Autonomous Mobile Robots with Lights*

Shantanu Das† Paola Flocchini‡ Giuseppe Prencipe§ Nicola Santoro¶
Masafumi Yamashita‖

Abstract

We consider the well known distributed setting of computational mobile entities, called robots, operating in the Euclidean plane in Look-Compute-Move (LCM) cycles. We investigate the computational impact of expanding the capabilities of the robots by endowing them with an externally visible memory register, called light, whose values, called colors, are persistent, that is they are not automatically reset at the end of each cycle. We refer to so endowed entities as luminous robots.

We study the computational power of luminous robots with respect to the three main settings of activation and synchronization: fully-synchronous, semi-synchronous, and asynchronous. We establish several results. A main contribution is the constructive proof that asynchronous robots, illuminated with a constant number of colors, are strictly more powerful than traditional semi-synchronous robots.

We also constructively prove that, for luminous robots, the difference between asynchrony and semi-synchrony disappears. This result must be contrasted with the strict dominance between the models without lights (even if the robots are enhanced with an unbounded amount of persistent internal memory).

Additionally we show that there are problems that robots cannot solve without lights, even if they are fully-synchronous, but can be solved by asynchronous luminous robots with $O(1)$ colors. It is still open whether or not asynchronous luminous robots with $O(1)$ colors are more powerful than fully-synchronous robots without lights.

We prove that this is indeed the case if the robots have the additional capability of remembering a single snapshot. This in turn shows that, for asynchronous robots, to have $O(1)$ colors and a single snapshot renders them more powerful than to have an unlimited amount of persistent memory (including snapshots) but no lights.

*Some of these results have been presented at the 32nd International Conference on Distributed Computing Systems (ICDCS).

†LIF, Aix-Marseille University & CNRS, France. Email: shantanu.das@lif.univ-mrs.fr

‡SEECS, University of Ottawa, Canada. Email: flocchin@site.uottawa.ca

§Dipartimento di Informatica, Università di Pisa, Italy. Email: prencipe@di.unipi.it

¶School of Computer Science, Carleton University, Canada. Email: santoro@scs.carleton.ca

‖Department of Informatics, Kyushu University, Japan. Email: mak@inf.kyushu-u.ac.jp

1 Introduction

1.1 Framework

Consider a distributed system composed of a team of mobile computational entities, called *robots*, placed in $\mathbb{R}^2$ where they can freely move, each endowed with a personal coordinate system, and operating in *Look-Compute-Move* (LCM) cycles. During a cycle, a robot first obtains a snapshot of the positions of the robots in its own local coordinate system (*Look*); using the snapshot as an input, the robot then executes a protocol, the same for all robots, to compute its destination (*Compute*); finally, moves towards the computed destination, if different from its current location (*Move*). After each cycle, a robot may be inactive for some time. The robots are *autonomous* (i.e., without a central control); they are *silent* (i.e., they have no means of direct communication of information to other robots); and typically they are *oblivious* (i.e., at the beginning of a cycle, a robot has no memory of any observation or computation of its previous cycles). What is computable in such systems has been the object of extensive research within distributed computing. The main goal has been to understand the computational limitations and powers of these robots for performing basic coordination tasks, and the focus has been on basic problems, such as *Pattern Formation* (e.g., [18, 20, 27, 29, 31]), *Gathering* (e.g., [1, 2, 5, 6, 7, 17, 27]), *Flocking* (e.g., [4, 21]), *Surrounding* (e.g., [11, 12, 16]), *Election* (e.g., [13]), etc. For a recent review see [15].

As in other areas of distributed computing, major computational differences exist depending on the level of synchronization of the computational entities. In the literature, three basic settings have been studied: *fully-synchronous* (`FSYNC`), *semi-synchronous* (`SSYNC`), and *asynchronous* (`ASYNC`). In `FSYNC`, the activations of all robots can be logically divided into global rounds; in each round, the robots are all activated, obtain the same snapshot, compute and perform their move; note that this is computationally equivalent to a fully synchronized system in which all robots are activated simultaneously and all operations are instantaneous. The semi-synchronous model `SSYNC` is like `FSYNC` where however not all robots are necessarily activated in each round. In `ASYNC`, there is no common notion of time, the robots are activated independently, and the duration of each activity and inactivity is finite but unpredictable.

The exploration of the limits of what is computable by such robots is ongoing; the research results obtained so far have greatly increased our basic knowledge and understanding of the relationship between the computability of the three synchronization settings. In particular, there are problems that are solvable in `FSYNC` but not in `SSYNC` (e.g. [27]) and, for non-oblivious robots, there are problems that are solvable in `SSYNC` but not in `ASYNC` (e.g. [25]); that is, the computational hierarchy (formalized later) between the three settings is strict:

$$\texttt{FSYNC} > \texttt{SSYNC} > \texttt{ASYNC}. \tag{1}$$

Among the basic questions, researchers have been investigating the impact that *additional* robots' capabilities have on the computational power of the system. In particular, a pressing question is what additional power would allow the robots to overcome the inherent difficulties and weaknesses of the `ASYNC` setting, the weakest one.

In this paper, we analyze the impact of enhancing the computational power of the entities with a simple mechanism for communication and memory. More precisely, each robot is equipped with a constant-sized register (called *light*) whose value (called *color*) is visible to all robots; the lights are persistent, that is, they are not automatically reset at the end of each cycle. Thus, in this new model of *luminous robots*, initially suggested by Peleg [14, 24], the entities are capable in each cycle of remembering (because of persistency) and communicating (because of visibility)

a constant number of bits. We denote this additional capability of the robots using a superscript representing the number of colors. Thus, $\texttt{ASYNC}^i$ denotes the ASYNC model when each robot is augmented by a light with i colors.

1.2 Main Contributions

In this paper, we study the computational relationship between luminous robots and the traditional models of robots without lights, focusing in particular on the impact lights have on asynchrony.

We prove that asynchrony with a constant number of colors is strictly more powerful than traditional semi-synchrony: $\texttt{ASYNC}^{O(1)} > \texttt{SSYNC}$. This result is obtained in two steps.

We first prove that, with three bits of externally visible memory, any problem solvable in SSYNC can be solved also in ASYNC. The proof is constructive: we design a $\texttt{ASYNC}^5$ protocol and prove that, for any given SSYNC protocol $\mathcal{P}$, our protocol produces a semi-synchronous execution of $\mathcal{P}$. We then show that there are problems unsolvable in SSYNC that can however be solved by asynchronous robots augmented with two bits of externally visible memory. Also this proof is constructive: we present a $\texttt{ASYNC}^4$ protocol that solves the rendezvous of two oblivious robots, a problem that is not solvable in SSYNC.

We also show that, when enhanced with visible lights, the difference between asynchrony and semi-synchrony disappears; in fact we prove that $\texttt{ASYNC}^{O(1)} \equiv \texttt{SSYNC}^{O(1)}$. This result must be contrasted with the strict dominance $\texttt{ASYNC} < \texttt{SSYNC}$ in absence of lights (see Expression 1). Summarizing, we prove that

$$\texttt{ASYNC}^{O(1)} \equiv \texttt{SSYNC}^{O(1)} > \texttt{SSYNC} \tag{2}$$

It should be noted that, as part of the proof, we provide a tool that allows designers of protocols for luminous robots to concentrate only on the semi-synchronous model rather than having to face the much greater difficulties of complete asynchrony. The price to pay is the increase by a constant factor in the number of colors.

As for the computational relationship between asynchrony with lights and full synchrony without lights, we show that there are problems solvable in $\texttt{ASYNC}^{O(1)}$ that cannot be solved in FSYNC. In other words, we prove that having a few visible bits allows asynchronous robots to perform tasks that are impossible even for fully-synchronous robots. It is still open whether $\texttt{ASYNC}^{O(1)}$ and FSYNC are incomparable, or $\texttt{ASYNC}^{O(1)} > \texttt{FSYNC}$.

Motivated by the possibility that $\texttt{ASYNC}^{O(1)}$ and FSYNC are incomparable, we also consider the additional power of remembering snapshots. We constructively prove that, if illuminated with a constant number of colors, asynchronous robots that can remember only a single snapshot are strictly more powerful than traditional fully-synchronous robots. This must be contrasted with the well known fact that, without lights, asynchronous robots are less powerful than semi-synchronous robots, even if they remember an unlimited number of previous snapshots. This also implies that, for asynchronous robots, having $O(1)$ colors and a single snapshot renders them more powerful than having an unlimited amount of persistent memory (including snapshots) but no lights.

2 The Model

The system is composed of a team of mobile entities, called *robots*, each modelled as a computational unit provided with its own local memory and capable of performing local computations.

The robots are placed in a spatial universe, here assumed to be $\mathbb{R}^2$, and they are viewed as points in $\mathbb{R}^2$. We assume that the robots always start from distinct points in the plane, but during the course of the algorithm multiple robots may occupy the same point in $\mathbb{R}^2$. Each robot has its own local coordinate system. A robot is endowed with sensorial capabilities and it observes the world by activating its sensors, which return a snapshot of the positions of the other robots in its local coordinate system.

The robots are *identical*: they are indistinguishable by their appearance and they execute the same protocol. The robots are *autonomous*, without a central control. The robots are *silent*, meaning that a robot cannot directly talk to another robot (e.g. using wireless communication). The robots can however communicate information indirectly e.g. using specific movements or using lights, as explained later.

Each robot is endowed with motorial capabilities, and can freely move in the plane.

At any point in time, a robot is either *active* or *inactive*. When *active*, a robot r executes a *Look-Compute-Move* (LCM) cycle performing the following three operations, each in a different state:

(i) **Look:** The robot observes the world by activating its sensor, which returns a snapshot of the positions of all robots with respect to its own coordinate system (since robots are viewed as points, their positions in the plane are just the set of their coordinates).

(ii) **Compute:** The robot executes its algorithm, using the snapshot as input. The result of the computation is a destination point.

(iii) **Move:** The robot moves towards the computed destination; if the destination is the current location, the robot stays still (performs a *null movement*).

When *inactive*, a robot is idle. All robots are initially inactive. The amount of time to complete a cycle is assumed to be finite, and the *Look* is assumed to be instantaneous.

The robots may or may not have distinct identities; if they are without identifiers that can be used during the computation they are said to be *anonymous*. The robots may or may not have *persistent* memory of the past, that is memory whose content is preserved from one cycle to the next; they are said to be *oblivious* if they do not, in which case they start each cycle without any information on the past. The local coordinate systems of the robots may or may not be consistent with each other. In a move, a robot might always reach its destination, in which case the move is said to be *rigid*; alternatively, the move might stop before reaching its destination, e.g., because of limits to its motion energy. In the non-rigid case, the distance traveled in a move is neither infinite nor infinitesimally small. More precisely, there exists an (arbitrarily small) constant $\delta > 0$ such that, if the destination point is closer than δ, the robot will reach it; otherwise, it will move towards it a distance of at least δ; the quantity δ might not be known to the robots.

We denote by $\mathcal{R}$ the set of all teams of robots satisfying the core assumptions (i.e., they are identical, silent, autonomous, and operate in LCM cycles), and denote by $R \in \mathcal{R}$ a team of robots having identical capabilities (e.g., common coordinate system, persistent storage, internal identity, rigid movements etc.). By $\mathcal{R}_n \subset \mathcal{R}$ we denote the set of all teams of size n.

With respect to the activation schedule of the robots and their *Look-Compute-Move* cycles, the most common *models* are the fully-synchronous, the semi-synchronous, and the asynchronous. In the *asynchronous* (`ASYNC`) model, the robots do not have a common notion of time; they are activated independently from each other, and the duration of each *Compute*, *Move* and inactivity is finite but unpredictable. As a consequence, robots perceived in a snapshot might

be performing any operation and, in particular, they might be moving; moreover, computations might be made based on obsolete observations. On the opposite side of the spectrum, in the *fully-synchronous* (FSYNC) model, the activations of all robots can be logically divided into global rounds; in each round, the robots are all activated, obtain the same snapshot, compute and perform their move. Note that this is computationally equivalent to a fully synchronized system in which all robots are activated simultaneously and all operations are instantaneous. The *semi-synchronous* (SSYNC) model is like the fully-synchronous model where however not all robots are necessarily activated in each round. Based on the fairness of the activation scheduler, sub-models can be obviously defined; we assume that every robot is activated infinitely often.

Given a model $X \in \{\texttt{FSYNC}, \texttt{SSYNC}, \texttt{ASYNC}\}$ and a team of robots $R \in \mathcal{R}$, let $Task(X, R)$ denote the set of problems solvable by R in X.

Given two models X and Y, we say that X is *computationally not less powerful than* Y, denoted by $X \geq Y$, if $\forall R \in \mathcal{R}$ we have $Task(Y, R) \subseteq Task(X, R)$.

We say that X is *computationally more powerful than* Y, denoted by $X > Y$, if $X \geq Y$ and $\exists R \in \mathcal{R}$ such that $Task(X, R) \setminus Task(Y, R) \neq \emptyset$.

If $X \geq Y$ and $Y \geq X$, we say that X and Y are *computationally equivalent*, denoted by $X \equiv Y$.

Finally, if $\exists R \in \mathcal{R}$ such that $Task(X, R) \setminus Task(Y, R) \neq \emptyset$ and $\exists R \in \mathcal{R}$ such that $Task(Y, R) \setminus Task(X, R) \neq \emptyset$, we say that X and Y are *orthogonal* (or *incomparable*), denoted by $X \perp Y$.

For simplicity of notation, let $\mathcal{A}(R)$, $\mathcal{S}(R)$, and $\mathcal{F}(R)$ denote $Task(\texttt{ASYNC}, R)$, $Task(\texttt{SSYNC}, R)$, and $Task(\texttt{FSYNC}, R)$, respectively.

A *luminous* robot r is a robot that, in addition to its capabilities, is endowed with a persistent and externally visible state variable $Light[r]$, called *light*, whose values are from a finite set C of states called *colors* (including the color that represents the state when the light is off). Note that if $|C| = 1$, then the light is always off; thus, this case corresponds to the traditional setting of robots without lights. In this paper, we assume $|C| > 1$ for (truly) luminous robots. 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. A light is *persistent* from one computational cycle to the next; while the robot r might be oblivious forgetting all other information from previous cycles, the color is not automatically reset at the end of a cycle.

We denote the availability of lights using a superscript representing the number of colors. In particular, we denote by X^i the model $X \in \{\texttt{ASYNC}, \texttt{SSYNC}, \texttt{FSYNC}\}$ when every robot is enhanced by a light with $i > 1$ colors. We denote by $\mathcal{A}^i(R)$, $\mathcal{S}^i(R)$, and $\mathcal{F}^i(R)$ the class of problems in $\texttt{ASYNC}^i$, $\texttt{SSYNC}^i$, $\texttt{FSYNC}^i$, respectively, solvable by team R when each $r \in R$ is enhanced by a light $Light[r]$ with $i > 1$ colors.

3 Luminous Asynchrony versus Semi-Synchrony

3.1 $\texttt{ASYNC}^{O(1)}$ is at least as powerful as SSYNC

In this section we show that asynchronous systems equipped with a light with $O(1)$ colors are at least as powerful as semi-synchronous systems without lights. More precisely, we have:

Theorem 3.1. *$\forall R \in \mathcal{R}$, $\mathcal{S}(R) \subseteq \mathcal{A}^5(R)$.*

The proof is constructive: we present a $\texttt{ASYNC}^5$ protocol Sim that produces a semi-synchronous execution of any SSYNC protocol $\mathcal{P}$.

State *Look*

- `Pos[r]`, the position on the plane of robot r (according to my coordinate system);
- `Light[r]`, the color of the light of robot r.

(**Note**: I am robot x)

State *Compute*

$p :=$ `Pos[x]`.
Case `Light[x]`:

- `T`
  If $\forall r \neq x$, `Light[r]` $\in \{$`T`, `S`$\}$ **Then**
  Execute $\mathcal{P}$.
  $p :=$ computed destination.
  `Light[x]` $:=$ `M`.
  If $(\exists r \neq x$ `Light[r]` $\in \{$`M`$\})$ **Then**
  `Light[x]` $:=$ `W`.
- `M`
  If $\forall r \neq x$, `Light[r]` $\notin \{$`T`$\}$ **Then**
  `Light[x]` $=$ `S`.
- `S`
  If $\forall r \neq x$, `Light[r]` $\in \{$`S`, `F`$\}$ **Then**
  `Light[x]` $=$ `F`.
- `F`
  If $\forall r \neq x$, `Light[r]` $\in \{$`F`, `T`$\}$ **Then**
  `Light[x]` $=$ `T`.
- `W`
  If $\forall r \neq x$, `Light[r]` $\notin \{$`M`$\}$ **Then**
  `Light[x]` $=$ `T`.

State *Move*

`Move(`p`)`.

Figure 1: Protocol Sim

3.1.1 Algorithm Description

The rules of the protocol are presented in Figure 1, while Figure 2 shows the transition diagram as the robots change colors.

The lights used by Sim 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 $\mathcal{P}$ (color `T`) and ends with all robots finishing the Mega-Cycle having executed $\mathcal{P}$ once (color `F`). All robots with light `F` then eventually turn their lights to `T`; when this process is completed, a new Mega-Cycle starts.

During each such Mega-Cycle, each robot gets a chance to execute one step of protocol $\mathcal{P}$. A robot r colored `T`, tries to execute one LCM cycle of protocol $\mathcal{P}$. However, the robot is allowed to move only if during its *Look* operation there were no robots colored `M` (i.e. robots that are potentially moving). If that is the case, robot r changes its color to `M` during the *Compute* part

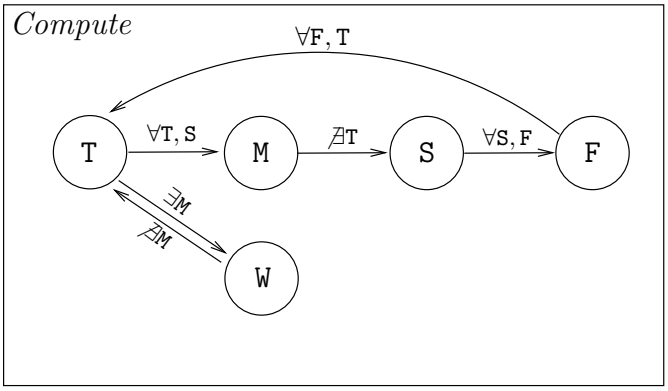

Figure 2: Transition diagram of protocol Sim. The label in the nodes represents the color of the light of the executing robot. The label of an edge expresses the condition on the lights of all the other robots that must be satisfied for the transition to occur. The notation "$\forall A, B$" means: "$\forall r \in R$, $\texttt{Light}[\texttt{r}] \in \{A, B\}$" (similarly for $\exists A$ and $\nexists A$).

of this cycle. On the other hand, if robot r sees any robot colored M during its *Look*, then it is not allowed to move; it changes its color to W and waits for the next turn (i.e., it waits until no robots are colored M). The robots colored M, after executing $\mathcal{P}$, will turn their own lights to S only when they see that no robot is colored T. Thus, after some time, each robot will be colored either S or W (the former are the robots that executed one step of $\mathcal{P}$ and the latter are those who missed their turn). The robots colored W will now turn their lights to T again and these robots get another chance to execute $\mathcal{P}$. Thus, a Mega-Cycle of protocol Sim consists of several stages such that, in each stage, a non-empty set of robots execute $\mathcal{P}$ while the other robots wait. Eventually all robots will be colored S (i.e., each robot has executed $\mathcal{P}$ once) and the Mega-Cycle ends with all robots changing to color F. At this point, all robots with light F turn their lights to T; when this process is completed, a new Mega-Cycle starts. Essentially, the transition of lights from T to W and back to T corresponds to a queue where robots that failed to enter the current stage wait for their turn.

3.1.2 Correctness

We now prove that Protocol Sim provides a fair and correct execution of any semi-synchronous protocol $\mathcal{P}$. We first define the concepts of *Mega-Cycles* and *Stages*, in terms of the global configuration of the robots. A *Mega-Cycle* begins at the time instant t when all robots are colored T, and the Mega-Cycle ends at the earliest time instant $t' > t$ when all robots are colored F. We shall show that, after a Mega-Cycle, the robots eventually return to color T so that the next Mega-Cycle can begin. A *Stage* of a Mega-Cycle begins at time t_0 when all robots are colored T or S (with at least one robot colored T). During the stage, some robots change to color M and the stage ends at the earliest subsequent time $t_2 > t_0$ when all robots are again colored T or S.

Lemma 3.1. *During each stage, the following holds:*
(i) At the beginning of the stage, one or more robots are colored T *and all others (if any) are colored* S*. (ii) At least one of those robots colored* T *executes $\mathcal{P}$ during this stage. (iii) All robots that execute $\mathcal{P}$ during this stage have the same snapshot. (iv) At the end of this stage, those*

robots that executed $\mathcal{P}$ in this stage are colored `S`*. (v) At the end of this stage, all robots are colored* `T` *or* `S`*.*

Proof. Parts (i) and (v) follow directly from the definition. Let t_0 be the earliest time when all robots were colored `T` or `S` (e.g. at the beginning), and let S be the snapshot at that time. Let $t_2 > t_0$ be the first time since t_0 when a robot changes from color `T` to color `M` (according to the rules of SIM this is the only possibility when all robots are colored `T`). Notice that this robot must have performed the *Look* operation at time t_1 such that $t_0 \leq t_1 < t_2$. Since during the period (t_0, t_2) the global configuration does not change, every robot that performs *Look* between t_0 and t_2 has the same snapshot S, and in this snapshot all robots are colored `T` or `S`. So, all these robots would eventually turn to `M` (and execute $\mathcal{P}$). Any robot r that performs *Look* at a time $\geq t_2$, would see some robot colored `M` and thus robot r would change to color `W` (such a robot does not execute $\mathcal{P}$ in this stage). This proves parts (ii) and (iii). Every robot will be eventually activated and will change to either `M` or `W` (depending on the time when they performed *Look*). At this point in time, no robot would have color `T`. According to the rules of the algorithm, those robots that are colored `M` and have executed one step of $\mathcal{P}$, would now change to color `S`. When all these robots have become `S`, the other robots that were colored `W` would now change to `T` again. This is the end of this stage and conditions (iv) and (v) hold at this time.

□

Lemma 3.2. *For each Mega-Cycle, the following holds:*
(i) At the beginning of a Mega-Cycle, all robots are colored `T`*. (ii) During the Mega-Cycle, each robot executes $\mathcal{P}$ exactly once. (iii) Each Mega-Cycle ends within a finite time. (iv) At the end of the Mega-Cycle all robots are colored* `F`*.*

Proof. Due to Lemma 3.1 we know that, in each stage, a non-empty subset of the robots execute $\mathcal{P}$ and these robots have the same snapshot. Thus each stage of protocol SIM corresponds to one activation round of an execution of $\mathcal{P}$ in the `SSYNC` model where the set of robots activated in that round corresponds to the set of robots that changed color from `T` to `M`, and eventually to `S`, during this stage of SIM. As long as there are more robots colored `T`, another stage of the algorithm SIM can begin. With each stage, the number of robots colored `T` decreases. Eventually all robots will be colored `S` and thus, each robot will have executed $\mathcal{P}$ exactly once. Now, according to the rules of algorithm SIM, the robots change color to `F`. At some time t, every robot will be colored `F`. At this time, the Mega-Cycle ends and the conditions of the Lemma 3.2 are satisfied. □

We can now prove the main result of this section.

Theorem 3.2. *Protocol* SIM *is correct, i.e. any execution of protocol* SIM *in* `ASYNC`^{5} *corresponds to a possible execution of $\mathcal{P}$ in* `SSYNC`*.*

Proof. First notice that after each Mega-Cycle, all robots are colored `F` (Lemma 3.2) and thus, according to the rules of protocol SIM, every robot would change color to `T`. Hence, after each Mega-Cycle ends, the next Mega-Cycle begins automatically. Since each Mega-Cycle terminates in finite time, the execution of protocol SIM is an infinite sequence of Mega-Cycles.

We have already shown that each stage within a Mega-Cycle of protocol SIM corresponds to a semi-synchronous activation round in some execution of $\mathcal{P}$ in the `SSYNC` model. We now need to show that the sequence of such activation rounds satisfies the fairness property. The fairness

property requires that in any infinite execution of $\mathcal{P}$, each robot must be activated infinitely often. We have shown that in each Mega-Cycle, each robot actively executes $\mathcal{P}$ once. Thus in any infinite execution, i.e. an infinite sequence of Mega-Cycles, each robot executes $\mathcal{P}$ infinitely many times. Hence, this execution simulates a possible execution of protocol $\mathcal{P}$. In fact, this particular execution also satisfies the stronger condition of 1-fairness.

Note that if protocol $\mathcal{P}$ is a terminating algorithm (i.e., it terminates for every execution), then during the simulation, the robots would stop moving in finite time. □

The above result provides the proof of Theorem 3.1.

3.2 $\texttt{ASYNC}^{O(1)}$ is more powerful than SSYNC

In the previous section we have shown that $\texttt{ASYNC}^{O(1)} \geq \texttt{SSYNC}$; that is, asynchronous robots, if illuminated with $O(1)$ colors, are at least as powerful as if they were semi-synchronous.

In this section we show that there are problems that robots cannot solve without visible bits, even if they are semi-synchronous and have an unbounded amount of persistent internal memory, but can be solved by asynchronous luminous robots with $O(1)$ colors.

Consider the extensively investigated *gathering* or *rendezvous* problem Gathering{k} of having k robots terminally gather in the same location, not previously known in advance. It is well known that the gathering of two oblivious anonymous robots cannot be guaranteed in the SSYNC model without any additional assumption:

Lemma 3.3 ([27])**.** $\exists R \in \mathcal{R}_2$, Gathering$\{2\} \notin \mathcal{S}(R)$.

We now prove that two anonymous robots can gather, when enabled with $O(1)$ visible lights, even if they have no other capabilities.

Theorem 3.3. $\forall R \in \mathcal{R}_2$, Gathering$\{2\} \in \mathcal{A}^4(R)$.

We prove the theorem constructively. Let x and y be the two robots each provided with a light with four colors and with no additional capabilities (that is, they are anonymous, oblivious, movements are non rigid, there is no consistency of coordinate systems, ...). Consider protocol TwoGatherLight shown in Figure 3. The protocol uses four colors: WHITE, RED, GREEN, and BLUE; initially the lights of both x and y are set to WHITE. The idea behind the protocol is as follows. If, after the beginning of the execution, both robots observe WHITE as the color of the other robots' light, then they both try to reach the point halfway between the two robots. On the other hand, if one robot begins execution earlier than the other, it will move towards the midpoint, turning its light RED before moving. If the second robot now performs a *Look* operation, it will see the RED light and know that the other robot is potentially moving. In this case the second robot waits for the first robot to change color from RED to BLUE.

When a robot sees the BLUE light on the other robot, it will try to move directly towards it. A robot with BLUE light waits until the second robot has also turned its light to BLUE. When both robots have BLUE lights, they turn their lights to GREEN to signal the end of one round of the algorithm (i.e., the robots synchronize with each-other at the end of each round). Now the robots turn their lights to WHITE to start the next round. As before, we will use the term Mega-Cycle to refer to the time period during which the robot has its light exactly once in each color starting from WHITE to RED, BLUE, GREEN and just before turning to WHITE again.

Based on the rules of the algorithm TwoGatherLight, the following properties can be shown:

Lemma 3.4. *In the execution of Algorithm* TwoGatherLight*:*

State *Look*

```
Pos[x] :=my current position;
Pos[y] := position of the other robot;
Light[x] := Value of my light;
Light[y] := Value of the light of the other robot.
```

State *Compute*

```
If Gather Then STOP.
p := Pos[x].
Case Light[x]:
  • WHITE
    If Light[y] = WHITE Then
      p := Half point between me and the other robot;
      Light[x] := RED.
    Else If Light[y] = BLUE Then
      p := Position of the other robot;
      Light[x] := RED.
  • RED
    Light[x] := BLUE.
  • BLUE
    If Light[y] ∈ {BLUE, GREEN} Then
      Light[x] := GREEN.
  • GREEN
    If Light[y] ∈ {GREEN, WHITE} Then
      Light[x] := WHITE.
```

State *Move*

```
Move(p).
```

Figure 3: TwoGatherLight, a protocol for gathering two robots in ASYNC^4.

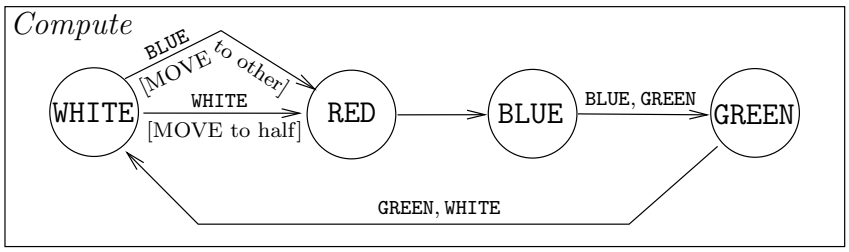

Figure 4: Transition diagram of protocol TwoGatherLight. The label of a node denotes the color of the executing robot; the label above an arc denotes the color of the other robot and the label below (if any) denotes the action corresponding to this transition.

(i) Unless the robots are already gathered, each Mega-Cycle completes in finite time (i.e. there are no deadlocks).

(ii) If the distance between the robots, $dist(x, y) > 2\delta$, then after each complete Mega-Cycle this distance decreases by at least 2δ and the robots never cross each-other.

(iii) If $dist(x, y) \leq 2\delta$, then the robots gather during the next Mega-Cycle.

Proof.

(i) An activated robot x stays in its current state if either (1) `Light[x]` is `WHITE` and `Light[y]` is `RED` or `GREEN`, or (2) `Light[x]` is `BLUE` and `Light[y]` is `RED` or `WHITE`, or (3) `Light[x]` is `GREEN` and `Light[y]` is `RED` or `BLUE`. Note that none of this three conditions can hold for more than one activation cycle for each robot. Thus, there will be no deadlock.

(ii) It is easy to see that during a complete Mega-Cycle, each robot is guaranteed to move (unless the robots have already gathered). Let robot x be the first robot to start moving during this Mega-Cycle. Let d be the distance between the robots at this time. Robot x may move only if `Light[y]` is `WHITE` or `BLUE` during the *Look* operation. If `Light[y]` is `BLUE`, then robot y has already moved during this Mega-Cycle contradicting the assumption. Thus, `Light[y]` is `WHITE`. Thus, robot x moves towards robot y by at least distance δ and at most distance $d/2$ during this Mega-Cycle. The distance moved by robot y depends on what robot y sees during the *Look* operation performed when its light was `WHITE`. If `Light[x]` was `WHITE` at that time, robot y moves at most by $d/2$ towards robot x (so they do not cross). Otherwise, if `Light[x]` was `BLUE`, then robot x has already finished moving and robot y moves a distance of d' which is at most the current distance between the robots. Thus, the robots do not cross each other. In both cases, the distance between the robots after the Mega-Cycle is at most $d - 2\delta$.

(iii) Let $d \leq 2\delta$ be the distance between the two robots and let p be the midpoint between the two locations. Without loss of generality, let x be the first robot to perform *Look* operation in the next Mega-Cycle. Robot x would decide to move a distance $d/2 \leq \delta$ and thus it would eventually arrive at location p. If robot y sees robot x when `Light[x]` is `WHITE`, then robot x has not moved yet and the distance between the robots is still d. Thus, robot y will also decide to move a distance $d/2$ and will eventually reach p.

The only other case is when robot y sees robot x when `Light[x]` is `BLUE`. In this case, robot x has already arrived at p. During the *Compute* operation, robot y will decide to

move directly to the other robot and it will also reach location p. Hence in all cases, the robots will gather at p during this Mega-Cycle.

□

We have shown that algorithm TwoGatherLight correctly solves the problem of gathering two robots when provided with a light of 4 colors. This completes the proof of Theorem 3.3.

By Theorem 3.1, Lemma 3.3, and Theorem 3.3 it follows that $\texttt{ASYNC}^{O(1)} \geq \texttt{SSYNC}$ and $\exists R \in \mathcal{R}$, $\mathcal{A}^{O(1)}(R) \setminus \mathcal{S}(R) \neq \emptyset$; that is,

Theorem 3.4. $\texttt{ASYNC}^{O(1)} > \texttt{SSYNC}$.

3.3 $\texttt{ASYNC}^{O(1)}$ is as powerful as $\texttt{SSYNC}^{O(1)}$

To complete this section we now prove that, when enhanced with a constant number of visible bits, luminous semi-synchronous robots are not more powerful than luminous asynchronous ones. We first show that $\mathcal{S}^{O(1)}(R) \subseteq \mathcal{A}^{O(1)}(R)$; more precisely.

Theorem 3.5. $\forall R \in \mathcal{R}, \forall k > 1$, $\mathcal{S}^k(R) \subseteq \mathcal{A}^{5k}(R)$

The proof is constructive. Let $\mathcal{P}$ be a protocol designed for the $\texttt{SSYNC}^k$ model. We show how to extend the simulation algorithm Sim to obtain an execution of $\mathcal{P}$ in $\texttt{ASYNC}^{5k}$.

Recall that algorithm Sim already simulates any semi-synchronous protocol without light, so we need only to consider the fact that $\mathcal{P}$ uses a light with k colors in its execution. Assume for the moment that we equip each robot r with a secondary light $L[r]$ of k colors, and that this light is clearly identified as such by all robots. We can then use the primary light $Light[r]$ to store the five colors used by Sim, and the secondary light $L[r]$ to record the color of the single light in $\mathcal{P}$. More precisely, the second light would initially be set to the same color as the robots executing $\mathcal{P}$ in $\texttt{SSYNC}^k$. During the *Compute* step of the simulation, whenever a robot computes a new destination, it also computes the new color according to $\mathcal{P}$, and sets the color of its secondary light to that color. All other steps of the algorithm are the same as in Sim.

From the correctness of Sim protocol it follows that the above algorithm correctly simulates any protocol for semi-synchronous robots with k lights. Notice that we can replace the two lights in the above simulation with a single light having $5k$ colors, proving Theorem 3.5.

Theorem 3.6. $\forall R \in \mathcal{R}$, $\mathcal{S}^{O(1)}(R) \equiv \mathcal{A}^{O(1)}(R)$.

Proof. The claim follows from Theorem 3.5 and the obvious relationship $\texttt{SSYNC}^{O(1)} \geq \texttt{ASYNC}^{O(1)}$. □

Thus, we have shown that: $\texttt{ASYNC}^{O(1)} \equiv \texttt{SSYNC}^{O(1)}$. In other words, when enhanced with lights, *the difference between asynchrony and semi-synchrony disappears*. This result must be contrasted with the strict dominance between the models without lights.

4 Luminous Asynchrony versus Full-Synchrony

In this section we investigate the relationship between asynchronous luminous robots and fully synchronous robots without lights.

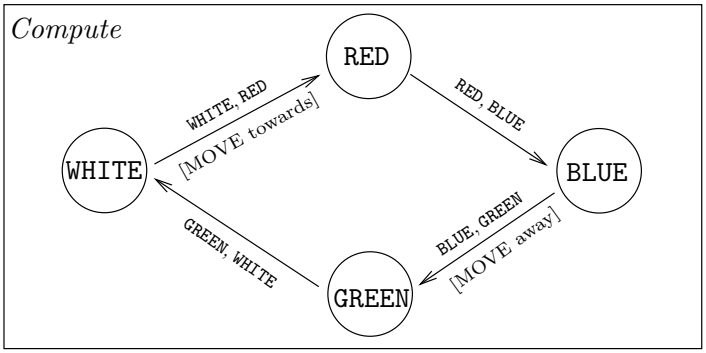

Figure 5: Transition diagram of protocol OSCILLATE. The label of a node denotes the color of the executing robot; the label above an arc denotes the color of the other robot and the label below (if any) denotes the action corresponding to this transition.

4.1 FSYNC is not more powerful than ASYNC$^{O(1)}$

We prove that there are problems that robots cannot solve without lights, even if they are fully-synchronous, but can be solved by asynchronous luminous robots with $O(1)$ colors.

Consider the problem OSCILLATING POINTS (OSP) requiring two robots, x and y, initially in distinct locations, to alternately come closer and move further from each other. More precisely, let $d(t)$ denote the distance of the two robots at time t. The OSP problem requires the two robots, starting from an arbitrary distance $d(0) > 0$ at time t_0, to move so that there exists a monotonically increasing infinite sequence of time instants $t_0, t_1, t_2, t_3, \ldots$ such that:
(1) $d(t_{2i+1}) < d(t_{2i})$, and $\forall h', h'' \in [t_{2i}, t_{2i+1}], h' < h'', d(h'') \leq d(h')$; and
(2) $d(t_{2i}) > d(t_{2i-1})$ and $\forall h', h'' \in [t_{2i-1}, t_{2i}], h' < h'', d(h'') \geq d(h')$.

Theorem 4.1. *$\exists R \in \mathcal{R}_2$, OSP $\notin \mathcal{F}(R)$.*

Proof. Consider a set R composed of two oblivious fully synchronous robots with no additional capabilities, a priori knowledge or agreement (e.g., common unit distance, unique ids, chirality, etc). The impossibility then follows from the fact that, regardless of full synchrony, the robots, upon become active, have no capability that allows them to remember whether in this cycle they must get closer or move further away. □

Theorem 4.2. *$\forall R \in \mathcal{R}_2$, OSP $\in \mathcal{A}^4(R)$.*

The proof of this theorem is constructive. Consider the simple protocol OSCILLATE shown in Figure 6 (refer also to the diagram reported in Figure 5); let x and y be the two robots. The protocol uses four colors: `WHITE`, `RED`, `GREEN`, and `BLUE`; initially both robots are inactive and their lights are set to `WHITE`.

Lemma 4.1. *Let both robots be* `WHITE` *and still at time t'. Then there exists a time $t'' > t'$ such that both robots are* `BLUE` *and still, $0 < d(t'') < d(t')$, and $\forall t, h \in (t', t''), t < h, d(h) \leq d(t)$.*

Proof. Let t_x and t_y denote the first time x and y become active and perform a *Look* operation after time t'; without loss of generality, let $t_x \leq t_y$. Then, according to the algorithm, x will change its color to `RED`, choose as its destination a point closer to y, and move towards that destination. Let $t'_x > t_x$ be the next time x becomes active again; if at that time y is still `WHITE`,

State *Look*

$\texttt{Pos}[\texttt{x}]$:=my current position;
$\texttt{Pos}[\texttt{y}]$:= position of the other robot;
$\texttt{Light}[\texttt{x}]$:= Value of my light;
$\texttt{Light}[\texttt{y}]$:= Value of the light of the other robot.

State *Compute*

$p := \texttt{Pos}[\texttt{x}]$.
Case $\texttt{Light}[\texttt{x}]$:

- WHITE
  If $\texttt{Light}[\texttt{y}] \in \{\texttt{WHITE}, \texttt{RED}\}$ **Then**
  $p :=$ point at distance $< |\texttt{Pos}[\texttt{x}] - \texttt{Pos}[\texttt{y}]|/2$ from me towards the other robot;
  $\texttt{Light}[\texttt{x}] := \texttt{RED}$.
- RED
  If $\texttt{Light}[\texttt{y}] \in \{\texttt{RED}, \texttt{BLUE}\}$ **Then**
  $\texttt{Light}[\texttt{x}] := \texttt{BLUE}$.
- BLUE
  If $\texttt{Light}[\texttt{y}] \in \{\texttt{BLUE}, \texttt{GREEN}\}$ **Then**
  $p :=$ point at distance $< |\texttt{Pos}[\texttt{x}] - \texttt{Pos}[\texttt{y}]|/2$ from me away from the other robot;
  $\texttt{Light}[\texttt{x}] := \texttt{GREEN}$.
- GREEN
  If $\texttt{Light}[\texttt{y}] \in \{\texttt{GREEN}, \texttt{WHITE}\}$ **Then**
  $\texttt{Light}[\texttt{x}] := \texttt{WHITE}$.

State *Move*

$\texttt{Move}(p)$.

Figure 6: Oscillate, a two robots solution for osp in $\texttt{ASYNC}^4$.

x will not perform any action (change of color or move) and wait for the next activation. In other words, x will stay still with color RED until y has changed its color. At time t_y, robot y is still WHITE; regardless of whether x has changed its color to RED or it is still WHITE, y will change its color to RED, choose as its destination a point closer to x, and move towards that destination. Let $t'_y > t_y$ be the next time y becomes active again; if at that time x is still WHITE, y will not perform any action (change of color or move) and wait for the next activation. In other words, y will stay still with color RED until x is no longer WHITE. It follows that, within finite time, both robots are still and have changed their color to RED; let $\hat{t}$ be the time when this happens.

Let $\hat{t}_x \geq \hat{t}$ and $\hat{t}_y \geq \hat{t}$ be the first time x and y become active since (and including) $\hat{t}$; without loss of generality, let $\hat{t}_x \leq \hat{t}_y$. Then, according to the algorithm, x will change its color to BLUE without moving, and it will stay still as long as it sees y RED. At time $\hat{t}_y$, robot y is still RED; regardless of whether x has changed its color to BLUE or it is still RED, y will change its color to BLUE, without moving, and it will stay still as long as it sees y RED. In other words, within finite time both robots become BLUE and are still; let t' be the time when this happens.

Since to become BLUE a robot must have performed the move towards the destination computed when it became RED, and the destination was less than the halfway point towards the other robot, then $0 < d(t'') < d(t')$; furthermore, since from time t' to time t'' the two robots were either still or moving towards each other, we have $\forall t, h \in [t', t''], t < h, d(h) \leq d(t)$. □

With a proof argument similar to that of the above lemma, we have:

Lemma 4.2. *Let both robots be* BLUE *and still at time t'. Then there exists a time $t'' > t'$ such that both robots are* WHITE *and still, $d(t'') > d(t')$, and $\forall t, h \in [t', t''], t < h, d(h) \geq d(t)$.*

Hence, since initially at time t_0 both robots are still and WHITE, by repeated applications of Lemmas 4.1 and 4.2, the claim of Theorem 4.2 follows.

Theorem 4.2 indicates that the availability of lights allows solving asynchronously at least one problem that is unsolvable even by fully-synchronous robots. This would suggest that luminous asynchrony with a constant number of colors is computationally stronger than full-synchrony without lights. However, at this time, there is no proof either supporting or dismissing such a suggestion.

Note that, to dismiss such a suggestion would require to prove the existence of problems that robots can solve without lights when operating in full synchrony, but that the same robots cannot solve asynchronously even when illuminated with a constant number of colors. (As shown by Theorem 3.3, the rendezvous of two robots, which might seem a natural candidate, is not such a problem.) Such a result, in view of Theorem 4.2, would imply that $\texttt{ASYNC}^{O(1)}$ and FSYNC are computationally *orthogonal* (i.e., incomparable).

4.2 Enhancing $\texttt{ASYNC}^{O(1)}$ to be more powerful than FSYNC

The possibility that $\texttt{ASYNC}^{O(1)}$ and FSYNC are computationally incomparable immediately opens the question of which additional property $\mathcal{P}$ would render $\texttt{ASYNC}^{O(1)}$ stronger than FSYNC.

We show that one such a property is the ability to remember a single snapshot, denoted by *Snapshot*$\{1\}$:

Theorem 4.3. $\texttt{ASYNC}^4 + \mathit{Snapshot}\{1\} > \texttt{FSYNC}$

We first prove that, if a set of robots can solve a problem in full-synchrony, then the same set of luminous robots with three colors and capable of storing one snapshot, possibly different at each cycle, can solve the same problem asynchronously. In other words, in $\texttt{ASYNC}^3 + \mathit{Snapshot}\{1\}$ it is possible to solve any problem solvable in FSYNC.

State *Look*

Take the snapshot of the positions of the robots, that returns for all robots $r \in R$:

- $Pos[r]$, the position on the plane of robot r (according to my coordinate system);
- $Light[r]$, the color of the light of robot r.

(**Note**: I am robot x)

State *Compute*

$p :=$ `Pos[x]`.
Case `Light[x]`

- `WHITE`
  If $\forall r \neq x$, `Light[r]` $=$ `WHITE` $\vee$ `Light[r]` $=$ `GREEN`, **Then**
  Store the current snapshot into the *non-volatile* array `Perm[]`.
  `Light[x]` $:=$ `GREEN`.
- `GREEN`
  If $\forall r \neq x$, `Light[r]` $=$ `GREEN` $\vee$ `Light[r]` $=$ `RED` **Then**
  Execute $\mathcal{P}$ using the snapshot in `Perm[]`.
  $p :=$ computed destination.
  `Light[x]` $:=$ `RED`.
- `RED`
  If $\forall r \neq x$, `Light[r]` $=$ `RED` $\vee$ `Light[r]` $=$ `WHITE` **Then**
  `Light[x]` $:=$ `WHITE`.

State *Move*

`Move(`p`)`.

Figure 7: Protocol SyncSim, that simulates `FSYNC` protocols in `ASYNC`^{3} $+$*Snapshot*$\{1\}$.

Also in this case the proof is constructive. Consider Protocol SyncSim, whose rules are shown in Figure 7 (refer also to Figure 8 for its transition diagram), which uses three colors: `WHITE`, `GREEN`, and `RED`; initially, all lights are `WHITE`. Similarly to Protocol Sim, protocol SyncSim enforces a sequence of Mega-Cycles $mc_0, mc_1, \ldots$: the difference here is that *all* robots execute $\mathcal{P}$ in each Mega-Cycle based on the same snapshot (we are simulating `FSYNC`). Each Mega-Cycle mc_i starts with all robots being `WHITE`; within finite time, all `WHITE` robots become `GREEN`; when a robot becomes `GREEN` in a Mega-Cycle, it has stored in a local array `Perm[]` a snapshot of the configuration it just observed: this is necessary to ensure that all robots will compute on the same configuration in this Mega-Cycle. After all robots become `GREEN`, the destination point is computed (using as configuration the one locally stored in `Perm[]`), and the robot starts to perform the *Move* operation, turning its light to `RED`. After a robot has completed the *Move*, it changes its light to `WHITE`. When the lights of all robots are `WHITE`, the current Mega-Cycle ends and the next one begins.

We now show that, for each Mega-Cycle mc_i, all robots execute $\mathcal{P}$ (exactly once) on the basis of the *same* snapshot.

Lemma 4.3. *For each Mega-Cycle, the following holds:*
(i) At the beginning of a Mega-Cycle, all robots are colored `WHITE`*. (ii) During the Mega-Cycle, every robot executes $\mathcal{P}$ exactly once on the basis of the same snapshot. (iii) Each Mega-Cycle*

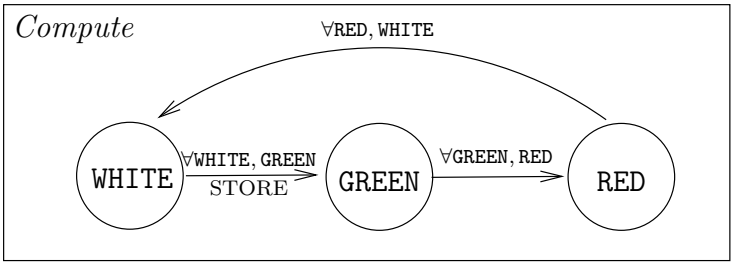

Figure 8: Transition diagram of protocol SyncSim. The label in the nodes represents the color of the light of the executing robot. The label of an edge expresses the condition on the lights of all the other robots that must be satisfied for the transition to occur. The notation "$\forall A, B$" means: "$\forall r \in R$, $\texttt{Light}[\texttt{r}] \in \{A, B\}$".

ends within a finite time. (iv) At the end of the Mega-Cycle all robots are colored WHITE.

Proof. Let all robots start the Mega-Cycle with color WHITE at time t. The first time a robot is activated, it takes a snapshot S, becomes GREEN, and from this moment on it waits until it no longer sees any WHITE robot. Hence, within finite time, at time $t' > t$, every robot becomes GREEN. Observe that, since in the interval of time from t to t' no robot moves, every robot has taken the same snapshot S upon becoming GREEN. Whenever a GREEN robot is activated after time t', according to protocol SyncSim, it will execute protocol $\mathcal{P}$ using snapshot S as input, computing a destination, changing color to RED, and then move towards the destination; after that, it does not move nor change color as long as it sees GREEN robots. Hence, within finite time, at time $t''' > t''$, every robot will have executed protocol $\mathcal{P}$ using snapshot S and become RED. Whenever a RED robot is activated after time t''', it becomes WHITE and it does not move nor change color as long as it sees RED robots. Hence, within finite time all robots are colored WHITE, and the Mega-Cycle ends. □

Since at the end of each Mega-Cycle, all robots are colored WHITE, according to the rules of protocol SyncSim, the next Mega-Cycle begins automatically. Since each Mega-Cycle corresponds to an execution of $\mathcal{P}$ by all robots using the same snapshot, it follows that

Theorem 4.4. *Protocol* SyncSim *is correct, i.e., any execution of protocol* SyncSim *in* $\texttt{ASYNC}^3$ $+Snapshot\{1\}$ *corresponds to a possible execution of* $\mathcal{P}$ *in* FSYNC.

In other words, $\texttt{ASYNC}^3 + Snapshot\{1\} \geq \texttt{FSYNC}$; hence, from Theorems 4.1 and 4.2, the correctness of Theorem 4.3 follows.

As we have seen, the capability of using external lights and remembering a single snapshot allows ASYNC robots to become more powerful than FSYNC robots. This should be contrasted with the fact that, without the use of external lights, SSYNC (and thus ASYNC) robots are incapable to achieve the power of the FSYNC model even if they can remember any number of snapshots [27]. The combination of these results thus yields the following:

$$\texttt{ASYNC} + Snapshot\{\infty\} < \texttt{FSYNC} < \texttt{ASYNC}^{O(1)} + Snapshot\{1\}. \tag{3}$$

5 Concluding Remarks and Open Problems

In this paper we started the examination of externally visible persistent memory in distributed computations by autonomous mobile robots.

We have shown that, using only a few bits of external memory, asynchronous robots can perform tasks which cannot be performed by semi-synchronous robots even with unbounded amount of internal memory. Moreover, a team of asynchronous robots empowered with lights is more powerful than an otherwise similar team of semi-synchronous robots. We have also shown that there are tasks that a team of asynchronous robots having the power of lights can perform while fully-synchronous robots cannot.

As part of the proof, we have provided a mechanism that allows designers of protocols for luminous robots to concentrate only on the semi-synchronous model rather than having to face the much greater difficulties of complete asynchrony. The price to pay is an increase by a constant factor in the number of colors.

Most of our algorithms can be made to work even if the robots start from an arbitrary state where the light on the robot has an arbitrary initial color. Thus, it is possible to have self-stabilizing algorithms for robots with lights. Indeed protocol TwoGatherLight is self-stabilizing. Protocols Sim,Oscillate, and SyncSim, which are not self-stabilizing as described, can be easily modified to have this property. In fact, it is easy to characterize the "illegal" configurations; we can then add to those protocols the rule that, if the result of a robot *Look* is an illegal configuration, then the robot turns its light to a default color, and waits until all lights have that color. From that moment on, the protocol behaves correctly.

The results presented here constitute only the start of the investigation into the newly introduced model of robots with lights. They open many problems and poses many questions.

First and foremost, it is still unknown whether or not there are problems solvable by fully-synchronous robots but not by asynchronous robots with lights; a positive answer implies $\texttt{ASYNC}^{O(1)} \perp \texttt{FSYNC}$ while a negative one implies $\texttt{ASYNC}^{O(1)} > \texttt{FSYNC}$.

We have shown that the availability of a snapshot would render asynchronous luminous robots more powerful than regular fully-synchronous one; should $\texttt{ASYNC}^{O(1)} \perp \texttt{FSYNC}$, an immediate research question is whether a property weaker than $Snapshot\{1\}$ would suffice to achieve the same result.

A large open research direction is the algorithmic re-examination of the classical problems (e.g., pattern formation, gathering, scattering, flocking, etc.) when the robots are luminous. Since the preliminary announcement of the model [9], some investigations in this direction have already started, and protocols for luminous robots have been designed: to solve the problem of forming a sequence of patterns [8] and to reach an obstruction-free configuration, in the setting where collinear robots can obstruct visibility [23].

Our main concern has been computational and our focus on feasibility; we did not attempt to minimize the number of colors used by the robots. Indeed, it is an interesting problem to determine the minimum number of lights sufficient to establish our results. In this direction, some attention has been given to the Gathering$\{2\}$ problem. It has been shown that the gathering of two robots can be solved using three colors but cannot be solved with only two colors [28]. A related result [19] indicated that gathering of two robots can be solved using $O(1)$ colors, even in more restricted models where each robot is allowed to see the lights of only the other robots and not its own light or, where the robot can see only its own light but not that of the others.

Finally, we have assumed a fault-free environment in this paper, i.e. the robots are not subject to failures. An important direction for future research is to consider the effect of failures (e.g. as in [1, 3, 10, 22, 26]), and also taking into account those failures that are peculiar to luminous robots. One type of failures is when the light of some of the robots are not persistent and turn off in the middle of computations. Such faults if they are temporary and non recurrent, may be tolerated using self-stabilizing algorithms as mentioned above. More severe types of

failures include stuck-at faults, where the light of a robot is permanently stuck at a particular color, or when there could be errors in detecting the color of the light of a distant robot. Another possible research direction is to consider the effect of adding lights to robots moving in three-dimensional space as in the recent paper [30].

Acknowledgements
This work has been supported in part by the Natural Sciences and Engineering Research Council of Canada through the Discovery Grant program; by Prof. Flocchini's University Research Chair; by the Scientific Grant in Aid by the Ministry of Education, Sports, Culture and Technology of Japan; by project ARS TechnoMedia (MIUR of Italy); and by project ANR-MACARON (anr-13-js02-0002) funded by ANR, France.

References

[1] N. Agmon and D. Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM Journal on Computing*, 36(1):56–82, 2006.

[2] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. A distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transaction on Robotics and Automation*, 15(5):818–828, 1999.

[3] Z. Bouzid, S. Das, and S. Tixeuil. Gathering of mobile robots tolerating multiple crash faults. In *33rd International Conference on Distributed Computing Systems (ICDCS)*, pages 337–346, 2013.

[4] D. Canepa and M. Gradinariu Potop-Butucaru. Stabilizing flocking via leader election in robot networks. In *9th International Conference on Stabilization, Safety, and Security of Distributed Systems (SSS)*, LNCS 4838, pages 52–66, 2007.

[5] S. Cicerone, G. Di Stefano, and A. Navarra. Minimum-traveled-distance gathering of oblivious robots over given meeting points. In *10th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS)*, pages 57–72, 2014.

[6] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing for mobile robots: Gathering. *SIAM Journal on Computing*, 41(4):829–879, 2012.

[7] R. Cohen and D. Peleg. Convergence properties of the gravitational algorithms in asynchronous robots systems. *SIAM Journal on Computing*, 34:1516–1528, 2005.

[8] S. Das, P. Flocchini, G. Prencipe, and N. Santoro. Synchronized dancing of oblivious chameleons. In *7th International Conference on Fun with Algorithms (FUN)*, pages 113–124, 2014.

[9] S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. The power of lights: Synchronizing asynchronous robots using visible bits. In *32nd International Conference on Distributed Computing Systems (ICDCS)*, pages 506–515, 2012.

[10] X. Défago, M. Gradinariu, S. Messika, and P. Raipin-Parvédy. Fault-tolerant and self-stabilizing mobile robots gathering. In *20th International Symposium on Distributed Computing (DISC)*, LNCS 4167, pages 46–60, September 2006.

[11] X. Défago and S. Souissi. Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity. *Theoretical Computer Science*, 396(1-3):97–112, 2008.

[12] Y. Dieudonné, O. Labbani-Igbida, and F. Petit. Circle formation of weak mobile robots. *ACM Transactions on Autonomous and Adaptive Systems*, 3(4:16), 2008.

[13] Y. Dieudonné, F. Petit, and V. Villain. Leader election problem versus pattern formation problem. In *International Symposium on Distributed Computing (DISC)*, LNCS 6343, pages 267–281, 2010.

[14] A. Efrima and D. Peleg. Distributed models and algorithms for mobile robot systems. In *33rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, LNCS 4362, pages 70–87, 2007.

[15] P. Flocchini, G. Prencipe, and N. Santoro. *Distributed Computing by Oblivious Mobile Robots*. Morgan & Claypool, 2012.

[16] P. Flocchini, G. Prencipe, N. Santoro, and G. Viglietta. Distributed computing by mobile robots: Solving the uniform circle formation problem. In *18th International Conference on Principles of Distributed Systems (OPODIS)*, LNCS 8878, pages 217–232, 2014.

[17] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of robots with limited visibility. *Theoretical Computer Science*, 337(1-3):147–168, 2005.

[18] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Arbitrary pattern formation by asynchronous oblivious robots. *Theoretical Computer Science*, 407:412–447, 2008.

[19] P. Flocchini, N. Santoro, G. Viglietta, and M. Yamashita. Rendezvous of two robots with constant memory. In *20th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 189–200, 2013.

[20] N. Fujinaga, Y. Yamauchi, S. Kijima, and M. Yamashita. Asynchronous pattern formation by anonymous oblivious mobile robots. In *26th International Symposium on Distributed Computing (DISC)*, pages 330–331, 2011.

[21] V. Gervasi and G. Prencipe. Coordination without communication: The case of the flocking problem. *Discrete Applied Mathematics*, 144(3):324–344, 2004.

[22] T. Izumi, S. Souissi, Y. Katayama, N. Inuzuka, X. Defago, K. Wada, and M. Yamashita. The gathering problem for two oblivious robots with unreliable compasses. *SIAM Journal on Computing*, 41(1):26–46, 2012.

[23] G.A. Di Luna, P. Flocchini, S. Gan Chaudhuri, N. Santoro, and G. Viglietta. Robots with lights: Overcoming obstructed visibility without colliding. In *16th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 150–164, 2014.

[24] D. Peleg. Distributed coordination algorithms for mobile robot swarms: New directions and challenges. In *7th International Workshop on Distributed Computing (IWDC)*, LNCS 3741, pages 1–12, 2005.

[25] G. Prencipe. The effect of synchronicity on the behavior of autonomous mobile robots. *Theory of Computing Systems*, 38(5):539–558, 2005.

[26] S. Souissi, X. Défago, and M. Yamashita. Using eventually consistent compasses to gather memory-less mobile robots with limited visibility. *ACM Transactions on Autonomous and Adaptive Systems*, 4(1):1–27, 2009.

[27] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999.

[28] G. Viglietta. Rendezvous of two robots with visible bits. In *Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS)*, pages 291–306, 2013.

[29] M. Yamashita and I. Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theoretical Computer Science*, 411(26-28):2433–2453, 2010.

[30] Y. Yamauchi, T. Uehara, S. Kijima, and M. Yamashita. Plane formation by synchronous mobile robots in the three dimensional euclidean space. In *29th International Symposium on Distributed Computing (DISC)*, 2015. (to appear).

[31] Y. Yamauchi and M. Yamashita. Randomized pattern formation algorithm for asynchronous oblivious mobile robots. In *28th International Symposium on Distributed Computing (DISC)*, pages 137–151, 2014.