Robots with Lights

Giuseppe Antonio Di Luna[1(✉)] and Giovanni Viglietta[2]

[1] Aix-Marseille Université, LIS, CNRS, Université de Toulon, Toulon, France
g.a.diluna@gmail.com
[2] JAIST, Nomi, Japan
johnny@jaist.ac.jp

Abstract. The classic Look-Compute-Move model of oblivious robots has many strengths: algorithms designed for this model are inherently resistant to a large set of failures that can affect the memory of the robots and their communication capabilities.

However, modern technologies allow for cheap and reliable means of communication and memorization. This is especially true if relatively low performances are needed, such as very limited communication bandwidth or constant memory. A theoretical model that expands the classic Look-Compute-Move by adding a minimal ability to communicate and remember is the model of robots with lights. In this model each robot carries a luminous source that it can modify at every cycle. The robot decides the color of its light during its Compute phase, and the light assumes such a color at the beginning of the next Move phase. Other robots can see the color of this light during their Look phases. The light will remain unaltered until the robot that carries it decides to change its color.

Typically, the number of available colors is very limited, i.e., it is constant with respect to the number of robots in the system.

In this chapter we will discuss the hierarchy of $\mathcal{F}$SYNC, $\mathcal{S}$SYNC, and $\mathcal{A}$SYNC models when lights are present, we call this model $\mathcal{LUMINOUS}$. Moreover, we will see how lights are applied to solve classic problems such as rendezvous and forming a sequence of patterns. Finally, we will see how lights have been exploited in models where the visibility of robots is limited by the presence of obstructions.

1 Introduction

In the classic Look-Compute-Move model of oblivious robots, the absence of persistent memory and explicit communication ensures that any algorithm for such weak model can be implemented in a wide range of harsh scenarios where communicating is not a reliable option, e.g., a hostile environment where communication jamming is a possibility. Moreover, algorithms designed for this model are inherently resistant to a large set of failures that can affect the memory of the robots and their communication capabilities.

Fortunately, modern technologies allow for cheap and reliable means of communication and storage. This is especially true if relatively low performances are needed, such as very limited communication bandwidth or constant memory.

P. Flocchini et al. (Eds.): Distributed Computing by Mobile Entities, LNCS 11340, pp. 252–277, 2019.
https://doi.org/10.1007/978-3-030-11072-7_11

A theoretical model that expands the classic `Look-Compute-Move` model by adding a minimal ability to communicate and remember is the model of robots with lights [7,8]. In this model each robot carries a luminous source that it can modify at every cycle. The robot decides the color of its light during its `Compute` phase, and the light assumes such a color at the beginning of the next `Move` phase. Other robots can see the color of this light during their `Look` phases. The light will remain unaltered until the robot that carries it decides to change its color.

Typically, the number of available colors is very limited, i.e., it is constant with respect to the number of robots in the system. Interestingly, using light to communicate is something feasible in the real world, and it has been used to implement real communication channels [19].

Essentially, lights allow robots to perform communication. Moreover, in case an robot can see its own light, the light itself also serves as memory. The impact of using lights with a constant number of colors is drastic, and greatly increases the computational power of mobile robots.

In this chapter we will discuss the hierarchy of $\mathcal{F}$SYNC, $\mathcal{S}$SYNC, and $\mathcal{A}$SYNC models when lights are present, that is when robots are $\mathcal{LUMINOUS}$. Moreover, we will see how lights are applied to solve classic problems such as Rendezvous and forming a sequence of patterns. Finally, we will see how lights have been exploited in models where the visibility of robots is limited by the presence of obstructions.

Chapter Outline. The outline of the Chapter is the following. We start with Sect. 2; devoted to formalising the $\mathcal{LUMINOUS}$ model. Specifically, the section describes how the `Look-Compute-Move` model is modified to incorporate colored lights. This amounts to letting each robot decide the color of its own light at each cycle, and stipulating that the snapshots taken by robots also contain light information. After formalising the model in Sect. 3, we investigate the computational power of lights. We discuss the relationship between $\mathcal{F}$SYNC, $\mathcal{S}$SYNC, and $\mathcal{A}$SYNC when robots are endowed with lights, focusing on robots on plane and mentioning the difference for agents on graph. Section 4 studies the Rendezvous problem with lights. Rendezvous is unsolvable without lights, but it becomes solvable when lights are introduced. In Sect. 5 we discuss how $\mathcal{LUMINOUS}$ robots can form a sequence of patterns. In Sect. 6 we study the problem of making obstructive robots mutually visible, showing how lights can be used to solve problems when the visibility is not unlimited. The Chapter terminates with the conclusive Sect. 7.

2 Model

When lights are introduced, the usual model of oblivious robots of Chap. 1, is extended in a natural way. Each robot carries a visible light, which at any time has a color chosen from a palette set $\mathcal{B}$. (The size of $\mathcal{B}$ will often be informally

referred to as the "number of lights".) In the basic model with lights, the palette set is the same for all robots, and it contains a special color *Off*, which is the color of every robot's light when the execution starts. Each robot can then freely alter the color of its own light, choosing it from $\mathcal{B}$, at each `Compute` phase.

More specifically, the usual `Look-Compute-Move` phases are modified as follows.

- `Look`: In the `Look` phase, a robot r receives an instant snapshot, which is a multiset of pairs of the form (p_i, c_i), where p_i is a point in the plane and $c_i \in \mathcal{B}$ is a color. The meaning of such a pair is that, at the time the snapshot was taken, the robot r could see a robot s located in p_i carrying a light with color c_i. As in the usual model, p_i is the position of s as expressed in the local coordinate system of r.
  In the basic model with lights, robots have full visibility, and therefore snapshots always contain information about every robot in the system. When more restrictive visibility models are considered, such as the obstructed visibility model of Sect. 6 of this chapter, a snapshot may contain less information.
- `Compute`: In the `Compute` phase, a robot, using the snapshot obtained in the most recent `Look` phase, chooses a destination point and a new color in $\mathcal{B}$ for its own light.[1]
- `Move`: During the `Move` phase, a robot first changes the color of its light to the color chosen in the most recent `Compute` phase (this action is atomic, i.e., it is instantly executed), and then it proceeds to move to its destination according to the classic `Look-Compute-Move` model.

With $M\text{ODEL}^m$, we indicate the model $M\text{ODEL}$ where robots carry lights whose colors are chosen from a palette set of size $|\mathcal{B}| = m$. For example, with $\mathcal{F}\text{SYNC}^{O(1)}$ we indicate the $\mathcal{F}\text{SYNC}$ model with lights of a constant number of colors (constant with respect to the number of robots in the system). When the set of problems solvable in model $M\text{ODEL}$ is included in the set of problems solvable in $M\text{ODEL}'$ we write $M\text{ODEL} \subseteq M\text{ODEL}'$ (we use $\subset$ to indicate the strict inclusion).

3 Computational Power of Lights

The presence of lights have an impact on the relative computational power of the $\mathcal{F}$SYNC-$\mathcal{A}$SYNC-$\mathcal{S}$SYNC models[2]. When we refer to robots without lights we have that, for $\mathcal{OBLOT}$ there are problems that are solvable in $\mathcal{F}\text{SYNC}$ but not in $\mathcal{S}\text{SYNC}$ (e.g., the *Gathering* problem). However, between $\mathcal{S}\text{SYNC}$ and $\mathcal{A}\text{SYNC}$ the only known result is the trivial A*sync* $\subseteq$ S*sync*.

In this section we investigate what happens when a constant number of lights is available.

[1] Note that setting the light at the end of the compute phase is equivalent to set the light at the beginning of the move phase.

[2] We say $\mathcal{F}\text{SYNC}$ model as shorthand for the model of $\mathcal{OBLOT}$ (or $\mathcal{LUMINOUS}$) robots with $\mathcal{F}\text{SYNC}$ scheduler.

3.1 The Relationship Between $\mathcal{A}\text{SYNC}^{O(1)}$ and $\mathcal{S}\text{SYNC}$

[8] shows that $\mathcal{A}\text{SYNC}^{O(1)}$ is more powerful than $\mathcal{S}\text{SYNC}$, that is $\mathcal{S}\text{SYNC} \subset \mathcal{A}\text{SYNC}^{O(1)}$. This is done in two steps, first it is shown how to simulate any algorithm for $\mathcal{S}\text{SYNC}$ in $\mathcal{A}\text{SYNC}^{O(1)}$ using a constant number of lights. Then it is shown that there exists a problem solvable in $\mathcal{A}\text{SYNC}^{O(1)}$ but not in $\mathcal{S}\text{SYNC}$.

Simulating $\mathcal{S}\text{SYNC}$ in $\mathcal{A}\text{SYNC}^{O(1)}$. The simulator of [8] uses five colors: Trying (T), Waiting (W), Moving (M), Stopping (S), Finished (F), and five states, one for each different color.

The idea is to synchronize robots in a simulated cycle, namely the Mega-Cycle. During a Mega-Cycle, each robot executes one step of the simulated algorithm $\mathcal{A}$. A new Mega-Cycle starts only when all robots terminated the execution of the previous Mega-Cycle. Each Mega-Cycle is structured as follows. Initially, all robots have light T. A robot with light T once activated tries to simulate one activation of $\mathcal{A}$, in doing so it first checks if all other robots have color in T or S. In such case, the robot sets its color to M and it executes $\mathcal{A}$. Otherwise, if there is an robot with color M, the robot from color T switches to color W and it does nothing. Intuitively, this check avoid that a robot provides to $\mathcal{A}$ a snapshot containing robots that are moving, this is consistent with $\mathcal{S}\text{SYNC}$ model.

If a robot has color M and no robot in its snapshot has color T, then it goes to color S and it does nothing. An robot with color W goes back to T only if there is no robot with color M. A robot with color S goes to color F if all robots have color in $\{S, F\}$. A robot with color F goes to color T if all robots have color in $\{T, F\}$. The previous rules ensure that, in a Mega-Cycle, each robot executes exactly one activation of $\mathcal{A}$. It is also easy to see that all robots execute the Mega-Cycle: until there is some robot in W or T, there will be at least one robots that enters in M. A new Mega-Cycle starts when some robot goes from F to T. The state diagram of the simulator is shown in Fig. 1.

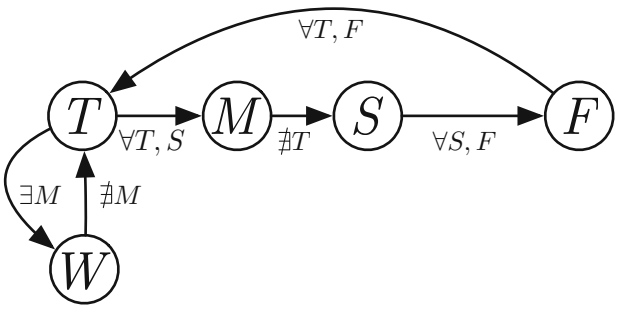


Fig. 1. State diagram of the simulator for $\mathcal{S}\text{SYNC}$ in $\mathcal{A}\text{SYNC}^{O(1)}$ of [8]. On each arrow there is label specifying a property of the snapshot that has to be verified to do the corresponding state transition. As example, the label between node T and node S is $\forall T, S$, that means: each robot in the snapshot is either in state T or S.

This simulator gives the following theorem:

Theorem 1 ([8]). $\mathcal{S}\text{SYNC} \subseteq \mathcal{A}\text{SYNC}^{O(1)}$.

The Additional Power of $\mathcal{A}\text{SYNC}^{O(1)}$. It is well known that Rendezvous of two oblivious anonymous robots cannot be solved in $\mathcal{S}\text{SYNC}$, see [26]. However, it is possible to create an algorithm that solves Rendezvous in $\mathcal{A}\text{SYNC}^{O(1)}$. The first paper showing this has been [8], where Rendezvous is solved using 4 colors. Other papers investigated the Rendezvous improving this first result in many directions. The Rendezvous will be the core of Sect. 4, the interested reader can refer to that Section.

The existence of several Rendezvous algorithms in $\mathcal{A}\text{SYNC}^{O(1)}$ and the Theorem 1, leads to:

Theorem 2 ([8]). $\mathcal{S}\text{SYNC} \subset \mathcal{A}\text{SYNC}^{O(1)}$.

3.2 The Relationship Between $\mathcal{A}\text{SYNC}^{O(1)}$ and $\mathcal{F}\text{SYNC}$

The relationship between $\mathcal{A}\text{SYNC}^{O(1)}$ and $\mathcal{F}\text{SYNC}$ is still not clear. It has been shown in [8] that there exist a problem solvable in $\mathcal{A}\text{SYNC}^{O(1)}$ and not in $\mathcal{F}\text{SYNC}$. The problem is the `Oscillating Points`.

In the `Oscillating Points` problem two robots, initially starting in distinct positions, have to move in such a way to alternate configurations in which their relative distance decreases (*near* configuration), and configurations in which their distance increases (*far* configuration). The intuitive reason why such problem is unsolvable in $\mathcal{F}\text{SYNC}$ is that the problem specification implicitly needs robots to remember weather they are in *near* or *far* configuration. This is not a problem when lights are present, being lights persistent a certain color can be associated with a specific configuration, in [8] 4 colors are used to solve `Oscillating Points`.

Theorem 3 ([8]). *The* `Oscillating Points` *problem is solvable in* $\mathcal{A}\text{SYNC}^{O(1)}$, *and is unsolvable in* $\mathcal{F}\text{SYNC}$.

However, it is still unknown if $\mathcal{F}\text{SYNC}$ can be simulated or not by $\mathcal{A}\text{SYNC}^{O(1)}$. Therefore, we do not know weather $\mathcal{F}\text{SYNC}$ is included in $\mathcal{A}\text{SYNC}^{O(1)}$, or weather the two models are orthogonal. Figure 2 shows the hierarchy of the models in the light of the results of [8].

The Power of Remembering. The relationship between $\mathcal{A}\text{SYNC}^{O(1)}$ and $\mathcal{F}\text{SYNC}$ is completely clear when we allows robots to remember the snapshot of the previous round (model $\mathcal{A}\text{SYNC}^{O(1)}$ + Snapshot). In this case, it is possible to simulate any $\mathcal{F}\text{SYNC}$ algorithm, see [8]. The interesting fact is to contrast this with what happens when lights are not available. In [26] it has been shown that $\mathcal{A}\text{SYNC}$ is weaker than $\mathcal{F}\text{SYNC}$ even when robots remember an unlimited amount of snapshots.

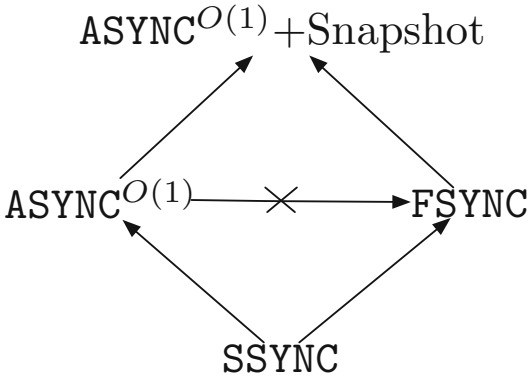


Fig. 2. Hiearchy of $\mathcal{A}\text{SYNC}^{O(1)}$, $\mathcal{F}\text{SYNC}$ and $\mathcal{S}\text{SYNC}$. An arrow indicates that the source model is included in the destination. A strikethrough arrow indicates that the source model is not included in the destination.

Robots on Graph with Lights. In [10] is investigated the relationship between $\mathcal{A}\text{SYNC}^{O(1)}$ and $\mathcal{F}\text{SYNC}$ when robots move in a discrete environment, that is modelled as a graph. Robots operate following the `Look-Compute-Move` cycle, and the snapshot is the entire graph. Two new problems are introduced to show the separation of $\mathcal{F}\text{SYNC}$ and $\mathcal{A}\text{SYNC}^{O(1)}$: the `Pattern Series Chasing` (solvable in $\mathcal{F}\text{SYNC}$ and not in $\mathcal{A}\text{SYNC}^{O(1)}$) and the `Forth and Back` (solvable in $\mathcal{A}\text{SYNC}^{O(1)}$ but not in $\mathcal{F}\text{SYNC}$). This implies that, when robots on graph are considered, $\mathcal{A}\text{SYNC}^{O(1)}$ and $\mathcal{F}\text{SYNC}$ are orthogonal models.

4 Rendezvous

Rendezvous is the special case of Gathering (see Chap. 4, Sect. 2) where the system consists of exactly two robots whose task is to move to the same point, no matter where and when, and then stop forever. This special case is surprisingly hard, due to the lack of "environmental landmarks" that may help the two robots agree on a common rendezvous point. In contrast, when more than two robots are present, it is relatively easy in most cases to implicitly agree on a small subset of robots that should gather first, while the others provide a visible static reference frame that helps circumvent limitations such as asynchrony and non-rigidity[3].

As shown in [26], the Rendezvous problem is unsolvable in $\mathcal{S}\text{SYNC}$. Indeed, suppose the local reference frames of the two robots are oriented symmetrically: since they have symmetric views, they always compute symmetric destination points. As long as the destination points they compute are different, the scheduler activates them both and lets them move. Whenever they compute the same destination point (which must be their midpoint, by symmetry), the scheduler

[3] In a rigid model robots always reach the destination when performing the move. In a non-rigid model robots may be stopped before reaching the destination, however they travel of at least a fixed unknown $\delta > 0$.

activates only one of them. To guarantee fairness, every time this happens, the scheduler activates a different robot, alternating.

Theorem 4 ([26]). *Rendezvous is unsolvable in* $\mathcal{S}$SYNC.

One way to cope with this impossibility is to use robots with lights: the first solution to Rendezvous in this model is found in [8], and was later improved in several directions. In the rest of this section, we will review some approaches to the Rendezvous problem for robots with lights.

4.1 Rendezvous with Fewest Colors

The focus of [28] is solving the Rendezvous problem for robots with lights using the minimum number of colors. The problem is solved in a variety of models, which combine different schedulers ($\mathcal{F}$SYNC, $\mathcal{S}$SYNC, $\mathcal{A}$SYNC), rigidity, and self-stabilization. The concept of rigidity is defined in Chap. 1, Sect. 2.4, while self-stabilization (see [14]) is the additional requirement that the two robots solve Rendezvous regardless of their initial colors (as opposed to stipulating that their lights have a specific predefined color when the execution starts).

Figure 3 summarizes the results of [28]: there is a hierarchy of 12 models obtained by combining the aforementioned parameters in all possible ways (an asterisk indicates that the solution has to be self-stabilizing). The number in parentheses after each model indicates that there is an algorithm that solves Rendezvous under that model using that many colors.

Under a certain assumption, all the numbers in Fig. 3 are minimal. The assumption is that the robots cannot use information about their distance to compute their destination points, but can only compute it as a function of their respective colors. More precisely, if a robot is located in p and performs a `Look` when the other robot is in q, then the destination point it computes must be of the form $(1-\lambda)p+\lambda q$, where $\lambda \in \mathbb{R}$ is computed as a function of the two robots' lights at the moment the `Look` was performed. All the algorithms presented in [28] are of this kind, as well.

The left side of Fig. 3 is easy to obtain, because in non-rigid $\mathcal{F}$SYNC there is a trivial algorithm that solves Rendezvous even in the basic model (i.e., with only one color): the algorithm makes each robot move to the midpoint of their current locations. Even if movements are non-rigid, the robots either meet or approach each other by at least 2δ at every turn, hence meeting in a finite number of turns.

For non-rigid $\mathcal{S}$SYNC and rigid $\mathcal{A}$SYNC there is an algorithm that uses only two colors, namely A and B, shown in Fig. 4. In the case of non-rigid $\mathcal{S}$SYNC, the algorithm is also self-stabilizing.

Labels on arrows indicate the color that is seen on the other robot and the λ parameter of the resulting move, i.e., the destination of the next `Move` with respect to the position of the other robot. So, "0" stands for "do not move", "1/2" means "move to the midpoint", and "1" means "move to the other robot". Roughly speaking, the idea of this algorithm is to make the robots approach each

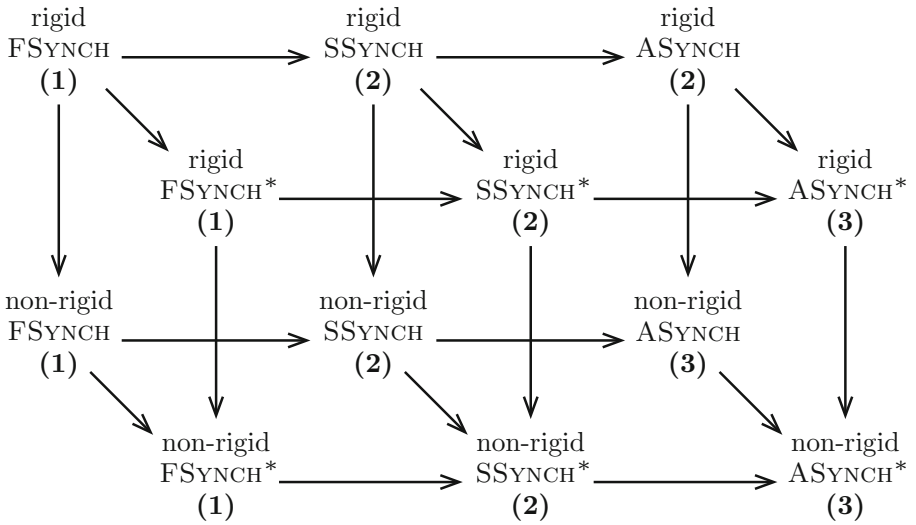


Fig. 3. Summary of results of [28]. For each model in the hierarchy, there exists a Rendezvous algorithm using the number of colors in parentheses. An asterisk indicates that the algorithm is self-stabilizing. If robots cannot use distance information in their computations, all these numbers are optimal.

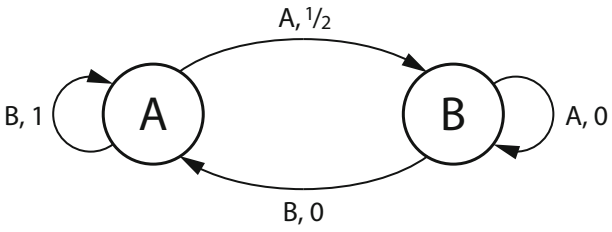


Fig. 4. Rendezvous algorithm from [28] for non-rigid $\mathcal{S}$SYNC (self-stabilizing) and rigid $\mathcal{A}$SYNC

other by moving toward the midpoint as long as their lights have the same color: if the scheduler keeps them synchronous, they eventually meet. If, on the other hand, the scheduler does not keep them synchronous, the two robots eventually see each other in different colors. Therefore the A-colored robot moves to the other robot's location, while the B-colored robot waits.

For non-rigid $\mathcal{A}$SYNC, the above algorithm fails. To see why, let r and s be the two robots, and let them both start with color A at distance greater than 2δ. If the scheduler lets them both perform an entire cycle but stops them as soon as they have moved by δ, they end up in color B a positive distance apart. Now, let both robots perform a `Look` phase, implying that both of them will eventually turn A. We let robot r finish the current cycle and perform a new `Look`, while the other robot s waits, still in color B. Hence, r will stay A and move to s's

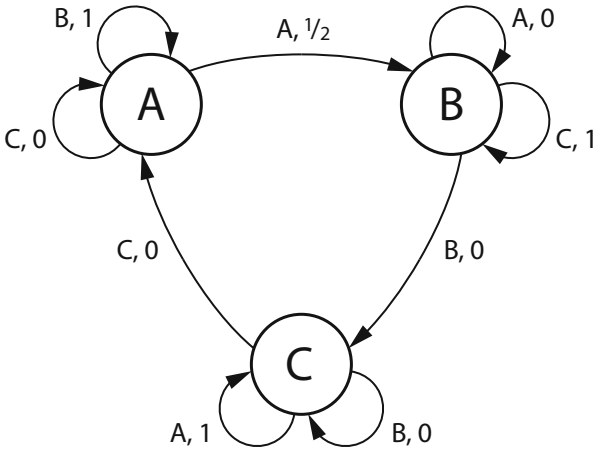


Fig. 5. Self-stabilizing Rendezvous algorithm from [28] for non-rigid $\mathcal{A}$SYNC

position. Now we let s finish the current cycle and perform a new `Look`. So s will turn B and move to the midpoint m. We let r finish the current cycle, thus reaching s, and perform a whole new cycle, turning B. Finally, we let s finish the current cycle, thus turning B and moving to m. As a result, both robots are again set to B, they are in a WAIT phase, both have executed at least one cycle, and their distance has halved. If the scheduler repeats the same pattern of activations, the robots will never gather.

Therefore, for non-rigid $\mathcal{A}$SYNC, the algorithm proposed in [28] uses three colors, A, B, C, and is self-stabilizing: see Fig. 5. The algorithm is an extension of that of Fig. 4, but its full analysis is somewhat technical.

Observe that the three algorithms outlined above are sufficient to establish all the color numbers indicated in Fig. 3. Indeed, due to the hierarchical structure of the models, if there is an arrow from model X to model Y in Fig. 3, then any algorithm for model Y also works for model X. The matching lower-bound proofs can be found in [28]. Summarizing, we have the following theorem.

Theorem 5 ([28])**.** *The Rendezvous problem is solvable in each of the 12 models of Fig. 3 using the number of lights indicated in parentheses under each model. If robots are not allowed to use distance information in their computations, these numbers are optimal.*

Finally, [28] shows how to refine the algorithm of Fig. 5 to detect termination: that is, to let the robots acknowledge that they have gathered, in order to turn off or "switch gears" and start performing a new task. Although this is not a requirement of the Rendezvous problem, it is a useful feature to add. The new algorithm still uses only three colors, but it also uses distance information, although robots only need distinguish between zero and non-zero distances.

Later improvements to [28] indicate that, if the robots are allowed to use distance information in their computations, they only require 2 colors to solve

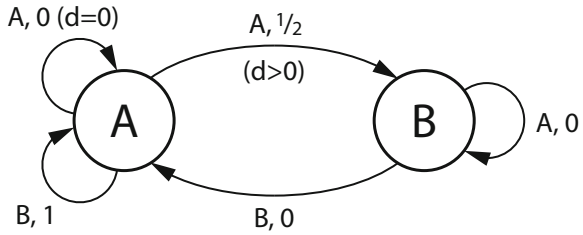


Fig. 6. Self-stabilizing Rendezvous algorithm from [18] for non-rigid $\mathcal{A}$SYNC using distance information

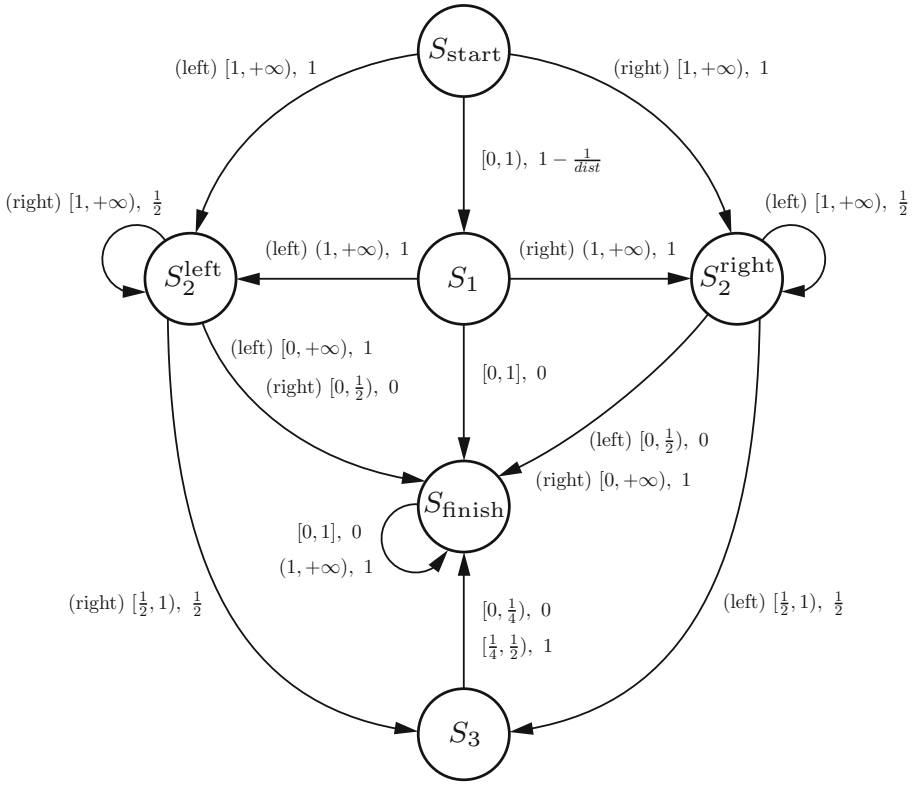


Fig. 7. Rendezvous algorithm from [16] for rigid $\mathcal{S}$SYNC robots in $\mathcal{F}$-STATE

Rendezvous, even under the non-rigid $\mathcal{A}$SYNC scheduler, and even in a self-stabilizing way. First, [20] proposed an algorithm that assumes the robots to know the value of the parameter δ related to non-rigid movements (see Chap. 1, Sect. 2.4). Then, [18] managed to drop even this assumption. The Rendezvous algorithm of [18] is shown in Fig. 6.

Observe that this algorithm is a simple modification of that of Fig. 4: the only difference is that, if a robot's color is A, it changes its color to B only if the other robot has color A and positive distance, $d > 0$ (i.e., if they have not gathered); otherwise, its color remains A. Although this algorithm uses distance information, it actually just needs to distinguish between zero and non-zero distances.

Theorem 6 ([18])**.** *The Rendezvous problem is solvable in a self-stabilizing way with 2 colors under the non-rigid* $\mathcal{A}$SYNC *scheduler, provided that distance information can be used in the computations.*

4.2 Rendezvous Under Weaker Light Models

Observe that visible lights offer a twofold advantage to robots: on one hand, a light serves as internal memory for the robot carrying it; on the other hand, it can be used to communicate information to other robots. In [16], these two aspects are decoupled, and two weaker light models are introduced: in the finite-state model ($\mathcal{F}$-STATE), each robot can see the color of its own light but not the color of the other lights; in the finite-communication model ($\mathcal{F}$-COMM), each robot can see the color of the other robots' lights, but not the color of its own light. In the case of a system with two robots, the latter model is equivalent to letting robots send each other messages and remember only the last received message (and be otherwise oblivious).

The Rendezvous problem is solved in [16] under these weaker light models. Specifically, if movements are rigid, the problem is solved with 6 colors in $\mathcal{F}$-STATE assuming the $\mathcal{S}$SYNC scheduler and with 12 colors in $\mathcal{F}$-COMM assuming the $\mathcal{A}$SYNC scheduler (no assumptions are made on the units of distance of the two robots, which may be different). If movements are non-rigid, the problem is solved in a self-stabilizing way with 3 colors in $\mathcal{F}$-COMM assuming the $\mathcal{S}$SYNC scheduler. If movements are non-rigid and, in addition, the robots know the value of δ, then the problem is solved with 3 colors in both $\mathcal{F}$-STATE assuming the $\mathcal{S}$SYNC scheduler and in $\mathcal{F}$-COMM assuming the $\mathcal{A}$SYNC scheduler (here, knowing δ implicitly gives the robots a common unit of distance).

The algorithm for $\mathcal{F}$-STATE robots in the rigid $\mathcal{S}$SYNC model is illustrated in Fig. 7, where circles denote the internal states of the two robots, and S_{start} is the initial state. An arrow with a label of the form $(d)I, \lambda$ denotes a state transition that applies when the other robot is seen in direction $d \in \{\text{left}, \text{right}\}$ and its observed distance lies in the interval $I \subset \mathbb{R}$. The parameter λ is as in Sect. 4.1, and defines the destination point with respect to the other robot's position. For example, a robot in state S_{start} perceiving the other at distance $\geqslant 1$ on the right will move to the position of the other robot and will change state to S_2^{right}. Note that a robot can arbitrarily assign a left and a right side to the line that connects it to the other robot, and this assignment does not change as the execution progresses.

According to the algorithm, the robots try to reach a configuration where they both observe each other at distance not smaller than 1 (i.e., their own unit

of distance). From this configuration, they attempt to meet in the midpoint. If they never meet because they are never activated in the same turn, eventually one of them notices that its observed distance is lower than 1. This implies a breakdown of symmetry that allows the robots to finally gather.

In order to reach the aforementioned desired configuration where they both observe a distance not smaller than 1, the two robots first move away from each other if they are too close. When they are far enough, they memorize the side on which they see each other (left or right), and try to switch positions. If only one of them is activated, they gather; otherwise they detect a side switch, and they can finally apply the above protocol, which leads to gathering.

This is complicated by the fact that the robots may disagree on the distances they observe, because they have different units of distance. To overcome this disagreement, they use their ability to detect a side switch to understand which distance their partner observed. If the desired configuration is not reached because of a disagreement, a breakdown of symmetry occurs, which is immediately exploited to gather anyway. As soon as the two robots are in the same location at the end of a cycle, they never move again, and Rendezvous is solved.

Theorem 7 ([16]). *The Rendezvous problem is solvable with 6 colors in* $\mathcal{F}$-STATE *under the rigid* $\mathcal{S}$SYNC *scheduler.*

Observe that the above algorithm makes a fundamental use of the fact that the scheduler is rigid and $\mathcal{S}$SYNC. For instance, the correct detection of a side switch by a robot relies on the fact that the other robot is not currently in the middle of a movement while it is observed (hence the scheduler is not $\mathcal{A}$SYNC), or it could be seen on a side and then switch side by the end of the current move. Similarly, the algorithm relies on the fact that the robots can reliably move away from each other and reach a distance not smaller than 1. In a non-rigid setting, they may be stopped too soon, in such a way that both end up in state S_1 but still detect a distance smaller than 1. From that point on, they will never move again, because each of them will incorrectly assume that the other robot will measure a distance greater than 1.

Now let us consider the $\mathcal{F}$-COMM model. The Rendezvous algorithm for $\mathcal{F}$-COMM robots in the rigid $\mathcal{A}$SYNC model is shown in Fig. 8, where the initial state of both robots is called "Test". The meaning of an arrow from state X to state Y is that if a robot observes that the light of the other robot has color X, then the first robot sets its own light to color Y. If an arrow has a label in the form of a predicate on d, it means that the transition only happens if the observed distance d between the two robots satisfies the predicate. Moreover, a boldface label λ on an arrow has the same meaning as in Sect. 4.1. If such a λ is followed by a predicate on the distance d in parentheses, the robot moves only if the predicate is satisfied, and stays still otherwise. For example, if a robot located in p sees the other robot located in q and in state "Both <1", and their distance d is positive, it assumes state "Moving Away". If, in addition, the distance d is less than 1, it also moves to the point $(1/2 + 1/d) \cdot p + (1/2 - 1/d) \cdot q$.

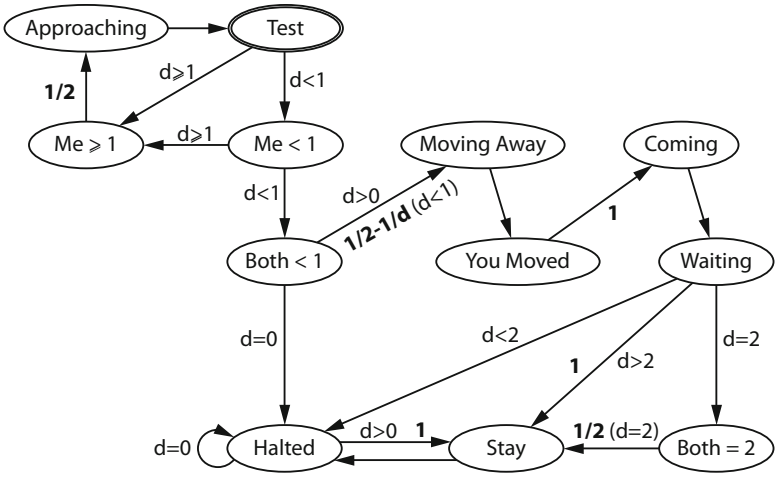


Fig. 8. Rendezvous algorithm from [16] for rigid $\mathcal{A}$SYNC robots in $\mathcal{F}$-COMM

According to the algorithm, the two robots try to reach a configuration where they both see each other at distance smaller than 1. To do so, they first communicate to each other whether or not the distance they observe is smaller than 1 (recall that they may disagree, because their units of distance may differ). If one robot acknowledges that its partner has observed a distance not smaller than 1, it reduces the distance by moving to the midpoint.

The process repeats until both robots observe a distance smaller than 1. At this point, if they have not gathered yet, they try to compare their units of distance in order to break symmetry. They move away from each other in such a way that their final distance is the sum of their respective units of distance. Before proceeding, they attempt to switch positions. If, due to asynchrony, they failed to be in the same state at any time before this step, they end up gathering. Instead, if their execution has been synchronous up to this point, they finally switch positions. Now, if the robots have not gathered yet, they know that their distance is actually the sum of their units. Because each robot knows its own unit, they can tell if one of them is larger. If a robot has a smaller unit, it moves toward its partner, which waits.

Otherwise, if their units are equal, they apply a straightforward protocol: as soon as a robot wakes up, it moves toward the midpoint and tells its partner to stay still. If both robots do so, they gather in the midpoint. If one robot is delayed due to asynchrony, it acknowledges the order to stay still and tells the other robot to come.

Theorem 8 ([16])**.** *The Rendezvous problem is solvable with 12 colors in* $\mathcal{F}$-COMM *under the rigid* $\mathcal{A}$SYNC *scheduler.*

Once again, the above algorithm crucially uses rigidity, for instance when the robots switch positions and assume that their current distance must be the sum

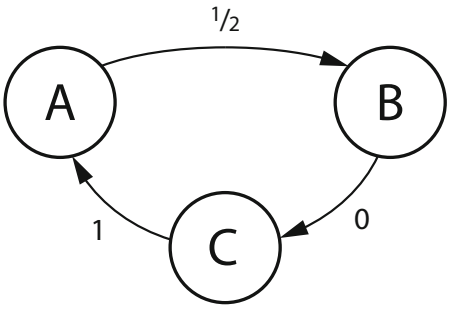


Fig. 9. Rendezvous algorithm from [16] for rigid $\mathcal{S}$SYNC robots in $\mathcal{F}$-COMM

of their units. In a non-rigid setting, they could be stopped too soon, and both detect a distance smaller than 2. From that point onward, if they are activated synchronously and rigidly, they keep switching positions without ever gathering.

For $\mathcal{F}$-COMM robots under the non-rigid $\mathcal{S}$SYNC scheduler, there is a simple self-stabilizing algorithm, shown in Fig. 9. The meaning of an arrow from state X to state Y is that if a robot observes that the light of the other robot has color X, then the first robot sets its own light to color Y. The label λ on each arrow has the same meaning as in Sect. 4.1.

Let us analyze this algorithm. Assume first that both robots start in the same state and both are activated at each turn. Then they always have equal states, and they cycle through states A, B, and C forever. Every time they are both in state A, they move toward the midpoint, and their distance reduces by at least 2δ. Eventually, it becomes so small that they actually gather.

Otherwise, if at any point the two robots are in different states, they will remain in different states forever. In this case their distance will never increase, and they will periodically be found in states B and C, respectively. Whenever this happens, the robot in state C retains its state and waits until the other robot is activated and moves toward it by at least δ. As soon as their distance becomes not greater than δ and they turn again B and C, they finally gather.

Theorem 9 ([16])**.** *The Rendezvous problem is solvable in a self-stabilizing way with 3 colors in* $\mathcal{F}$-COMM *under the non-rigid* $\mathcal{S}$SYNC *scheduler.*

Finally, if the robots are non-rigid but know the value of δ, they can solve Rendezvous with 3 colors both in $\mathcal{F}$-STATE under the $\mathcal{S}$SYNC scheduler and in $\mathcal{F}$-COMM under the $\mathcal{A}$SYNC scheduler. The two algorithms are relatively simple, because not only do the robots know at what point they can assume that all movements will be rigid (i.e., when their distance is at most δ), but knowing δ in their respective reference frames also implicitly gives them a common unit of distance. The details of the two algorithms are found in [16].

Theorem 10 ([16])**.** *The Rendezvous problem is solvable with 3 colors in* $\mathcal{F}$-STATE *and under the non-rigid* $\mathcal{S}$SYNC *scheduler, provided that the robots know the value of* δ*.*

Theorem 11 ([16]). *The Rendezvous problem is solvable with 3 colors in* $\mathcal{F}$-COMM *and under the non-rigid* $\mathcal{A}$SYNC *scheduler, provided that the robots know the value of* δ.

Whether Rendezvous can be solved at all in $\mathcal{F}$-STATE under the rigid $\mathcal{A}$SYNC scheduler is left in [16] as an open problem.

5 Sequence of Patterns

Forming a specific pattern has been a prototypical problem in the oblivious robot model, see [15,17,29] and Chap. 3. The general version of this problem specifies that a set of robot has to form a specific pattern (up to rotation or scaling). In this section we use the concepts of symmetricity and of equivalence class of robots, shortened in class, defined in Chap. 3.

5.1 Sequence of Patterns Without Light

A natural extension of the pattern formation is the one in which robots have to form a sequence of patterns, see [9]. Let $\mathcal{S} = \langle S_0, \ldots, S_{m-1} \rangle$ be a sequence of distinct patterns. A set of robots forms $\mathcal{S}$, starting from a configuration Γ, if it forms the infinite periodic sequence $\mathcal{S}^\infty = \langle S_0, S_2, \ldots, S_{m-1} \rangle^\infty$, obtained by repeating forever the sequence $\mathcal{S}$. It is not hard to see that in the oblivious model there are several restrictions on $\mathcal{S}$, depending on the configuration Γ. Clearly, the relationship between the symmetricity of Γ and the one of any pattern in $\mathcal{S}$ has to be the same of the classic pattern formation (see Chap. 3).

Moreover, the following conditions are necessary: the symmetricity of each pattern in $\mathcal{S}$ is the same; the number of points in each pattern has to be the same. It is not hard to see why the previous conditions are necessary: once two robots share the same position, they could be bonded forever by always activating them at the same time, thus it is not possible for the number of points in the patterns to change; the condition on the symmetricity is also obvious and it comes from similar considerations.

Interestingly, in $\mathcal{S}$SYNC with **rigid** robots the above conditions are also sufficient, see [9].

5.2 Sequence of Patterns with Light

Forming a sequence of patterns in the $\mathcal{A}$SYNC model with **non-rigid** luminous robots has been studied in [6]. Note that, in [6] a pattern in $\mathcal{S}$ is only a set of robots positions, without any restriction on lights color, i.e. two patterns are the same even if the colors of the robots in each pattern are completely different.

In the following we will refer to positional class to indicate a set of points that share the same view when colors are not considered, and we will refer to chromatic class to indicate a set of points that share the same view when colors are considered. When lights are available, it is possible to form a sequence $\mathcal{S}$ even

if: there are repeating pattern in $\mathcal{S}$; the number of points in each pattern is not the same; and, patterns have different symmetricity. The only constraint on $\mathcal{S}$ is that the symmetricity of any pattern in $\mathcal{S}$ has to be divided by the symmetricity of the initial configuration Γ. This in contrast with what happens when lights are not available.

In the algorithm of [6] colors are used to synchronize the various phase of the algorithm, and to keep the symmetry broken when robots from distinct positional classes move in such a way to end up in the same positional class, e.g. two robots go to the same point but have two different colors. Moreover, colors are also used to encode information about the sequence. Specifically, if the same pattern S appears in two different positions in $\mathcal{S}$ it will have a different coloring allowing robots to distinguish which instance of S is.

More in details, the algorithm is divided in five phases:

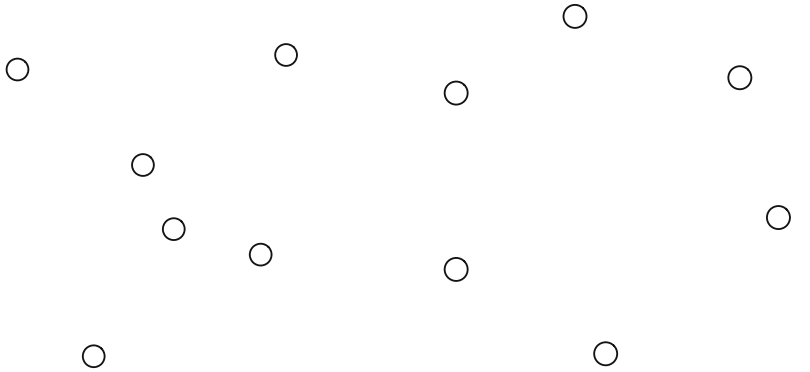
- Leader Identification: In this phase the robots elect one class as leader class. A light with color *Gold* is used to uniquely mark this class in the next phases of the algorithm.
- Pattern Identification: During the pattern identification the leader class moves in such a way to uniquely identify the specific pattern S_j of $\mathcal{S}$ that has to be formed. In this phase colors are used for synchronization.
- Separation: In the separation step robots that are in the same positional class, but in different chromatic classes, separate to form different positional class. Colors are used to symmetry breaking purpose. At the of this phase each chromatic class is on a different circle. All circles are concentric.
- Rotation and composition: Finally, in this two phases the robots dispose themselves to build pattern S_j.

Theorem 12 ([6]). *A set of* **non-rigid** *luminous robots in* ASYNC *starting from an initial configuration Γ can form a sequence of pattern $\mathcal{S}$ if for any $P \in \mathcal{S}$, the symmetricity of P is divided by the symmetricity of Γ.*

6 Mutual Visibility

In this Section we consider the setting where robots obstruct the visibility of each other, and we focus on the Mutual Visibility problem. Such problem has been the object of several recent papers [11–13,22–25,27], and a variety of solutions have been proposed investigating trade-offs between time complexity and number of lights. For this reason, it is interesting to study the Mutual Visibility problem to understand how lights can be used in the design of sophisticated algorithms.

Problem Statement. In the Mutual Visibility problem a set of robots initially positioned in an arbitrary configuration have a to reach a final configuration $\mathcal{C}_f$, where for each pair of distinct robot positions p_j, p_i in $\mathcal{C}_f$ it does not exist any robot r_s on the segment connecting p_i and p_j. We say that robots are *mutually visible* in configuration $\mathcal{C}_f$. See Fig. 10.



(a) Configuration where robots are not mutually visibile.

(b) Configuration where robots are mutually visibile.

Fig. 10. Example of initial and final configuration for Mutual Visibility.

Preliminary Definitions. Given a configuration $\mathcal{C}_t$, at time t, $\mathcal{H}(t)$ denotes the convex hull of $\{p_1(t), p_2(t), \cdots, p_n(t)\}$ at time t. The robots lying on its boundary are the *external robots*, the ones lying in its interior are the *internal robots*. Note that, a robot may not know where the convex hull's vertices are located, because its view may be obstructed by other robots. However, it can easily determine whether it is an external or an internal robot, i.e. a robot r is *external* when there are two robots in its snapshot such that the angle formed by r and them is at least π and there is no other robot in that angle.

6.1 Main Strategies

Analysing the literature, it is possible to identify three main meta-strategies *Shrink*, *Contain* and *Local*. Roughly, each of these techniques works as follow:

- *Shrink*: In this strategy the external robots shrink towards a single point. In doing so internal robots progressively become external. The strategy terminate when all robots are vertices of the convex hull.
- *Contain*: This strategy is based on two phases *interior depletion* and *vertex adjustment*. In the first phase, the internal robots move towards the convex hull. In the second phase the robots on the convex hull move to become vertices.
- *Local*: In the *Local* strategy each robot does a constant number of steps, sometimes a single step, based on its local view and moving of a small distance from its initial position. Doing this it tries to decrease the number of collinear robots.

These three strategy have been proposed in [11]. In the first two strategies the final configuration does not only ensure Mutual Visibility but it also solves the

Convex Formation problem, arranging the robots as vertices of a convex polygon. The existing algorithms for Mutual Visibility use some variation of these strategies, where the modifications are used to obtain special properties such as fast solutions, optimal number of moves, or resilience to faulty robots.

In the following we will assume that the initial configuration is not a line. In case the initial configuration is a line, it is easy recognizable by each robots, and they can run a simple custom algorithm to move themselves in a configuration where there is a proper convex hull.

Strategy *Shrink*. The main idea of strategy *Shrink*, first proposed in [11,13], is to move the external robots on the vertices of the convex hull towards the inside of the convex hull. The final purpose is to shrink the convex hull (see Fig. 11) while not decreasing the number of external robots. The robots use two colors: *Off*, *Vertex*. Initially all robots have a pre-defined color *Off*, the robots that are also vertices become *Vertex* to signal their special positioning. By shrinking the convex hull, internal robots become external, and, eventually, vertices of the convex hull.

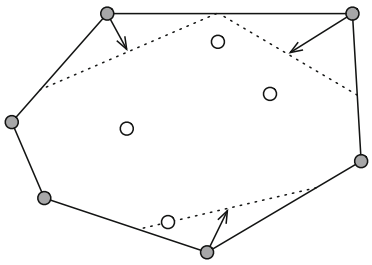
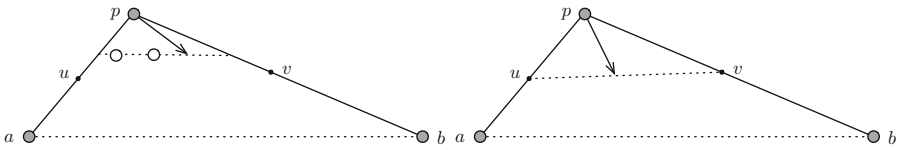


Fig. 11. Motion of vertices in *Shrink*.

Details of Shrink. We will explain the details of the strategy *Shrink* of [11] designed for **rigid** $\mathcal{S}$SYNC. A vertex robot r in position p moves inside the triangle formed by itself and its own two neighbors on the convex hull's boundary[4] (note that such neighbors are necessarily visible). This triangle is $\triangle pab$ of Fig. 12. The only robots moving are the ones on the vertices of the hull. The move is designed in a careful way, and the robot r moves according to this three rules:

- (Rule 1) To avoid collision with other moving vertices r does not go outside the triangle $\triangle puv$, where u, v are the midpoints of edges pa and pb.
- (Rule 2) In order to keep being external, r does not cross any line passing through a robot inside $\triangle puv$ and parallel to ab. In case there is a robot in $\triangle puv$, robot r moves on the closest of such lines. In this last case the number of external robots is increased, see Fig. 12(a).

[4] The neighbors are the adjacent robots on the convex hull boundary.



(a) robot r in position p moves carefully to increase external robots and to remain external.

(b) Default move of robot r in position p to shrink the convex hull.

Fig. 12. Movements of an external robot according to the presence or not of an internal robot in the area of movement, $\triangle puv$.

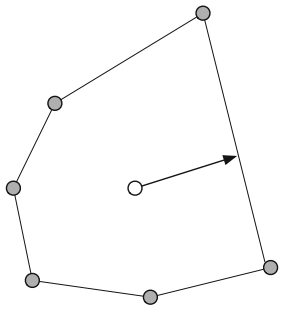


Fig. 13. Special case of unique internal robot.

– (Rule 3) When there are not robots inside $\triangle puv$, then r moves on the line uv remaining a vertex of the convex hull. This move allows [11] to prove that the convex hull shrinks in a such a way to converge to a single point. This convergence property is a key point to prove that all internal robots eventually became external.

Notice that, if a vertex robot r moves using Rule 3 and one of its neighbor, let it be a, is not a vertex then a becomes a vertex. With this observation it is clear that once all robots are external, they eventually become vertices of the convex hull.

There is only one special case: when there exists an unique internal robots. In this case the robot has to move because it could be positioned in the converge point of the shrinking procedure. Thus, it will be never reached by the others.

A custom move is needed in this case. When an internal robots sees that all the other robots are vertices, then it knows that it is the unique internal robots, and it moves to an edge of the convex hull, see Fig. 13.

Robots terminate when they reach a strictly convex configuration, that is when they all see each other with color *Vertex*. Notice, that in this algorithm lights are used for termination detection.

It has been shown that a protocol based on *Shrink* correctly terminates in **rigid** $\mathcal{S}$SYNC using two colors. Moreover, it is possible to slightly modify it so

to solve Mutual Visibility also in the model without lights, but when robots have knowledge of n (the total number of robots in the system). Since lights are only used for termination, without them and with knowledge of n a robots terminate when it sees a strictly convex configuration, and n other robots. It is easy to see that, when n is unknown, *Shrink* uses an optimal number of colors.

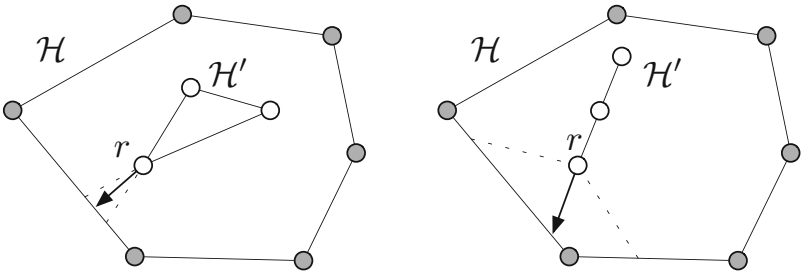
Theorem 13 ([11])**.** *Protocol* Shrink *solves* Mutual Visibility *by* **rigid** *robots in* $\mathcal{S}$SYNC *with* 2 *colors, or with* no *colors if the robots know* n*.*

The strength of *Shrink* strategy is that it requires a minimal number of colors. However, the convergence shown in Theorem 13 does not work in a **non-rigid** or $\mathcal{A}$SYNC model.

Strategy *Contain*. The meta-strategy *Contain*, first presented in [11,12] for **non-rigid** robots in $\mathcal{S}$SYNC, consists of two successive stages: *interior depletion* and *vertex adjustment*. The algorithm uses three colors: *Off*, *External*, *Adjusting*. In the interior depletion stage, the internal robots move towards the boundary of the convex hull. At the end of this stages all robots are external. In the vertex adjustment stage, the external robots make small adjustments to finally reach a strictly convex configuration. Let $\mathcal{H}'(t)$ be the convex hull of the internal robots at time $t \in \mathbb{N}$, see Fig. 14(a).

Details of Contain. More precisely, strategy *Contain* works as follows.

- *Interior depletion*: Initially, all robots have lights *Off*. Once an external robot is activated it switches light to *External* and it does nothing. The robots that move are the one on the border $\mathcal{H}'$. Eventually, a robot realizes to be on the border of $\mathcal{H}'$, this can be done locally once enough external robots have been activated. A vertices r of $\mathcal{H}'$ moves on the border of $\mathcal{H}$. There are three possible way for the robot to move:
  1. When r is the only internal robots, it moves towards the midpoint of the closest edge.
  2. When r believes to be a vertex of a non degenerate $\mathcal{H}'$, i.e. the robots in $\mathcal{H}'$ do not form a line, then r try to move to $\mathcal{H}$. It does so, only when it is able to identify correctly identify an edge of $\mathcal{H}$ where it can move without colliding with other moving robots. As example, if r is a vertex forming an acute internal angle of $\mathcal{H}'$, then it moves to $\mathcal{H}$ by remaining inside the zone delimited by the extension of the edges of $\mathcal{H}'$ to which it belongs, see Fig. 14(a).
  3. When r is an extreme of the line $\mathcal{H}'$, it moves towards $\mathcal{H}$ by using a direction that has a right angle away oriented away from $\mathcal{H}'$, see Fig. 14(b).
- *Vertex adjustment*: When a robots r, vertex of $\mathcal{H}$, with light *External* sees only robots with light *External*, then it makes an adjustment move, the same of strategy *Shrink* of Fig. 12(b). Before doing the move it sets its light to *Adjusting*. After this adjustment, the neighbors of r on $\mathcal{H}$ will be vertices, if they were not both vertices before the move. This *Adjusting* light is used by r to remember it adjusted itself. A robot with light *Adjusting*, once activated it switches to *External* and it terminates.



(a) Case of a non degenerate $\mathcal{H}'$. (b) Case of a degenerate $\mathcal{H}'$.

Fig. 14. Movements of a robot on the border of $\mathcal{H}'$ in *Contain*.

Theorem 14 ([11]). *Protocol* Contain *solves* Mutual Visibility *by* **non-rigid** *robots in* $\mathcal{S}$SYNC *with* 3 *colors.*

Note that *Contain* works even when the system is **non-rigid**, this is in contrast with *shrink*. However, it does have a greater number of lights and the algorithm is more complex.

In [11] slight variations of protocol *Contain* are presented to solve the problem under various conditions and knowledge. Let δ be the minimum distance travelled by a robot.

Theorem 15 ([11]). Mutual Visibility *can be solved in* $\mathcal{S}$SYNC *by* **non-rigid** *robots with* no *colors, if they know* δ *and* n*; it can be solved with* 2 *colors, if the robots know only* δ. Mutual Visibility *can be solved in* $\mathcal{A}$SYNC *by* **rigid** *robots with* 3 *colors, and in* $\mathcal{A}$SYNC *by* **non-rigid** *robots, if they agree on the direction of one coordinate axis.*

Strategy *Local*. The meta-strategy *Local* has been proposed in [11]. *Local* is the only one of the three meta-strategies in which the final solution does not solve Convex Formation. In this strategy, the first time a robot is activated, it makes a small move to a new position avoiding to stop on, or to trespass, any line connecting two visible robots, see Fig. 15. Only two colors are used: *Off*, *Moved* that is turned on before a robot move. This simple idea solves Mutual Visibility using two colors for the sequential scheduler $\mathcal{S}$EQUENTIAL (a particular case of $\mathcal{S}$SYNC where a single robot is activated at each step). Also notice that using this strategy, any robot moves at most once, in the entire execution.

Theorem 16 ([11]). *Protocol* Local *solves* Mutual Visibility *by* **non-rigid** *robots in* $\mathcal{S}$EQUENTIAL *using 2 colors.*

6.2 State of the Art for the Number of Lights

The solutions above have been improved, in number of colors employed and in the model where Mutual Visibility is solvable. In [22] an Algorithm following the

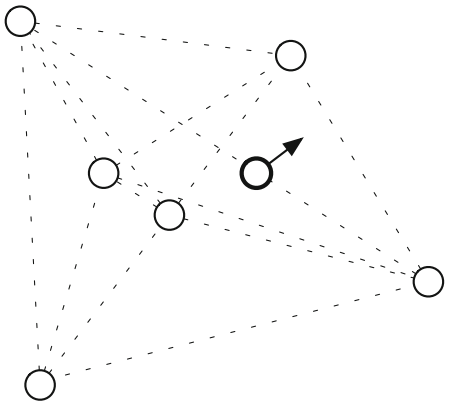


Fig. 15. Motion of a robot in *Local*: the robot with a bold circle moves in such a way to never cross or reach a segment connecting two visible robots.

Contain meta-strategy has been proposed, which allows to solve the problem in $\mathcal{A}$SYNC **non-rigid** with no colors but agreement on one axis, and in $\mathcal{S}$SYNC **non-rigid** with only 2 colors.

Theorem 17 ([22])**.** Mutual Visibility *can be solved in* $\mathcal{S}$SYNC *by* **non-rigid** *robots, in* $\mathcal{A}$SYNC *by* **rigid** *robots, and in* $\mathcal{A}$SYNC *by* **non-rigid** *robots, if they agree on the direction of one coordinate axis, using* 2 *colors.*

Moreover, another solution has been proposed based on the *Local* strategy [3]. This algorithm solves Mutual Visibility (but not Convex Formation) in $\mathcal{A}$SYNC **non-rigid** with 7 colors. The main idea is to enforce an order between the robots movements, by allowing only robots that are not collinear with other robots (the *terminal* robots) to move; in this way a local move monotonically increases the number of terminal robots.

Theorem 18 ([3])**.** Mutual Visibility *can be solved in* $\mathcal{A}$SYNC *by* **non-rigid** *robots using* 7 *colors.*

6.3 Time Complexity and Fault Tolerance

Apart from minimising the number of colors in Mutual Visibility solutions, other works have focused on designing time efficient solutions [23–25,27], or solution for environments where the robots may fails [1], or when robots are fat [21].

Time Complexity. When we consider $\mathcal{F}$SYNC the time complexity of an algorithm is measured by the maximum number of rounds needed to terminate. In case of $\mathcal{S}$SYNC or $\mathcal{A}$SYNC the definition is not straightforward. [24,25] proposes the concept of epoch. Each epoch is an interval of time. A new epoch starts when the previous epoch ends, and an epoch ends as soon as all robots have been

activated at least once since the start of the previous epoch (or time $t = 0$ for the first epoch), see [4]. In the following, the complexity of an algorithm $\mathcal{S}$SYNC or $\mathcal{A}$SYNC is measured as the maximum number of epochs needed to terminate.

Failures and Fat Robots. When a robot experiments a crash failure, it stops moving and it remains in the same position forever. A fat robot is modelled as circular entities with unit radius [5], this in contrast with the classical model where robots are points. For fat robots a robot r_i sees a robot r_j if there exists a not obstructed segment from any point of the circle modelling robot r_i to any point of the circle modelling robot r_j [2,21].

Time Efficient Solutions. As we said, the *Contain* and *Shrink* strategies solve the Convex Formation problem. The *Local* strategy solves only the Mutual Visibility. The *Local* solution proposed in [3] solves Mutual Visibility in $\mathcal{A}$SYNC by **non-rigid** robots in $\mathcal{O}(n)$ steps and a constant number of moves for each robot.

Efficient Solutions for Convex Formation. The first work proposing an efficient solution for Convex Formation has been [27], an algorithm for $\mathcal{F}$SYNC **rigid** uses the *Contain* meta-strategy to solve the problem in $\mathcal{O}(\log n)$ rounds (the algorithm, however, allows collisions). This runtime is done by proposing an *interior depletion* phase where internal robots goes on the convex hull in $\mathcal{O}(\log n)$ rounds. The *vertex adjustment* also ends in $\mathcal{O}(\log n)$ rounds, and in this case robots on the edge of the $\mathcal{H}$ move to create a strictly convex configuration. This is in contrast with the classic *Contain* where vertex move.

In [24] the authors propose an algorithm for $\mathcal{S}$SYNC **rigid** that follows the *Contain* meta-strategy and solves Mutual Visibility in $\mathcal{O}(1)$ epochs. The algorithm uses an initial step, *corner moving*, in which vertex of $\mathcal{H}$ does one adjustment movement. After this adjustment, the vertex will be visible to all internal robots. The movement is similar to the one used by *Shrink*, with the additional idea of never cross, or reach, locations where a collinearity with internal robots can be created. After the adjustment, the convex hull $\mathcal{H}$ will be detectable by internal robots, that can move on its border in a constant number of epochs. In the last phase, the robots on the edges of $\mathcal{H}$ move to create a strictly convex configuration.

A similar strategy, with some modifications, works in $\mathcal{A}$SYNC **rigid** with time complexity $\mathcal{O}(\log n)$ [25].

Finally, in $\mathcal{F}$SYNC **rigid**, when n is known, a linear time solution exists, without using any light [23]. The current state of the art from a time complexity perspective is summarized in Table 1, where only collision-less solutions are considered.

Fault-Tolerance and Fat Robots. In [1] it is investigated how to solve Mutual Visibility in $\mathcal{S}$SYNC **rigid**, with agreement on the axes, when a single robot may experience a crash failure. The algorithm uses 3 colors, and it is based on a variation of the *Shrink* strategy. Notice that, if there is a crashed robot placed

Table 1. Collisions-less solutions for Convex Formation, for $\mathcal{F}$SYNC the time complexity is measured using the number or rounds. For $\mathcal{S}$SYNC and $\mathcal{A}$SYNC, is the number of epochs.

Paper	Scheduler	Time	# Colors
[23]	$\mathcal{F}$SYNC **rigid**	$\mathcal{O}(n)$	0
[24]	$\mathcal{S}$SYNC **rigid**	$\mathcal{O}(1)$	12
[25]	$\mathcal{A}$SYNC **rigid**	$\mathcal{O}(\log n)$	25

exactly in the convergence point of the *Shrink* procedure, then the classic *Shrink* strategy could fail. It could be possible for pairs of robots on the convex hull to be collinear with the faulty robot, and to keep this collinearity by moving in a symmetric way. This is solved in [1], by doing a special move once a configuration with a single internal robot is reached.

Finally, Mutual Visibility has been solved, in $\mathcal{F}$SYNC **rigid**, when robots are fat using 10 colors, in time $\mathcal{O}(n)$ [21].

7 Conclusions

We have seen that lights greatly enhance the capability of robots. They create a new computational landscape where the relationships between $\mathcal{A}$SYNC, $\mathcal{S}$SYNC and $\mathcal{F}$SYNC are different from the usual oblivious model. Besides this new relationship, a wide set problems can now be solved. A prototypical example is the Rendezvous problem.

Another advantage is the possibility to solve old problems under weaker assumptions, see the reduction of the restrictions needed on the sequences of formable patterns by luminous robots.

Finally, the uses of lights allows the designing of fast and fault tolerant algorithms in models where robots obstruct each other.

The model of luminous robot is still relatively new, and there are many open problems.

References

1. Aljohani, A., Sharma, G.: Complete visibility for mobile robots with lights tolerating a faulty robot. In: Proceedings of the 32nd IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPS Workshops), pp. 834–843 (2017)
2. Aljohani, A., Poudel, P., Sharma, G.: Fault-tolerant complete visibility for asynchronous robots with lights under one-axis agreement. In: Rahman, M.S., Sung, W.-K., Uehara, R. (eds.) WALCOM 2018. LNCS, vol. 10755, pp. 169–182. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-75172-6_15

3. Bhagat, S., Mukhopadhyaya, K.: Optimum algorithm for mutual visibility among asynchronous robots with lights. In: Spirakis, P., Tsigas, P. (eds.) SSS 2017. LNCS, vol. 10616, pp. 341–355. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69084-1_24
4. Cord-Landwehr, A., et al.: A new approach for analyzing convergence algorithms for mobile robots. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6756, pp. 650–661. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22012-8_52
5. Czyzowicz, J., Gasieniec, L., Pelc, A.: Gathering few fat mobile robots in the plane. Theor. Comput. Sci. **410**, 81–499 (2009)
6. Das, S., Flocchini, P., Prencipe, G., Santoro, N.: Forming sequences of geometric patterns with oblivious mobile robots. In: Proceedings of the 7th International Conference on FUN with Algorithms (FUN), pp. 113–124 (2014)
7. Das, S., Flocchini, P., Prencipe, G., Santoro, N., Yamashita, M.: The power of lights: synchronizing asynchronous robots using visible bits. In: 32nd IEEE International Conference on Distributed Computing Systems (ICDCS), pp. 506–515 (2012)
8. Das, S., Flocchini, P., Prencipe, G., Santoro, N., Yamashita, M.: Autonomous mobile robots with lights. Theor. Comput. Sci. **609**, 171–184 (2016)
9. Das, S., Flocchini, P., Santoro, N., Yamashita, M.: Forming sequences of geometric patterns with oblivious mobile robots. Distrib. Comput. **28**, 131–145 (2015)
10. D'Emidio, M., Frigoni, D., Navarra, A.: Synchronous robots vs asynchronous lights-enhanced robots on graphs. Electron Notes Theor. Comput. Sci. **322**, 169–180 (2016)
11. Di Luna, G.A., Flocchini, P., Gan Chaudhuri, S., Poloni, F., Santoro, N., Viglietta, G.: Mutual visibility by luminous robots without collisions. Inf. Comput. **254**, 392–418 (2017)
12. Di Luna, G.A., Flocchini, P., Gan Chaudhuri, S., Santoro, N., Viglietta, G.: Robots with lights: overcoming obstructed visibility without colliding. In: Felber, P., Garg, V. (eds.) SSS 2014. LNCS, vol. 8756, pp. 150–164. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11764-5_11
13. Di Luna, G.A., Flocchini, P., Poloni, F., Santoro, N., Viglietta, G.: The mutual visibility problem for oblivious robots. In: Proceedings of the 26th Canadian Computational Geometry Conference (CCCG), pp. 348–354 (2014)
14. Dijkstra, E.W.: Self-stabilizing systems in spite of distributed control. Commun. ACM **17**, 643–644 (1974)
15. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Arbitrary pattern formation by asynchronous oblivious robots. Theor. Comput. Sci. **407**, 412–447 (2008)
16. Flocchini, P., Santoro, N., Viglietta, G., Yamashita, M.: Rendezvous with constant memory. Theor. Comput. Sci. **621**, 57–72 (2016)
17. Fujinaga, N., Yamauchi, Y., Ono, H., Kijima, S., Yamashita, M.: Pattern formation by oblivious asynchronous mobile robots. SIAM J. Comput. **44**, 740–785 (2015)
18. Heriban, A., Defago, X., Tixeuil, S.: Optimally gathering two robots. In: Proceedings of the 19th International Conference on Distributed Computing and Networking (ICDCN), pp. 3:1–3:10 (2018)
19. Khan, L.U.: Visible light communication: applications, architecture, standardization and research challenges. Dig. Commun. Netw. **2**, 78–88 (2017)
20. Okumura, T., Wada, K., Katayama, Y.: Optimal asynchronous rendezvous for mobile robots with lights. Arxiv, CoRR abs/1707.04449 (2017)

21. Sharma, G., Alsaedi, R., Bush, C., Mukhopadyay, S.: The complete visibility problem for fat robots with lights. In: Proceedings of the 19th International Conference on Distributed Computing and Networking (ICDCN), pp. 21:1–21:4 (2018)
22. Sharma, G., Busch, C., Mukhopadhyay, S.: Mutual visibility with an optimal number of colors. In: Bose, P., Gąsieniec, L.A., Römer, K., Wattenhofer, R. (eds.) ALGOSENSORS 2015. LNCS, vol. 9536, pp. 196–210. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28472-9_15
23. Sharma, G., Bush, C., Mukhopadyay, S.: Brief announcement: complete visibility for oblivious robots in linear time. In: Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), pp. 325–327 (2017)
24. Sharma, G., Vaidyanathan, R., Trahan, J.L., Busch, C., Rai, S.: Complete visibility for robots with lights in $O(1)$ time. In: Bonakdarpour, B., Petit, F. (eds.) SSS 2016. LNCS, vol. 10083, pp. 327–345. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49259-9_26
25. Sharma, G., Vaidyanathan, R., Trahan, J.L., Bush, C., Rai, S.: O(log n)-time complete visibility for asynchronous robots with lights. In: Proceedings of the 32nd IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 513–522 (2017)
26. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: formation of geometric patterns. SIAM J. Comput. **28**, 1347–1363 (1999)
27. Vaidyanathan, R., Bush, C., Trahan, J.L., Sharma, G., Rai, S.: Logarithmic-time complete visibility for robots with lights. In: Proceedings of the 29th IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 375–384 (2015)
28. Viglietta, G.: Rendezvous of two robots with visible bits. In: Flocchini, P., Gao, J., Kranakis, E., Meyer auf der Heide, F. (eds.) ALGOSENSORS 2013. LNCS, vol. 8243, pp. 291–306. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-45346-5_21
29. Yamashita, M., Suzuki, I.: Characterizing geometric patterns formable by oblivious anonymous mobile robots. Theor. Comput. Sci. **411**, 2433–2453 (2010)