Improved Protocols for Luminous Asynchronous Robots[⋆]

Mattia D'Emidio[1], Gabriele Di Stefano[2], Daniele Frigioni[2], Alfredo Navarra[3]

[1] Gran Sasso Science Institute (GSSI)
Viale Francesco Crispi 7, I–67100, L'Aquila, Italy.
`mattia.demidio@gssi.infn.it`

[2] Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica,
Università degli Studi dell'Aquila, Via Vetoio, I–67100 L'Aquila, Italy.
`{gabriele.distefano,daniele.frigioni}@univaq.it`

[3] Dipartimento di Matematica e Informatica, University of Perugia,
Via Vanvitelli 1, I–06123, Perugia, Italy.
`alfredo.navarra@unipg.it`

Abstract. The distributed setting of computational mobile entities, called robots, that have to perform tasks without global coordination has been extensively studied in the literature. A well-known scenario is that in which robots operate in *Look-Compute-Move* (LCM) cycles. LCM cycles might be subject to different temporal constraints dictated by the considered schedule. The classic models for the activation and synchronization of mobile robots are the well-known *fully-synchronous*, *semi-synchronous*, and *asynchronous* models.
In this paper, we concentrate on the weakest asynchronous model, and propose improved and general protocols to solve tasks when the robots are endowed with lights, i.e. they are *luminous.*

1 Introduction

The distributed setting of computational mobile robots that have to perform tasks without global coordination has been extensively studied in the literature. A well-known scenario is that in which robots operate in *Look-Compute-Move* (LCM) cycles (see [1,2,11,12] and references therein). During each cycle, a robot acquires

⋆ The work has been partially supported by the European project "Geospatial based Environment for Optimisation Systems Addressing Fire Emergencies" (GEO-SAFE), contract no. H2020-691161, and by the Italian project "RISE: un nuovo framework distribuito per data collection, monitoraggio e comunicazioni in contesti di emergency response", Fondazione Cassa Risparmio Perugia, code 2016.0104.021.

V. Biló, A. Caruso (Eds.): ICTCS 2016, Proceedings of the 17th Italian Conference on Theoretical Computer Science, 73100 Lecce, Italy, September 7–9 2016, pp. 136–148 published in CEUR Workshop Proceedins Vol-1720 at `http://ceur-ws.org/Vol-1720`

a snapshot of the surrounding environment (*Look* phase), then executes an appropriate algorithm by using the obtained snapshot as input (*Compute* phase), and finally moves toward a desired destination, if any (*Move* phase). Look-Compute-Move cycles might be subject to different temporal constraints dictated by the considered schedule. The classic models for the activation and synchronization of mobile robots are the well-known *fully-synchronous* (FSYNC), *semi-synchronous* (SSYNC), and *asynchronous* (ASYNC) models (see, e.g., [3,5,8]).

- **Fully-synchronous** (FSYNC): The *activation* phase (i.e. the execution of an LCM cycle) of all robots can be logically divided into global rounds. In each round all the robots are activated, obtain the same snapshot of the environment, compute and perform their move. Notice that, this assumption is computationally equivalent to a fully synchronized system in which robots are activated simultaneously and all operations happen instantaneously.
- **Semi-synchronous** (SSYNC): It coincides with the FSYNC model, with the only difference that not all robots are necessarily activated in each round.
- **Asynchronous** (ASYNC): The robots are activated independently, and the duration of each *Compute*, *Move* and inactivity phase is finite but unpredictable. As a result, robots do not have a common notion of time. Moreover, they can be seen while moving, and computations can be made based on obsolete information about positions.

Recently, a new model has been introduced by Das et al. in [4], extending the classic ones. In detail, given a model $\mathcal{M} \in \{\text{FSYNC}, \text{SSYNC}, \text{ASYNC}\}$, the authors define model $\mathcal{M}^c$, where each robot operating in $\mathcal{M}$ is equipped with a *light* that is visible to itself and to the other robots during the *Look* phase. The light associated with a robot can generate c different colors (for some constant integer $c > 0$), and can be updated by a robot during its *Compute* phase. The light is assumed to be *persistent*, i.e. despite robots can be oblivious, their lights are not automatically reset at the end of a LCM-cycle. Light-enhanced robots, introduced for the first time in [9,13], are usually referred as *luminous* robots (see, e.g., [10]). Note that, depending on the considered scenario, a robot might have visibility of the lights of either all other robots or just of a subset of them.

A first comprehensive evaluation of the computational power of robots operating in the LCM model and moving within the *Euclidean plane*, under different levels of synchronization, has been proposed in [4]. In detail, the authors provide a series of results that prove relations between classic models and variations of them, including the possibility that robots are *luminous.*

In [6] a characterization of the computational power of robots moving on *graphs* has been proposed. In particular, the authors first show relations among the three classic activation and synchronization models; second, they compare the models where robots are endowed with lights against the models without lights; third, they highlight the relations among the different models concerning luminous robots; finally, they provide a detailed comparison of the proposed results with the case of robots moving in the Euclidean plane.

1.1 Our Contribution

In this paper, we extend the work done in [6] and [7] as follows. First, we propose a reviewed and improved version of the proof given in [7] to show that FSYNC is not more powerful than $\text{ASYNC}^{O(1)}$. To this aim, we introduce a new algorithm which uses less colors to solve the same problem considered in the proof, thus showing it is solvable in ASYNC^3 rather than in ASYNC^4. Second, we generalize the newly introduced algorithm into a general algorithmic framework that for any $k > 2$ allows (any number of) robots operating in ASYNC^k to address all tasks within class of *basic formation problems* having k states (BFP_k from now on). In such problems, described for the first time in [6], the focus is on allowing robots to achieve specific sequences of k placements regardless of the movements they perform to reach each disposal. Robots might move on graphs or on the Euclidean plane. Finally, we show, by means of a specific distributed task, that there is a non-trivial subset of problems in BFP_k for which using robots in ASYNC^3 instead of ASYNC^k suffice to reach the corresponding goal.

1.2 Structure of the Paper

The paper is organized as follows. In Section 2, we provide the necessary notation for the considered problems. In Section 3, we give the improved version of the algorithm given in [7], while in Section 4 we introduce its generalization. In Section 5, we discuss on how ASYNC^3 robots can be used to solve all tasks within a non-trivial subset of BFP_k. Finally, Section 6 concludes the paper.

2 Preliminaries

We consider a system composed of mobile entities, called *robots*, that operate in LCM-cycles. In particular, each robot is modeled as an independent computational unit, capable of performing local computations. The robots are placed in a spatial environment which is assumed to be either the Euclidean plane or an undirected graph $G = (V, E)$, i.e. robots are placed on the nodes of the graph. Each robot has its own local perception of the surrounding environment, which means it can detect all other robots either as points in the plane with respect to its own coordinate system or perceiving a graph isomorphic to G and understand whether a node is occupied by a robot or not. Each robot is equipped with sensing capabilities that return a *snapshot* of the relative positions of all other robots with respect to its location.

In the remaining of the paper, we assume that robots are *anonymous* and *identical*, i.e. they are indistinguishable by their appearance, and execute the same algorithm. Unless differently specified, robots are assumed to be *oblivious*, i.e. they have no memory. Moreover, we consider robots acting without a central control, i.e. they are assumed to be *autonomous* and not able to directly communicate information (e.g. by a wireless interface) with other robots, i.e. they are *silent*. Each robot is endowed with motor capabilities and can freely move. However, when moving on graphs, the movement along one edge is considered instantaneous, so that each time a robot perceives the snapshot of the current configuration, it sees all other robots always on the nodes of the graph. We will specify different assumptions if required by the context.

At any point in time, a robot is either *active* or *inactive*. All robots are initially inactive, i.e. they are idle. When active, a robot executes an LCM-cycle by performing the following three operations in sequence, each of them associated with a different state:

– **Look:** The robot observes the environment. The result of this phase is a snapshot of the positions of all robots with respect to its own perception.
– **Compute:** The robot executes its own algorithm, using the data sensed in the *Look* phase as input. The result of this phase is a target node among the neighbors of the node in which the robot currently resides (at most one edge per cycle can be traversed by a robot).
– **Move:** The robot moves toward the computed target. If the target is the current position, then the robot stays still, i.e. it performs what is called a *null* movement.

The amount of time to complete a full LCM-cycle is assumed to be finite but unpredictable.

3 Algorithm 3-FORTHBACK

In this section, we propose a reviewed and improved version of the proof given in [7] to show that FSYNC is not more powerful than ASYNC$^{O(1)}$. To this aim, we make use of the following task for the proof.

Forth and Back (FB)	
***Input*:**	Two anonymous robots placed at two distinct internal nodes of a path P at some distance d (in terms of number of edges).
***Solution*:**	A distributed algorithm that ensures the two robots to alternate their distance between d and $d+2$.

In [6] it has been already shown that the FB problem cannot be solved in FSYNC. They also show that FB can be solved in ASYNC$^{O(1)}$, by an algorithm (namely FORTHBACK) which requires each robot to be equipped with a light that can assume *four* colors. Here, we propose an enhanced algorithm, named Algorithm 3-FORTHBACK (see Algorithm 1), which is able to solve the problem even if the robots are endowed with a light that can generate only *three* colors.

The new strategy has its own practical interest since it improves over the algorithm given in [6]. In particular, it reduces the number of colors needed for solving the problem, which can be easily imagined as a proxy for the use of power/communication resources. In addition, the intuition behind its design opens new perspectives for devising general strategies dedicated to robots operating in ASYNC^{k}. We will discuss in the next sections on such aspect. In details,

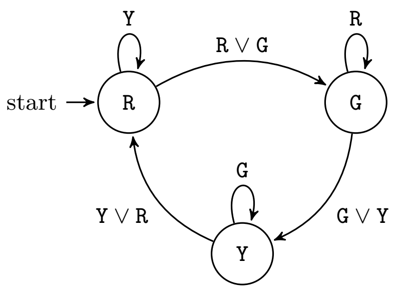

Fig. 1. Finite state machine associated to algorithm 3-ForthBack.

we will show that the technique underlying Algorithm 3-ForthBack can be generalized into an algorithmic framework that allows async^k robots to address a class of problems of significant interest.

We now proceed with the description of the new algorithm. The three colors used by Algorithm 3-ForthBack are Red (R), Green (G), and Yellow (Y), with the following meanings:

- Red indicates that the robot is ready to move for the next step and the previous distance must be increased;
- Green indicates that the robot is ready to move for the next step and the previous distance must be decreased;
- Yellow indicates that the robot has moved to decrease the previous distance and it is ready for the next step.

In what follows, we denote by $L[r]$ the light associated with a given robot r. At the beginning, both robots start with the lights set to Red. As we will see, Algorithm 3-ForthBack ensures that, whenever one of the robots has the light set to Red (to Green, respectively), the other robot will eventually turn its light to Red (to Green, respectively), i.e. they are always able to be synchronized at some point. In other words, Algorithm 3-ForthBack solves FB in async^3 by exploiting the parity encoding of the current step by means of the light. We describe the algorithm as it is executed by a generic given robot r. For the sake of clarity, we also summarize the behavior of each robot via a finite state machine (see Figure 1), where the label within a node represents the color of the light $L[r]$ of the executing robot, while the label above an edge represents the condition on the light $L[r']$ of the other robot r' that triggers the transition to occur.

Theorem 1. *Algorithm* 3-ForthBack *correctly solves the* FB *problem in* async^3 .

Proof. First of all, notice that, if d is the initial distance between the two robots (when both lights are Red), then $d+2$ defines the final placement of the two robots after the first step. Therefore, in order to reach the requested configuration,

Algorithm 1: Algorithm 3-ForthBack performed by a generic robot r to solve FB in ASYNC^3.

```
1  Let $r'$ be the other robot;
2  Let $\delta$ be my distance from $r'$;
3  if $L[r] = $ Red then
4      if $L[r'] = $ Red $\vee L[r'] = $ Green then
5          $L[r] :=$ Green;
6          Let $v$ be the neighbor at distance $\delta + 1$ from $r'$;
7          The new position is $v$;
8          Exit;
9  if $L[r] = $ Green then
10     if $L[r'] = $ Green $\vee L[r'] = $ Yellow then
11         $L[r] :=$ Yellow;
12         Let $v$ be the neighbor at distance $\delta - 1$ from $r'$;
13         The new position is $v$;
14         Exit;
15 if $L[r] = $ Yellow then
16     if $L[r'] = $ Yellow $\vee L[r'] = $ Red then
17         $L[r] :=$ Red;
18         Exit;
```

when a robot has to move to increase its distance (i.e. its light is Red), if the current distance is d', then the target distance has to be set to $d' + 1$ (see Line 7), since both robots contribute of one edge.

Similarly, if $d + 2$ is the distance between the two robots after an increasing step, then d defines the placement of the two robots after the current step. Hence, in order to reach the requested configuration, when a robot has to move to decrease its distance (i.e. its light is Green), if the current distance is d', then the target distance has to be set to $d' - 1$ (see Line 13), since both robots contribute of one edge.

Now, if $L[r]$ is either Red or Green, then r is ready to accomplish a movement, which must either increase the distance of the previous placement ($L[r] = $ Red case) or decrease it ($L[r] = $ Green case). The robot r can decide which is the case by looking at the light of the other robot.

On the one hand, if $L[r] = $ Red and $L[r']$ is either Red or Green (see Line 4), then r must move away from r' of one edge, as robot r' is either ready to move to increase the distance ($L[r'] = $ Red) or is ready to move to decrease the distance($L[r'] = $ Green). On the other hand, if $L[r] = $ Green and $L[r']$ is either Green or Yellow (see Line 10), then r must move closer to r' of one edge since robot r' is either ready to move to decrease the distance ($L[r'] = $ Green) or has already accomplished a movement that has decreased it ($L[r'] = $ Yellow).

If $L[r]$ is Yellow and $L[r']$ is either Red or Yellow, then r can conclude that r' is either ready to move to increase the distance ($L[r'] = $ Red) or has already terminated a movement whose purpose was that of decreasing the distance

($L[r'] =$ YELLOW). Robot r, in this case, just switch its light to RED (see Line 17). If r' has performed the above step before r, i.e. $L[r] =$ YELLOW and $L[r']$ is GREEN, then, r simply keeps $L[r]$ to YELLOW. □

4 Generalizing Algorithm 3-ForthBack

In this section, we propose a general algorithmic paradigm that allows robots operating in ASYNC^k, $k > 2$ to address a whole class of problems, namely the basic formation problems BFP$_k$.

BFP$_k$ problems can be informally defined as the class of problems where a set of k static configurations have to be (possibly cyclically) reached, in order to achieve the goal, regardless of the movements they perform to reach each disposal (see [6] for a more thorough discussion). We remind that the class of BFP$_k$ problems can be defined for robots moving on both graphs and the Euclidean plane.

More formally, problems are in BFP$_k$ if and only if their dynamics can be completely described by a finite state machine (o by any subset of it) with the following characteristics:

- there are k distinct states;
- there exists a total (strict) ordering among the k states, i.e. for every i–th state there exists a transition that brings the system from state i to a state $(i + 1) \mod k$.

In other words, the problem asks the robots to change from a state to another one in a sequential manner, according to some criteria. An example of finite state machine of the above kind is reported in Figure 2, where label x_i within a node represents the i–the state in the ordering. Clearly the above class of problems satisfies BFP$_1 \subseteq$ BFP$_2 \subseteq \cdots \subseteq$ BFP$_k$ for any k.

Trivially, the FB problem, discussed in Section 3, belongs to BFP$_2$, since the robots have to cyclically oscillate between $k = 2$ distinct configurations, i.e. those in which the robots are at distance $d' + 1$ and $d' - 1$. However, to solve FB we required 3 colors in order to synchronize the robots. Whereas, when $k > 2$ there is no need for an extra color.

Note that, problems in BFP$_k$ might be solvable in FSYNC or not. This possibility depends on the capability of the robots of distinguishing, by only observing the surrounding environment, two states that are adjacent in the sequence (and therefore are connected by a transition in the finite state machine). If robots are capable of doing so, it is easy to see that (any number of) FSYNC robots can solve problems of BFP$_k$ in $\Theta(k)$ time steps by the following very simple strategy. Starting from an initial state x_0, at each synchronous time step, all robots wake up and perceive the very same snapshot of the surrounding environment. Then, given the state they perceived, say x_i, they all check the criterion associated to the arcs outgoing x_i in the finite state machine (i.e. they perform the compute phase) and, according to the result they perform the move phase that either brings them into $x_{(i+1) \mod k}$ or keeps them into x_i. An example

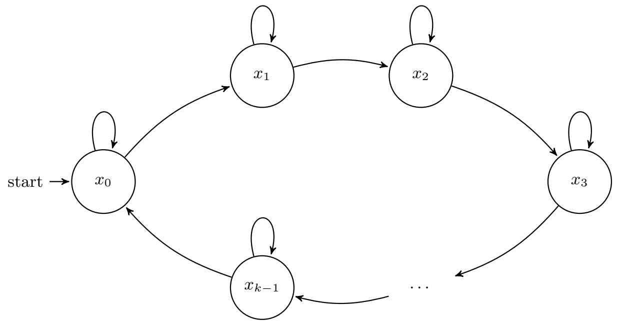

Fig. 2. Finite state machine associated to problems in BFP_k.

of problem in BFP_k that is not solvable in FSYNC is indeed problem FB, since the knowledge obtained by the acquired snapshot is not enough to determine whether the target distance has to be set to $d'+1$ or to $d'-1$ and hence to decide the next state. On the contrary, an example of problem in BFP_k that is solvable in FSYNC, by the above strategy, is the so-called *Pattern Series Chasing* (PSC) problem whose first description can be found in [7], and is reported below.

Pattern Series Chasing (PSC)	
Input:	An undirected and complete graph G with nodes labeled from 1 to n. An array A of c patterns, for some integer constant $c > 0$, each involving $k < n$ nodes of G, such that $A[i] \neq A[j]$, for every $0 \leq i \neq j < c$. A set of k robots forming $A[0]$ in G.
Solution:	A distributed algorithm that ensures robots to form pattern $A[(i+1) \mod c]$ after $A[i \mod c]$, $i \in \mathbb{N}$.

Regarding FB, in the previous section we have shown that it can be solved by two ASYNC^3 robots by Algorithm 1. By following the intuition underlying such algorithm, we now give a generalization of it that allows (any number of) ASYNC^k robots to solve problems in BFP_k for $k > 2$. The main idea behind the algorithmic paradigm we are proposing is based on the fact that, when considering BFP_k problems, an implicit ordering can be provided by the lights of ASYNC^k robots.

The strategy for solving any BFP_k by ASYNC^k robots can be summarized as follows (see Algorithm 2 for more details). Suppose we are given a generic problem in BFP_k and a set $\mathcal{R} = r_1, r_2, \ldots, r_n$ of n robots. We denote by $L[r_x]$ the color assumed by the light of robot r_x. We equip the robots with lights assuming k colors with the following meaning: color COL_i indicates that the robot is in state x_i and it is ready to move to state $x_{(i+1) \mod k}$, and to accomplish the associated (possibly null) move phase. Moreover, we assume that colors exhibit a total (strict) ordering, i.e. for each color x_i there exist two colors $x_{(i\pm1) \mod k}$ such that $x_{(i-1) \mod k} < x_i < x_{(i+1) \mod k}$. To solve the problem is then enough

Algorithm 2: Algorithmic paradigm that allows (any number of) ASYNC^k robots to solve problems in BFP$_k$, $k > 2$.

```
1 Let $r_i$ be the robot executing the algorithm;
2 if $L[r_i] = \text{COL}_{x_i}$ then
3     foreach $r_j \in \mathcal{R} \; : \; i \neq j$ do
4         if $L[r_j] < L[r_i]$ then
5             Exit;
6     $L[r_i] := \text{COL}_{x_{(i+1) \bmod k}}$;
7     Perform move phase of state $x_{(i+1) \bmod k}$;
8     Exit;
```

to exploit the ordering of the colors to take coherent decisions about the state to be reached. In particular, each robot r_i performs its (possibly null) move phase if and only if its current color $L[r_i] = \text{COL}_{x_i}$ is less or equal than the colors of the lights of all other robots. If so, it also switches its light to $\text{COL}_{x_{(i+1) \bmod k}}$. By the above discussion, the following result can be stated.

Theorem 2. ASYNC^k *robots can solve any problem in* BFP$_k$*,* $k > 2$*.*

It is worth noticing that Algorithm 3-FORTHBACK, given in the Section 3, can be easily derived by Algorithm 2 by considering the different move phases associated to each of the two states of problem FB.

As a final remark, note that an algorithm for solving problem PSC by ASYNC^c robots, where c is the cardinality of the array A, can be derived as well by Algorithm 2 by customizing each move phase according to the pattern to be reached.

5 Further Applications of Algorithm 3-FORTHBACK

In this section, we show that there is a non-trivial category of problems in BFP$_k$ for which something better than using ASYNC^k robots can be done. To this aim, we first define the main characteristics of such category and show they can be solved by weaker ASYNC^3 robots. Consider again Algorithm 3-FORTHBACK defined in Section 3, the three colors have been though to encode the static configurations, but they can be used to simply encode an *advancing* in the current computational process. This is not the case for the PSC problem as shown in [7]. In order to better understand the intuition, we now consider a variant of the classical *patrolling problem*, where three robots must infinitely traverse a circle but ensuring some specific configurations where they are all idle. It means that during the patrolling, robots must ensure some predefined *static* configurations.

Given a circle C, let $arc(x, y)$ be the smallest arc of C from x to y.

Patrolling With Stops (PWS)

***Input*:** Three anonymous robots r_1, r_2 and r_3 placed on a circle C with center c, such that $arc(r_1, r_2) = d$, $arc(r_2, r_3) = 2d$, $d < \frac{\pi}{6}$.

***Solution*:** A distributed algorithm that ensures the three robots to patrol C by forming a static configuration similar to C each time the angle α in c defined by the initial position of r_1, c and the current position of r_1 is $p \cdot \frac{\pi}{3}$, for any integer p.

In PWS, we have considered six static configurations that must be reached cyclically, by setting the angle α as a multiple of $\frac{\pi}{3}$. Clearly an arbitrary number of configurations can be considered by reducing α. According to Theorem 2, we know how to solve PWS in ASYNC^6. We now show that an algorithm similar to Algorithm 3-FORTHBACK can be defined to solve the PWS problem in ASYNC^3.

Theorem 3. *There exists an algorithm in* ASYNC^3 *that solves the* PWS *problem.*

Proof. The proof proceeds by providing three subroutines of the same algorithm, each one executed by a different robot, according to its role among r_1, r_2 and r_3. Although the three robots are anonymous, by looking to their relative positioning and lights, they can always deduce who they are. As we are going to show, our algorithm never changes the role of a robot. The meaning of the lights is the same for all robots, that is:

– RED: ready to reach the new static configuration
– GREEN: moving to the computed target
– YELLOW: target reached

The minimum arc of C containing all three robots is always divided into two sub-arcs. The middle robot is always r_2. Initially (when all robots assume light RED), the closest robot to r_2 is r_1, while the other is r_3. The proposed algorithm makes r_3 move always as first, increasing its distance from r_2. This is done by switching $L[r_3]$ to GREEN and evaluating the next position where a static configuration must be guaranteed. Movements are always performed along the circumference of C as the patrolling requires.

While $L[r_3]$ = GREEN, r_1 and r_2 do not move. Once $L[r_3]$ = YELLOW, r_2 can start its movement toward the next position. Due to the adversary, each time a robot moves toward a target position, it can reach it or it can be stopped before. Nevertheless, our algorithm is designed so as the same robot will be the unique one allowed to move until it reaches the desired destination, eventually.

Once both r_2 and r_3 have reached their current destinations and $L[r_2]$ = $L[r_3]$ = YELLOW, the last robot that has to move deduces it is r_1.

Once all robots have reached their destinations, $L[r_1]$ = $L[r_2]$ = $L[r_3]$ = YELLOW and the configuration is static, that is a stop has been performed and the next positioning can start. This is done first by switching all the lights to RED, but in a sequential order, starting from r_3, then r_2 and finally r_1. Then, the whole process is repeated. □

Algorithm 3: Algorithm PWS $\{r_1\}$ performed by r_1

```
1  Let $x$ be the point on $C$ between $r_1$ and $r_2$ such that $|arc(x, r_3)| = \frac{3}{2}|arc(r_2, r_3)|$;
2  if $(L[r] = \text{Red} \wedge L[r_2] = L[r_3] = \text{Yellow}) \vee (L[r] = \text{Green} \wedge |arc(r, r_3)| > x)$
   then
3      $L[r] := \text{Green}$;
4      The new position is $x$;
5      Exit;
6  if $L[r] = \text{Green}$ then
7      $L[r] := \text{Yellow}$;
8      Exit;
9  if $L[r] = \text{Yellow} \wedge L[r_2] = L[r_3] = \text{Red}$ then
10     $L[r] := \text{Red}$;
11     Exit;
```

Algorithm 4: Algorithm PWS $\{r_2\}$ performed by r_3

```
1  Let $x$ be the point on $C$ between $r_2$ and $r_3$ such that
   $|arc(x, r_3)| = \frac{2}{3}\left(|arc(r_1, r_3)| - \arcsin\frac{\pi}{3}\right)$;
2  if $(L[r] = \text{Red} \wedge L[r_3] = \text{Yellow}) \vee (L[r] = \text{Green} \wedge |arc(r, r_3)| > x)$ then
3      $L[r] := \text{Green}$;
4      The new position is $x$;
5      Exit;
6  if $L[r] = \text{Green}$ then
7      $L[r] := \text{Yellow}$;
8      Exit;
9  if $L[r] = L[r_1] = \text{Yellow} \wedge = L[r_3] = \text{Red}$ then
10     $L[r] := \text{Red}$;
11     Exit;
```

6 Conclusion

In this paper, we have considered the problem of devising protocols for luminous asynchronous robots. In details, we have extended the work done in [6] and [7] in three directions. We first have proposed a reviewed, improved version of the proof given in [7] to show that FSYNC is not more powerful than $\text{ASYNC}^{O(1)}$. To do so, we have introduced a new more efficient algorithm for solving the FB problem that requires less colors to work with respect to to its previous counterpart of [7]. Apart from the desirable property of being optimized in terms of colors, the new algorithm has driven us also to the design of its generalization that allows (any number of) ASYNC^k robots to address any problem in BFP_k, for any $k > 2$. As a final contribution, we have shown that there exists a non-trivial subset of problems in BFP_k for which weaker ASYNC^3 robots are powerful enough to achieve the considered goal.

Algorithm 5: Algorithm PWS $\{r_3\}$ performed by r_3

```
1  Let $x$ be the point on $C$ such that $|arc(r_1, x)| = 3|arc(r_1, r_2)| + \arcsin \frac{\pi}{3}$;
2  if $(L[r] = \textsc{Green} \wedge |arc(r_1, r)| < |arc(r_1, x)|) \vee L[r] = L[r_1] = L[r_2] = \textsc{Red}$
     then
3      $L[r] := \textsc{Green}$;
4      The new position is $x$;
5      Exit;
6  if $L[r] = \textsc{Green}$ then
7      $L[r] := \textsc{Yellow}$;
8      Exit;
9  if $L[r] = L[r_1] = L[r_2] = \textsc{Yellow}$ then
10     $L[r] := \textsc{Red}$;
11     Exit;
```

References

1. A. Chatterjee, S. G. Chaudhuri, and K. Mukhopadhyaya. Gathering asynchronous swarm robots under nonuniform limited visibility. In *11th International Conference on Distributed Computing and Internet Technology (ICDCIT)*, volume 8956 of *Lecture Notes in Computer Science*, pages 174–180. Springer, 2015.
2. S. Cicerone, G. Di Stefano, and A. Navarra. Minmax-distance gathering on given meeting points. In *9th Int. Conference on Algorithms and Complexity (CIAC)*, volume 9079 of *Lecture Notes in Computer Science*, pages 127–139. Springer, 2015.
3. G. D'Angelo, G. Di Stefano, and A. Navarra. Gathering on rings under the look-compute-move model. *Distributed Computing*, 27(4):255–285, 2014.
4. S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. Autonomous mobile robots with lights. *Theoretical Computer Science*, 609:171–184, 2016.
5. B. Degener, B. Kempkes, T. Langner, F. Meyer auf der Heide, P. Pietrzyk, and R. Wattenhofer. A tight runtime bound for synchronous gathering of autonomous robots with limited visibility. In *23rd ACM Symp. on Parallelism in algorithms and architectures (SPAA)*, pages 139–148. ACM, 2011.
6. M. D'Emidio, D. Frigioni, and A. Navarra. Characterizing the computational power of anonymous mobile robots. In *36th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 293–302, 2016.
7. M. D'Emidio, D. Frigioni, and A. Navarra. Synchronous robots vs asynchronous lights-enhanced robots on graphs. In *16th Italian Conference on Theoretical Computer Science (ICTCS)*, volume 322 of *Electronic Notes in Theoretical Computer Science*, pages 169–180. Elsevier, 2016. doi:10.1016/j.entcs.2016.03.012.
8. S. Devismes, F. Petit, and S. Tixeuil. Optimal probabilistic ring exploration by semi-synchronous oblivious robots. In *16th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 5869 of *Lecture Notes in Computer Science*, pages 195–208, 2009.
9. A. Efrima and D. Peleg. Distributed models and algorithms for mobile robot systems. In *33rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, volume 4362 of *Lecture Notes in Computer Science*, pages 70–87. Springer, 2007.

10. P. Flocchini. Computations by luminous robots. In *14th International Conference on Ad-hoc, Mobile, and Wireless Networks (ADHOC-NOW)*, volume 9143 of *Lecture Notes in Computer Science*, pages 238–252. Springer, 2015.
11. P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing by oblivious mobile robots. *Synthesis Lectures on Distributed Computing Theory*, 3(2):1–185, 2012.
12. S. Kamei, A. Lamani, F. Ooshita, and S. Tixeuil. Gathering an even number of robots in an odd ring without global multiplicity detection. In *Mathematical Foundations of Computer Science (MFCS)*, pages 542–553. Springer, 2012.
13. D. Peleg. Distributed coordination algorithms for mobile robot swarms: New directions and challenges. In *7th International Workshop on Distributed Computing (IWDC)*, volume 3741 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005.