Synchronizing asynchronous mobile robots with limited visibility using lights

D Das¹, S Mukhopadhyaya² and Mrityunjay Ghosh³

¹Assistant Professor, National Institute of Technology, Durgapur, Mahatma Gandhi Avenue, Durgapur, West Bengal, India
²Associate Professor, Birla Institute of Technology, Mesra, Lalpur Extension Center, Ranchi, India
³QRD LAB, Calcutta University, Kolkata, India.

E-mail: deepanwita@cse.nitdgp.ac.in

Abstract. There are three models of automated robots based on their activation schedules and time notions. They are FSYNC, SSYNC and ASYNC Model. The FSYNC-fully synchronous robots have the common notion of time and follow a global round of time. In every turn, all the robots are active and perform their job in synchronization. All the robots have the same notion of time in the SSYNC-semi-synchronous model, whereas in this model, all robots may not be active in each round. The activation time of ASYNC-asynchronous is independent and unpredictable. Research is ongoing to vanish the difference between asynchrony, semi-synchrony and synchrony by equipping the robots with various combination of lights and snapshots. Most of the research has been done on robots with unlimited visibility capabilities. In these works, a robot can view the colors of all other robots and based on that the decision is taken to change its own color thus achieves synchronization. Decision taking becomes challenging if the robots have limited view of the other robots. This paper proposed a technique of synchronizing the robots which are asynchronous by nature, using observable colored lights, where the robots have limited range of visibility.

Keywords: Robot Swarm, Asynchronous Robots, Limited visibility, Colored Lights

1. Introduction

Nature inspires the collective effort of weak entities through collaboration and communication that exceeds the capabilities of one individual. The entities have only some basic capabilities and local information, based on which they outperform intelligent singles. For example, ants can be seen to carry a large dead insect in comparison to their own size, towards their nest through collaboration. No single ant could do the job. Inspired by such natural examples, distributed systems consisting of small weak robots, capable of doing big tasks through collaboration like moving heavy goods, environment monitoring or battlefield surveillance [1], searching or cleaning terrains, identify the fire sources [2], demining of landmines [3], area exploration, surveillance etc., are becoming the center of interest in the field of swarm related research. The robots are simple with very limited capacity in terms of memory, communication and viewing power, and they operate in CORDA [4] computational model. In this model, the robots execute a series of computational rounds. There are three phases, called look(L), compute(C) and move(M) or LCM in each round.

Achieving coordination is a big challenge for such simple robots with limited capabilities. The study is going on to explore the computation capabilities of such weak robots based on various activation schedules and timing notions. The models based on the synchrony are named as full, semi and
asynchronous that is FSYNC, SSYNC and ASYNC model. The robots are said to be fully synchronous (FSYNC) when they have the common notion of time and follow a global round of time. All the robots perform their LCM cycle in a synchronous manner. In the SSYNC model also, all the robots have the same notion of time. But in this model, all robots may not be active in each round. However, when they are active, they are all synchronous. The activation schedule of ASYNC robots is independent, different from each other and unpredictable. To complete the LCM cycle or remain inactive, the ASYNC robots take a finite amount of time.

Let us consider $R(M, R)$ is the set of all problems, solvable via a set of robots $R$ in the model $M$. Now if $R(M, R) \subseteq R(N, R) \forall R \in \mathbb{R}$, then we can say, Model $N$ is not less powerful than $M$. Trivially, $FSYNC \geq SSYNC$, and $SSYNC \geq ASYNC$. Therefore, combining these two, we get, $FSYNC \geq SSYNC \geq ASYNC$. But, in the context of designing a robot, asynchronous robots are more practical than the other two.

2. Literature Review

The objective of this research is to vanish the difference between asynchrony, semi-synchrony and synchrony by equipping the robots with various combination of lights and snapshots. A number of research results were reported in this context. The use of external signals or lights for capability enhancement of the robots was first introduced by Peleg [5]. In another work, Peleg [6] has proposed the use of visible lights during partitioning a robot swarm. In this domain of work, Das et al. [7], [8], [9] have contributed several results. They have considered the robots with observable lights, named as luminous robots. The lights can take a different number of colors that represents various states, and they do not reset automatically at the end of a cycle. They have also assumed that the robots can store a certain number of snapshots which enables the robots to produce better result considering various timing models. They have used a notation to signify the robot is in which form of synchrony and embedded with visible lights or able to store snapshots or both. The notation is $\text{TimingModel}_{a, b}$ where $\text{TimingModel}$ represents the model of synchrony under consideration, $a$ and $b$ signifies the number of color lights and the number of snapshots to be stored by the robots, respectively. The robots considered are identical, autonomous, with no communication, with or without persistent memory and follows CORDA model.

In [7], they have investigated the use of visible lights. In [8], [9] Das et al. have some contributions which are categorized into two types: (I) Luminous asynchrony versus semi-synchrony and (II) Luminous asynchrony versus fully-synchrony. They have tried to establish the facts that (I) An asynchronous robot embedded with numbers of persistent colored lights is as powerful as a semi-synchronous robot. (II) An asynchronous robot embedded with numbers of persistent colored lights along with some stored snapshots is as powerful as a fully-synchronous robot. However, even-if equipped with only snapshots it won't be as powerful as semi-synchronous if colored lights are not used.

To investigate the first type of contribution, they have considered two algorithms $SIM$ and $TOGETHERLIGHT$. In algorithm $SIM$ they have proved that, illuminated with fixed numbers of colors an $ASYNC$ robot produced semi-synchronous execution of any protocol. They have considered 6 colors for this to prove that $ASYNC$ is at least as powerful as: $ASYNC^{O(1)} \equiv SSYNC^{O(1)}$ or $ASYNC^{O} \equiv SSYNC$ [8]. The algorithm $SIM$ which uses 6 colors that signifies 6 phases of a megacycle. The phases are: W-waiting, N-next, T-trying, S-stopped, F-finished, M-moving. Transition diagram for algorithm $SIM$ in [8] is shown in Figure 1. Each megacycle starts with color T and ends with color F. Initially a robot is having light T. If a robot observes all other robot is having T or S, they start executing the protocol and turns their light M and eventually turns light S. If a robot with T light observes some of the robots with M light (not all having T or S), then they are not allowed to execute the protocol. They need to wait by turning W light until all the robots turn their S light. When all turn their S light on the waiting robots turn their N light and then T light. Thus, they become eligible for executing the protocol in the next phase. This scenario is similar to semi-synchronous execution of the protocol. Here not all the robots are active and executing the protocol at a time, but when they are eligible, they are executing the protocol at the same time. However, in [9], they have slightly modified the algorithm by using 5 colors...
instead of 6, which is presented in Figure 2. Algorithm TOGETHERLIGHT gathers two robots in ASYNC⁴. This algorithm proves that enabled with O(1) visible lights, two asynchronous and oblivious robots can gather. They have considered robots with 4 colors, namely OFF, RED, GREEN and BLUE. Two robots are considered as x and y.

![Figure 1. The transition diagram for algorithm SIM.](image1)

![Figure 2. The transition diagram of algorithm SIM Modified in [6].](image2)

Initially, both are OFF. After activation, they reach the midway between them by following the algorithm. If one robot x begins earlier than y (OFF) then it turns the light to RED and moves to the halfway between x and y and turns to BLUE while reached. If y begins earlier and x finds y as RED it understands that y is still moving. x waits until y becomes BLUE. If y is BLUE, x reaches up to y and eventually BLUE. A robot with BLUE light waits for another to become BLUE and then turn GREEN to signify the end. The author proved that in this situation, ASYNC is more powerful than SSYNC while embedded with colored lights. Figure 3 represents the transition diagram for algorithm TWOGATHERLIGHT. Combining both the results of SIM and TWOGATHERLIGHT, they have claimed that ASYNCO(1) is as powerful as SSYNCO(1).
Figure 3. The transition diagram of algorithm TWOGATHERLIGHT.

In the second part of their work Das et al. discussed about the connection between asynchronous robots with light and fully synchronous robots without light. They have proved that if any ASYNC robot having 3 different colors and capability of memorizing one snapshot, can resolve any problem that is solvable in FSYNC. That is $ASYNC_3 \equiv FSYNC$. Additionally, they have proved that some problems cannot be solved by FSYNC robots without lights but can be solved with ASYNC robot with $O(1)$ colors. They have proposed an algorithm called SYNCSIM which signifies $ASYNC_3 \equiv FSYNC$.

The robots are equipped with lights with three colors, namely OFF, RED and GREEN. Initially, all the robots are having their color as OFF. After some time, they turn on their GREEN lights. While turning into GREEN, the robots immediately store a snapshot in their memory. Until all the robots turn GREEN, they are not allowed to execute the protocol. Thus, it ensures that every robot will execute the protocol based on the same snapshot. This signifies the synchronous execution of the protocol. Finally, at the termination of the protocol, the robot turns their light into RED and moves to the destination. Upon reaching it turns into OFF again. Figure 4 represents the transition diagram for algorithm SYNCSIM which claims that while embedded with lights and stored snapshot, an ASYNC robot is at least as powerful as FSYNC.

Figure 4. The transition diagram of algorithm SYNCSIM

Two algorithms BLINKING and OSCILLATE has been proposed to claim that an ASYNC robot is more powerful than FSYNC. In BLINKING algorithm, the robots need to perform two sub-tasks alternatively. In the first sub-task, they need to form a circle, in the next sub-task they need to gather at a point and again they have to form the circle by going back to the same location in the next alternative batch of sub-tasks. They have proved that once gathered the robots cannot go back to the previous circular formation in FSYNC as no snapshots are stored, while with a stored snapshot, this can be achieved with ASYNC robots. In algorithm OSCILLATE two robots need to alternatively come closure or go further from each other. This can be achieved with ASYNC with 4 colored lights. But in the FSYNC system the robots have no capability to remember whether they have to come closer or move far away in a specific cycle. OSCILLATE claims that $ASYNC_4 \equiv FSYNC$.

In all the above contributions, it can be said that while embedded with colored lights and capacity to remember snapshots asynchronous robots can be as powerful as semi-synchronous and fully synchronous. In all the above contribution visibility of the robots are considered to be unlimited. No such work is found where the effort has been given to eliminate the synchrony problem among the robots that have limited range of perceptibility; however, robots with limited range of visibility is an assumption which is closer to reality. In literature, some application of colored lights is found in [10], [11] etc. Ooshita et al. [10] and Nagahama et al. [11] have investigated ring exploration via independent
robots with visibility up to a fixed distance and having lights that emit fixed number of colors. Result of [10] shows that in the number of robots required for perpetual exploration is 2 (for FSYNC) and 3 (for SSYNC and ASYNC), whereas for terminating exploration, 3 (for FSYNC) and 4 (for SSYNC and ASYNC). They have investigated the importance of colored lights as, without lights terminating exploration is not possible to achieve in ASYNC and SSYNC systems, whereas 5 robots are required in FSYNC. In [11], the authors show that the number of lights can be reduced by one than the previous work [10].

In another work by Bramas et al. [12] have considered the problem of infinite grid exploration (IGE) using some synchronous opaque robots which only agree on a common chirality. The result of their research has shown that using five colored lights and a visibility range one; five robots are required to solve the (exclusive) IGE problem. They have also demonstrated that the non-exclusive IGE problem can be solved using six robots with visibility one and with three colors and the exclusive IGE problem using seven robots with visibility two.

However, all the results of [10], [11] and [12] show that there are restrictions on the number of robots to achieve the solution in various timing models. However, in all these applications, no effort is made to generalize the solutions that have no restrictions on the timing models, configuration as well as the numbers of robots. We aim to eliminate the notion of synchronicity so that a single solution can work for various environments in which the solutions differ due to the timing models. In this proposed work, we have designed an algorithm for autonomous, asynchronous mobile robots with limited view so that they can perform their protocol in a synchronous manner.

We increase the capabilities of the asynchronous robots by providing two additional features. A light bulb is embedded on the robots, which emits 4 different colors. Compared to the literature [algorithm SYNCSIM which claims ASYNC_{1}^{4} ≡ FSYNC with unlimited visibility], increasing only one bit of light can solve a problem of asynchrony while considering limited visibility to the robots. The second capability is that each robot is able to retain a single snapshot from the previous cycle. The color of bulb light is updated by the robot during computation. As a result, the colors of all the visible robots those falls within the visibility circle of a particular robot can be detected by the during the Look operation. The lights are assumed to be persistent; they do not switch off automatically when the cycle comes to an end. Thus, it acts like a form of persistent and external memory of the robots. In our proposed algorithm, we consider that the lights the robots have can generate YELLOW, GREEN, RED and WHITE colors and the algorithms claims ASYNC_{1}^{4} ≡ FSYNC under limited visibility.

3. Proposed Model and Assumptions
In this work the system is generated by a distributed group of independent robots. The robots are placed in a 2D plane. The position of a robot is determined by a point on that plane. Every robot is equipped with a sensor. A robot can detect the other robots on the plane within a certain range, and the sensor results a view generated based on those robots’ locations with reference to the local coordinate system. The sensing range of the robot is called the visibility circle. Therefore, the visibility of the robot is limited. The robots are enabled to compute in its local memory with the help of their local coordinate system. They follow a full compass model, i.e. they all fully agree on the direction as well as the orientation of both the axes. The robots have a motoric capability to freely move from one point to another point on the two-dimensional plane. During a move, a robot travels a finite distance. The robots are identical in all respect, mainly with respect to their computational power. The robots perform the same algorithm. The robots are autonomous and there is no selected leader within the team of robots. The robots are silent, as there is no direct communication or explicit communication with the other robots. They have a limited range of view, so they only able to view the robots that are within their visibility range. The visibility graph for all the robots are presumed to be connected before any LCM cycle. At any point in time, the robot is either active or inactive. When a robot is active, then it performs a Look-Compute-Move cycle, which consists of three different stages.
Look - A robot observes the environment within its visibility circle by its sensor. The sensor results a view of the robots’ locations which are positioned within its sensing range. A snapshot is generated with respect to one time instance. 

Compute – In this phase, the robot computes the next point to move on, based on the perceived snapshot. 

Move – In this phase, the robot moves to the target point calculated in the former phase. In case the present positional point is same as the calculated point, then the robot makes a null move. Initially, all the robots are assumed to be inactive. When the robots are active, they take a finite time to complete LCM cycle. 

In our algorithm, we consider each robot as Rx and all other robots except Rx as Rr. In the Figure 5, the visibility circle of Rx is ABCD. It is denoted by Vx. Let a vertical straight line through Rx according to local coordinate system intersect Vx (ABCD) on the points A and C respectively. We introduce four functions top(Rx), bottom(Rx), left(Rx), right(Rx), which returns parts of the snapshot, after processing the stored snapshot from the previous cycle. 

top(Rx) - returns the partial snapshot of all robot Rr on the straight line ARx. 
bottom(Rx) - returns the partial snapshot of all robot Rr on the straight line RxC. 
left(Rx) - returns the partial snapshot of all robot Rr in the region ACD. 
right(Rx) - returns the partial snapshot of all robot Rr in the region ABD. 

![Figure 5. Visibility circle of Rx.](image)

4. Algorithm and Its Correctness 

4.1. The Proposed Algorithm 

In this algorithm, a system of ASYNC robots with limited visibility are required to generate equivalent performance as a system of FSYNC robots. The algorithm proposed here takes 4 different persistent colors and considers a single stored snapshot to achieve synchronicity. The rules of the algorithm are given below and Figure 6 represents the transition diagram of the robots while they change color while performing the computational cycles. 

At the beginning of each LCM cycle, the lights are set to WHITE. During Look state a robot Rs views the locations of robots that are positioned within Vx, stores their positions in an array Position[] and their corresponding color of the lights in an array Light[] at that cycle. Next, the robot Rs changes its own color light based on the observations and values from Position[], Light[] pairs.
Figure 6. Workflow of Robot $R_x$

Algorithm **SYNCvis**

**State Look**: Capture the view/snapshot of the robots locations where $R_i \in V_x$

- $Position[r]$ - is the location of robot $R_i$ (w.r.t the coordinate system of robot $R_x$)
- $Light[r]$ - is the light’s color of robot $R_i$

**State Compute**:

- $dest = Position[x]$
- $case: Light[x]$

**Case: WHITE**

If ($\exists no R_r$ such that $R_r \in top(R_x) \lor R_r \in left(R_x)) \lor \ (\forall R_r \in top(R_x) \lor left(R_x), Light[x] == YELLOW)$, then store the current snapshot in an array $args[]$.

- $Light[x] = YELLOW$

**Case: YELLOW**

If ($\exists no R_r$ such that $R_r \in bottom(R_x) \lor R_r \in right(R_x)) \lor \ (\forall R_r \in bottom(R_x) \lor right(R_x)$, $Light[x] == GREEN)$,

- $Light[x] = GREEN$

**Case: GREEN**

If ($\exists no R_r$ such that $R_r \in top(R_x) \lor R_r \in left(R_x)) \lor \ (\forall R_r \in top(R_x) \lor left(R_x)$, $Light[x] == RED)$,

then perform the protocol PRO using snapshot in $args[]$.

- $dest = computed destination$

- $Light[x] = RED$

**Case: RED**

If ($\exists no R_r$ such that $R_r \in bottom(R_x) \lor R_r \in right(R_x)) \lor \ (\forall R_r \in bottom(R_x) \lor right(R_x)$, $Light[x] == WHITE)$,

- $Light[x] = WHITE$

**State Move**:

- Move($dest$)
The rules are given above based on the color of the lights the robots have during different states of their LCM cycles. Next section discusses the correctness of the algorithm.

4.2. Correctness Proof

In this section, we aim to present that the algorithm produces a full synchronous execution by the \( ASY NC \) robots.

**Lemma I:** All robots grabs the same snapshot of their initial distribution before executing their \( Compute \) state.

**Proof:** Initially, the robots have WHITE lights. Due to the rules discussed in algorithm \( SYNCvis \), among all the robots in the domain, the toprmost or the leftmost robot(s) in the domain is always the first one which converts its light to YELLOW. After that, the other robots gradually convert their lights from WHITE to YELLOW, starting from top-left towards the bottom-right. The rightmost or bottom-most robot will be the last one to convert their lights to YELLOW. Converting to YELLOW also signifies the capture of the current snapshot. Thus, all the robots eventually converting to YELLOW, have a snapshot of the other robots. Now the question is, do they have a same and static snapshot of their initial distribution or not?

We know that the robots are allowed to move only after the execution of \( Compute \) state, which results in the next destination. Using that snapshot a robot can compute the next point to move in \( Move \) state. Due to the rules, the robots converting to YELLOW can store the snapshot, but they are permitted to \( Compute \) only after all of them become GREEN; thus, they are not able to move immediately. The first robot that has converted their light to YELLOW need to wait before executing its \( Compute \) state while others are converting from WHITE to YELLOW. Even if all are converted to YELLOW, they need to wait until all of them become GREEN. Again, according to the rules, at first the bottom-most or the rightmost robot will convert its light from YELLOW to GREEN and gradually all the other robots starting from bottom right convert to GREEN towards the top left. The topmost or leftmost robot is the last one to be converted from YELLOW to GREEN.

This ensures that the robots will be able to capture the same static snapshot similar to their initial distribution before executing the \( Compute \) as they are not able to move during conversion from WHITE to YELLOW and YELLOW to GREEN even when the robots are \( ASY NC \) and have limited visibilities.

**Lemma II:** Starting from an initial asynchronous \( Looks \), all the robots execute fully synchronous \( Compute \) and \( Move \) states.

**Proof:** According to Lemma I, robots will always be able to capture the same static snapshot similar to their initial distribution before executing the \( Compute \). Even if the robots are asynchronous by Lemma I, they have the same snapshot before \( Compute \). Robots are permitted to start \( Compute \) only after all of them become GREEN. Similarly, they are permitted to \( Move \) only after all of them become RED. Thus, results in synchronous \( Compute \) and \( Move \) for all the robots. So, this can be claimed that:

**Theorem:** Under limited visibility capabilities, \( ASY NC \) robots loaded with 4 colored lights and one snapshot eventually become \( FSYNC \) in finite time, i.e. \( ASY NC_4 \equiv FSYNC \).

5. Conclusion and Future Scope

This paper studies various models based on synchrony. The study also includes a detailed comparison between the models and possible uses of lights and snapshots to make a weak model powerful than a comparatively higher model.

Previous studies consider robots with unlimited visibilities. In those works, to achieve synchrony various computation and decision making was done by the robots based on the color of lights of all other visible robots. But in a practical scenario, robots may have limited visibility. If robots have limited
visibility, it is not possible for a robot to view all other robots and their lights. So, achieving synchrony is a challenging job in case of limited visibility of the robots. The uniqueness of this work is to show that when the \textit{ASYNC} robots with limited range of visibilities are equipped with a number of colored lights and snapshots of the environment, they are as powerful as \textit{FSYNC} robots. The algorithm completes in finite time. It is also shown that even if there is no communication among the robots, eventually they become synchronous by following the algorithm. We have embedded 4 colored lights and the capability to remember only one snapshot to the robots. As a future scope of work, this algorithm can be studied for real-world applications. Also, research can be done to reduce the number of lights.

References

[1] Heo, N and Varshney, P K 2003 A distributed self spreading algorithm for mobile wireless sensor networks. \textit{In IEEE Wireless Communications and Networking}, 3, pp 1597-1602.
[2] Marjovi, A, Nunes, J G, Marques, I and Almeida, A d 2009 Multi-robot exploration and fire searching. \textit{In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009}, pp 1929-1934.
[3] Antoni, D, Ban, E and Agar, M 2001 Demining robots - requirements and constraints. \textit{ATKAFF}, 42, 3-4, pp 189-197.
[4] Prencipe, G 2001 Corda: Distributed coordination of a set of autonomous mobile robots. \textit{In: 4th European Research Seminar on Advances in Distributed Systems, Bertinoro, Italy}, pp 185-190.
[5] Peleg, D 2005 Distributed coordination algorithms for mobile robot swarms: new direction and challenges, \textit{In: 7th International Workshop on Distributed Computing (IWDC), Ser. LNCS 3741}, pp 1-12.
[6] Efrima, A and Peleg, D 2007 Distributed models and algorithms for mobile robot systems, \textit{In: 33rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM), Ser. LNCS 4362}, pp 70-87.
[7] Das, S, Flocchini, P, Prencipe, G, Santoro and N, Yamashita, M 2010 On the computational power of oblivious robots: forming a series of geometric patterns, \textit{In: Proceedings of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing (PODC ’10)}, ACM, NY, USA, pp 267-276.
[8] Das, S, Flocchini, P, Prencipe, G, Santoro, N and Yamashita, M 2012 The power of lights: synchronizing asynchronous robots using visible bits, \textit{In: IEEE 32nd International Conference on Distributed Computing Systems, Macau}, pp 506-515.
[9] Das, S, Flocchini, P, Prencipe, G, Santoro, N and Yamashita, M 2016 Autonomous mobile robots with lights, \textit{Theoretical Computer Science}, 609, Part 1, pp 171-184.
[10] Ooshita F and Tixeuil S 2018 Ring exploration with myopic luminous robots. \textit{In: Izumi T., Kuznetsov P. (eds) Stabilization, Safety, and Security of Distributed Systems. SSS 2018. LNCS 11201, Springer, Cham}, pp 301-316.
[11] Nagahama S, Ooshita, F and Inoue, M 2019 Ring Exploration of myopic luminous robots with visibility more than one. \textit{In: Ghaffari M., Nesterenko M., Tixeuil S., Tucci S., Yamauchi Y. (eds) Stabilization, Safety, and Security of Distributed Systems. SSS 2019. LNCS 11914, Springer, Cham}, pp 256-271.
[12] Bramas Q, Devismes, S and Lafourcade, P 2019 Infinite grid exploration by disoriented robots. \textit{In: Censor-Hillel K., Flammini M. (eds) Structural Information and Communication Complexity. SIROCCO 2019. LNCS 11639, Springer, Cham}, pp 340-344.