Uniform Circle Formation

Giovanni Viglietta(✉)

JAIST, Nomi, Japan
johnny@jaist.ac.jp

Abstract. We treat the second of the two patterns that are formable in the $\mathcal{OBLOT}$ model from every initial configuration of n robots: `Uniform Circle`, i.e., the pattern where the robots are located at the vertices of a regular n-gon. The algorithm presented in this chapter solves the `Uniform Circle` Formation Problem in the standard $\mathcal{OBLOT}$ model under the $\mathcal{A}$SYNC scheduler.

Keyword: Uniform Circle Formation

1 Introduction

In this chapter we treat the second of the two patterns that are formable in $\mathcal{OBLOT}$ from every initial configuration of n robots: `Uniform Circle`, i.e., the pattern where the robots are located at the vertices of a regular n-gon. The algorithm presented in this chapter asssumes that no two robots are initially in the same location, and solves the `Uniform Circle` Formation Problem in the standard $\mathcal{OBLOT}$ model under the $\mathcal{A}$SYNC scheduler, and has been published in [12,15].

The main body of the algorithm is described in Sect. 2, and deals with the case of $n > 5$ robots. The cases with fewer robots are solved with ad-hoc algorithms, and are discussed in Sect. 3.

Several related papers have appeared, studying the `Uniform Circle` Formation Problem or its variants, and solving it in special cases or under different assumptions [2,4–8,10,16]. One of the earliest results on the problem appeared in [18], where it is shown that robots can form `Uniform Circle` under the $\mathcal{S}$SYNC scheduler, provided that they can remember past observations. Another early algorithm, given in [5], makes a swarm of oblivious robots converge towards `Uniform Circle` (possibly without ever forming it) under the $\mathcal{S}$SYNC scheduler. Other algorithms from the same period can be combined to prove that `Uniform Circle` is actually formable under the $\mathcal{S}$SYNC scheduler: by concatenating the algorithm in [14], for forming a biangular configuration, with the one in [7], for forming `Uniform Circle` from a biangular starting configuration, it is possible to form `Uniform Circle` from any initial configuration (the case with four robots has been solved separately in [8]). Observe, however, that the two algorithms can be concatenated only because the scheduler is $\mathcal{S}$SYNC: under the $\mathcal{A}$SYNC scheduler, this technique would be ineffective.

P. Flocchini et al. (Eds.): Distributed Computing by Mobile Entities, LNCS 11340, pp. 83–108, 2019.
https://doi.org/10.1007/978-3-030-11072-7_5

Some results for the $\mathcal{A}$SYNC scheduler assume implicit agreements among all the robots. As shown in [10], if the local coordinate systems of all the robots have the same orientation (i.e., the system has chirality), there is a simple algorithm to form `Uniform Circle`. This has later been improved by a general result of [13], which only assumes that all local coordinate systems are right-handed. Then, in [11], a `Uniform Circle` Formation algorithm was given for $\mathcal{A}$SYNC robots with no assumptions on their local coordinate systems, but allowing them to move along circular arcs, as well as straight line segments. A solution for $n \neq 4$ robots under the $\mathcal{A}$SYNC scheduler without extra assumptions (i.e., in the standard $\mathcal{OBLOT}$ model) was finally given in [12], and the solution for the special case $n = 4$ appeared in [15].

Related results about the formation of `Uniform Circle` include a study of the problem of minimizing the maximum distance traveled by a robot, where robots have one bit of internal memory [1], an algorithm for transparent robots with extent that are $\mathcal{S}$SYNC, perform rigid movements, and agree on one axis [17], and an algorithm for opaque $\mathcal{LUMINOUS}$ robots in $\mathcal{F}$SYNC [9].

2 General Algorithm for $n > 5$ Robots

If the swarm of robots is not too small, i.e., if $n > 5$, there is a general `Uniform Circle` Formation algorithm that works in all cases [12]. Recall that the goal of the robots is to position themselves on the vertices of a regular n-gon and stop moving: we call this type of configuration *Regular*.

A fundamental geometric tool used in the algorithm is the concept of *smallest enclosing circle (SEC)* of the swarm of robots: this is the circle of smallest radius that contains all robots. It is well known that such a circle exists, is unique, and its center and radius can be effectively computed by the robots.

The algorithm can be outlined as follows: the general strategy is to make the robots move to the SEC, determine their final "target points", and then move to such points to form a *Regular* configuration. The robots always move in an orderly fashion, and the ones that are allowed to move at each step are carefully chosen in such a way that the SEC does not change during their movements. The only exception to this protocol is when the robots form, either "intentionally" or "accidentally", a particular type of configuration called *Pre-regular*: in this case, they follow a special procedure that ignores the SEC.

The details of the algorithm are given below.

2.1 Special Cases: *Biangular* and *Pre-regular* Configurations

We first consider the *Biangular* configurations, exemplified in Fig. 1(a). In such a configuration, the number of robots n is even, and the set of their locations has exactly $n/2$ axes of symmetry. Note that a *Biangular* configuration can be partitioned into two *Regular* configurations of size $n/2$. This is a particularly interesting situation, because the robots may all have the exact the same view, provided that their axes are oriented symmetrically, as Fig. 1(a) shows. In this scenario, the scheduler may force all robots to perform the same computation and move at the same time, which will make the configuration remain *Biangular* at all times (or become *Regular*). So, any `Uniform Circle` Formation algorithm must ensure that this type of synchronous behavior from a *Biangular* configuration indeed results in the formation of a *Regular* configuration. Nonetheless, the algorithm must also take into account that movements may be asynchronous and non-rigid: while moving toward their destinations, the robots may also form different and possibly asymmetric intermediate configurations, and asynchronously compute new destinations from those configurations. Therefore, it is clearly desirable that the robots preserve some geometric invariant as they move, so that any such intermediate configuration is treated coherently with the *Biangular* case.

A solution to the problem of forming a regular polygon starting from a *Biangular* configuration is given in [7], where the robots identify a "supporting regular polygon" (see Fig. 1(b)), and each robot moves toward the closest vertex of such a polygon. Any intermediate configuration possibly formed while the robots move asynchronously towards the vertices of the supporting polygon is called *Pre-regular* (note that all *Biangular* configurations are also *Pre-regular*). While executing this procedure from a *Pre-regular* configuration, the supporting polygon remains invariant (e.g., see Fig. 1(c)). So, whenever the configuration is perceived as *Pre-regular* by *all* the robots, moving toward the appropriate vertex of the supporting polygon results in the formation of a *Regular* configuration.

Note that a robot can effectively determine whether the observed configuration is *Pre-regular*. Moreover, thanks to the following lemma, all robots in a *Pre-regular* configuration implicitly agree on the same supporting polygon and behave coherently with one another.

Lemma 1 ([12]). *The supporting polygon of a* Pre-regular *configuration is unique.*

2.2 General Strategy: SEC and Analogy Classes

Consider now a starting position of the robots that is not *Pre-regular* (and hence not *Biangular*). Recall that the robots have no common reference frame, and there are no "environmental" elements that can help them orient themselves. This difficulty may prevent the robots from coordinating their movements and act "consistently" from one cycle to another. To overcome this, the SEC of the robots' positions is identified (see Fig. 2(a)), and the algorithm ensures that the robots move in such a way as to keep the SEC fixed. This will hold true so long as the configuration is not *Pre-regular*. If the configuration happens to become *Pre-regular* during the execution, then the procedure of Sect. 2.1 will be executed, and the SEC will no longer be preserved.

As a preliminary step, the general algorithm attempts to make all robots reach the perimeter of the SEC. So, let us consider a configuration that is not *Pre-regular* and in which all robots lie on the perimeter of SEC. In this situation, we identify pairs of robots r_1 and r_2 that are located in "symmetric" positions, i.e.,

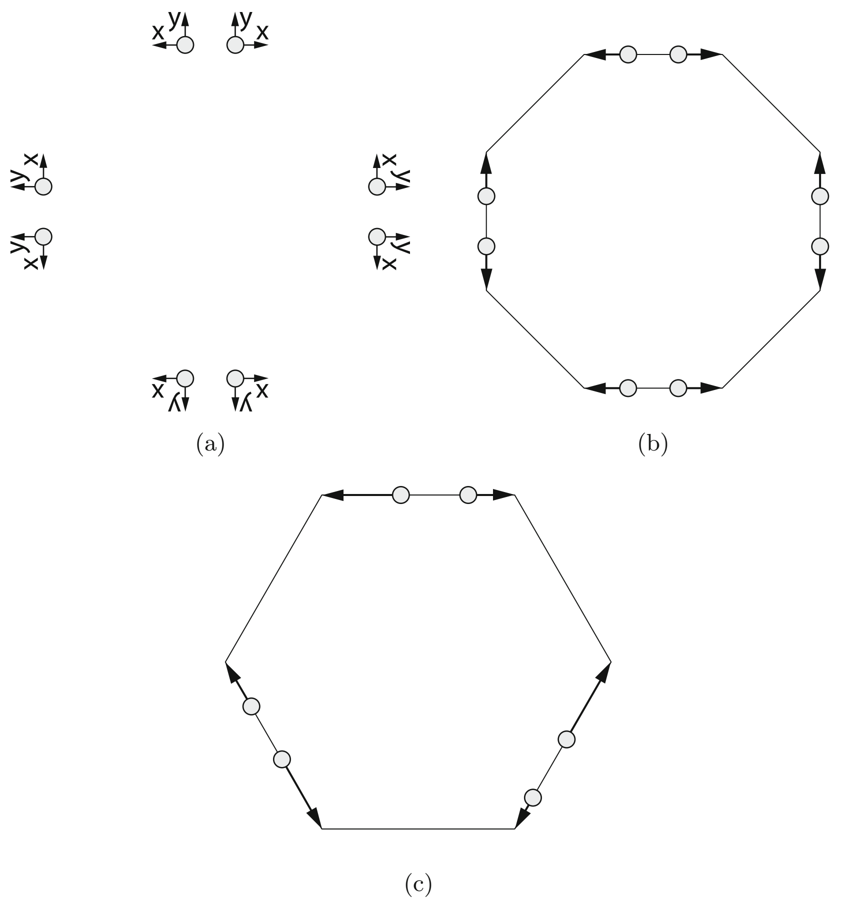


Fig. 1. (a) A *Biangular* configuration with local axes oriented in such a way that all robots have the same view. (b) How the algorithm resolves a *Biangular* configuration. (c) A generic *Pre-regular* configuration with its supporting polygon, which remains invariant as the robots move along the arrows. (Source: [12])

such that there is an isometry of the plane that permutes the robots' locations switching the positions of r_1 and r_2. We call two such robots *analogous*, and the swarm is thus partitioned into *analogy classes* of analogous robots (see Fig. 2(b)). In general, an analogy class has either the shape of a *Regular* set or of a *Biangular* set (with some degenerate cases, such as a single point or a pair of points).

Similarly to the *Biangular* case (cf. Sect. 2.1), the scheduler may force all the robots in an analogy class to perform the same computation and move at the same time, thus occupying symmetric positions again, and potentially forever. To accommodate this, the algorithm incorporates this type of behavior and makes all analogous robots always *deliberately* move together in the same fashion.

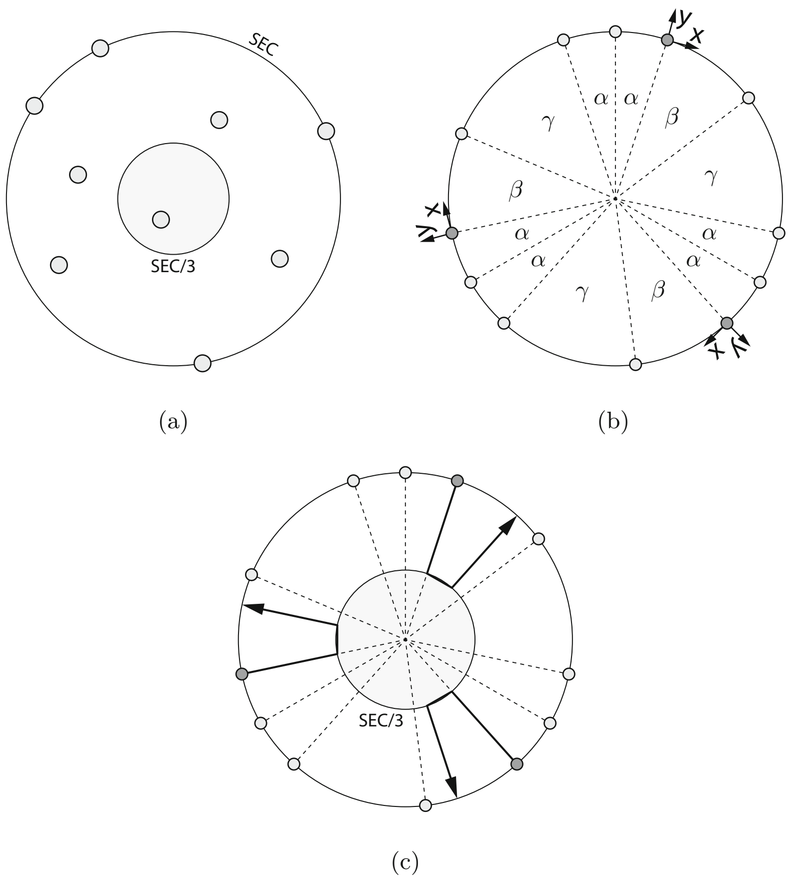


Fig. 2. (a) A swarm of robots with its SEC and SEC/3. (b) The three highlighted robots form an analogy class. If their axes are oriented as indicated, the three robots have the same view. (c) The three dark-shaded robots are selected as walkers and move according to the arrows. At the end, for each walker there is a non-walker at an angular distance of $\pi/3$ from it (note that $\pi/3$ is a multiple of $2\pi/n = \pi/6$). (Source: [12])

More specifically, the algorithm lets only one analogy class move at any given time, while all the other robots wait on the SEC (see Fig. 2(c)). The robots in the analogy class that is allowed to move are called *walkers*. When the walkers have been chosen, they move radially to SEC/3, which is the circle whose radius is 1/3 of the radius of the SEC and concentric with the SEC. Once they are all on the perimeter of SEC/3, they move to their so-called *finish set*, while staying within SEC/3 (or in its interior). When they are all in their finish set, they move radially to the SEC again. After that, a new analogy class of walkers is chosen, and so on. The walkers and the finish set are chosen in such a way that, when the walkers are done moving, some kind of "progress" toward a *Regular* configuration is made. For instance, progress is made when two analogy classes merge and become one (note that a *Regular* configuration has only one analogy class), or when the angular distance between two robots on the SEC becomes a multiple of $2\pi/n$ (note that in a *Regular* configuration all angular distances are multiples of $2\pi/n$).

Of course, as the walkers move, they need a strategy to "wait for each other" and make sure to reach a configuration where they are once again analogous. Also, different analogy classes should plan their movements "coherently", in such a way that their combined motion eventually results in the formation of a *Regular* configuration. Note that this is complicated by the fact that, when a class of walkers starts moving, some of the "reference points" the robots were using to compute their destinations may be lost. Moreover, it may be impossible to select a class of walkers in such a way that some "progress" is made when they reach their destinations, and in such a way that the SEC remains fixed as they move. In this case, the configuration is said to be *locked*, and some special moves have to be made.

Finally, as the robots move according to the general algorithm that has just been outlined, they may end up forming a *Pre-regular* configuration "by accident". So, the robots need a technique to stop immediately whenever this happens, so that they can all recognize the *Pre-regular* configuration and start executing the procedure of Sect. 2.1 (note that some robots may be in the middle of a movement when a *Pre-regular* configuration is formed accidentally, and countermeasures have to be taken to prevent this, or else different robots may end up executing different protocols, and the swarm will behave incoherently).

All the aforementioned aspects will be discussed in the rest of this section. Next we will show how the robots can reach the SEC from any initial configuration, as a preliminary step.

2.3 Preliminary Step: Reaching the SEC

A simple way to make all robots reach the SEC without colliding is to make each of them move radially, away from the center, as in Fig. 3(a). This protocol only works assuming that no two robots are *co-radial*, i.e., on the same half-line extending from the center of the SEC. A special case is the *Central* configuration, in which one robot lies at the center of the SEC. *Central* configurations are easily resolved by simply making the central robot move to SEC/3, in such a way as not to become co-radial with any other robot.

The *Co-radial* configurations that are not *Central* are handled as follows. First of all, if there is any robot in the interior of SEC/3 that is not co-radial with any other robot, it moves radially to SEC/3 (note how the evolution of a *Central* configuration naturally blends with this protocol). Then, for each maximal set of at least two mutually co-radial robots, the robot that is closest to the center of the SEC moves radially toward the center until it is in SEC/3 (see Fig. 3(b)). Finally, the most internal co-radial robots make a lateral move to become non-co-radial, as in Fig. 3(c). The lateral move is within SEC/3 (or its interior) and it is "sufficiently small", in order to prevent collisions. A sufficiently small move

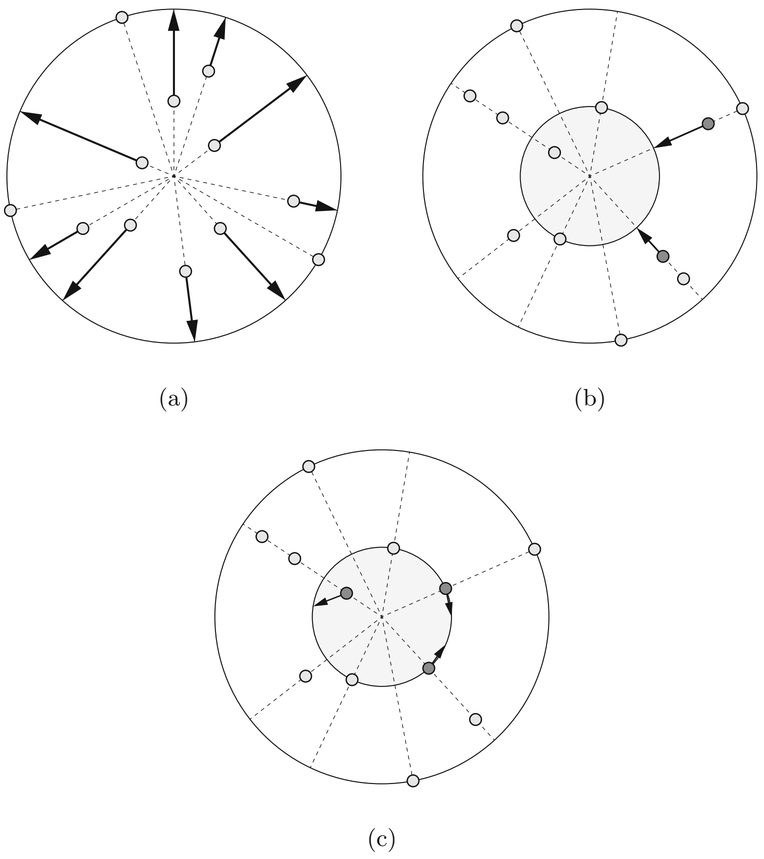


Fig. 3. (a) All robots move radially to reach the SEC. (b) The most internal co-radial robots move radially to SEC/3. (c) When they are in SEC/3, they make a small lateral move. (Source: [12])

is, for instance, a move that reduces the angular distance to any other robot by no more than 1/3.

The reason why we make robots reach SEC/3 before performing lateral moves is because we want to prevent the accidental formation of *Pre-regular* configurations. We will discuss this aspect later, in Sect. 2.9.

It is easy to see how this strategy makes the robots coordinate themselves and avoid collisions. Indeed, as soon as a robot r makes a lateral move and stops being co-radial with other robots, it is seen by the other robots as a non-co-radial robot lying in the interior of SEC/3. Hence, no other robot will take initiatives, and will just wait until r has reached SEC/3 and has stopped there. This guarantees that, when a robot decides to perform a lateral move, no other robot is in the middle of a lateral move.

Also, no matter how many robots lie on the same line through the center of the SEC, the innermost will always move first, and then the others will follow in order, after the first has stabilized on SEC/3. When this procedure is completed, there are no more co-radial robots and no robots in the interior of SEC/3. At this point, the robots can safely move toward the SEC radially.

Lemma 2 ([12]). *If no* Pre-regular *configurations are ever formed, the algorithm will make all robots eventually reach the perimeter of the SEC and stop there.*

After this phase of the algorithm has been completed, no two robots will ever become co-radial again. We will achieve this through a careful selection of *walkers* and *target points*, and by making walkers move appropriately.

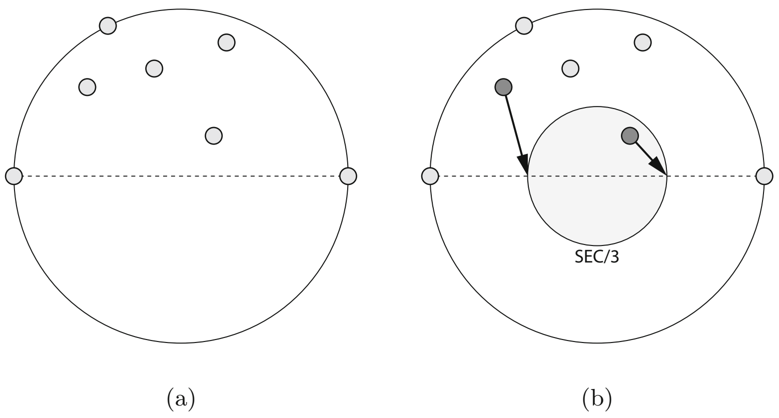
2.4 ***Half-Disk*** Configurations

Another special initial case has to be resolved: the *Half-disk* case. In this configuration, all the robots lie in one half-disk of the SEC, and the diameter of such a half-disk is called *principal diameter* (see Fig. 4(a)). The reason why it is convenient to resolve these configurations immediately and separately from all others will be explained in the following, when discussing locked configurations.

Half-disk configurations are resolved by making some robots move from the "occupied" half-disk of the SEC to the "non-occupied" one. Note that, while doing so, some robots have to cross the principal diameter. Also, as a consequence of the definition of SEC, the principal diameter must contain robots on both endpoints. These two robots, r_1 and r_2, must stay in place in order to maintain the SEC stable. Hence, exactly two other robots, which have smallest angular distances from r_1 and r_2 respectively, move to the two points where the principal diameter intersects SEC/3 (see Fig. 4(b)). Once they are both there, they move into the non-occupied half-disk, remaining inside SEC/3, as in Fig. 4(c). (More precisely, if the principal diameter already contains some robots on or inside SEC/3, such robots do not preliminarily move to the perimeter of SEC/3, because it is unnecessary and it may even cause collisions; in this case, they move into the unoccupied half-disk right away.)

A very special *Half-disk* case is the one where all robots lie on the same line. This case is handled as a generic *Half-disk*, with two robots first moving on SEC/3 (if they are not already on it or in its interior), and then moving away from the principal diameter. If they move in opposite directions, the configuration is no longer *Half-disk*. If they move in the same direction, they form a generic *Half-disk*, which is then resolved normally.

When analyzing the possible evolutions of a *Half-disk* configuration, one has to keep in mind that it transitions into a different configuration while one or two robots are still moving. This turns out to be relatively easy, since the moving robots are inside SEC/3 (like the robots that move laterally in the *Co-radial* case) and move in a very predictable and controlled way. When the configuration ceases to be *Half-disk*, the robots will move on SEC as described before, and they will never form a *Half-disk* configuration again.



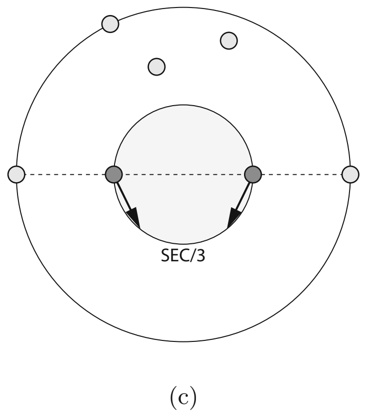


Fig. 4. (a) A *Half-disk* configuration and the principal diameter. (b) Two robots move to the intersection between the principal diameter and SEC/3. (c) The same two robots move to the non-occupied half-disk. (Source: [12])

Lemma 3 ([12]). *If no* Pre-regular *configurations are ever formed, the algorithm will make all robots eventually reach the perimeter of the SEC and stop there, in such a way that in every half-circle of the SEC there is at least one robot.*

2.5 Identifying Targets

Suppose now that all robots lie on the perimeter of the SEC, and the configuration is not *Pre-regular* and not *Half-disk*. In this case we can define a *target set*, which represents the final *Regular* configuration that the robots are attempting to form. Each element of the target set is called a *target*, and corresponds to some robot's intended destination. Hence the target set is a *Regular* set of n points arranged on the SEC in such a way that it can be computed by all robots, regardless of their local coordinate system (i.e, regardless of the orientation of their local axes, their handedness, and their unit of distance). Next we describe how the target set is defined, depending on the configuration of the robots.

Assume that the configuration has an axis of symmetry ℓ. Then ℓ must necessarily be an axis of symmetry of the target set. If one robot r lies on ℓ, then the target of r coincides by definition with r's location, and the other targets are defined accordingly (see Fig. 5(a)). If no robot lies on ℓ, then no target is placed on ℓ, either. In this case, the correspondence between robots and targets is as in Fig. 5(b). It is not hard to prove that this definition is independent of the choice of an axis of symmetry ℓ, and that therefore the same target set is computed by all robots.

Lemma 4 ([12]). *Even if the configuration has more than one axis of symmetry, it has a unique target set, and the robot-target correspondence is uniquely determined.*

Assume now that the configuration has no axes of symmetry. In this case we say that two robots are *concordant* if their angular distance is of the form $2k\pi/n$, for some integer k, and between them there are exactly $k-1$ robots. In other terms, two concordant robots have the "correct" angular distance, and between them there is the "correct" number of robots. This is an equivalence relation that partitions the robots into *concordance classes*. The largest concordance class determines the target set: by definition, each robot in this class coincides with its own target. Even if the largest concordance class is not unique, it turns out that there is always a way to choose one of them unambiguously, in such a way that all robots implicitly agree on it. Once some targets have been fixed, the other targets and correspondences are determined accordingly, as Fig. 5(c) shows.

Lemma 5 ([12]). *In every configuration with all robots on the perimeter of the SEC, the target set and the robot-target correspondence are uniquely determined.*

2.6 Identifying Walkers, Locked Configurations

When the target set has been identified, then the *walkers* can be defined. The walkers are simply the analogy class of robots that are going to move next.

Typically, the algorithm will attempt to move an analogy class of robots to their corresponding targets. The robots that currently lie on their targets are called *satisfied*, and these robots should not move. Moreover, the walkers should be chosen in such a way that, when they abandon the perimeter of the SEC and move into its interior, they do not cause the SEC to change. An analogy class of robots with this property is called *movable*. Finally, no two robots should become co-radial as a result the walkers' movements. This means that the walkers should be chosen in such a way that, as they move toward their targets, they do not become co-radial with other robots. The targets of such robots are said to be *reachable*.

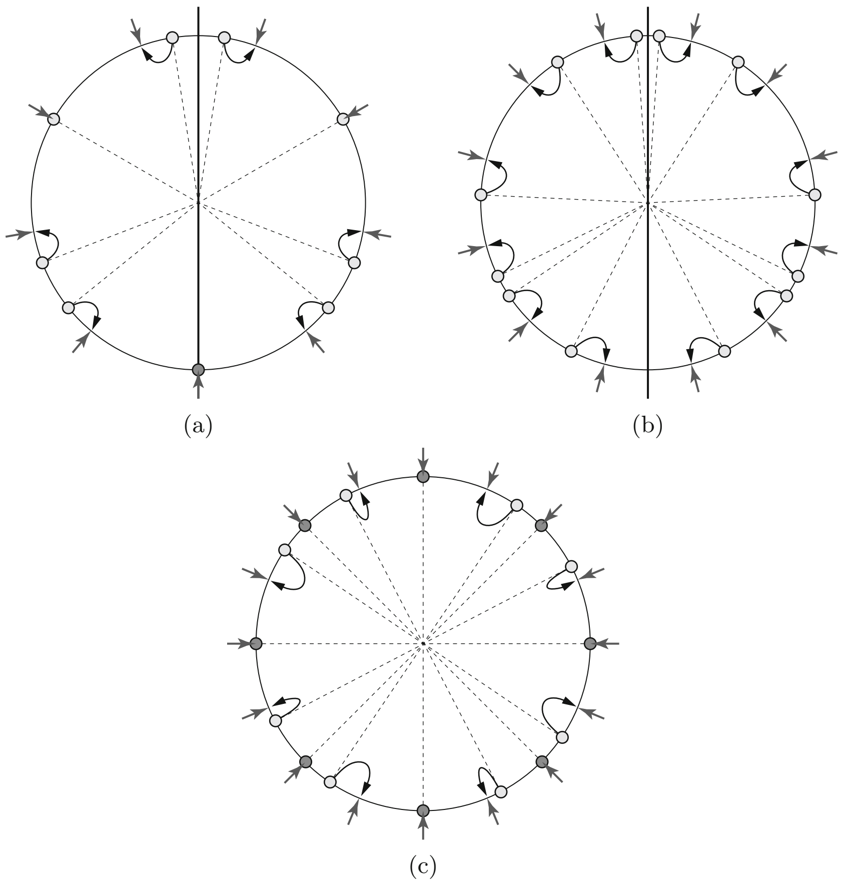


Fig. 5. The outer arrows indicate targets, and the inner arrows indicate correspondences between robots and targets. (a) The dark-shaded robot lies on an axis of symmetry. (b) There are some axes of symmetry, none of which contains a robot. (c) There are no axes of symmetry, and the dark-shaded robots form the largest concordance class. (Source: [12])

Therefore, the walkers are selected to be a movable analogy class of robots that are not satisfied and can reach their targets without ever becoming co-radial with other robots. If such a class is not unique, one can always be chosen unambiguously.

There are special cases where no such analogy class or robots exists: these configurations are said to be *locked* (see for instance Fig. 6(a)). In a locked configuration, the walkers are chosen with a different criterion: they are an analogy class that is movable and not satisfied, and that is adjacent to some non-movable analogy class. Such an analogy class is called *unlocking*. The goal of these walkers is not to reach their targets (if they could, the configuration would not be locked), but to move in such a way as to “unlock” the configuration (as in Fig. 6(b)), thus allowing other robots, which were previously non-movable, to reach their targets (as in Fig. 6(c)).

Lemma 6 ([12]). *In a locked configuration, each analogy class consists of at most two robots. Also, there are at most two robots that are non-movable, and they are adjacent.*

It follows that there are only one or two walkers in a locked configuration, and each of them is adjacent to some non-movable robot.

Lemma 7 ([12]). *In every configuration with all robots on the perimeter of the SEC, the walkers and their destinations are well defined.*

2.7 Identifying *Valid* Configurations

Next we describe the journey that the walkers have to take to reach their destinations. First they move radially to the perimeter of SEC/3, and they wait for each other there. Once they are all on SEC/3, they start moving laterally, remaining within SEC/3 and its interior, until they reach their *finish set*, which is simply the set their destinations on SEC/3. Once they are in their finish set, they move back to the perimeter of the SEC, radially.

The reason why the walkers move all the way to SEC/3, instead of going directly to their destinations, is two-fold. It makes it easier to foresee and prevent the accidental formation of *Pre-regular* configurations (see Sect. 2.9), and it clearly separates the robots that should move from the ones that should wait, so that none gets confused as the configuration changes.

Note that it is easy to recognize a configuration in which the walkers are moving radially to SEC/3 or back to the SEC, because the analogy classes (and hence the walkers) are defined only based on angular distances between robots. Thus, if all robots are on the SEC, except for a few analogous robots that are between SEC and SEC/3, then the configuration is recognized as a “consistent”, or *Valid* one, in which the walkers are either moving to SEC/3 or are moving back to the perimeter of the SEC (see Fig. 7(a)).

If the walkers have already started moving laterally in SEC/3, then recognizing the configuration as a *Valid* one is more difficult. This can be done by “guessing” where the internal robots were located when they were still on the SEC and they have been selected as walkers. If there is a way to re-position the internal robots within their respective “sectors” of the SEC in such a way as to make them become a full analogy class, then the configuration is considered *Valid*, and the internal robots are considered walkers (see Fig. 7(b)). Otherwise, it means that the execution is in one of the earlier stages, and the robots still have to make their preliminary move to the SEC.

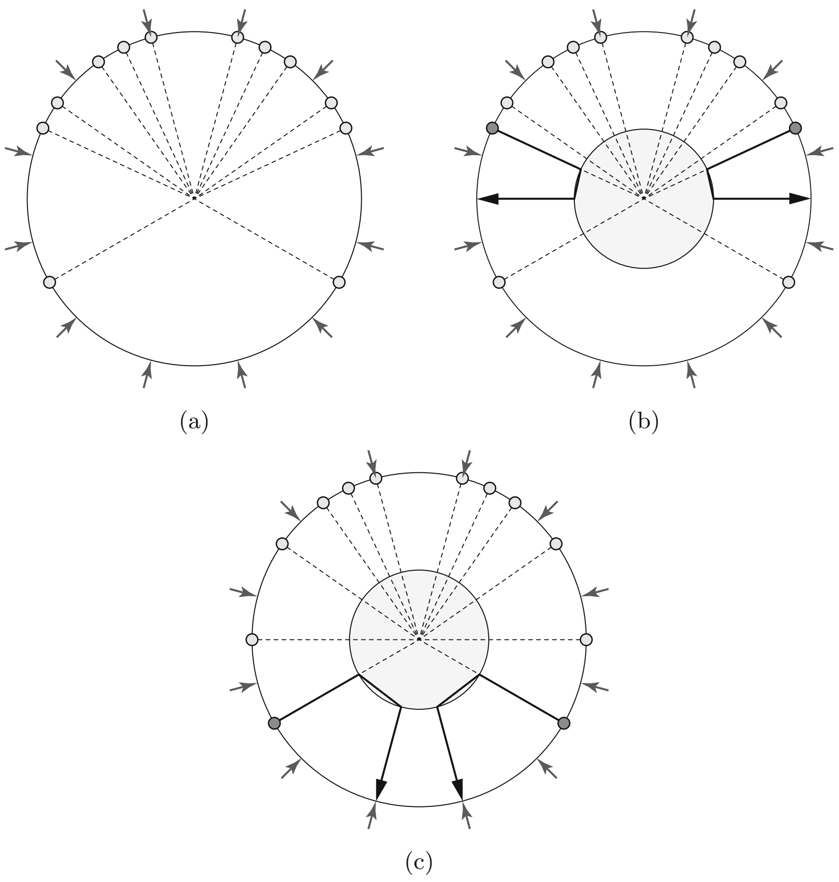


Fig. 6. (a) A locked configuration: the topmost robots are satisfied, the bottommost robots are non-movable, and all other robots would become co-radial in the process of reaching their targets. (b) A preliminary move is made to unlock the configuration. (c) When the configuration is unlocked, the bottommost robots become movable. (Source: [12])

2.8 Identifying the Finish Set

Once the configuration has been recognized as *Valid* and all walkers are in SEC/3, they compute their *finish set*. Recall that this is the set of their destinations on SEC/3, which they want to reach before moving back to SEC.

In order to understand where they should be going, the walkers have to recompute their targets. Indeed, note that the original targets have been computed when the walkers were on the perimeter of SEC. As they are now in SEC/3, in the process of moving laterally to their destinations, they need a robust way to define targets. This means that different walkers should compute the same target set, and the target set should not change as the walkers move within SEC/3.

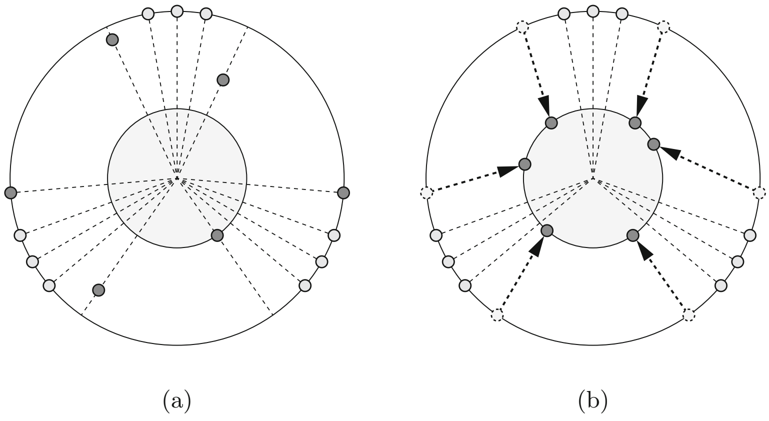


Fig. 7. Two types of *Valid* configurations. (a) Some analogous robots lie between SEC and SEC/3, and all other robots are on the SEC. (b) All robots are on the SEC or on SEC/3, and the distribution of the internal robots is compatible with a possible initial configuration in which they were all on the SEC, forming an analogy class. (Source: citeFPSV17)

Of course it may not be possible to reconstruct the original walkers' positions on the SEC and recompute the original targets, and therefore once again the walkers have to "take a guess". The default guess is that, when they were still on the perimeter of the SEC, each walker was equidistant from its two adjacent robots, as in Fig. 8(a). This position of the walkers is referred to as the *principal relocation*, and of course it can be computed unambiguously by all robots.

Now the robots compute the finish set as follows. First of all, if the principal relocation is not a full analogy class, but just a subset of one, then the walkers know that it could not possibly be their initial position on the SEC (see Fig. 8(b)). In this case, the finish set is defined to be the principal relocation itself. The reason is that, by moving to their principal relocation, the walkers all join some bigger analogy class: this is not an "ineffective" move, because it makes progress toward having a unique analogy class.

On the other hand, if the principal relocation forms in fact an analogy class, then the walkers assume that to be their original position on the SEC. Hence they compute the new targets based on that configuration, with the usual algorithm (see Fig. 8(c)). Now, if the walkers can reach their respective targets from inside SEC/3 (that is, without becoming co-radial with other robots), then the finish set is the set of their targets. Otherwise, the walkers are confused, and by default their finish set is the principal relocation again.

Now that the finish set has been defined, the robots move there, always remaining within SEC/3, and without becoming co-radial with each other. There is only one exception: suppose that the walkers reach their finish set and move radially to the perimeter of the SEC: let R be the set of the final positions of

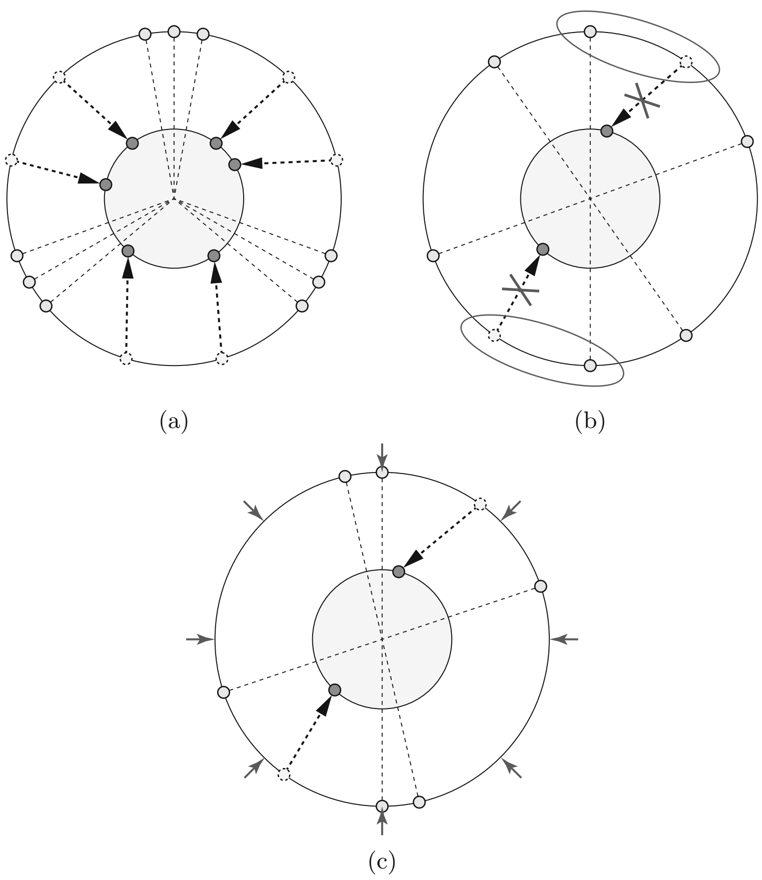


Fig. 8. (a) The principal relocation of the internal robots. (b) If the principal relocation is a proper subset of an analogy class, it cannot be the original position of the internal robots, or else a larger set of walkers would have been selected. (c) If the principal relocation forms an analogy class, it is used to determine the target set. Such targets remain fixed as the internal robots move within their respective sectors. (Source: [12])

the walkers on the SEC. If the new configuration is locked, and the robots in R happen to form an unlocking analogy class, then it was not a correct move for the walkers to go to R. Indeed, this would cause them to become walkers again (unless there are two unlocking analogy classes and the other one is chosen), and the execution would enter an infinite loop. In this special case, the walkers have to do something to unlock the configuration, instead of reaching R. The strategy is as follows: if the walkers are two, they move to two antipodal points (as in Fig. 6(b)); if there is a unique walker, it becomes antipodal with some non-movable robot currently located on the SEC. Note that this type of move would not be possible in a *Half-disk* configuration: this is precisely why the algorithm makes sure to resolve *Half-disk* configurations early on.

Lemma 8 **([12]).** *Suppose that an unlocking analogy class of walkers is chosen in a locked configuration, and said walkers move to their destinations. Then, the resulting configuration is not locked, and all its analogy classes are movable.*

2.9 Accidental Formation of *Pre-regular* Configurations

The algorithm has still one unresolved issue. Recall that, every time a robot computes a new destination, it first checks if the configuration is *Pre-regular*. If it is, it executes the special protocol given in Sect. 2.1; otherwise it proceeds normally. So, let us consider what happens if the swarm is executing the non-*Pre-regular* protocol, and suddenly a *Pre-regular* configuration is formed "by accident". If a robot happens to perform an observation right at that time, it is going to execute the *Pre-regular* protocol, while all the other robots are still executing the other one, and maybe they are in the middle of a move (see Fig. 9(a)). This leads to an incoherent behavior that will likely disrupt the "flow" of the entire algorithm.

To resolve this issue, we have to avoid the unintended formation of *Pre-regular* configurations whenever possible. If in some cases it is not easily avoidable, then we have to make sure that the whole swarm stops moving, or *freezes*, whenever a *Pre-regular* configuration is formed. This way, all robots will transition into the new configuration, and all of them will coherently execute the *Pre-regular* protocol in their next cycles.

Fortunately, certain configurations are safe:

Lemma 9 **([12]).** *No* Central *or* Co-radial *or* Half-disk *configuration can be* Pre-regular.

So, in these initial phases, no *Pre-regular* configuration can be formed accidentally. Another important observation is the following:

Lemma 10 **([12]).** *In a* Pre-regular *configuration, no robot can be in SEC/3.*

This explains why we make our walkers move radially to SEC/3 first, and we allow them to move laterally only within SEC/3.

Hence, the only moves that have to be analyzed are the radial ones, which are performed by the walkers between SEC and SEC/3 or by the robots that are reaching the perimeter of the SEC during the preliminary step. We can conveniently simplify the problem if we let only one analogy class of robots move at a time. Note that this is already the case when the moving robots are the walkers, and in the other cases there is always a way to totally order the analogy classes unambiguously. If only one analogy class is moving radially, it is then easier to keep the swarm's behavior under control and analyze all possible outcomes.

The general protocol that is used for radial moves is called *cautious move*, and is described next.

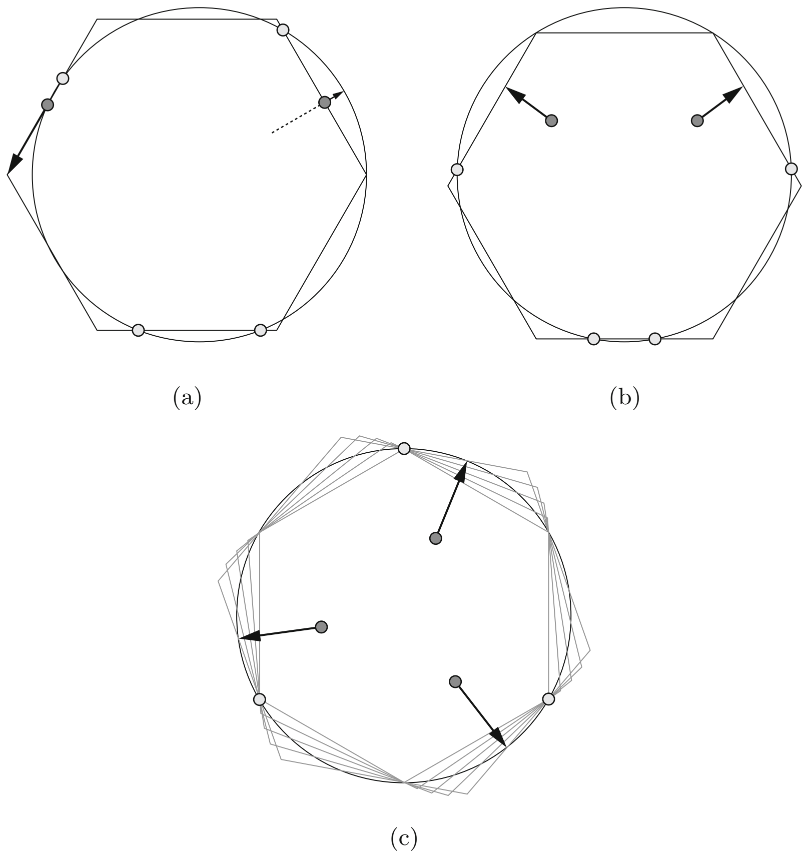


Fig. 9. (a) As the robot on the right moves to the SEC, a *Pre-regular* configuration is accidentally formed. The robot on the left recognizes a *Pre-regular* configuration, and starts executing the corresponding protocol, which is inconsistent with the other robot's move. (b) To prevent this behavior, enough critical points are added. Now the swarm is guaranteed to stop as soon as a *Pre-regular* configuration is formed. (c) A case in which infinitely many *Pre-regular* configurations are formable. Still, only the innermost is relevant, because it can be reached before all the others. (Source: [12])

2.10 Cautious Moves

In a cautious move, there is an analogy class of robots that have to move radially, either all from SEC/3 to SEC or all from SEC to SEC/3, while the other robots wait. Each moving robot has a final destination and is given as input a finite set of *critical points* along its path. Collectively, the moving robots execute a protocol that makes them move in such a way as to freeze whenever they are all located at a critical point (see for instance Fig. 9(b))[1].

[1] A similar concept has been used in [3], with some technical differences.

The procedure first augments the set of input critical points with a finite set of "auxiliary" critical points, and then lets a robot move toward the next critical point (auxiliary or not) along its path, provided that some conditions are met. The details are as follows.

- The endpoint of each robot's path is added to the set of critical points.
- For every robot r and every critical point p lying on any other robot's path, a critical point is added on r's path at the same distance from the center of the SEC as p.
- Then, for each pair of consecutive critical points on each robot's path, the midpoint is added as a critical point.
- The robots that are not farthest from the endpoints of their respective paths are not allowed to start moving.
- The robots that are farthest from the endpoints of their respective paths move to the next critical point along their respective paths.

Lemma 11 **([12]).** *If the robots execute the cautious move protocol from a frozen initial configuration, they either reach their final destinations or they freeze in a configuration where all of them are in a critical point.*

(Recall that a configuration is said to be *frozen* if no robot is moving.)

So, if the potentially formable *Pre-regular* configurations are used to generate the critical points of a cautious move, it is indeed guaranteed that the robots will freeze as soon as they form one. This is not always possible, because the formable *Pre-regular* configurations may be infinitely many (as in Fig. 9(c)), while the critical points must be finite. However, it can be shown that, in all cases, either there is a finite number of *Pre-regular* configurations that will be formed before all the others, or suitable critical points can be chosen in such a way as to prevent the formation of *Pre-regular* configurations altogether. Hence, it turns out that it is always possible to choose a finite set of critical points for all cautious moves, and guarantee that the swam is frozen whenever it transitions into a *Pre-regular* configuration.

Lemma 12 **([12]).** *Let an analogy class of robots perform a radial cautious move from SEC to SEC/3 or vice versa, with suitable critical points. Then, either all robots reach their destinations, or they freeze in a* Pre-regular *configuration.*

2.11 Correctness of the Algorithm

All the elements of the `Uniform Circle` Formation algorithm have been presented. When a robot executes the algorithm, it determines the current configuration type and executes the corresponding procedure to compute a destination point. Observe that some configurations fall in more than one category (for instance, the *Central* configurations are also *Co-radial*), and so the order in which such categories are tested matters. The order is the following:

- *Regular*,
- *Pre-regular*,
- *Central*,
- *Half-disk*,
- *Co-radial*,
- *Valid*,
- *Invalid*.

All these classes have already been defined, except *Invalid*, which is the set of configurations that do not fall in any other class. In an *Invalid* configuration, all robots simply perform a cautious move toward the perimeter of the SEC.

The correctness of the *Pre-regular* case of the algorithm, as well as the *Central*, *Co-radial*, and *Half-disk* cases is relatively straightforward to prove. Other parts of the algorithm, however, need a more careful analysis: these include a characterization of the locked configuration and the determination of critical points in every configuration where a radial move is made, in order to avoid the accidental formation of *Pre-regular* configuration.

These theoretical tools allow to finally tackle the *Valid* case, and so analyze the main "loop" of the algorithm. It can be shown that the different phases of the execution "hinge together" as intended: all the walkers reach SEC/3 and freeze there (unless a *Pre-regular* configuration is formed in the process), then they all move to their finish set, freeze again, and finally they move back to the perimeter of the SEC. As the execution continues and more iterations of this phase are made, it is necessary to study how the target set changes, in order to make sure that a *Pre-regular* configuration is eventually formed.

To this end, it can be proven that, at each iteration, some "progress" is made toward a *Regular* or *Biangular* configuration. This could mean that the walkers join another analogy class (thus reducing the total number of analogy classes), or that a new axis of symmetry is acquired, or that more robots become satisfied. Of course the configuration may also be locked: in this case it can be proved that, after one iteration, either the configuration is no longer locked, or some analogy classes have merged, or a previously non-movable analogy class has become movable.

Also, by design, the algorithm never allows an analogy class to split (because the walkers constitute an analogy class when they are selected, and are again all analogous when they reach their finish set), and it never causes a symmetric configuration to become asymmetric from one iteration to the next. However, it is true that the targets may change, and thus the number of satisfied robots may actually decrease. However, this can happen only when some analogy classes merge or when the configuration becomes symmetric, and thus it can happen only finitely many times.

So, either a *Pre-regular* configuration is formed by accident (and this case leads to a quick resolution), or eventually there will be only one analogy class left, and hence the configuration will be *Regular* or *Biangular*. Figure 10 shows the possible transitions between configuration types that the algorithm allows. Observe that every possible flow ends in a *Regular* configuration.

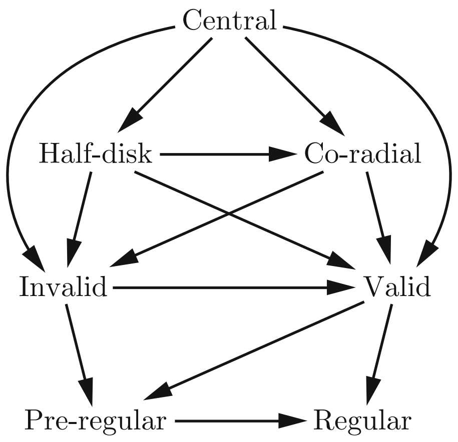


Fig. 10. Possible transitions between configurations in the `Uniform Circle` Formation algorithm. (Source: [12])

Theorem 1 **([12]).** *The* `Uniform Circle` *Formation Problem is solvable by $n > 5$ robots in the standard $\mathcal{OBLOT}$ model under the $\mathcal{A}$*SYNC *scheduler.*

3 Special Algorithms for $n \leq 5$ Robots

For small values of n, i.e., $n \leq 5$, the `Uniform Circle` Formation algorithm of Sect. 2 fails, and ad-hoc algorithms have been designed for these cases. If $n \leq 3$ the problem is relatively simple, and the case $n = 5$ is obtained by modifying the general algorithm of Sect. 2. The case $n = 4$, on the other hand, requires a special treatment.

3.1 $n \neq 4$ Robots

If $n = 1$ or $n = 2$, the pattern `Uniform Circle` is automatically formed, and nothing has to be done. If $n = 3$, the algorithm is as follows:

- if the three distances between pairs of robots are all distinct and robots r_1 and r_2 are farthest apart, then robot r_3 moves parallel to r_1r_2 toward the axis of r_1r_2;
- otherwise, if $r_1r_3 = r_2r_3$, then r_3 moves to the closest point that forms an equilateral triangle with r_1 and r_2 (in case there are two such points, one is chosen arbitrarily).

The analysis of this algorithm is straightforward.

Theorem 2 **([12]).** *The* `Uniform Circle` *Formation Problem is solvable by $n \leq 3$ robots in the standard $\mathcal{OBLOT}$ model under the $\mathcal{A}$*SYNC *scheduler.*

If $n = 5$, it turns out that the algorithm of Sect. 2 works as it is, except in one situation: there are locked configurations where an unlocking analogy class cannot be found in the usual way, because all the classes that are adjacent to some unmovable class are satisfied. This cannot happen if $n > 5$, and in this type of configuration the algorithm is undefined.

The algorithm can be modified to encompass these cases as follows. Suppose that the configuration is *Valid*, with all robots on the perimeter of the SEC, and without axes of symmetry. Two robots are said to be *antipodal* if they occupy antipodal points of the SEC.

– If no two robots are antipodal, a movable robot is chosen unambiguously, it becomes a walker, and moves to become antipodal with some other robot. It can be shown that there exists one such robot that can complete its movement without becoming co-radial with any other robot.
– If exactly two robots are antipodal, there is a unique robot that is adjacent to both of them (recall that a *Valid* configuration cannot be *Half-disk*). Such a robot is movable, and it becomes the walker and moves to become antipodal with another robot.
– If two pairs of robots are antipodal, the non-antipodal robot is movable, it becomes the walker, and moves to the midpoint of its adjacent robots, thus creating an axis of symmetry.

When the configuration has an axis of symmetry ℓ, since $n = 5$ is odd, there must be a robot r on ℓ. If ℓ is not unique, the configuration is *Regular*; so, assume that ℓ is unique. If the two analogous robots s and s' that are farthest from r are not satisfied, the other two robots q and q' make a preliminary move to become antipodal (on the diameter of the SEC that is perpendicular to ℓ), thus making s and s' able to move to their targets. When s and s' are satisfied, q and q' move to their targets to form a *Regular* configuration.

Observe that, since $n = 5$ is an odd number, no *Pre-regular* configuration can be formed, and thus the cautious moves require no ad-hoc critical points.

Theorem 3 ([12]). *The* `Uniform Circle` *Formation Problem is solvable by* $n = 5$ *robots in the standard* $\mathcal{OBLOT}$ *model under the* $\mathcal{A}$SYNC *scheduler.*

3.2 $n = 4$ Robots

The case with $n = 4$ robots presents difficulties that make it unique, and has been approached with radically different techniques. This case has been resolved in [15].

When trying to apply the algorithm of Sect. 2 to $n = 4$ robots, it is immediate to recognize that the *Biangular* configurations are now rectangular, and for such configurations the supporting polygon defined in Sect. 2.1 is not a unique square, but there are infinitely many of them. Even if the robots tried to implicitly agree on one such square with a common criterion, the square would shift as the robots move asynchronously to its vertices, yielding a convergence algorithm at best (as opposed to a formation algorithm).

The approach adopted in [15] is, roughly speaking, to "tilt" the supporting square by 45°. That is, a square is found such that each robot lies on a distinct edge of it (or on the extension of an edge), and the target of each robot is the midpoint of the edge on which it lies. Once again there is more than one such square, but the construction in Fig. 11 unambiguously produces one.

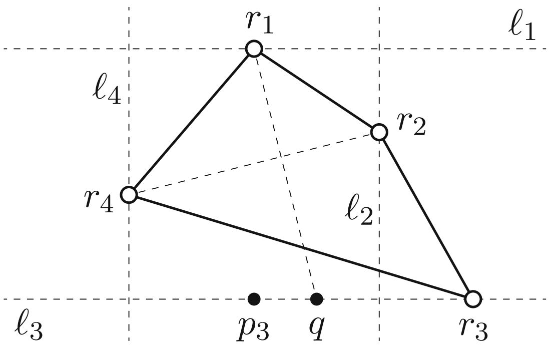


Fig. 11. Construction of the supporting square. (Source: [15])

Let $r_1r_2r_3r_4$ be a strictly convex quadrilateral whose diagonals r_1r_3 and r_2r_4 are not perpendicular. Let q be the unique point such that $r_1q = r_2r_4$, the lines r_1q and r_2r_4 are perpendicular, and the ray emanating from r_1 and passing through q intersects the line r_2r_4. Let ℓ_3 be the line through r_3 and q, and let ℓ_1 be the line through r_1 parallel to ℓ_3. (Since r_1r_3 and r_2r_4 are not perpendicular, ℓ_3 is well defined and is distinct from ℓ_1.) Let ℓ_2 be the line through r_2 perpendicular to ℓ_1, and let ℓ_4 be the line through r_4 parallel to ℓ_2. By construction, these four lines intersect at four points that are vertices of a square Q, the supporting square. In turn, the midpoints of the edges of Q form a second square Q', whose vertices are the target points of the robots. Referring to Fig. 11, the target of r_3 is p_3, and the segment r_3p_3 is called r_3's *pathway*, etc.

Lemma 13 ([15]). *All robots agree on the same supporting square, which remains fixed as all robots move to their target points asynchronously. Moreover, no two robots' pathways intersect.*

As the above construction assumes that the robots form a convex quadrilateral with non-perpendicular diagonals, special protocols are needed in these cases. If the quadrilateral is non-convex, there is a unique robot that is contained in the triangle formed by the other three: this robot moves to the foot of an altitude of such triangle, thus forming a loosely convex configuration with perpendicular diagonals. On the other hand, whenever the quadrilateral is (loosely) convex and its diagonals are perpendicular, then the robots that are closest to the intersection point of the diagonals move away from it until the configuration becomes a square.

The above protocol has yet one exception: it does not apply when all four robots are on the same line. In this case, the two non-extremal robots move to either side of the line by a small amount. As they move asynchronously in this fashion, their supporting square changes wildly, and so a "safe region" has to be defined in which the robots do not rely on their supporting square but follow a different protocol. The safe region has the form of a *thin hexagon*, depicted in Fig. 12. The proportions of the hexagon are carefully chosen for reasons that will be clear later.

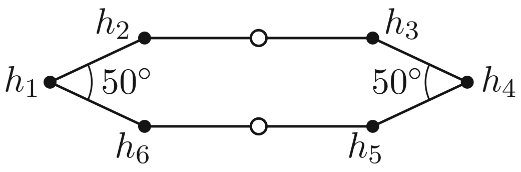


Fig. 12. Thin hexagon: the angles at h_1 and h_4 are 50°, and $h_1h_4 = 4 \cdot h_1h_2$. (Source: [15])

If there are robots in h_1 and h_4 and two internal robots on different sides of the diagonal h_1h_4, then the internal robots move to the white dots of Fig. 12, turning the configuration into one with perpendicular diagonals. If the internal robots are on the same side of the diagonal h_1h_4, say above it in Fig. 12, then they move to the vertices h_2 and h_3 respectively, and they wait for each other. When they both arrive and stop, the supporting square can be computed without ambiguity, and the protocol can safely switch to the normal one, which makes all robots move to their target points on the supporting square.

Of course, if all robots asynchronously move toward their target points, they may accidentally form configurations that are not strictly convex or have perpendicular diagonals or are thin hexagons, much like they could accidentally form *Pre-regular* configurations in the algorithm of Sect. 2. Again, the issue is resolved by limiting the number of robots that move concurrently and by employing a concept of critical point and cautious move similar to that of Sect. 2.10.

To define the critical points, some definitions are needed. Two robots are *concordant* if they have to move around the center of the supporting square in the same "direction" (i.e., clockwise or counterclockwise) as they go toward their targets. They are *discordant* if they move in opposite directions, and a robot is said to be *finished* when it is on its target point. Let the guidelines of two discordant robots r_i and r_j intersect in a point g. If p_i lies on the segment r_ig and p_j lies on the segment r_jg, then r_i and r_j are said to be *convergent*; otherwise, they are *divergent* (if both r_i and r_j are finished, they are both convergent and divergent). If the pathway of r_i intersects the segment r_jr_k in v, then r_i is said

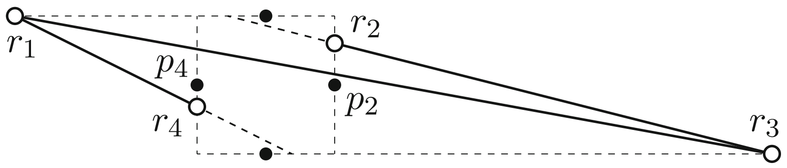


Fig. 13. Robot r_2 is blocked; robots r_1 and r_3 are hindered. (Source: [15])

to be *blocked* at v. If the pathway of r_i intersects an extension of the segment r_jr_k in v, then r_i is said to be *hindered* at v (see Fig. 13).

Of course, the points v defined above, which make a robot blocked or hindered, constitute natural critical points for the robots' movements, as they determine transitions to special classes of configurations (i.e., not strictly convex or with perpendicular diagonals).

Whenever possible, the algorithm makes only one robot move while the others wait. In this case asynchrony is not an issue, and the robot will either move to its target or to the first critical points on its pathway. An example of such a situation is when one robot is discordant with the other three: then, the discordant one will move before the other three. Another example is when all robots are concordant: in this case, the two robots on opposite sides of the supporting square that are closest to each other will move (as they move toward their targets, their distance will keep being the shortest). It can be proven that these two robots cannot be both blocked or both hindered. Thus, if one of them is blocked, it is chosen as the only robot to move; if none of them is blocked but one is hindered, it is the one to move. In all situations in which a single robot cannot be distinguished in a robust way, it is always possible to choose two robots to move while the other two robots wait. Suitable critical points also exist for these situations.

A notable fact is that, when looking for critical points, in almost all cases it is unnecessary to take into account the accidental formation of thin hexagon configurations. Indeed, in most of the relevant configurations, if the robots to move are selected properly, no thin hexagon can ever be formed, as Fig. 14 exemplifies. This is due to the proportions of the thin hexagon detailed in Fig. 12, which have been chosen specifically for this purpose. For instance, suppose that robots r_1 and r_2 are divergent and robots r_3 and r_4 are divergent, as well. Further assume that r_1 is the only robot that does not lie on an edge of the supporting square, but on the extension of it: in this configuration, r_1 is chosen to move toward its target. It is easy to verify that, if r_2 is the robot closest to r_1, then the two angles $\angle r_1r_3r_4$ and $\angle r_1r_4r_3$ are both greater than $25°$. Hence, if r_1 is on an acute vertex of a thin hexagon and r_3 (respectively, r_4) is on the opposite vertex, then r_4 (respectively, r_3) cannot be contained in the same thin hexagon.

As in Sect. 2, for the algorithm to work properly, it is necessary to identify to which class the observed configuration belongs, by checking them one by one in the correct order, because some classes have intersections. The order is the following:

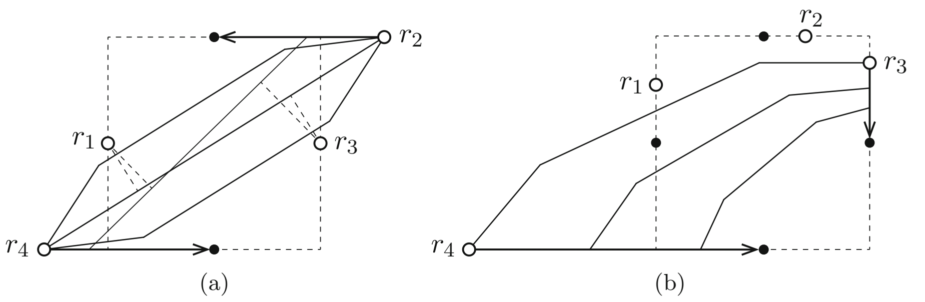


Fig. 14. (a) If r_1 and r_3 are on their targets and r_2 and r_4 move, no thin hexagon is formed. (b) As r_3 and r_4 move, no thin hexagon is formed. (Source: [15])

- Perpendicular diagonals,
- Thin hexagon,
- Non-convex,
- All robots concordant,
- Two robots convergent, two other robots divergent,
- Two robots divergent, two other robots divergent,
- Three robots concordant, one discordant.

To each class corresponds a different set of actions. The algorithm is designed in such a way that, whenever a class transition occurs, no robot is moving: thus, all robots will "witness" the transition and will coherently execute the instructions for the new class. Also, when the configuration transitions from class A to class B, the class B comes before A in the list above: this ensures that some progress toward the solution is always made. The last rule has only one exception: when all robots are on the same side of a thin hexagon, they will eventually be found on four consecutive vertices of it. This configuration is then interpreted as one with two pairs of divergent robots, which is resolved by the normal algorithm without possibility of forming a thin hexagon again.

We conclude that `Uniform Circle` is formable by $n = 4$ robots, as well.

Theorem 4 (**[15]**)**.** *The* `Uniform Circle` *Formation Problem is solvable by* $n = 4$ *robots in the standard* $\mathcal{OBLOT}$ *model under the* $\mathcal{A}$SYNC *scheduler.*

References

1. Bhagat, S., Mukhopadhyaya, K.: Optimum circle formation by autonomous robots. In: Chaki, R., Cortesi, A., Saeed, K., Chaki, N. (eds.) Advanced Computing and Systems for Security. AISC, vol. 666, pp. 153–165. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-8180-4_10
2. Chatzigiannakis, I., Markou, M., Nikoletseas, S.: Distributed circle formation for anonymous oblivious robots. In: Ribeiro, C.C., Martins, S.L. (eds.) WEA 2004. LNCS, vol. 3059, pp. 159–174. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24838-5_12

3. Cieliebak, M., Flocchini, P., Prencipe, G., Santoro, N.: Distributed computing by mobile robots: gathering. SIAM J. Comput. **41**(4), 829–879 (2012)
4. Défago, X., Konagaya, A.: Circle formation for oblivious anonymous mobile robots with no common sense of orientation. In: Proceedings of the ACM International Workshop on Principles of Mobile Computing (POMC), pp. 97–104 (2002)
5. Défago, X., Souissi, S.: Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity. Theor. Comput. Sci. **396**(1–3), 97–112 (2008)
6. Dieudonné, Y., Labbani-Igbida, O., Petit, F.: Circle formation of weak mobile robots. ACM Trans. Auton. Adapt. Syst. **3**(4), 16:1–16:20 (2008)
7. Dieudonné, Y., Petit, F.: Swing words to make circle formation quiescent. In: Prencipe, G., Zaks, S. (eds.) SIROCCO 2007. LNCS, vol. 4474, pp. 166–179. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72951-8_14
8. Dieudonné, Y., Petit, F.: Squaring the circle with weak mobile robots. In: Hong, S.-H., Nagamochi, H., Fukunaga, T. (eds.) ISAAC 2008. LNCS, vol. 5369, pp. 354–365. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-92182-0_33
9. Feletti, C., Mereghetti, C., Palano, B.: Uniform circle formation for swarms of opaque robots with lights. In: Izumi, T., Kuznetsov, P. (eds.) Stabilization, Safety, and Security of Distributed Systems. LNCS, vol. 11201, pp. 317–332. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03232-6_21
10. Flocchini, P., Prencipe, G., Santoro, N.: Self-deployment algorithms for mobile sensors on a ring. Theor. Comput. Sci. **402**(1), 67–80 (2008)
11. Flocchini, P., Prencipe, G., Santoro, N., Viglietta, G.: Distributed computing by mobile robots: solving the uniform circle formation problem. In: Aguilera, M.K., Querzoni, L., Shapiro, M. (eds.) OPODIS 2014. LNCS, vol. 8878, pp. 217–232. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-14472-6_15
12. Flocchini, P., Prencipe, G., Santoro, N., Viglietta, G.: Distributed computing by mobile robots: uniform circle formation. Distrib. Comput. **30**(6), 413–457 (2017)
13. Fujinaga, N., Yamauchi, Y., Kijima, S., Yamashita, M.: Asynchronous pattern formation by anonymous oblivious mobile robots. SIAM J. Comput. **44**(3), 740–785 (2015)
14. Katreniak, B.: Biangular circle formation by asynchronous mobile robots. In: Pelc, A., Raynal, M. (eds.) SIROCCO 2005. LNCS, vol. 3499, pp. 185–199. Springer, Heidelberg (2005). https://doi.org/10.1007/11429647_16
15. Mamino, M., Viglietta, G.: Square formation by asynchronous oblivious robots. In: Proceedings of the 28th Canadian Conference on Computational Geometry (CCCG), pp. 1–6 (2016)
16. Miyamae, T., Ichikawa, S., Hara, F.: Emergent approach to circle formation by multiple autonomous modular robots. J. Robot. Mechatron. **21**(1), 3–11 (2009)
17. Mondal, M., Chaudhuri, S.G.: Uniform circle formation by mobile robots. In: Proceedings of the Workshop Program of the 19th International Conference on Distributed Computing and Networking (ICDCN), pp. 20:1–20:2 (2018)
18. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: formation of geometric patterns. SIAM J. Comput. **28**(4), 1347–1363 (1999)