Robots with Lights: Overcoming Obstructed Visibility Without Colliding

G. A. Di Luna¹, P. Flocchini², S. Gan Chaudhuri³, N. Santoro⁴, and G. Viglietta²

¹ Dipartimento di Ingegneria Informatica, Automatica e Gestionale Antonio Ruberti, Università degli Studi di Roma “La Sapienza”, Rome, Italy, diluna@dis.uniroma1.it
² School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa ON, Canada, flocchin@site.uottawa.ca, viglietta@gmail.com
³ Department of Information Technology, Jadavpur University, Kolkata, India, srutiganc@it.jusl.ac.in
⁴ School of Computer Science, Carleton University, Ottawa ON, Canada, santoro@scs.carleton.ca

Abstract. Robots with lights is a model of autonomous mobile computational entities operating in the plane in Look-Compute-Move cycles: each agent has an externally visible light which can assume colors from a fixed set; the lights are persistent (i.e., the color is not erased at the end of a cycle), but otherwise the agents are oblivious. The investigation of computability in this model, initially suggested by Peleg, is under way, and several results have been recently established. In these investigations, however, an agent is assumed to be capable to see through another agent.

In this paper we start the study of computing when visibility is obstructable, and investigate the most basic problem for this setting, Complete Visibility: The agents must reach within finite time a configuration where they can all see each other and terminate. We do not make any assumption on a-priori knowledge of the number of agents, on rigidity of movements nor on chirality. The local coordinate system of an agent may change at each activation. Also, by definition of lights, an agent can communicate and remember only a constant number of bits in each cycle. In spite of these weak conditions, we prove that COMPLETE VISIBILITY is always solvable, even in the asynchronous setting, without collisions and using a small constant number of colors. The proof is constructive. We also show how to extend our protocol for COMPLETE VISIBILITY so that, with the same number of colors, the agents solve the (non-uniform) CIRCLE FORMATION problem with obstructed visibility.

1 Introduction

1.1 Framework

In the traditional model of distributed computing by mobile entities in the plane, called robots or agents, each entity is modelled as a point; it is provided with a local coordinate system (not necessarily consistent with that of the other agents); it has sensorial capabilities, called vision, enabling it to determine the position (within its own coordinate system) of the other agents. The agents are anonymous, they are indistinguishable, and they execute the same code.

Agents operates in Look-Compute-Move cycles: when becoming active, an agent uses its sensing capabilities to get a snapshot of its surroundings (Look), then this snapshot is used to compute a destination point (Compute), and finally it moves towards
this destination (Move); after that, the agent becomes inactive. In the majority of investigations, the agents are assumed to be oblivious: at the beginning of each cycle, an agent has no recollection of its past observations and computations \[11\]. Depending on the assumptions on the activation schedule and the duration of the cycles, three main settings are identified. In the fully-synchronous setting, all agents are activated simultaneously, and each cycle is instantaneous. The semi-synchronous setting is like the fully synchronous one except that the set of agents to be activated is chosen by an adversary, subject only to a fairness restriction: each agent will be activated infinitely often. In the asynchronous setting, there is no common notion of time, and no assumption is made on timing of activation, other than fairness, nor on the duration of each computation and movement, other than it is finite.

Vision and mobility provide the agents with stigmergy, enabling the agents to communicate and coordinate their actions by moving and sensing their relative positions. The agents are otherwise assumed to be silent, without any means of explicit direct communication \[11\]. This restriction enables deployment in extremely harsh environments where communication is not possible, i.e. an underwater deployment or a military scenario where wireless communication are impossible or can be jammed. Nevertheless, in many other situations it is possible to assume the availability of some sort of direct communication. The theoretical interest is obviously for weak communication capabilities.

A model employing a weak explicit communication mechanism is that of robots with lights: in this model, each agent is provided with a local externally visible light, which can can assume colors from a fixed set; the agents explicitly communicate with each other using these lights \[5, 6, 10, 12, 14, 16\]. In this model, the lights are persistent (i.e., the color is not erased at the end of a cycle), but otherwise the agents are oblivious.

The classical model of silent entities and the more recent model of entities with visible lights share a common assumption, that visibility is unobstructed. That is, three or more collinear agents are assumed to be mutually visible. It can be easily argued against such an assumption, and for the importance of investigating computability when visibility is obstructed by presence of the agents: given three collinear agents, the one in the middle blocks the visibility between the other two and they cannot see each other.

Nothing is known on computing with obstructed visibility except for the investigations on the so-called fat agents model, where agents are not points but unit discs, and collisions are allowed\[5\] and can be used as an explicit computational tool (e.g., \[1, 2, 4\]); and for the study of uniformly spreading agents operating in a one dimensional space (i.e., on a line) \[3\]. In this paper we start to fill this void, and focus on agents with visible lights in presence of obstructed visibility.

The problem we investigate is perhaps the most basic in a situation of obstructed visibility, and it is the one of the agents reaching a configuration of complete unobstructed visibility. More precisely, this problem, that we shall call COMPLETE VISIBILITY, requires the agents, starting from an arbitrary initial configuration where

\[5\] In pointilinear models, collisions create unbreakable symmetries; thus, unless this is the required outcome of the problem, their avoidance is required by all solution protocols. In addition, in real world implementations, collisions (e.g., of two quad copters) may have unpredictable outcomes that might be better avoided.
they are in distinct points but might be unable to see everybody and might not know the
total number of agents\footnote{The actual number of agents may be unknown for several reasons; e.g., if the deployment of agents has been done by an airplane, a subset of agents may be lost or destroyed during the landing process.}. to reach within finite time a configuration in which every agent
is in a distinct location from which it can see all other agents, and no longer move.

Among the configurations that achieve complete visibility, a special class is that
where all agents are on the perimeter of a circle (not necessarily equally spaced). The
problem of forming any such a configuration is called CIRCLE FORMATION and it has
been extensively studied both in the classical model of silent agents and in the ones with
visible lights (e.g., \cite{7,9,13,15}). Unfortunately, none of these investigations consider
obstructed visibility, and their algorithms do not work in the setting considered here.

\section{Our Contributions}

In this paper we study solving COMPLETE VISIBILITY by robots with lights. That is,
we consider autonomous and anonymous agents, each endowed with a visible light that
can assume a constant number of persistent colors, that are otherwise oblivious, and
whose visibility is obstructed by other agents in the line of sight; and we investigate
under what conditions they can solve COMPLETE VISIBILITY and at what cost (i.e.,
how many colors).

We do not make any assumptions on a-priori knowledge on the number of agents,
nor on agreement on coordinate systems, unit of distance and chirality; actually, the lo-
cal coordinate system of an agent may change at each activation. Neither we make any
assumption on rigidity of movements; that is, a move may be stopped by an adversary
before the agent reaches its destination; the only constraint is that, if interrupted before
reaching its destination, the agent moves at least a minimum distance $\delta > 0$ (other-
wise, no destination can ever be reached). Also, by definition of lights, an agent can
communicate and remember only a constant number of bits in each cycle.

In spite of these weak conditions, we prove that COMPLETE VISIBILITY is always
solvable, even in the asynchronous setting, without collisions and using a small constant
number of colors. The proof is constructive. We first design a protocol that achieves
complete visibility with six colors under a semi-synchronous scheduler. We then show
how to transform it into an asynchronous algorithm with only four additional colors.
We also show how to extend the protocol so that, under the same weak conditions and
without increasing the number of colors, the agents can position themselves on the
perimeter of a circle. In other words, we also show how to solve the (non-uniform)
CIRCLE FORMATION problem with obstructed visibility.

Due to lack of space, some of the proofs are sketched and some omitted.

\section{Model and Definitions}

Consider a set of mobile anonymous agents $\mathcal{A} : \{a_1, a_2, \ldots, a_n\}$. Each agent $a_i$ has a
persistent state variable $s_i$, which may assume any value in a finite set of colors $C$. 
We denote by \( x_i(t) \in \mathbb{R}^2 \) the position occupied by agent \( a_i \) at time \( t \) expressed in some global coordinate system (used only for description purposes, and unknown to the agents); when no ambiguity arises, we omit the indication of time. A configuration \( C \) is a set of \( n \) tuples in \( C \times \mathbb{R}^2 \) each defining the position and color of an agent; let \( C_t \) denote the configuration at time \( t \).

Each agent \( a_i \) has its own system of coordinates centered in itself, which does not necessarily agree with those of the other agents, i.e. there is no common unit of measure and not common notion of clockwise orientation. Agents \( a_i \) and \( a_j \) are visible to each other at time \( t \) if and only if the segment \( x_i(t) x_j(t) \) does not contain any other agents. Let \( C_t[a_i] \) denote the set of the positions and colors of the agents visible to \( a_i \) at time \( t \). We shall call such a set local view. A configuration \( C \) is said to be obstruction-free if \( \forall a_i \in A \) we have \( |C[a_i]| = n \); that is, if all agents can see each other. Two agents \( a_i \) and \( a_j \) are said to collide at time \( t \) if \( x_i(t) = x_j(t) \).

At any time, agents can be active or inactive. When activated, an agent \( a_i \) performs a sequence of operations called Look-Compute-Move: it activates the sensors to obtain a snapshot (called local view) of the positions of the visible agents expressed in its own coordinate system (Look); it then executes an algorithm (the same for all agents) based on its local view, which returns a destination point \( x \in \mathbb{R}^2 \) and a color \( c \in C \) (Compute); it then sets its own state variable to \( c \) and moves towards \( x \) (Move). The movement may be stopped by an adversary before the agent reaches its destination; the only constraint on the adversary is that, if interrupted before reaching its destination, a robot moves at least a minimum distance \( \delta > 0 \) (otherwise, no destination can ever be reached).

We consider two schedulers for the activation of the agents: Semi-Synchronous (SSYNC) and Asynchronous (ASYNC). In SSYNC, the time is discrete; at each time instant \( t \) (called round) a subset of the agents is activated and performs its operational cycle instantaneously. The choice of the activation is done by an adversary, which however activates each agent infinitely often. In ASYNC, there is no common notion of time; each agent is activated independently, and each Compute and Move operation can take an unpredictable (but bounded) amount of time, unknown to the agent.

At the beginning (time \( t=0 \)), the agents start in an arbitrary configuration \( C_0 \) occupying different positions, and they are black (the state variable of each one is set to a special symbol \( b \)). The goal is for the agents to reach, in finite time, an obstruction-free configuration without ever colliding. We call this problem COMPLETE VISIBILITY. An algorithm is said to solve the problem if it always achieves complete visibility regardless of the choices of the adversary, and from any initial configuration.

Let \( H \) be the convex hull defined by \( C_t \), let \( \partial H = V_t \cup B_t \) denote the agents on the border of \( H \), where \( V_t : \{v_1, . . . , v_k\} \subseteq A \) is the set of agents (corner-agents) located at the corners of \( H \) and \( B_t : \{b_1, . . . , b_t\} \) is the set of those located on the edges of \( H \) (edge-agents); let \( I_t \) be the set of agents that are interior of \( H_t \) (interior-agents). Let \( n_t = |V_t| \) be the number of corners in \( H \). Given an agent \( a_i \in A \), we denote by \( H_t[a_i] \) the convex hull of its local view \( C_t[a_i] \). Let \( C_t^c \) indicate the set of agents in \( C_t \) with color \( c \) at time \( t \), similarly we define \( H_t^c[a_i] \) as the convex hull, of \( C_t^c[a_i] \). Analogously defined are the extensions of \( V_t, B_t, I_t \). Given a configuration \( C \), we indicate by \( SEC(C) \) the smallest enclosing circle containing \( C \) (when no ambiguity
arises we just use the term SEC). Given two points \( x, y \in \mathbb{R}^2 \) with \( xy \) we indicate the line that contains them, and we use the operator \( \cap \) to indicate the intersection of lines and segments. Let \( d(x, y) \) indicate the Euclidean distance between two points (or a segment and a point); moreover, given \( x, y, z \in \mathbb{R}^2 \) we use \( \angle xyz \) to indicate the angle with vertex \( y \) and sides \( xy, yz \). In the following, with an abuse of notation, when no ambiguity arises, we use \( a_i \) to denote both the agent and its position.

3 Complete Visibility in SSYNC

In this Section we provide an Algorithm that reaches Complete Visibility in the semi-synchronous setting. The algorithm is described assuming \( |V_0| \geq 3 \); we will then show how the agents can easily move to reach this condition starting from a configuration with \( |V_0| = 2 \).

Our algorithm works in two phases: (1) Interior Depletion (ID) and (2) Edge Depletion (ED). The purpose of the Interior Depletion phase is to reach a configuration \( C_{ID} \) in which there are no interior-agents. In this phase, the interior-agents move towards an edge they perceive as belonging to the border of the convex hull, and they position themselves between two corner-agents. At the end of this phase, all agents are on \( \partial H_0 \).

The goal of the Edge Depletion phase is to have all agents in \( B_{ID} \) to move so to reach complete visibility.

3.1 Phase 1: Interior Depletion Phase

Initially all agents are black. The objective of this phase is to have all agents on \( \partial H_0 \), with the corner-agents colored red and the edge-agents colored brown.

Notice that corner (resp. edge) agents are able to recognize their condition in spite of possible obstructions. In fact, if a black agent \( a_i \) is activated at some round \( r \), and it sees that \( C_r[a_i] \) contains a region of plane that is free of agents and wider than \( 180^\circ \), then \( a_i \) knows it is a corner and sets its variable \( s_i \) to red. A similar rule is applied to edge-agents; in this case, an edge-agent \( a_i \) sets its variable \( s_i \) to brown if \( C_r[a_i] \) contains a region of plane free of agents and wide exactly \( 180^\circ \) (see Coloring Case of Figure 1).

In the ID phase, corner-agents color themselves red, and no longer move, while edge-agents color themselves brown. Each interior-agent \( a \) moves to position itself on one of its nearest visible edges of \( \partial H_0 \); note that an edge of \( \partial H_0 \) can be recognized in \( a \)'s local view once it is occupied only by brown and red agents. To prevent collisions, the interior-agent moves towards the chosen edge \( e \) perpendicularly if and only if it is one with minimum distance to \( e \) and its destination on \( e \) is empty; otherwise it does not move. An edge-agent on the destination of an interior one, slightly moves to make room for the interior-agent. The INTERIOR DEPLETION algorithm is detailed in Figure 1.

It is easy to see that at the end of this phase, all the agents will be positioned on a convex hull.

Lemma 1. For any initial configuration \( C_0 \) there exists a round \( r \in \mathbb{N}^+ \) such that in \( C_r \) we have that \( I_r = \emptyset \); furthermore, this occurs without collisions.
Algorithm \textsc{Interior Depletion} (for the generic agent \(a_i\) activated at round \(r\))

- Coloring Case: if \((s_i = \text{black})\) then:
  
  \begin{itemize}
    \item If \((a_i\) is a corner-agent in \(\mathcal{H}_r[a_i]\)) then \(a_i\) sets \(s_i = \text{red}\)
    \item If \((a_i\) is an edge-agent in \(\mathcal{H}_r[a_i]\)) then \(a_i\) sets \(s_i = \text{brown}\)
  \end{itemize}

- Interior Case: if \((a_i\) is interior in \(\mathcal{H}_r[a_i]\)) and \(s_i = \text{black}\) then:
  
  \begin{itemize}
    \item \(a_i\) uses its local view \(C_i[a_i]\) to determine the edges of \(\partial \mathcal{H}_r[a_i]\).
    \item If \((\exists e \in \partial \mathcal{H}_r[a_i]\) such that \(\forall a_j \in \mathcal{I}_r[a_i]\), \(d(a_j, e) \leq d(a_i, e)\)) then
      
      \begin{itemize}
        \item \(a_i\) computes a point \(x\) of \(e\) such that \(\overrightarrow{a_i x} \perp e\); if \(x\) is empty, then \(a_i\) moves toward \(x\).
      \end{itemize}
  \end{itemize}

- Obstructing Edge Case: if \((s_i = \text{brown})\) then:
  
  \begin{itemize}
    \item Let \(e\) be the edge to which \(a_i\) belongs; if \((\exists a_j \in \mathcal{I}_r[a_i] \land \overrightarrow{a_j a_i} \perp e)\), then \(a_i\) moves toward the nearest point \(x \in e\) such that \(\forall a_k \in \mathcal{I}_r[a_i], \overrightarrow{a_k x} \not\perp e\).
  \end{itemize}

Fig. 1: Algorithm for the Interior Depletion Phase

\textbf{Theorem 1.} There is a round \(r \in \mathbb{N}^+\) such that the agents occupy different positions on \(\mathcal{H}_r\). Moreover, the corner-agents are red, and the edge-agents are brown.

\subsection{Phase 2: Edge Depletion -ED}

The purpose of the ED phase is to move the edge-agents out of the current convex hull to reach a final configuration whose convex hull includes \(\mathcal{H}_0\) and all agents are on the corners, thus achieving complete visibility.

The algorithm makes an edge-agent move from its edge \(e = \overrightarrow{v_0v_1}\) to a point out of the current convex hull, but within a \textit{safe zone}. Safe zones are calculated so to guarantee that \textit{red} agents never cease to be located on corners of the current convex hull, in spite of the movement of the edge-agents. More precisely, the safe zone \(S(e)\) of \(e\) consists of the portion of plane outside the current convex hull, such that \(\forall x \in S(e)\) we have \(\angle xv_0v_1 < \frac{180^\circ - \angle v_0v_1v_2}{4}\) and \(\angle v_0v_1x < \frac{180^\circ - \angle v_0v_1v_2}{4}\) (see Figure 2a).

Note that, due to the mutual obstructions that lead to different local views, edge-agents cannot always compute \(S(e)\) exactly (see Figure 2b). In fact, only when there is a single edge-agent between the two \textit{red} corner-agents on \(e\), the computation of \(S(e)\) is exact; in any case, we can show that the safe area \(S'(e)\) computed by an agent is \(S'(e) \subseteq S(e)\) and thus still safe.

The migration of edge-agents and their transformation into corner-agents occurs in steps: in fact, if the edge \(e\) contains more than one edge-agent, our algorithm makes them move in turns, starting from the two agents \(b_1\) and \(b_0\) that are immediate neighbors of the corners \(v_1\) and \(v_0\), respectively. Only once they are out of the convex hull and they are corner of a new edge \(e'\), other agents on \(e\) will follow, always moving perpendicularly to \(e'\). Careful changes of colors are required to coordinate this process. In fact, once the first pair is in position, the two agents will become \textit{blue} to signal the other \textit{brown} agents on \(e\) that it is their turn to move out; they will set their color to \textit{red}.
only when there is no interior-agent in the space delimited by $e'$ and $e$. Once red, their color will never change until completion.

Due to the different estimations of $S$, to semi-synchronicity, and to the unpredictable distance traversed by an agent (possibly stopped before destination), a variety of situations could disrupt this ideal behaviour. In particular, it could happen that only one of the two agents, say $b_1$, moves while the other stays still, or that $b_1$ moves further from $e$ than $b_0$. In both cases this leads to a configuration in which $b_0$ becomes a interior or edge-agent. This problem is however adjusted by $b_1$ that, when noticing the situation, moves towards $v_1$ until $b_0$ becomes a corner in $\mathcal{H}[b_1]$. A further complication is that $b_1$ might wrongly perceive $b_0$ as a corner and thus decide not to move; this occurs if $v_0 b_0$ happens to be collinear with $b_1$ obstructing visibility; such a case is however detected by $b_0$ itself, which uses a different color (orange) to signal that $b_1$ has to move further towards $v_1$ to transform $b_0$ into a corner (see Figure 2d).

Fig. 2: Edge Depletion Phase

The detailed algorithm for the ED phase is reported in Figure 3.

3.3 The case of $|\mathcal{V}_0| = 2$

The strategy of the previous Section works for $|\mathcal{V}_0| > 2$. It is however simple to have the agent move to reach such a condition from $|\mathcal{V}_0| = 2$, as described below.

When $|\mathcal{V}_0| = 2$ the agents are necessarily disposed forming a line and $|A| \geq 2$. First notice that an agent $a$ can detect that the configuration is a line, and whether it is an extremity (i.e., it sees only one other agents $a'$), or an internal agent (i.e., it is
Algorithm Edge Depletion

For agent \( a_i \) activated at round \( r \), to be executed if and only if \( \tilde{h}(black, a_i) \in C_r[a_i] \).

- Execute \text{Compute Order} \ and \ appropriate \ case \ from \ the \ list \ below.

- **Brown Edge Case:** \( a_i \) belongs to an edge \( e \) of \( C_r[a_i] \) and \( s_i = brown \).
  - If \( a_i \) is the only agent on \( e \) then
    - \( a_i \) computes the angles \( \alpha = 180^\circ - \angle v_{o}v_{a}v_{o}, \beta = 180^\circ - \angle v_{a}v_{1}v_{2}, \) and \( \gamma = \min\left(\frac{\alpha}{2}, \frac{\beta}{2}\right) \); it then computes a point \( x \) such that \( \angle xv_{1}a_{i} < \gamma \) and \( \angle xv_{0}a_{i} < \gamma \).
    - \( a_i \) sets \( s_i = yellow \).
    - \( a_i \) moves perpendicularly to \( e \) with destination \( x \).
  - If \( a_i \) is not the only non-red \ agent \ on \( e \) and one of its neighbors on \( e \) is red \ (by \ routine \ \text{ComputeOrder}, \ this \ agent \ is \( v_j \)) then: let \( b \) be its other neighbor;
    - \( a_i \) computes the two angles \( \alpha = 180^\circ - \angle v_{a}v_{1}v_{2}, \beta = 180^\circ - \angle v_{a}v_{1}a_{i} \), and \( \gamma = \min\left(\frac{\alpha}{2}, \frac{\beta}{2}\right) \); it then computes a point \( x \) such that \( \angle xv_{1}a_{i} < \gamma \) and \( \angle xba_{i} < \gamma \).
    - \( a_i \) sets \( s_i = yellow \).
    - \( a_i \) moves perpendicularly to \( e \) with destination \( x \).

- **Yellow Case:** \( s_i = yellow \).
  - if there is another yellow or blue \ agent \( a_j \) \ with \( e^{a_i} = e^{a_j} \) then
    - if \( a_i \cap a_j \not\in C_r[a_i] \) then \( a_i \) sets \( s_i = yellow \).
    - if \( a_i \not\in C_r[a_i] \) then \( a_i \) moves towards \( v_1 \) along \( \alpha_{1v_1} \) of \( \frac{d(a_i,v_{1})}{2} \).
    - \( a_i \) sets \( s_i = orange \).
  - else if \( \tilde{h}(s_j, a_j) \in C_r[a_i] \) with \( a_j \neq a_i \) and \( e^{a_i} = e^{a_j} \) and \( \tilde{h}(s_j, a_j) \in C_r[a_i] \cap e^{a_i} \), then
    - \( a_j \) set \( s_i = red \).

- **Orange Case:** \( s_i = orange \).
  - if there is another blue \ agent \( a_j \) \ with \( e^{a_i} = e^{a_j} \) then
    - if \( a_j \not\in C_r[a_i] \) then \( a_i \) sets \( s_i = blue \).

- **Blue Case:** \( s_i = blue \) and \( a_i \in e \) \ with \( e \) edge of \( C_r[a_i] \).
  - if there is another \ orange \ agent \( a_j \) \ with \( e^{a_i} = e^{a_j} \) then
    - \( a_i \) moves along \( \alpha_{a_i} \) in direction \( e_{1} \) \ towards \ the \ point \ at \ distance \( \frac{d(a_i,v_{1})}{2} \).
    - else if \( \tilde{h}(brown, a_j) \in C_r[a_i] \) such that \( a_j \) could move to \( e \) then
      - \( a_j \) sets \( s_i = red \).

- **Brown Interior Case:** \( a_i \) is such that \( s_i = brown \) and \( a_i \in I_r[a_i] \).
  - if there exists and edge \( e' = \alpha_{a_i}a_{j} \) \ with \( s_2 = s_y = blue \) and \( a_i \) could move perpendicularly towards \( e' \) \ without crossing any segment delimited by two \ red \ agents, \ then \( a_i \) moves towards \( e' \).
  - if \( a_i \in e = \alpha_{a_i}a_{j} \) \ with \( e \in \partial H_{r}^{(red,brown)}[a_i] \) and \( \exists x \in \mathbb{R}^2 \) such that \( a_0a_i \perp \alpha_{a_1} \) and \( b_{a_1} \in \angle a_0a_1x \) or \( b_{a_j} \in \angle a_1a_1x \) \ then \( a_i \) executes the second subcase \ of \ the \ Brown \ Edge \ Case.

- **Corner Case:** \( a_i \) is a corner of \( C_r[a_i] \) and \( s_i = red \).
  - \( a_i \) can check \ local \ termination \ and \ the \ global \ termination
    - \( a_i \) locally terminates when \( s_i = red \)
    - \( a_i \) detects the global termination of ED phase when \( \tilde{h}(s_j, a_j) \in C_r[a_i] \) with \( s_j \neq red \)

Fig. 3: Edge Depletion Phase algorithm
Procedure **Compute Order**

- if \( a_i \) belongs to an edge \( e \) of \( \mathcal{H}_r[a_i] \) and \( s_i = \text{brown} \), it orders the red agents in its local view in a circular order, starting from the closest, \( (v_1, v_2, \ldots, v_0) \).
- if \( s_i \in \{\text{orange, blue, yellow}\} \), then \( a_i \) determines which of its current neighbors was \( v_1 \) in its previous computation and the edge \( e_{a_i} = v_1v_0 \) to which it belonged:
  - \( a_i \) computes the nearest edge \( e = \{u,v\} \in \mathcal{H}_r^{red}[a_i] \)
  - \( a_i \) computes the point \( x \in \mathbb{R}^2 \) such that \( \overrightarrow{uv} \perp \overrightarrow{a_ix} \)
  - \( a_i \) sets \( v_1 = u, v_0 = v \) if \( \exists a_j \in \angle uxva_i \), otherwise it sets \( v_1 = v, v_0 = u \).
  - \( a_i \) sets \( e_{a_i} = \overrightarrow{v_1v_0} \)

| Color    | Meaning                                      | Transition to:               |
|----------|----------------------------------------------|------------------------------|
| Black    | initial color of all agents                  | \{Red, Brown\}              |
| Brown    | agents on edges or having to move to a new edge of \( \mathcal{H} \) | Yellow                      |
| Yellow   | agents moving out of \( \mathcal{H} \) to form a new edge | \{Blue, Orange, Red\}       |
| Orange   | agents needing to be transformed into corners| Blue                        |
| Blue     | corner-agent now forming a new edge \( e \), waiting for other agents to move to \( e \) | Red                         |
| Red      | a stable corner-agent                         |                              |

Fig. 4: Colors used in the **Complete Visibility** algorithm.

between two collinear agents). If \( a \) is internal, it does not move; if it is an extreme, \( a \) sets its color to red and moves perpendicular to the segment \( \overrightarrow{a'a} \). This means that, as soon as at least one of the extremes is activated, it will move (or they will move) creating a configuration with \( |V| > 2 \). At that point, the algorithm previously described is applied.

**Correctness of the ED phase.** With the following lemma we show that the global absence of interior-agents with respect to the initial convex hull, can be locally detected by each agent.

**Lemma 2.** Given an agent \( a_i \in \mathcal{A} \) with \( s_i \in \{\text{red, brown}\} \) and a round \( r \in \mathbb{N}^+ \), if \( \#(\text{black}, a_j) \in \mathcal{C}_r[a_i] \) then \( \mathcal{C}_r \) does not contain interior-agents with respect to \( \mathcal{H}_0 \).

**Proof.** By contradiction, assume that \( \#(\text{black}, a_j) \in \mathcal{C}_r[a_i] \) but there exists at least an interior-agent \( a \) with respect to \( \mathcal{H}_0 \). By the rules of the ID phase, agent \( a \) cannot change its color from black to another because it can detect it is neither a corner nor a border. Thus, \( a \) is not in \( \mathcal{C}_r[a_i] \) because \( \mathcal{C}_r[a_i] \), by assumption, does not contain black agents. Thus, it must exist an agent \( a_k \) that has color different from black and \( a_k \in \bigcup_{r \in \mathbb{N}^+} \mathcal{C}_r \). But since \( a \) is interior then also \( a_k \) is interior, and so \( s_k = \text{black} \). \( \square \) **Lemma 2**
We now show that the safe area $S'(e)$ computed by an edge-agent on $e$ is such that $S'(e) \subseteq S(e)$ and thus its movement is still safe (it does not transform a red corner into an interior or edge-agent).

**Lemma 3.** Given a configuration $C_r$ and an edge $e = \overrightarrow{vu}$ of $H_r$, if an agent $a_j \in e$ moves from $e$, it moves inside the safe zone $S(e)$

**Proof.** The case when there is a single edge-agent $b \in e$ is trivial because $b$ can compute exactly $S(e)$. Consider now the case when there are two or more edge-agents on $e$; among those, let $b_0$ and $b_1$ be the two that are neighbors of $v_0, v_1$. Those agents move only when executing the Brown Edge Case or Brown Interior Case. Let us consider the movement of the first that is activated, say $b_1$. Agent $b_1$ has two neighbors on $e$: a brown neighbor $b$ and the red corner $v_1$. Agent $b_1$ orders the corners in its view from $v_1$ to $v_{last}$, according to its local notion of clockwise, where $v_{last}$ is the last corner before $b$, i.e. $v_{-1}$ in Figure 2b. Following the rules of the algorithm, $b_1$ computes: $\alpha = 180^\circ - \angle v_{last}bb_1$, $\beta = 180^\circ - \angle b_1v_1v_2$, and $\gamma = \min(\frac{\alpha}{4}, \frac{\beta}{4})$. Angle $\angle v_{last}bb_1$ is an upper bound on $\angle v_{last}v_0b_1$, otherwise we could get a contradiction since $v_{last}b$ and $v_{last}v_0$ will intersect in two points: one is $v_{last}$ and the other one is after the intersection of $v_{last}v_0$ and $v_0v_1$, that is impossible. Thus, $\alpha$ is a lower bound on the angle that a single agent would compute on $e$, which implies that $b_1$ will move inside $S(e)$. The same holds for $b_0$. Notice that, given two points $x$ and $y$ inside the safe zone, any point $z \in \overline{xy}$ is still inside the safe zone, thus any agent that moves on the lines connecting two agents inside $S(e)$ will still be in $S(e)$, completing the proof. □ Lemma 3.

The next lemma shows that the moves of our algorithm cannot transform any red corner-agents into an interior-agent.

**Lemma 4.** Consider a corner-agent $v_1$ of $H_r$ with $s_1 = red$, we have that $\forall r \in \mathbb{N}^+$ with $r > r'$, $v_1$ is also a red corner-agent of $H_r$.

**Proof.** It is easy to see that during the ID phase we have that $H_r = H_0$ since the interior-agents will never trespass the edges of $H_0$, so the hypothesis holds. We have to show that the same holds during the ED phase. We have that $v_1$ never moves after it sets $s_1 = red$ so if $v_1$ is a corner it cannot become interior as a consequence of its own move. Consider the two edges adjacent to $v_1$: $e_1 = \overrightarrow{v_0v_1}$ and $e_2 = \overrightarrow{v_1v_2}$. Assume, by contradiction, that there exists a round $r$ in which the moves of a set $X$ of agents on these two edges is such that $v_1$ is a corner-agent in $H_{r−1}$ but not in $H_r$. From Lemma 3, we have that agents in $X$ move to points inside the safe zones $S(e_1)$ and $S(e_2)$ of $e_1, e_2$. Let us consider two points $x \in S(e_1)$ and $y \in S(e_2)$, such that agents on them will make $v_1$ interior. If $v_1$ is interior in $H_r$, we have that $\angle v_1xy > 180^\circ$. It is easy to see that $\angle v_0v_1x < \gamma$ (see Brown Edge Case and Brown Interior Case of Figure 3) and that $\gamma \leq \frac{180^\circ - \angle v_0v_1v_2}{4}$, since $\gamma = \min(\frac{\alpha}{4}, \frac{\beta}{4})$, and that at least one of the two among $\beta, \alpha$ is a lower bound on $180^\circ - \angle v_0v_1v_2$. The same holds for $y$, so we have $\angle v_2v_1y \leq \frac{180^\circ - \angle v_0v_1v_2}{4}$. Thus, we have $\angle v_0v_1x + \angle v_2v_1y + \angle v_0v_1v_2 < 180^\circ$ and then $\angle v_1xy < 180^\circ$, which is a contradiction. So, $v_1$ cannot be interior in $H_r$. The same arguments hold if at round $r−1$ we consider a set of agents $X$ on two edges $e', e''$ that are not adjacent to $v_1$: this is easy to see since, given $x \in S(e')$ and $y \in S(e'')$ we
have $\angle xv_1y \leq \angle v_0v_1v_2 < 180^\circ$, which is another contradiction to the hypothesis of $v_1$ being interior in $\mathcal{H}_r$.

In the next sequence of lemmas, we show that, given an edge $e$ in a configuration $C$ of the ED phase, all edge-agents in $e$ will eventually become red corners.

**Lemma 5.** Given a configuration $C_r$ and an edge $e$ of $\mathcal{H}_r$ with a single brown agent $b$ on $e$, eventually $b$ will be a red corner.

**Proof.** Since red corners never move and no interior-agents can be moving on $e$, while inactive, agent $b$ maintains its single position inside $e$. When activated at some round $r'$, agent $b$ executes the Brown Edge Case with a single agent. Thus $b$ switches color to yellow and it moves perpendicularly to $e$ of at least $\min(d(b, x), \delta)$. At round $r' + 1$, $b$ is a corner-agent of $\mathcal{H}_{r'+1}$; in the next activation, after executing the Yellow Case code, $b$ becomes red.

**Lemma 6.** Given a configuration $C_r$ and an edge $e$ of $\mathcal{H}_r$ with exactly two brown agents $b_0, b_1$ on $e$, eventually they will set their state variable to yellow and they will move outside $e$.

**Proof.** Let $b_1$ be the first to be activated at some round $r' \geq r$. At that time, $b_1$ switches its color to yellow and it moves perpendicularly to $e$ (see Brown Edge Case). Agent $b_0$ will do the same, no matter if it is activated in round $r'$ or in some successive rounds (see Brown Edge Case and Brown Interior Case).

**Lemma 7.** Given a configuration $C$, any agent $b_1$ with $s_1 = \text{yellow}$ eventually becomes corner and will set its state variable to red.

**Proof.** If $b_1$ is yellow then $a_1$ has moved from an edge $e = \overline{v_0v_1}$. If $b_1$ was not the only agent on $e$ that could move, then there is (or there will be) another yellow agent $b_0$ moving from $e$. By construction, $b_1$ waits until it sees the other yellow agent $b_0$ (see Yellow Case). If both $b_1$ and $b_0$ realize to be corners of the current convex hull, then they eventually set their color to blue and then to red, thus the lemma is proved. However, due to the non-rigidity or the different local views of $b_1$ and $b_0$, the pathological case of Figure 2c may arise where one of the two, say $b_0$, becomes an interior-agent. This case is adjusted by the Yellow Case rule: each time $a_1$ is activated, it will move towards $v_1$ until a round $r''$ is reached when $b_0$ is not interior anymore in $C_{r''}[b_1]$. Note that, since $b_1$ moves always half of the distance $d(b_1, v_1)$, and the number of rounds until the next activation of $b_0$ is finite, we have that $b_1$ will never touch $v_1$. Two possible sub-cases may happen at round $r''$: (i) $b_1v_1 \cap b_0v_0 \notin (b_0v_0 \cup b_1v_1)$; in this case, in the subsequent activations, $b_1$ and $b_0$ will set their colors to blue; (ii) $b_1 \in b_0v_0$; this might not be detected by the local view of $b_1$, but it is detected by $b_0$ that sets its color to orange; in the next activations $b_1$ will move so to transform $b_0$ into a corner and, after this move, an activation of $b_0$ will set $s_0 = \text{blue}$. So, in both sub-cases we eventually reach a configuration in which $b_1$ and $b_0$ are blue corner-agents. In the subsequent activations, they will set their color to red, proving the lemma.

**Lemma 8.** Given a configuration $C$, let $e = \overline{v_0v_1}$ be an edge with $q > 2$ edge-agents on it. Eventually all these agents will become corners and set their color to red.
Proof. The two edge-agents \( b_0, b_1 \in e \) that are neighbors of red corners, execute the same code described in the previous lemma. So, they will reach a configuration \( C_{r'} \) in which \( b_0 \) and \( b_1 \) are blue corner-agents. In this case, they wait until all the agents on \( e \) move on the segment \( b_0b_1 \); then, they set their color to red (see the rule 3 of Blue Case). It is straightforward to see that each remaining agent on \( e \) will move now towards this new edge without colliding, since all movements to the same edge are on parallels trajectories. It follows that, in finite time, a new edge \( e' \) is formed with \( q - 2 \) agents. Iterating the reasoning we will end up in a case where the number of edge-agents on the same edge is at most 2, hence, by Lemmas 5-7 the lemma follows. \( \square \) Lemma 8

Theorem 2. The Complete Visibility problem is solvable in SSYNC by a team of oblivious, obstructable agents, using five colors without creating any collision.

Proof. From Theorem 1 we have that from any configuration \( C_0 \) we reach a configuration \( C_{ID} \) where \( I_{ID} = \emptyset \). This is locally detected by agents (see Lemma 2), that start executing the ED phase. By Lemma 4 we have that the number of red corners is not decreasing during the execution of the algorithm. From Lemmas 5-8 we have that eventually each edge-agent \( a \) of \( H_{ID} \) will became a red corner. So we will reach a configuration \( C_{final} \) in which all agents are corner of \( H_{final} \), thus, they cannot obstruct each other. Moreover, It is easy to see that each agent is able to detect not only local termination, when it sets its color to red, but also global termination of ED phase, and thus of the algorithm, when each agent in its local view is red. \( \square \) Theorem 2

4 Complete Visibility in ASYNC

In this section we consider the asynchronous model (ASYNC), where there is no common notion of time or rounds, there are no assumptions on time, on activation, on synchronization; moreover, each Compute and Move operation and inactivity may take an unpredictable (but finite) amount of time, unknown to the agent. As a consequence, agents can be seen while moving, and their computations and movements may be based on obsolete information.

Asynchronous Interior Depletion phase. The Interior Depletion algorithm of Sec. 3.1 works also in ASYNC without modifications. We only need to show that the asynchronous behaviour of the agents, and in particular the asynchronous assignment of colors, cannot induce a collision among interior-agents. Since agents always move perpendicularly to the closest edge, it is easy to see that this does not happen and thus Lemmas 1 and Theorem 1 hold also in the asynchronous case.

Asynchronous Edge Depletion phase. The Edge Depletion phase has to be modified for ASYNC. To see why the Edge Depletion algorithm would not work, consider, for example, the Yellow Case in Algorithm 3. It is possible that a moving yellow agent is seen by another yellow agent, this could lead to scenarios in which an agent assumes color red while it is on the edge of the convex hull and not on a corner.

The source of inconsistencies is the fact that agents can be seen while in transit. To prevent this problem we use new colors (yellow\(_{moving}\) and blue\(_{moving}\)) to signal
that the agents are in transit; those agents will take color yellow (resp. blue) once as the movement is completed. Using these intermediate colors, we can simulate the ED phase of the previous Section (for $|V_0| > 2$).

More precisely, in the Edge Depletion algorithm of Figure 3, instead of becoming yellow, a brown agent becomes yellow moving, turning yellow at the next activation. Similarly, instead of becoming blue, a yellow agent becomes blue moving, turning blue only when seeing that the “companion” agent is blue moving or blue.

It is not difficult to see that, with these additional colors, since agents will always move inside the safe zones of $H$, the validity of Lemmas 4, 7-8 holds also in ASYNC.

The case of $|V_0| = 2$. When the agents initially form a line, the algorithm described for SSYNC where the agents first move to a configuration $|V_0| > 2$, and then apply the general Algorithm, would not work. Consider, for example, the following scenario: both extreme agents compute and their destination is in opposite direction, but only one of them actually moves. At this point, the agents on the line set their color to red or brown, but they will become interior-agents as soon as the slower extreme agent moves from the line towards its destination, thus changing the convex hull.

The idea is to use a completely different algorithm in ASYNC when the initial configuration is a line (refer to Figure 5). Two additional special colors (line-extreme and line-moving) are used. The color line-extreme is taken by the two agents $a_1$ and $a_2$ located at the extreme points of the line, $x_1$ and $x_n$, when activated; this color is used to acknowledge the line condition, and to define the smallest enclosing circle $SEC$ with diameter $x_1, x_n$. Notice that, due to obstructed visibility, the diameter, and thus $SEC$, is unknown to the agents. The two extreme agents will never move.

The general strategy is to have the other agents move to points on $SEC$. First notice that an agent $a$ can detect that the configuration is a line, either by geometric conditions (i.e., it sees only one or two collinear agents), or by the special color of some visible agents (line-extreme or line-moving). If an uncolored agent $a$ located in $x$ sees a line-extreme agent (say $a_1$), then $a$ changes its color to line-moving and it moves perpendicularly to $xx_1$ toward the perimeter of the circle whose diameter is identified by $a_1$ and the closest agent $b \neq a_1$ on the line $xx_1$ (note that there must be at least one, possibly the other extreme). A line moving agent follows similar rules; if it can detect $SEC$ (e.g., it sees two line-extreme) it continues its perpendicular move towards it. Otherwise, it does not move. It can be shown that, at any time, there is at least one agent that, if activated, can move. A non extreme agent switches its color to red when it sees only agents on the SEC; an extreme agents switches its color to red when it sees only red or line-extreme agents.

It is not difficult to see that this set of rules will allow the agents to reach $SEC$ in finite time becoming red, and thus to solve the COMPLETE VISIBILITY problem.

**Theorem 3.** The COMPLETE VISIBILITY problem is solvable in ASYNC by a team of oblivious, obstructable agents, using eight colors without creating any collision.
5 Circle Formation in ASYNC

When executing the previous algorithm, the agents reach a configuration \( C_{\text{final}} \) in which all agents are corners of \( H_{\text{final}} \). Starting from this particular configuration it is possible to arrange the agents in such a way to reach a configuration \( C_{\text{circ}} \) in which each agent is positioned on the \( \text{SEC}(C_{\text{final}}) \). Note that the solution of the \text{COMPLETE VISIBILITY} problem when \( |V_0| = 2 \) already form a circle, hence we focus on the case \( |V_0| > 2 \).

(a) \text{CIRCLE FORMATION}: Agent \( a \) is neighbor of an agent \( b \) on \( \text{SEC} \), so it moves on line \( ca \) in direction of \( \text{SEC} \). During this movement, the corner-agents of the convex hull are not modified, and visibility with the other agents is preserved.

(b) \text{ASYNCH FORMATION for } |V_0| = 2: \text{the two extreme agents signal the line configuration with color } \text{line-extreme}, \text{the other agents move perpendicularly to them until they reach the } \text{SEC} \text{ whose diameter is defined by the extreme agents.}

Notice that, when all agents are on \( \partial H_{\text{final}} \) they can compute the same \( \text{SEC}(C_{\text{final}}) \) since all the local views are consistent. Moreover, there exists a set of agents \( \mathcal{X} \subseteq \mathcal{A} \) that are already on \( \text{SEC} \), and \( |\mathcal{X}| \geq 2 \). The idea of the algorithm is to move all agents on \( \text{SEC} \) in such a way that in each point of their trajectories they can see a subset of the \( \mathcal{Y} \) such that \( \text{SEC}(\mathcal{Y}) = \text{SEC}(\mathcal{X}) = \text{SEC}(C_{\text{final}}) \). More precisely, the moving rule allows agents to move towards \( \text{SEC} \) if they are “neighbors” (i.e., neighboring corner) of some agent on \( \text{SEC} \) in \( C_{\text{final}} \) (see Figure 5a). Let \( a \) be neighbor of some \( b \) already on \( \text{SEC} \), and let \( c \) be its other neighbor: \( a \) will move toward \( \text{SEC} \) on line \( ca \) guaranteeing that the corner-agents of the convex hull stay corner-agents, and do not loose visibility with any other agent. Note that, unless in final position, there is always at least one agent that can move. The algorithm terminates when all the agents are on \( \text{SEC} \).

It is not difficult to see that:

\textbf{Theorem 4.} \text{Starting from a configuration } \mathcal{C}_{\text{final}} \text{ in which all the agents are corners, there is an algorithm in ASYNC that makes the agents reach a configuration } \mathcal{C}_{\text{circ}} \text{ in which each agent occupies a different position on } \text{SEC}(C_{\text{final}}) \text{ without colliding.}
Acknowledgements. This work has been supported in part by the National Science and Engineering Research Council of Canada, under Discovery Grants, and by Professor Flocchini’s University Research Chair.

References

1. C. Agathangelou, C. Georgiou, and M. Mavronicolas. A distributed algorithm for gathering many fat mobile robots in the plane. In Proceedings of the 32nd ACM Symposium on Principles of Distributed Computing (PODC), 250–259, 2013.
2. K. Bolla, T. Kovacs, and G. Fazekas. Gathering of fat robots with limited visibility and without global navigation. In Int. Symp. on Swarm and Evolutionary Comp., 30–38, 2012.
3. R. Cohen and D. Peleg. Local spreading algorithms for autonomous robot systems. Theoretical Computer Science, 399:71–82, 2008.
4. J. Czyzowicz, L. Gasieniec, and A. Pelc. Gathering few fat mobile robots in the plane. Theoretical Computer Science, 410(67):481 – 499, 2009.
5. S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. The power of lights: Synchronizing asynchronous robots using visible bits. In Proceedings of the 32nd International Conference on Distributed Computing Systems (ICDCS), 506–515, 2012.
6. S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. Synchronized dancing of oblivious chameleons. In Proc. 7th Int. Conf. on FUN with Algorithms (FUN), 2014.
7. S. Datta, A. Dutta, S. Gan Chaudhuri, and K. Mukhopadhyaya. Circle formation by asynchronous fat robots. In Proceedings of the 9th international conference on Distributed Computing and Internet Technology (ICDCIT), 195-207, 2013.
8. X. Défago and S. Souissi. Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity. Theor. Comp. Sci., 396(1,3):97–112, 2008.
9. O. Petit F. Dieudonné, Labbani-Igbida. Circle formation of weak mobile robots. ACM Transactions on Autonomous and Adaptive Systems, 3(4):1–16, 2008.
10. A. Efrima and D. Peleg. Distributed models and algorithms for mobile robot systems. In Proceedings of the 33rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM), 70–87, 2007.
11. P. Flocchini, G. Prencipe, and N. Santoro. Distributed Computing by Oblivious Mobile Robots. Morgan & Claypool, 2012.
12. P. Flocchini, N. Santoro, G. Viglietta, and M. Yamashita. Rendezvous of two robots with constant memory. In Proceedings of the 20th International Colloquium on Structural Information and Communication Complexity (SIROCCO), 189–200, 2013.
13. B. Katreniak. Biangular circle formation by asynchronous mobile robots. In Proc. 12th Int. Coll. on Structural Information and Communication Complexity (SIROCCO), 185–199, 2005.
14. D. Peleg. Distributed coordination algorithms for mobile robot swarms: New directions and challenges. In Proc. 7th Int. Workshop on Distr. Comp. (IWDC), 1–12, 2005.
15. K. Sugihara and I. Suzuki. Distributed motion coordination of multiple mobile robots. In Proceedings of 5th IEEE Int. Symposium on Intelligent Control, 138–143, 1990.
16. G. Viglietta. Rendezvous of two robots with visible bits. In Proc. 9th Symp. on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS), 291–306, 2013.