Computations by Luminous Robots

Paola Flocchini

School of Electrical Engineering and Computer Science
University of Ottawa, Canada
`flocchin@site.uottawa.ca`

Abstract. The study of computability issues by a system of simple, autonomous, oblivious, mobile robots, operating in the plane in LOOK-COMPUTE-MOVE cycles, has been the object of intensive investigations. These robots do not have explicit communication mechanisms, but they implicitly cooperate towards a common goal.
This paper focuses on *luminous robots*, a recently introduced model where the robots are equipped with a light that can take a constant number of different colors. The light is visible to the observing robots and stays lit from a computation cycle to the next. The availability of lights, which provides little communication and memory, has clearly a great impact on the system of robots. We review the recent results, highlighting the many open problems and research directions.

1 Introduction

Consider a set of autonomous mobile computational entities, called *robots*, which operate in the Euclidean plane initially occupying distinct points. The robots are provided with (possibly different) local coordinate systems centered in themselves, and with sensors that allow them to perceive the positions of the other robots. They operate in cycles of LOOK-COMPUTE-MOVE activities: when active, a robot observes the others, it computes a destination point, and it moves towards it. Once a cycle is completed, however, a robot *forgets* any previous information, and it starts the next cycle from scratch, making a computation solely on the basis of its current observation. Such forgetful behaviour is called *obliviousness*. Robots operate synchronously (`FSYNC`), if they perform simultaneously all their *Look-Compute-Move* activities in synchronized rounds; semi-synchronously (`SSYNC`), if only subsets of robots are activated in synchronized rounds; or asynchronously (`ASYNC`), if there is no synchronization and each robot operates at its own pace.

Autonomous oblivious robots have been extensively investigated, considering a variety of assumptions on their characteristics (e.g., limited vs. full visibility, level of synchrony, availability of a common coordinate system, etc.) and studying how these assumptions impact the robots' computational power on basic coordination tasks. Typical tasks in this setting are, for example, *Pattern Formation*, where the robots must place themselves so to form a given shape (e.g., see [13, 16, 21, 24]), *Gathering*, where the robots must move to the same point (e.g., see [1, 2, 8, 14, 18]), *Scattering*, where robots have to move from each other so to cover the space (e.g., see [3, 7, 17]). For a recent account of the current investigations, see [12].

Obliviousness is very limiting for the robots, which can rely only on the environment to decide their next step towards a common goal. On the other hand, it is a very desirable characteristic because it provides a form of fault-tolerance and self-stabilization.

Recently, to partially overcome the limits of obliviousness while maintaining some of its advantages, a stronger model has been introduced where oblivious robots carry a "light" that can take different colors (a constant number of them), can be seen by the observing robots, and stays on from cycle to cycle. Clearly, the availability of lights drastically changes the computational power of the robots, which have now the means for a little memory (by coding information to be stored into the light), and a little communication (by using the light to transmit information to the other robots). Although the idea was suggested some time ago [19], the actual model has been formalized only recently [6], and has already generated several interesting results.

The first natural question is to understand the impact that the availability of lights has on the computability power of the robots depending on their level of synchrony. This capability is quite strong; in fact, ASYNC luminous robots can perform any task solvable by SSYNC non-luminous ones. Moreover, in contrast with the situation in the classical setting, in the luminous realm there is no computational difference between the ASYNC and the SSYNC models. Several results in this regards have been derived [6], but the full computational picture describing the impact of lights on the various models is still to be completed.

Another interesting issue is to de-couple the concept of lights into two quite distinct components: the one that provides some memory to the robots (an internal state that persists from cycle to cycle), and the one that allows some form of communication (an external visible sign). Understanding the computational power of each component versus the power of their combination is a challenging and important issue. A first step in this direction has been done by studying the impact of the two individual capabilities on the rendezvous problem [15] and leaving a wealth of open problems still to be addressed.

This paper reviews the recent discoveries on luminous robots indicating the research directions emerged by the study of this new model.

2 The Robots

Let $\mathcal{R} = \{r_1, r_2, \cdots, r_n\}$ be a set of *robots*, operating in $\mathbb{R}^2$. Each robot is the centre of its own coordinate system and the robots have no agreement on the orientation of the system, on its handedness, or on their unit of distance. We denote by $r(t) \in \mathbb{R}^2$ the position occupied by robot $r \in \mathcal{R}$ at time t (for description purposes, the positions are expressed in a global coordinate system, which is unknown to the robots). Two robots r and s are said to *collide* at time t if $r(t) = s(t)$.

The robots are provided with sensors that allow them to perceive the positions of the other robots. If they can sense the whole environment, we say that they have *full visibility*, if they are able to perceive other robots only within a certain visibility radius, we say that they have *limited visibility*.

The robots are autonomous (i.e., without any external control), anonymous (i.e., without internal identifiers), indistinguishable (i.e., without external markings), without any direct means of communication.

At any time, robots can be active or inactive, and initially they are all inactive. When activated, a robot performs a LOOK-COMPUTE-MOVE sequence of operations: it first obtains a snapshot of the positions, expressed in its local coordinate system, of all visible robots (LOOK); using the last obtained snapshot as an input, the robot executes an algorithm (the same for all robots) to compute a destination point $x \in \mathbb{R}^2$ (COMPUTE); finally, it moves towards x (MOVE). It then stays inactive until the next activation.

The robots are *oblivious* in the sense that, when a robot becomes inactive, all its local memory is reset. In other words, upon becoming active again, a robot has no memory of past computations and snapshots.

A *luminous* robot r is a robot that, in addition to the above capabilities, is endowed with a persistent and externally visible state variable $Light[r]$, called *light*, whose values are from a finite set C of *colors*. The value of $Light[r]$ (i.e., its color) can be changed in each cycle by r at the end of its COMPUTE operation. A light is *externally visible* in the sense that its color at time t is visible to all robots that perform a LOOK operation at that time in its visibility radius. A light is *persistent* in the sense that, while r is oblivious and forgets all other information from previous cycles, the color is not automatically reset at the end of a cycle.

With regards to the activation and timing of the robots, there are two basic settings: *semi-synchronous* (SSYNC) and *asynchronous* (ASYNC). In SSYNC, the time is discrete; at each time instant t (a *round*) a subset of the robots is activated and performs its operations atomically, ending at time $t + 1$. At any given round, any subset of robots may be activated. In particular, if all robots are activated at every round, the system is *fully synchronous* (FSYNC). At the opposite spectrum, in ASYNC, there is no common notion of time; each robot is activated independently, the LOOK operation is instantaneous, but the COMPUTE and MOVE operations can take an unpredictable (but finite) amount of time, unknown to the robot.

The choice of the activations is done by an *adversary*, which, for fairness, activates each robot infinitely often. In ASYNC, the adversary choses also the (finite) duration of each operation. The adversary might or might not have the power to interrupt the movement of a robot before it reaches its destination in the MOVE operation. If it does, the system is said to be NON-RIGID; the only constraint on the adversary is that there exists a constant $\delta > 0$ such that, if interrupted before reaching its destination, a robot moves at least a distance δ, not known to the robot itself. Notice that, otherwise, the adversary would be able to prevent a robot from reaching any given destination in a finite number of turns. If movements are not under the control of the adversary, and every robot reaches its destination at every turn, the system is said to be Rigid.

3 Luminous robots in FSYNC, SSYNC, and ASYNC

3.1 The Setting

A natural research investigation when considering luminous robots is to understand the impact of lights on the computability power of the robots, with respect to the different synchrony levels of their scheduler. It is well known that, in absence of lights, the three models form a strict hierarchy in that there exist problems solvable in FSYNC but not in SSYNC (for oblivious robots [21]), as well as problems solvable in SSYNC but not in ASYNC (for non-oblivious robots [20]). The first question is whether such a strict dominance exists also in the context of luminous robots. Interestingly, it turns out that the availability of lights does not preserve it: the difference between asynchrony and semi-synchrony disappears making the two model equally powerful [6].

In the following, when robots are luminous we will refer to luminous ASYNC (resp. luminous SSYNC, luminous FSYNC) models, and we denote by $\mathtt{ASYNC}^m$ (resp. $\mathtt{SSYNC}^m$, $\mathtt{FSYNC}^m$) luminous models using m colors.

3.2 Luminous ASYNC vs. SSYNC

This Section describes the relationship between luminous $\mathtt{ASYNC}^{O(1)}$ versus non-luminous SSYNC, as well as the one between luminous $\mathtt{ASYNC}^{O(1)}$ and $\mathtt{SSYNC}^{O(1)}$.

a) Luminous ASYNC is at least as powerful as SSYNC. First of all, synchronous systems equipped with lights (with a constant number of colors) are at least as powerful as semi-synchronous systems without lights. In fact, as shown below, any problem solvable in SSYNC without lights is also solvable by asynchronous luminous robots.

Given an algorithm $\mathcal{P}$ that solves a problem in SSYNC, there is a simulation protocol for luminous robots in ASYNC in which every execution is equivalent to a SSYNC execution of $\mathcal{P}$. The lights used by the simulation protocol can have five colors: T(rying), M(oving), S(topped), F(inished), W(aiting). At the beginning, all lights are set to T. The protocol is a sequence of Mega-Cycles, each of which starts with all robots trying to execute protocol $\mathcal{P}$ (with color T) and ends with all robots finishing the Mega-Cycle (with color F) having executed $\mathcal{P}$ once. All robots with light F eventually turn their lights to T and when this process is completed, a new Mega-Cycle starts. The protocol is designed in such a way that, during each Mega-Cycle, every robot executes exactly one Look-Compute-Move step of algorithm $\mathcal{P}$, changing light to M when performing the move prescribed by algorithm $\mathcal{P}$. The execution of the simulation is semi-syncronous because robots allowed to execute the step concurrently in the same Mega-Cycle are guaranteed to have observed the same snapshot (i.e., while nobody was moving). The transitions describing the change of color of the robots in a Mega-Cycle are depicted in Figure 1.

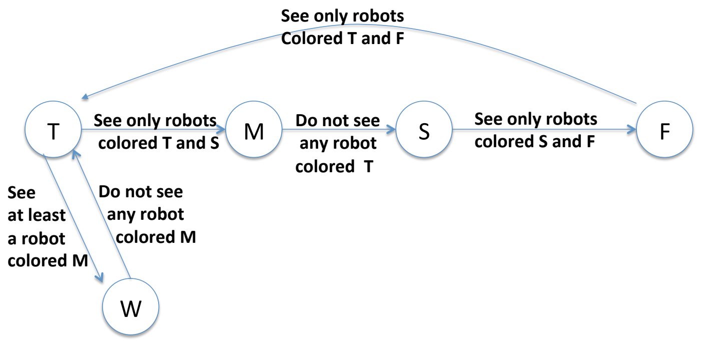

See only robots Colored T and F
T
See only robots colored T and S
M
Do not see any robot colored T
S
See only robots colored S and F
F
See at least a robot colored M
Do not see any robot colored M
W

Fig. 1. `ASYNC` simulation of a `SSYNC` algorithm (executing $\mathcal{P}$ on the current snapshot only when transitioning from `T` to `M`). 1) The content of a circle represents the color of the computing robot; 2) the caption on an arrow describes a condition on the colors seen by the robot; 3) the transition between two colors indicates the local change of color corresponding to the given condition.

Theorem 1. *[6] Let $\mathcal{P}$ be an algorithm that solves a problem in* `SSYNC`*. There exists an algorithm $\mathcal{S}(\mathcal{P})$ in luminous* `ASYNC` *in which every execution is equivalent to a* `SSYNC` *execution of $\mathcal{P}$.*

b) Luminous `ASYNC` is more powerful than `SSYNC`. Luminous robots in `ASYNC` turn out to be actually *more powerful* than robots without lights in `SSYNC`. In fact, there exist problems that can be solved by asynchronous luminous robots with $O(1)$ colors, but that are unsolvable by semi-synchronous robots without colors. One such a problem is rendezvous of two oblivious robots where the robots, initially located in different points of the plane, need to gather exactly in the same point. In fact, it is well known that rendezvous is unsolvable in `SSYNC` [21]. The impossibility in the classical model without colors is due to the fact that the adversary could schedule the activations in such a way that the robots are forced to oscillate and never meet. However, there exists a simple solution in luminous `ASYNC` that uses four colors for anonymous, oblivious robots with non rigid movement and no common coordinate systems [6]. A luminous robot can exploit the lights to decide when to move towards the midpoint and when instead to move towards the other robot, without ever risking to switch position. The solution makes use of four colors: $\{a, b, c, d\}$, and it is depicted in Figure 2.

Theorem 2. *[6]* $\texttt{ASYNC}^{O(1)}$ *is more powerful than* `SSYNC`.

c) Luminous $\texttt{ASYNC}^{O(1)}$ is as powerful as luminous $\texttt{SSYNC}^{O(1)}$. Finally, in contrast with the dominance of `SSYNC` versus `ASYNC` observable without lights, the difference between asynchrony and semi-synchrony disappears when they are both enhanced with lights. This can be seen by showing that, given an algorithm $\mathcal{P}$ designed for $\texttt{SSYNC}^k$, there is an execution of the simulation protocol described

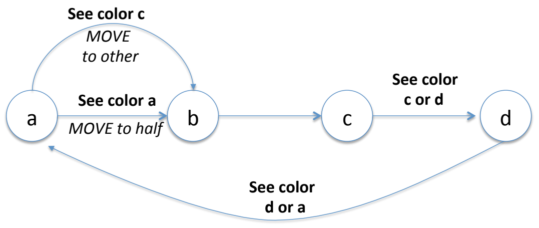


Fig. 2. Rendezvous for luminous robots in `ASYNC`. 1) The color in a circle represents the color of the computing robot; 2) a transition between a color and another indicates a change in the robot's color; 3) the caption on top of a transition (if any) indicates what the computing robot sees; 4) the caption below a transition describes the corresponding movement (if any) prescribed by the algorithm.

earlier, which uses precisely $O(k)$ colors (thus works in $\texttt{ASYNC}^{O(k)}$). The result then follows.

Theorem 3. *[6]* $\texttt{ASYNC}^{O(1)}$ *is as powerful as* $\texttt{SSYNC}^{O(1)}$

3.3 Luminours `ASYNC` vs. `FSYNC`

For the case of fully synchronous system the only known result is that there exist problems that robots cannot solve without lights, even if they are fully-synchronous, but that can be solved by asynchronous luminous robots with $O(1)$ colors. One such problem is the Oscillating Points Problem requiring two robots, x and y, initially in distinct locations, to alternately come closer and move further from each other.

Theorem 4. *[6]* `FSYNC` *is not more powerful than* $\texttt{ASYNC}^{O(1)}$

3.4 Open Problems

First and foremost, it is still unknown whether or not there are problems solvable by fully-synchronous robots but not by asynchronous luminous robots; a positive answer would imply that $\texttt{ASYNC}^{O(1)}$ and `FSYNC` are incomparable, while a negative one would imply that $\texttt{ASYNC}^{O(1)}$ is more powerful than `FSYNC`.

Moreover, it is known that the availability of a snapshot renders asynchronous luminous robots more powerful than regular fully-synchronous one [6]; whether a property weaker than a snapshot would suffice to achieve the same result is still open.

4 Computing by Luminous Robots: Mutual Visibility

4.1 The Setting and the Problem

In this Section we consider a variant of the model, where robots cannot see through other robots; in other words, r can *see* another robot s (equivalently, s

is *visible* to r) at time t if and only if no other robot lies in the segment $r(t)s(t)$ at that time. Moreover, collisions are not permitted. In such a setting, the color of robot r at time t can be seen by all robots visible by r at that time.

The Mutual Visibility problem requires the robots, initially located in different position, to terminate in a configuration where they are still in distinct locations with no three of them being collinear.

Let $\mathcal{H}(t)$ denote the convex hull of $\{r_1(t), r_2(t), \cdots, r_n(t)\}$ at time t. The robots lying on its boundary are called *external robots* at time t, while the ones lying in its interior are the *internal robots* at time t. 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.

4.2 Solutions

We describe two solutions, Algorithm *Shrink* and Algorithm *Contain* whose goal is to allow the robots to position themselves at the vertices of a convex polygon, thus solving Mutual Visibility. These algorithms are based on different strategies, and are tailored for different situations. Protocol *Shrink* uses two colors and requires rigid movements, while protocol *Contain* uses more colors but operates also with non-rigid movements [9–11].

a) Algorithm *Shrink*

Consider RIGID robots in SSYNC. The main idea of Algorithm *Shrink* is to make the external robots move towards the inside of the convex hull formed by the robots, so to shrink it (see Figure 3). Initially the robots have a pre-defined colour *Off* and they become *Vertex* to signal termination. Eventually, all the robots reach a strictly convex configuration, they all see each other with colour *Vertex*, and they terminate.

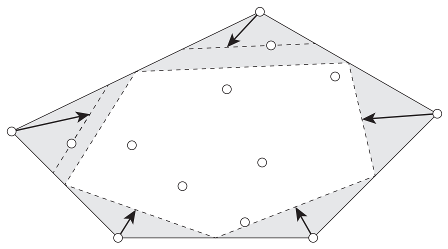
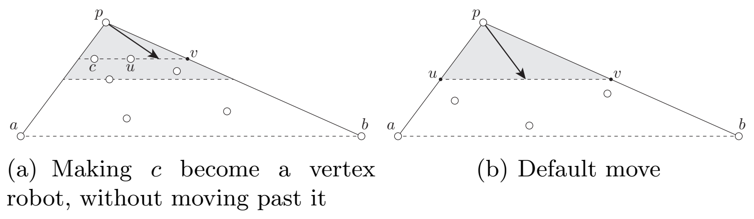


Fig. 3. From [10]: Combined motion of all vertex robots in Algorithm *Shrink*

More precisely, the behaviour of the robots is as follows: a vertex robot moves inside the triangle formed by itself and its own two neighbors on the convex hull's boundary (note that they are necessarily visible). The goal of the move is to make the convex hull shrink, and possibly to increase the number of vertex robots. The destination inside the triangle is carefully calculated so to avoid collisions with other robots that may be moving at the same time, and to prevent the moving robot to become a non-vertex robot. A few other technicalities are required to treat the special case when the initial configuration is a line and to avoid deadlocks when all robots visible to an internal one are coloured *Vertex* [10].



(a) Making c become a vertex robot, without moving past it
(b) Default move

Fig. 4. From [10]: Move of an external robot p in two cases (robots' locations are indicated by small circles), where a and b are the locations of p's two neighbors on $\mathcal{H}$.

Algorithm *Shrink* correctly terminates in Rigid SSYNC using two colors. Moreover, it is possible to slightly modify it so to solve Mutual Visibility also when the two colors are not available, but robots have knowledge of n (the total number of robots in the system).

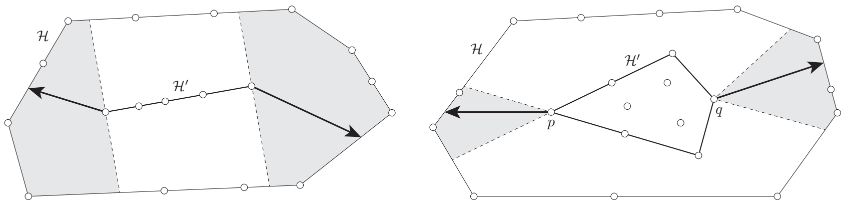
Theorem 5. *[10] Protocol* Shrink *always solves* Mutual Visibility *by* Rigid *robots in* SSYNC *with* 2 *colors, or with* no *colors if the robots know their number, n.*

b) Algorithm *Contain*

Consider Non-Rigid robots in SSYNC. Algorithm *Contain* consists of two phases: an *interior depletion* phase and a *vertex adjustments* phase, to be executed in succession. In the first phase, the internal robots move towards the boundary of the convex hull signalling that they become external by changing their light, and in the second phase the robots (who are now all external) make small adjustments to finally reach a strictly convex configuration.

More precisely, let $\mathcal{H}'(t)$ be the convex hull of the positions of the internal robots at time $t \in \mathbb{N}$. When a robot r understands that it lies on a vertex of $\mathcal{H}'$, it moves towards the boundary of $\mathcal{H}$, part of which is identifiable by r. The destination point depends on the position of r within $\mathcal{H}'$ (whether r is the only internal robot, or $\mathcal{H}'$ is a line segment and r occupies one endpoint, or $\mathcal{H}'$ is a non-degenerate polygon and r it lies on one of its vertices). Some of these cases are depicted in Figure 5. It can be shown that eventually all robots become *External* and the interior of the convex hull is depleted.

The vertex adjustments phase proceeds as follows: when a robot lies at a vertex of $\mathcal{H}$ and it sees only robots with light sets to *External*, it makes the "default move" depicted in Figure 4(b). It also sets its light to *Adjusting*, to remember



(a) Case with collinear internal robots (b) General case

Fig. 5. From [11]: Interior depletion phase of Algorithm *Contain*.

it already adjusted. When the adjustment is done, the robots at a and b are guaranteed to occupy vertices of $\mathcal{H}$. Each external robot becomes a vertex robot at some point, then it adjusts its position while remaining a vertex, possibly making its adjacent robots become vertices as well, and it terminates. When all robots have terminated, the configuration is strictly convex, and therefore Mutual Visibility is solved.

Theorem 6. *[11] Protocol* Contain *always solves* Mutual Visibility *by* NON-RIGID *robots in* SSYNC *with* 3 *colors.*

Slight variations of protocol *Contain* can solve the problem under a variety of combination of conditions and knowledge.

Theorem 7. *[11] In* SSYNC*,* Mutual Visibility *can be solved by* NON-RIGID *robots with* no *colors, if the robots know δ (the minimum distance traversed by a robot) and their number n, and with* 2 *colors, if the robots know δ. In* ASYNC*,* Mutual Visibility *can always be solved with* 3 *colors in* ASYNC *by* RIGID *robots, and in* ASYNC *by* NON-RIGID *robots, if they agree on the direction of one coordinate axis.*

An interesting issue that has been investigated in regards to the Mutual Visibility problem in FSYNC is the time complexity of a solution. Indeed, a RIGID logarithmic time algorithm has been proposed [22] and such an algorithm would improve the time complexity of protocol *Contain* if it were to be executed in a synchronous system with rigid movements: note that *Contain* is designed for the weaker models of SSYNC (NON-RIGID) or ASYNC (RIGID).

4.3 Open Problems

Some questions follow immediately from the above solutions: Can Mutual Visibility be solved in ASYNC without additional assumptions (like agreement on direction) ? What is the power of rigidity ? is it necessary in ASYNC ? Is there a NON-RIGID solution that employs less than 3 colors ? Can Mutual Visibility be solved without colors and without additional information ? Note that the two colors of protocol *Shrink* are used only for termination purposes, and they could be traded by knowledge of n.

5 Partial Luminosity: Communication versus Memory

5.1 The Setting and the Problem

All the results discussed so far assume the availability of lights that are externally visible to robots performing their LOOK operation, as well as locally persistent from one computation cycle to the next. Visibility of lights is mostly used by the robots to *communicate* pieces of information, while persistence is providing them with limited *memory* in an otherwise oblivious system. As seen in the previous Sections, the combination of these capabilities is understandably quite powerful.

To better understand the power of communication versus the one of memory in robots' computation, in [15] the two capabilities have been considered separately. More precisely, two robots' models have been proposed. For silent *finite-state* robots (FSTATE), the light of a robot is visible only to the robot itself. For *finite-communication* robots (FCOMM), a robot's light is visible only to the other robots. The two settings have been studied in relation to the classical rendezvous problem.

rendezvous consists of having two robots meet in the same point. It is well known that in the oblivious robots' model without any light, the problem is solvable only in `FSYNC` [21]. The main problem is "conscious" symmetry breaking, which cannot be achieved under `SSYNC` and `ASYNC` schedulers. On the other hand, it is solvable for luminous robots (with full lights); in fact there is a 2-colors algorithm for `SSYNC`, which is optimal [23], and a 4 colors algorithm in `ASYNC` (described in Section 3.2) [6].

When considering FSTATE versus FCOMM robots, several components have emerged. For example, it has become more evident that the power of memory versus the one of communication is highly connected to other capabilities of the robots. In particular, the rigidity of their movements, and the level of synchrony of the scheduler. It is indeed the combined use of these features that allows rendezvous to be solved.

5.2 FSTATE **rigid Robots in** `SSYNC`

When the lights provide only constant memory, an algorithm using six colors (internal states) has been devised for rigid robots in `SSYNC`.

The main idea of the algorithm is to make use of the (possibly different) units of distance of the robots to have them gather either as a result of a series of "symmetric" rules, or by exploiting any accidental symmetry breaking that might happen due to lack of synchrony or to disagreements on the unit distance. The role of internal states is to allow the robots to recognize such symmetry breaking, should it occur.

Intuitively, the attempted behaviour of the robots is the following: they try to reach a configuration in which they both observe the other robot at distance greater than (or equal to) their own unit. They then try to switch position and, when this happens, they attempt to gather by meeting in the mid-point between them. Note that the internal state can be used to differentiate the various stages of this attempted behaviour and, in particular, to memorize the other's position (left/right) and thus detect a switch. It can be shown that if the attempted behaviour is indeed reached, gathering is eventually achieved. On the other hand, if the attempted behaviour is not reached (because of disagreement on the unit distance or asymmetries in the activation scheduling), a detectable symmetry breaking occurs, which can be immediately exploited to gather anyway.

Theorem 8. *[15] In* `SSYNC`*, Rendezvous of two* FState *rigid robots is solvable with six internal states.*

5.3 FComm rigid Robots in `ASYNC`

Also in `ASYNC` for FState robots, the algorithm uses the local unit distance as a computational tool, but in a rather different way, since a robot cannot remember its own color and has to infer information by observing the other robot's light.

As before, the robots attempt to coordinate a behaviour that would eventually result in gathering if their schedule happens to be synchronized and/or their unit distances are the same, but that would achieve gathering also for any break of symmetry given by a deviation from the attempted behaviour. Intuitively, the two robots try to reach a configuration in which they both see each other at distance lower than their unit distance. At this point they try to compare their unit distance, in order to break symmetry. They do that by attempting the creation of a configuration where their distance is equal to the sum of the respective unit distances. When this is accomplished, each robot can infer the other's unit and compare it with its own. If a robot has a smaller unit, it moves towards its partner, which waits. Otherwise, if their units are equal, as soon as a robot wakes up, it moves towards the mid-point and orders its partner to stay still. If both robots do so, they gather in the middle. If one robot is delayed due to asynchrony, it acknowledges the order to stay still and tells the other robot to come.

The coordination required to accomplish this overall attempted behaviour relies on the communication through the external lights, as well as on some level of synchrony that may or may not occur depending on the scheduler. If it does not occur, i.e., if the robots find themselves in different states, symmetry is broken and it can be shown that gathering is guaranteed anyway.

Theorem 9. *[15] In* `ASYNC`*, Rendezvous of two rigid* FComm *robots is solvable with twelve colors.*

5.4 FComm Robots in `SSYNC`

The situation of FComm Robots in `SSYNC` is quite different and much simpler. Indeed, there exist an algorithm using 3 colors, which is optimal and does not even require rigidity of movement. The robots can assume three colors: a, b, and c. If a robot sees the other colored a, it colors itself b and it moves to the mid-point between them, if a robot sees the other colored b, it colors itself c and it stay still, finally, if a robot sees the other colored c, it colors itself a and moves to the other robot's position (the algorithm is depicted in Figure 6). It is easy to see that such a simple algorithm solves the problem.

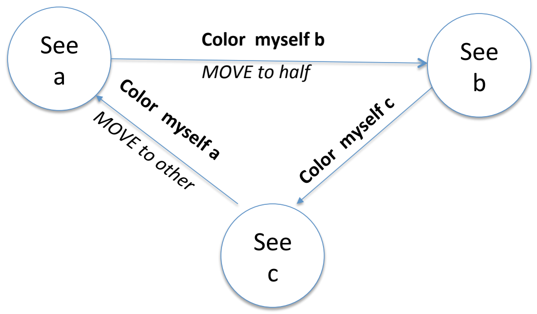


Fig. 6. SSYNC rendezvous in FComm. 1) the content of a circle represents what the computing robot sees; 2) the caption on top of a transition describes the change of the robot's color; 3) the caption below a transition describes the corresponding movement (if any) prescribed by the algorithm.

Theorem 10. *[15] In* ASYNC, *Rendezvous of two* FState *robots is solvable with three colors.*

5.5 Open Problems

The above results open more questions than they close. The general question on the relationship between the computability power of external lights versus internal lights, and thus on whether it is better to communicate or to remember is still open. The striking difference between the simplicity of the SSYNC solution for the FComm model and the complexity of the one for the FState model, as well as the inability to find a SSYNC solution for FState, seem to indicate that FComm is more powerful that FState.

Another parameter that plays an important role in robots' computational power is represented by rigidity of movement, which appears to be crucial to be able to find solutions in FState and in ASYNC in the FComm model. Is rigidity necessary ?

6 Computing by Luminous Robots: Sequences of Patterns

6.1 The Setting and the Problem

Pattern formation by oblivious robots is one of the basic problems and it has been studied extensively in various settings (e.g., see [13, 16, 21, 24]). A pattern is given to each robots as a set of point in its own coordinate system, the robots must place themselves in the plane so to form the pattern, possibly scaled, rotated or translated.

Forming a sequence of patterns is the natural next step. The main challenge is to have the oblivious robots create some form of memory in the environment to be able to move from one pattern to the next. Let $\mathcal{S} =< S_0, \ldots, S_{m-1} >$ be the sequence of patterns to be formed, and let Γ be the initial configuration of the robots. A set of robots are said to form $\mathcal{S}$ if they form the infinite periodic sequence $\mathcal{S}^\infty = \langle S_0, S_2, \ldots, S_{m-1}\rangle^\infty$, starting from an arbitrary pattern $P \in \mathcal{S}$.

6.2 Solutions

Not surprisingly, both with and without lights, the sequences of patterns formable by the robots highly depend on symmetry considerations. Intuitively, in the model of robots without lights, it is not possible to form sequences where the patterns have different levels of symmetry ("symmetricity"). The reason is that robots located in symmetric points have the same view of the environment and they can be forced to behave exactly in the same way, never breaking their symmetry class. In particular the "symmetricity" of each pattern must divide that of the starting configuration.

Without lights, a sequence of patterns can be formed by rigid robots in `SSYNC` if and only if *i*) the level of symmetry of each pattern is the same; *ii*) the patterns contain all the same number of points; *iii*) no pattern can appear more than once in the sequence [4].

For luminous robots, a quite different characterization holds for the weaker `ASYNC` model. The "symmetricity" of each pattern must still divide the one of the starting configuration. However, in striking contrast with the previous case, a sequence of patterns can be formed by luminous robots also allowing: *i*) the level of symmetry of each pattern to change; *ii*) the patterns to contain different numbers of points (*contractions*); *iii*) the sequence to contain repeated patterns (*repetitions*) [5].

The availability of lights is clearly very powerful, and it is exploited mainly to allow symmetry breaking in various situations: for example, to permanently identify a set of leaders, to color robots that need to move on the same point for forming a pattern and later need to move away from each other, to color differently instances of a same pattern that repeats in the sequence. Interestingly, the necessary number of colors $f(\mathcal{S})$ for forming sequence $\mathcal{S}$, although difficult to express in a closed formula, is easily computable and bounded as follows:

$$\mu(\mathcal{S})^{\frac{1}{\alpha_M(\mathcal{S})}} \cdot (\lceil \alpha_M(\mathcal{S})/\alpha_m(S_i)\rceil) > f(\mathcal{S}) \geq \max\{\mu(\mathcal{S})^{\frac{1}{\alpha_M(\mathcal{S})}}, \lceil \alpha_M(\mathcal{S})/\alpha_m(S)\rceil\}$$

where $\mu(\mathcal{S})$ is the maximum number of occurrences of some pattern in $\mathcal{S}$, and $\alpha_M(\mathcal{S})$ (resp $\alpha_m(\mathcal{S})$) is the maximum (resp. minimum) number of classes of robots with the same view of the environment ("symmetricity") in some pattern in $\mathcal{S}$.

Theorem 11. *[5] Any sequence of patterns $\mathcal{S}$ can be performed by robots with $O(f(\mathcal{S}))$ colors.*

7 Conclusions

In this paper we have reviewed the existing results on computations by luminous robots. The investigations have just started and the current results leave many new research directions and open problems.

First of all, the computational picture of the power of luminous robots is not fully understood because some gaps still exist: for example, the relationship between `ASYNC` luminous robots and `FSYNC` non-luminous ones is still unknown. It is clear that the availability of lights highly increases the robots' capabilities; what is not clear is the impact that other assumptions have in combination with lights. For example, rigidity of movement seems to be crucial in some settings for the robots to exploit the presence of lights: this is especially true for lights that provides only internal states (FState) and lights that are instead only visible to others (FComm), but a formal proof is still missing. An other important question left unanswered is whether there exists a computational difference between FState and FComm: is it more useful for the robots to remember or to communicate ?

Acknowledgements. I would like to thank the researchers who have collaborated with me in the investigations on luminous robots: Shantanu Das, Giuseppe Di Luna, Sruti Gan Chaudhuri, Federico Poloni, Giuseppe Prencipe, Nicola Santoro, Giovanni Viglietta and Masafumi Yamashita

References

1. M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing by mobile robots: Gathering. *SIAM Journal on Computing*, 41(4): 829-879, 2012.
2. R. Cohen and D. Peleg. Convergence properties of the gravitational algorithm in asynchronous robot systems. *SIAM Journal on Computing*, 34:1516–1528, 2005.
3. R. Cohen and D. Peleg. Local spreading algorithms for autonomous robot systems. *Theoretical Computer Science*, 399:71–82, 2008.
4. S. Das, P. Flocchini, N. Santoro, and M. Yamashita. Forming sequences of geometric patterns with oblivious mobile robots. *Distributed Computing*, in press, 2015.
5. S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. Synchronized dancing of oblivious chameleons. In *7th Int. Conference on FUN with Algorithms*, 113-124, 2014.
6. S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. The power of lights: synchronizing asynchronous robots using visible bits. In *32nd Int. Conference on Distributed Computing Systems* (ICDCS), 506–515, 2012.
7. Y. Dieudonné, and F.Petit. Scatter of weak mobile robots. *Parallel Processing Letters*,19(1):175–184, 2009.
8. Y. Dieudonné and F. Petit. Self-stabilizing gathering with strong multiplicity detection. *Theoretical Computer Science*, 428(13), 2012.

9. G.A. Di Luna, P. Flocchini, S. Gan Chaudhuri, N. Santoro, and G. Viglietta. Robots with lights: overcoming obstructed visibility without colliding. In *16th Int. Symposium on Stabilization, Safety, and Security of Distributed Systems* (SSS), 150–164, 2014.
10. G.A. Di Luna, P. Flocchini, F. Poloni, N. Santoro, and G. Viglietta. How oblivious mobile robots can achieve mutual visibility. In *Proc. of 26th Canadian Computational Geometry Conference* (CCCG), 2014.
11. G.A. Di Luna, P. Flocchini, S. Gan Chaudhuri, F. Poloni, N. Santoro, and G. Viglietta. Mutual visibility by luminous robots without collisions. http://arxiv.org/abs/1503.04347, 2015.
12. P. Flocchini, G. Prencipe, and N. Santoro. *Distributed Computing by Oblivious Mobile Robots*. Morgan & Claypool, 2012.
13. P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Arbitrary pattern formation by asynchronous oblivious robots. *Theoretical Computer Science* 407(1-3), 412–447, 2008.
14. P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science*, 337(1-3):147–168, 2005.
15. P. Flocchini, N. Santoro, G. Viglietta, and M. Yamashita. Rendezvous of two robots with constant memory. In *20th Int. Colloquium on Structural Information and Communication Complexity* (SIROCCO), 189–200, 2013.
16. N. Fujinaga, Y. Yamauchi, S. Kijima and M. Yamashita. Asynchronous pattern formation by anonymous oblivious mobile robots. In *26th Int. Symposium on Distributed Computing* (DISC), 312-325, 2012.
17. T. Izumi, M. Gradinariu Potop-Butucaru, and S. Tixeuil. Connectivity-preserving scattering of mobile robots with limited visibility. In *12th Int. Symposium on Stabilization, Safety, and Security of Distributed Systems* (SSS), 319–331, 2010.
18. J. Lin, A.S. Morse, and B.D.O. Anderson. The multi-agent rendezvous problem. parts 1 and 2. *SIAM Journal on Control and Optimization*, 46(6):2096–2147, 2007.
19. D. Peleg, Distributed coordination algorithms for mobile robot swarms: new directions and challenges. In *7th Int. Workshop on Distributed Computing* (IWDC), 1–12, 2005.
20. G. Prencipe. The effect of synchronicity on the behavior of autonomous mobile robots. *Theory of Computing Systems*, 38(5):539–558, 2005.
21. I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, vol. 28, pp. 1347–1363, 1999.
22. R. Vaidyanathan, C. Busch, J. Trahan, G. Sharma, and S. Rai. Logarithmic-time complete visibility for robots with lights. In *29th IEEE Int. Parallel and Distributed Processing Symposium* (IPDPS), 2015.
23. G. Viglietta. Rendezvous of two robots with visible bits. In *9th Int. Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics* (ALGOSENSORS), 291-306, 2013.
24. M. Yamashita and I. Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theoretical Computer Science*, 411(26-28):2433 – 2453, 2010.