9] Serafino Cicerone, Gabriele Di Stefano, Alfredo Navarra, "Semi-asynchronous": a new scheduler for robot based computing systems, in: Proc. 38th IEEE Int.'l Conf. on Distributed Computing Systems, (ICDCS), IEEE, 2018, pp. 176–187.
[10] Serafino Cicerone, Gabriele Di Stefano, Alfredo Navarra, Asynchronous arbitrary pattern formation: the effects of a rigorous approach, Distrib. Comput. 32 (2) (2019) 91–132.
[11] Serafino Cicerone, Gabriele Di Stefano, Alfredo Navarra, Asynchronous robots on graphs: gathering, in: Paola Flocchini, Giuseppe Prencipe, Nicola Santoro (Eds.), Distributed Computing by Mobile Entities, Current Research in Moving and Computing, in: LNCS, vol. 11340, Springer, 2019, pp. 184–217.

[12] Serafino Cicerone, Gabriele Di Stefano, Alfredo Navarra, Embedded pattern formation by asynchronous robots without chirality, Distrib. Comput. 32 (4) (2019) 291–315.
[13] Serafino Cicerone, Gabriele Di Stefano, Alfredo Navarra, Gathering synchronous robots in graphs: from general properties to dense and symmetric topologies, in: Proc. 26th Int.'l Colloquium on Structural Information and Communication Complexity (SIROCCO), in: LNCS, vol. 11639, Springer, 2019, pp. 170–184.
[14] Serafino Cicerone, Gabriele Di Stefano, Alfredo Navarra, On gathering of semi-synchronous robots in graphs, in: Stabilization, Safety, and Security of Distributed Systems - 21st International Symposium, (SSS), in: LNCS, vol. 11914, Springer, 2019, pp. 84–98.
[15] Mark Cieliebak, Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, Distributed computing by mobile robots: gathering, SIAM J. Comput. 41 (4) (2012) 829–879.
[16] Andreas Cord-Landwehr, Matthias Fischer, Daniel Jung, Friedhelm Meyer auf der Heide, Asymptotically optimal gathering on a grid, in: Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2016, Asilomar State Beach/Pacific, Grove, CA, USA, July 11–13, 2016, ACM, 2016, pp. 301–312.
[17] Gianlorenzo D'Angelo, Gabriele Di Stefano, Ralf Klasing, Alfredo Navarra, Gathering of robots on anonymous grids and trees without multiplicity detection, Theor. Comput. Sci. 610 (2016) 158–168.
[18] Gianlorenzo D'Angelo, Gabriele Di Stefano, Alfredo Navarra, Gathering asynchronous and oblivious robots on basic graph topologies under the look-compute-move model, in: Search Theory: a Game Theoretic Perspective, Springer, 2013, pp. 197–222.
[19] Gianlorenzo D'Angelo, Gabriele Di Stefano, Alfredo Navarra, Gathering on rings under the look-compute-move model, Distrib. Comput. 27 (4) (2014) 255–285.
[20] Gianlorenzo D'Angelo, Gabriele Di Stefano, Alfredo Navarra, Gathering six oblivious robots on anonymous symmetric rings, J. Discret. Algorithms 26 (2014) 16–27.
[21] Gianlorenzo D'Angelo, Gabriele Di Stefano, Alfredo Navarra, Nicolas Nisse, Karol Suchan, Computing on rings by oblivious robots: a unified approach for different tasks, Algorithmica 72 (4) (2015) 1055–1096.
[22] Gianlorenzo D'Angelo, Alfredo Navarra, Nicolas Nisse, A unified approach for gathering and exclusive searching on rings under weak assumptions, Distrib. Comput. 30 (1) (2017) 17–48.
[23] Shantanu Das, Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, Masafumi Yamashita, Autonomous mobile robots with lights, Theor. Comput. Sci. 609 (2016) 171–184.
[24] Shantanu Das, Paola Flocchini, Nicola Santoro, Masafumi Yamashita, Forming sequences of geometric patterns with oblivious mobile robots, Distrib. Comput. 28 (2) (2015) 131–145.
[25] MattiaD'Emidio, Gabriele Di Stefano, Daniele Frigioni, Alfredo Navarra, Characterizing the computational power of mobile robots on graphs and implications for the Euclidean plane, Inf. Comput. 263 (2018) 57–74.
[26] Mattia D'Emidio, Daniele Frigioni, Alfredo Navarra, Characterizing the computational power of anonymous mobile robots, in: Proc. 36th IEEE Int.'l Conf. on Distributed Computing Systems, (ICDCS), IEEE, 2016, pp. 293–302.
[27] Gabriele Di Stefano, Pietro Montanari, Alfredo Navarra, About ungatherability of oblivious and asynchronous robots on anonymous rings, in: Proc. 26th Int.'l WS on Combinatorial Algorithms (IWOCA), in: LNCS, vol. 9538, Springer, 2016, pp. 136–147.
[28] Gabriele Di Stefano, Alfredo Navarra, Gathering of oblivious robots on infinite grids with minimum traveled distance, Inf. Comput. 254 (2017) 377–391.
[29] Gabriele Di Stefano, Alfredo Navarra, Optimal gathering of oblivious robots in anonymous graphs and its application on trees and rings, Distrib. Comput. 30 (2) (2017) 75–86.
[30] P. Erdös, M. Kac, The Gaussian law of errors in the theory of additive number theoretic functions, Am. J. Math. 62 (1) (1940) 738–742.
[31] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, Peter Widmayer, Arbitrary pattern formation by asynchronous, anonymous, oblivious robots, Theor. Comput. Sci. 407 (1–3) (2008) 412–447.
[32] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro (Eds.), Distributed Computing by Mobile Entities, Current Research in Moving and Computing, LNCS, vol. 11340, Springer, 2019.
[33] Swapnil Ghike, Krishnendu Mukhopadhyaya, A distributed algorithm for pattern formation by autonomous robots, with no agreement on coordinate compass, in: Proc. 6th Int.'l Conf. on Distributed Computing and Internet Technology, (ICDCIT), in: LNCS, vol. 5966, Springer, 2010, pp. 157–169.
[34] Samuel Guilbault, Andrzej Pelc, Gathering asynchronous oblivious agents with local vision in regular bipartite graphs, Theor. Comput. Sci. 509 (2013) 86–96.
[35] Godfrey H. Hardy, Edward M. Wright, An Introduction to the Theory of Numbers, 6th ed., Oxford University Press, 2008.
[36] Tomoko Izumi, Taisuke Izumi, Sayaka Kamei, Fukuhito Ooshita, Time-optimal gathering algorithm of mobile robots with local weak multiplicity detection in rings, IEICE Trans. 96-A (6) (2013) 1072–1080.
[37] Ralf Klasing, Adrian Kosowski, Alfredo Navarra, Taking advantage of symmetries: gathering of many asynchronous oblivious robots on a ring, Theor. Comput. Sci. 411 (2010) 3235–3246.
[38] Ralf Klasing, Euripides Markou, Andrzej Pelc, Gathering asynchronous oblivious mobile robots in a ring, Theor. Comput. Sci. 390 (2008) 27–39.
[39] Brendan D. McKay, Practical graph isomorphism, in: Congressus Numerantium, 30, 1981, pp. 45–87.
[40] Takunari Miyazaki, The complexity of McKay's canonical labeling algorithm, in: Groups and Computation, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, June 7–10, 1995, 1995, pp. 239–256.
[41] Fukuhito Ooshita, Sébastien Tixeuil, On the self-stabilization of mobile oblivious robots in uniform rings, in: Proc. 14th Int.'l Symp. on Stabilization, Safety, and Security in Distributed Systems (SSS), in: LNCS, vol. 7596, Springer, 2012, pp. 49–63.
[42] Arun Sadhu, Madhumita Sardar, Deepanwita Das, Srabani Mukhopadhyaya, Gathering in the discrete domain: state of the art, in: Jyotsna Kumar Mandal, Dhananjay Bhattacharyya, Nitin Auluck (Eds.), Advanced Computing and Communication Technologies, Springer, Singapore, 2019, pp. 11–18.
[43] Nicola Santoro, Design and Analysis of Distributed Algorithms, John Wiley & Sons, 2007.
[44] Peter J. Slater, Medians of arbitrary graphs, J. Graph Theory 4 (4) (1980) 389–392.
[45] Ichiro Suzuki, Masafumi Yamashita, Distributed anonymous mobile robots: formation of geometric patterns, SIAM J. Comput. 28 (4) (1999) 1347–1363.
[46] Masafumi Yamashita, Ichiro Suzuki, Characterizing geometric patterns formable by oblivious anonymous mobile robots, Theor. Comput. Sci. 411 (26–28) (2010) 2433–2453.
[47] Yukiko Yamauchi, Taichi Uehara, Shuji Kijima, Masafumi Yamashita, Plane formation by synchronous mobile robots in the three dimensional Euclidean space, in: Proc. 29th Int.'l Symp. on Distributed Computing (DISC), in: LNCS, vol. 9363, Springer, 2015, pp. 92–106.