On the Computational Power of Energy-Constrained Mobile Robots: Algorithms and Cross-Model Analysis*

Kevin Buchin[1][0000−0002−3022−7877], Paola Flocchini[2][0000−0003−3584−5727], Irina Kostitsyna[3][0000−0003−0544−2257], Tom Peters[3][], Nicola Santoro[4][0000−0002−7954−3918], and Koichi Wada[5][0000−0002−5351−1459]

[1] TU Dortmund, Germany, kevin.buchin@tu-dortmund.de
[2] University of Ottawa, Canada, pflocchi@uottawa.ca
[3] TU Eindhoven, The Netherlands
i.kostitsyna@tue.nl, t.peters1@tue.nl
[4] Carleton University, Canada, santoro@scs.carleton.ca
[5] Hosei University, Tokyo, Japan, wada@hosei.ac.jp

Abstract. We consider distributed systems of identical autonomous computational entities, called *robots*, moving and operating in the plane in synchronous *Look-Compute-Move* (*LCM*) cycles. The algorithmic capabilities of these systems have been extensively investigated in the literature under four distinct models ($\mathcal{OBLOT}$, $\mathcal{FSTA}$, $\mathcal{FCOM}$, $\mathcal{LUMI}$), each identifying different levels of memory persistence and communication capabilities of the robots. Despite their differences, they all always assume that robots have unlimited amounts of energy.

In this paper, we remove this assumption and start the study of the computational capabilities of robots whose energy is limited, albeit renewable. We first study the impact that memory persistence and communication capabilities have on the computational power of such energy-constrained systems of robots; we do so by analyzing the computational relationship between the four models under this energy constraint. We provide a complete characterization of this relationship.

We then study the difference in computational power caused by the energy restriction and provide a complete characterization of the relationship between energy-constrained and unrestricted robots in each model. We prove that within $\mathcal{LUMI}$ there is no difference; an integral part of the proof is the design and analysis of an algorithm that in $\mathcal{LUMI}$ allows energy-constrained robots to execute correctly any protocol for robots with unlimited energy. We then show the (apparently counterintuitive) result that in all other models, the energy constraint actually provides the robots with a computational advantage.

Keywords: oblivious robots · luminous robots · energy-constrained robots · comparison of models.

* This work was supported in part by JSPS KAKENHI No. 20K11685 and 21K11748, Israel & Japan Science and Technology Agency (JST) SICORP (Grant#JPMJSC1806), and by the Natural Sciences and Engineering Research Council of Canada (NSERC) under Discovery Grants A2415 and 203254.

1 Introduction

In this paper, we consider distributed systems composed of identical autonomous computational entities, viewed as points and called *robots*, moving and operating in the Euclidean plane in synchronous *Look-Compute-Move* (*LCM*) cycles. In each synchronous round, a non-empty set of (possibly all) robots is activated, each performs its *LCM* cycle simultaneously and terminates by the end of the round. Each cycle is composed of three phases: in the *Look* phase, an entity obtains a snapshot of the robots' configuration showing the positions of all the other robots; in the *Compute* phase, the robot executes its algorithm (the same for all robots) and computes a destination point using the snapshot as input; in the *Move* phase, the robot moves towards the computed destination. Repeating these cycles, the robots can collectively perform some tasks and solve some pattern formation problems.

The activation of robots is controlled by an adversarial scheduler, who selects which robots are activated in each round. This general setting is usually called *semi-synchronous* (SSYNCH); the special restricted setting where every robot is activated in every round is called *fully-synchronous* (FSYNCH) .

These distributed robot systems have been extensively investigated within distributed computing. The research aim has been to understand the nature and the extent of the impact that factors, such as *memory persistence* and *communication capability*, have on the solvability of problems and thus on the computational power of the system. To this end, four models have been identified and investigated: $\mathcal{OBLOT}$, $\mathcal{FSTA}$, $\mathcal{FCOM}$, and $\mathcal{LUMI}$.

In the most common (and weakest) model, $\mathcal{OBLOT}$, in addition to the standard assumptions of anonymity and uniformity (robots have no IDs and run identical algorithms), the robots are *oblivious* (they have no persistent memory to record information of previous cycles) and they are *silent* (without explicit means of communication). The restrictions imposed by the absence of persistent memory and the incapacity of explicit communication severely limit what the robots can do. Computability in this model has been the object of intensive research since its introduction in [25] (e.g., see [1,4,5,6,12,19,25,13,16,25,28,29,3,17], as well as the recent book [10]).

In the stronger $\mathcal{LUMI}$ model, formally introduced and defined in [7], robots are provided with some (albeit limited) persistent memory and means for communication. In this model each robot is equipped with a constant-sized memory (called *light*), whose value (*color*) can be set during the *Compute* phase. The light is visible to all the robots and is persistent between the robot activations. Hence, these luminous robots are capable of both remembering and communicating a constant number of bits. Design of algorithms and feasibility of solving problems for luminous robots have been extensively studied [7,14,18,8,20,21,26,22,23,27]; for a recent survey, see [9]. The availability of both persistent memory and communication, however limited, clearly renders luminous robots more powerful than oblivious robots (see e.g., [7]).

To better understand the computational power of persistent memory and communication individually, models $\mathcal{FSTA}$ and $\mathcal{FCOM}$ (which fall in between

$\mathcal{OBLOT}$ and $\mathcal{LUMI}$) were introduced in [14] (and studied in [20,26]). In the first model, $\mathcal{FSTA}$, the light of a robot is "internal", i.e., visible only to that robot, while in the second model, $\mathcal{FCOM}$, the light of a robot is visible only to the other robots but not to the robot itself. Thus in $\mathcal{FSTA}$, the color merely encodes an internal state, and the robots are *finite-state* and *silent*. On the contrary, in $\mathcal{FCOM}$, a robot can communicate to the other robots by means of the light but forgets the content of its transmission by the next cycle; that is, robots are *oblivious* and *finite-communication*.

To understand the computational power of these distributed systems, one needs to explore and determine the computational power of the robots within each of these models, as well as (and more importantly) with respect to each other. This type of cross-model investigation has been taking place but is rather limited in scope (e.g., [14]). Recently, a substantial step has been taken in [15] where, by integrating existing bounds and establishing new results, a comprehensive map of the computational relationship between the four models, $\mathcal{OBLOT}$, $\mathcal{FSTA}$, $\mathcal{FCOM}$, and $\mathcal{LUMI}$, has been drawn (and hence the computational impact of the presence/absence of persistent memory and/or communication capabilities has been established) for the two fundamental synchronous settings: fully-synchronous and semi-synchronous.

The energy problem. In the vast existing literature, surprisingly, no consideration has been made so far on the energy required for the robots to be able to operate. Existing works share the same implicit assumption that the robots have an unlimited amount of energy enabling them to perform their activities in every activation round. In this paper, we remove this assumption and initiate the study of the robots whose energy is limited, albeit renewable. More precisely, we consider systems where an activated entity uses all its energy to execute an *LCM* cycle, and once this happens, the robot is not operational and cannot be activated in the next round; the energy, however, can be restored through a period of inactivity. This would be the case if, for example, the robot's power is provided by a battery rechargeable by energy harvesting (as it is done in conceptually related systems such as wireless mobile sensors [24]).

The immediate natural questions are: *what is the computational power of these energy-constrained robots*? and, in particular, *what is the impact of the crucial factors (memory and communication) in this case*? In this paper, we start investigating these questions.

Contributions. We consider systems where the energy of a robot is sufficient to execute exactly one *LCM* cycle, and the depleted energy is restored after one round of inactivity. We investigate the computational power of the distributed robot systems described by the four models when the robots are subject to such energy constraint.

We establish an equivalence between a system of energy-constrained robots under the semi-synchronous (SSYNCH) scheduler and a system of classic robots with unlimited energy under a new scheduler, which we call RSYNCH. By our definition of RSYNCH, the sets of robots activated in any two consecutive rounds

are required to be disjoint. This direct correspondence enables us to reduce the cross-model investigation of the energy-constrained robots to the cross-model investigation of energy-unbounded robots under the RSYNCH scheduler. Furthermore, it allows us to determine the change (if any) in computational power due to the energy restriction, by determining the relationship between RSYNCH and the general unrestricted SSYNCH scheduler.

Let M^S and M^{RS} denote the systems of unlimited-energy robots defined by model $M \in \{\mathcal{LUMI}, \mathcal{FCOM}, \mathcal{FSTA}, \mathcal{OBLOT}\}$ under SSYNCH and RSYNCH, respectively (the latter being equivalent to the systems of energy-constrained robots under SSYNCH). We first study the impact that memory persistence and communication capabilities have on the computational power of energy-constrained systems of robots; we do so by analyzing the computational relationship between the four models under RSYNCH scheduler. We provide a complete characterization:

$$\mathcal{LUMI}^{RS} \equiv \mathcal{FCOM}^{RS} > \mathcal{FSTA}^{RS} > \mathcal{OBLOT}^{RS},$$

where (as formally defined in Section 2) $\mathcal{X} > \mathcal{Y}$ denotes that model $\mathcal{X}$ is strictly more powerful than $\mathcal{Y}$, and $\mathcal{X} \equiv \mathcal{Y}$ denotes that $\mathcal{X}$ and $\mathcal{Y}$ are computationally equivalent. An integral part of the proof that $\mathcal{FCOM}^{RS}$ is more powerful than $\mathcal{FSTA}^{RS}$ (that is, it is better to communicate than to remember), is the design and analysis of an algorithm that allows robots in $\mathcal{FCOM}^{RS}$ to correctly execute any protocol for the more powerful $\mathcal{LUMI}^{RS}$.

We then study what impact on computational power is created by restricting the energy of the robots, by comparing the computational difference between energy-constrained and unrestricted robots in each of the four models (i.e., between M^{RS} and M^S for each $M \in \{\mathcal{LUMI}, \mathcal{FCOM}, \mathcal{FSTA}, \mathcal{OBLOT}\}$). We provide a complete characterization. In particular, we prove that for $\mathcal{LUMI}$ robots, the strongest model, there is no difference between energy-constrained and unlimited-energy robots; i.e., $\mathcal{LUMI}^{RS} \equiv \mathcal{LUMI}^{S}$. An integral part of the proof is the design and analysis of an algorithm that allows energy-constrained robots in $\mathcal{LUMI}^{RS}$ to correctly execute any protocol for robots with unlimited energy in $\mathcal{LUMI}^{S}$. In all other models, we prove that restricting energy actually provides the robots with a definite computational advantage; this apparently counterintuitive result is due to the fact that the energy restriction reduces the adversarial power of the activation scheduler. Let us stress that the established characterization covers all the cross-model and cross-scheduler relationships.

Finally, we complete the study of systems of energy-constrained robots by analyzing the relationship between their computational power and that of robots with unlimited energy under the most benign synchronous activation scheduler FSYNCH (i.e., fully synchronous). In this case, perhaps not surprising, we prove that, in each robot model, energy-constrained robots are strictly less powerful than fully-synchronous ones with unbounded energy. We again cover all the cross-model and cross-scheduler relationships.

The details of omitted parts and proofs are shown in a full paper [2].

2 Models and preliminaries

The basics. The system consists of a set $R = \{r_0, \ldots, r_{n-1}\}$ of computational entities, called robots, modeled as geometric points, that live in $\mathbb{R}^2$, where they can move freely and continuously. The robots are autonomous without a central control. They are indistinguishable by their appearance, do not have internal identifiers, and execute the same algorithm. Each robot has its own local coordinate system, which may be inconsistent with the coordinate systems of the other robots. A robot perceives itself at the origin of its coordinate system, and is capable of observing the positions of the other robots in it.

The robots operate in *Look-Compute-Move* (LCM) cycles. When activated, a robot executes a cycle by performing the following three operations:

Look The robot obtains a snapshot of the positions occupied by robots expressed with respect to its own coordinate system, and their colors. This operation is assumed to be instantaneous.

Compute The robot executes the algorithm using the snapshot as input; the result of the computation is a destination point.

Move The robot moves towards the computed destination. If the destination is the current location, the robot stays still.

The system is *synchronous*. That is, time is divided into discrete intervals, called *rounds*. In each round a robot is either active or inactive. The robots active in a round perform their LCM cycle in perfect synchronization; if not active, the robot is idle in that round. All robots are initially idle. In the following, we use round and time interchangeably.

Each robot has a bounded amount of *energy*, which is totally consumed whenever it performs a cycle; its energy however is restored after being idle for a round. A robot with depleted energy cannot be active.

Movements are said to be *rigid* if the robots always reach their destination. They are said to be *non-rigid* if they may be unpredictably stopped by an adversary whose only limitation is the existence of $\delta > 0$, unknown to the robots, such that if the destination is at distance at most δ the robot will reach it, else it will move at least δ towards the destination.

There might not be consistency between the local coordinate systems and their unit of distance. The absence of any a priori assumption on consistency of the local coordinate systems is called *disorientation*. The type of disorientation can range from *fixed*, where each local coordinate system remains the same through all the rounds, to *variable* where the direction, the orientation, and the unit distance of a robot may vary between successive rounds. In this paper we consider only fixed disorientation.

Let $x_i(t)$ denote the location of robot r_i at time t in a global coordinate system (unknown to the robots), and let $X(t) = \{x_i(t) : 0 \leq i \leq n-1\} = \{x_0(t), x_1(t), \ldots, x_{m-1}(t)\}$; note that $|X(t)| = m \leq n$ since several robots might be at the same location at time t. A *configuration* $C(t)$ at time t is the multi-set of the n pairs of the $(x_i(t), c_i(t))$, where $c_i(t)$ is the color (formally defined below) of robot r_i at time t.

The robots are said to have *chirality* if they share the same circular orientation of the plane (i.e., they agree on "clockwise" direction). Notice that, in presence of chirality, at any time t, there would exist a unique circular ordering of the locations $X(t)$ occupied by the robots at that time; let `suc` and `pred` be the functions denoting the ordering and, without loss of generality, let $\texttt{suc}(x_i(t)) = x_{i+1 \bmod m}(t)$ and $\texttt{pred}(x_i(t)) = x_{i-1 \bmod m}(t)$ for $i \in \{0, 1, \ldots, m-1\}$.

The computational models. In $\mathcal{OBLOT}$ model the robots are *silent*: they have no explicit means of communication; furthermore they are *oblivious*: at the start of a cycle, a robot has no memory of observations and computations performed in previous cycles.

In $\mathcal{LUMI}$ model each robot r is equipped with a persistent visible state variable $Light[r]$, called *light*, whose values are taken from a finite set C of states called *colors* (including the color that represents the initial state when the light is off). The color of the light is set by r at the end of its *Compute* operation. The lights are *persistent* from one computational cycle to the next: the color is not automatically reset at the end of a cycle; the robot is otherwise oblivious, forgetting all other information from previous cycles. In $\mathcal{LUMI}$ the *Look* operation produces a colored snapshot; i.e., it returns the set of pairs $(position, color)$ of the other robots. Note that if $|C| = 1$, then the light is not used, and the model is equivalent to $\mathcal{OBLOT}$.

As mentioned above, the lights provide simultaneously persistent memory and direct means of communication, although both limited to a constant number of bits per cycle. The two sub-models $\mathcal{FSTA}$ and $\mathcal{FCOM}$ of $\mathcal{LUMI}$ each offers only one of these two capabilities. In $\mathcal{FSTA}$ model a robot can only see the color of its own light; that is, the light is *internal* and its color merely encodes an internal state. Hence the robots are *silent* as in $\mathcal{OBLOT}$; but are *finite-state* as in $\mathcal{LUMI}$. Observe that a snapshot in $\mathcal{FSTA}$ is the same as in $\mathcal{OBLOT}$.

In $\mathcal{FCOM}$ the lights are *external* and visible only to the other robots: a robot can communicate to the others by setting its color, but forgets it by the next cycle; that is, robots are *finite-communication* but *oblivious*. A snapshot that robot r perceives in $\mathcal{FCOM}$, as in $\mathcal{LUMI}$, contains the information about robots' colors, except that the color $Light[r]$ is omitted from the set of colors associated with the position of r.

Activation schedulers and energy restriction. In each synchronous round, some robots become active and they execute their *LCM* cycle in complete synchrony. The choice of which non-empty subset of the robots is activated in a specific round is under the control of an adversarial *activation scheduler* constrained to be fair; that is, every robot will become active infinitely often. Given a synchronous scheduler $\mathcal{S}$ and a set of robots R, an *activation sequence* of R under $\mathcal{S}$ is an infinite sequence $E = \langle e_1, e_2, \ldots, e_i, \ldots \rangle$, where $e_i \subseteq R$ denotes the set of robots activated in round i, satisfying the *fairness constraint*:

$$\forall r \in R, i \geq 1 \, \exists j > i : r \in e_j \, .$$

Let $\mathcal{E}(\mathcal{S}, R)$ denote the set of all activation sequences of R by $\mathcal{S}$. In the standard synchronous scheduler (SSYNCH), first studied in [25] and often called *semi-synchronous*, each sequence $E = \langle e_1, e_2, \ldots, e_i, \ldots \rangle \in \mathcal{E}(\text{SSYNCH}, R)$ satisfies the *basic condition*

$$\forall i \geq 1, \varnothing \neq e_i \subseteq R\,.$$

The special *fully-synchronous* (FSYNCH) setting, where every robot is activated in every round, corresponds to further restricting the activation sequences by imposing $\forall i \geq 1, e_i = R$. Notice that, in this setting, the activation scheduler has no adversarial power. Another special setting is defined by the well-known *round robin* scheduler (e.g., see [26]), whose generalized definition corresponds to adding the restriction:

$$[\exists p > 1 : (\bigcup_{1 \leq i \leq p} e_i = R) \ \textbf{and}\ (\forall 1 \leq i \neq j \leq p, [e_i \cap e_j = \varnothing])\ \textbf{and}\ (\forall i \geq 1, [e_i = e_{i+p}])]\,.$$

We study systems of *energy-constrained robots* under the standard synchronous activation scheduler. More precisely, we study systems where a robot (i) has just enough energy to execute a cycle, (ii) it cannot be activated in a round unless it has full energy, and (iii) its depleted energy is regenerated after one round. These three conditions clearly have an impact on the possible activation sequences of the robots. In particular, since a robot with depleted energy cannot be activated, the basic condition on e_i becomes

$$\forall i \geq 1, e_i \subseteq R^*[i] \ \textbf{and}\ R^*[i] \neq \varnothing \Rightarrow e_i \neq \varnothing\,, \tag{1}$$

where $R^*[i] \subseteq R$ denotes the set of robots with full energy in round i. Furthermore, since it takes a round to regenerate depleted energy, e_i must also satisfy

$$\forall i \geq 1, (e_i \cap e_{i+1} = \varnothing)\,.$$

Notice that, since $e_i \subseteq R^*[i]$, it is possible that $e_i = R$ when $R^*[i] = R$. Should this be the case, since a robot has just enough energy to execute a cycle, then $R^*[i+1] = \varnothing$ and thus $e_{i+1} = \varnothing$. Furthermore, due Equation (1),

$$\exists i, (\varnothing \neq e_i \neq R) \Rightarrow \forall j \geq i, (\varnothing \neq e_j \neq R)\,.$$

That is, if fewer than $|R|$ but a positive number of robots are activated in any round i, then the set of robots with full energy $R^*[j] \neq R$, and thus $\varnothing \neq e_j \neq R$, for all $j \geq i$. Thus, the activation sequences of the energy-constrained robots are infinite sequences where the prefix is a (possibly empty) alternating sequence of R and $\varnothing$, and, if the prefix is finite, the rest are non-empty sets satisfying the constraint $(e_i \cap e_{i+1} = \varnothing)$. Notice that this set of sequences, denoted by $\mathcal{E}(\text{SSYNCH}_{res}, R)$, is not a proper subset of $\mathcal{E}(\text{SSYNCH}, R)$ since some sequences might have empty sets in their prefix.

Consider now the synchronous scheduler, we shall call RSYNCH, obtained from SSYNCH by adding the following *restricted-repetition condition* to its activation

sequences:

$$\Big[\forall i \geq 1, e_i = R\Big] \textbf{ or}$$
$$\Big[\exists p \geq 0 : \Big(\big[\forall i \leq p, (e_i = R)\big] \textbf{ and } \big[\forall i > p, (\varnothing \neq e_i \neq R \textbf{ and } e_i \cap e_{i+1} = \varnothing)\big]\Big)\Big],$$

that is, $\mathcal{E}(\text{RSYNCH}, R)$ is composed of sequences where the prefix is a (possibly empty) sequence of R and, if the prefix is finite, the rest are non-empty sets satisfying the constraint $(e_i \cap e_{i+1} = \varnothing)$.

There is an obvious bijection ϕ between $\mathcal{E}(\text{SSYNCH}_{res}, R)$ and $\mathcal{E}(\text{RSYNCH}, R)$, where $\phi(E)$ corresponds to removing all empty sets from $E \in \mathcal{E}(\text{SSYNCH}_{res}, R)$. Thus computation performed by R under E is equivalent to one performed under $\phi(E)$. Informally, under SSYNCH_{res}, if all robots are activated in the same round i, they will be all idle in round $i+1$, and they will all be with full energy in round $i+2$. Since no activity takes place in round $i+1$, we can ignore all such empty rounds and assume that round $i+2$ occurs right after round i. Thus, the computation by a set of energy-constrained robots R under the standard synchronous scheduler SSYNCH is equivalent to the one if the robots in R were energy-unbounded but the activation was controlled by scheduler RSYNCH. This restricted-repetition setting has never been studied before; observe that it includes both fully synchronous FSYNCH and round robin as special cases.

Computational relationships. Let $\mathcal{M} = \{\mathcal{LUMI}, \mathcal{FCOM}, \mathcal{FSTA}, \mathcal{OBLOT}\}$ be the set of models under investigation, and $\mathcal{S} = \{\text{FSYNCH}, \text{RSYNCH}, \text{SSYNCH}\}$ be the set of activation schedulers under consideration. We denote by $\mathcal{R}$ the set of all teams of robots satisfying the core assumptions (i.e., they are identical, autonomous, and operate in *LCM* cycles), and $R \in \mathcal{R}$ a team of robots having identical capabilities (e.g., common coordinate system, persistent storage, internal identity, rigid movements etc.). By $\mathcal{R}_n \subset \mathcal{R}$ we denote the set of all teams of size n.

Given a model $M \in \mathcal{M}$, a scheduler $S \in \mathcal{S}$, and a team of robots $R \in \mathcal{R}$, let $\mathit{Task}(M, S; R)$ denote the set of problems solvable by R in M under adversarial scheduler S. Let $M_1, M_2 \in \mathcal{M}$ and $S_1, S_2 \in \mathcal{S}$.

- We say that model M_1 under scheduler S_1 is *computationally not less powerful than* model M_2 under S_2, denoted by $M_1^{S_1} \geq M_2^{S_2}$ if $\forall R \in \mathcal{R}$ we have $\mathit{Task}(M_1, S_1; R) \supseteq \mathit{Task}(M_2, S_2; R)$.
- We say that M_1 under S_1 is *computationally more powerful than* M_2 under S_2, denoted by $M_1^{S_1} > M_2^{S_2}$, if $M_1^{S_1} \geq M_2^{S_2}$ and $\exists R \in \mathcal{R}$ such that $\mathit{Task}(M_1, S_1; R) \setminus \mathit{Task}(M_2, S_2; R) \neq \varnothing$.
- We say that M_1 under S_1 and M_2 under S_2 are *computationally equivalent*, denoted by $M_1^{S_1} \equiv M_2^{S_2}$, if $M_1^{S_1} \geq M_2^{S_2}$ and $M_2^{S_2} \geq M_1^{S_1}$.
- Finally, we say that M_1 under S_1 and M_2 under S_2 are *computationally orthogonal* (or *incomparable*), denoted by $M_1^{S_1} \perp M_2^{S_2}$, if $\exists R_1, R_2 \in \mathcal{R}$ such that $\mathit{Task}(M_1, S_1; R_1) \setminus \mathit{Task}(M_2, S_2; R_1) \neq \varnothing$ and $\mathit{Task}(M_2, S_2; R_2) \setminus \mathit{Task}(M_1, S_1; R_2) \neq \varnothing$.

For brevity, for a model $M \in \mathcal{M}$, let M^F, M^{RS}, and M^S, denote M^{FSYNCH}, M^{RSYNCH}, and M^{SSYNCH}, respectively. Furthermore, with a slight abuse of notation, let $M^F(R)$, $M^{RS}(R)$, and $M^S(R)$, denote $\mathit{Task}(M, \text{FSYNCH}; R)$, $\mathit{Task}(M, \text{RSYNCH}; R)$, and $\mathit{Task}(M, \text{SSYNCH}; R)$, respectively. Trivially, for all $M \in \mathcal{M}$,

$$M^F \geq M^{RS} \geq M^S,$$

and, for all $P \in \mathcal{S}$,

$$\mathcal{LUMI}^P \geq \mathcal{FSTA}^P \geq \mathcal{OBLOT}^P \textbf{ and } \mathcal{LUMI}^P \geq \mathcal{FCOM}^P \geq \mathcal{OBLOT}^P .$$

3 Computational relationship between RSYNCH and SSYNCH

We begin by studying the impact that constraining the energy has on the computational capability of the robots. We do so by analyzing the computational relationship between RSYNCH and SSYNCH in each of the four models.

3.1 Power of RSYNCH in $\mathcal{FCOM}$, $\mathcal{FSTA}$ and $\mathcal{OBLOT}$

First, we show that RENDEZVOUS problem (RDV) [11], where two robots a and b must gather in the same location not known in advance, cannot be solved in RSYNCH. Recall that $\mathcal{R}_2 \subset \mathcal{R}$ is the set of all teams of robots of size 2.

Lemma 1. $\exists R \in \mathcal{R}_2$, RDV $\notin \mathcal{OBLOT}^{RS}(R)$. *This result holds even in presence of chirality and rigidity of movement.*

The problem SHRINKING ROTATION (SRO) was introduced to show that $\mathcal{OBLOT}^F > \mathcal{OBLOT}^S$, and that models $\mathcal{OBLOT}^F$ and $\mathcal{FCOM}^S$ (or $\mathcal{FSTA}^S$) are incomparable [15].

Lemma 2 ([15]). $\exists R \in \mathcal{R}_2$, SRO $\notin \mathcal{FCOM}^S(R) \cup \mathcal{FSTA}^S(R)$. *This result holds even in presence of chirality and rigidity of movement.*

This problem can also play a role in showing $\mathcal{OBLOT}^{RS} > \mathcal{OBLOT}^S$ and orthogonality of $\mathcal{OBLOT}^{RS}$ and $\mathcal{FCOM}^S$ (or $\mathcal{FSTA}^S$).

Lemma 3. $\forall R \in \mathcal{R}_2$, SRO $\in \mathcal{OBLOT}^{RS}(R)$, *assuming common chirality and rigid movement.*

Lemma 1 and the fact that RDV can be solved by $\mathcal{FCOM}$ and $\mathcal{FSTA}$ in SSYNCH [14], and Lemmas 2 and 3 imply:

Theorem 1. *The following relations hold:* $\mathcal{OBLOT}^{RS} \perp \mathcal{FCOM}^S$, $\mathcal{OBLOT}^{RS} \perp \mathcal{FSTA}^S$, $\mathcal{FCOM}^{RS} > \mathcal{FCOM}^S$, $\mathcal{FSTA}^{RS} > \mathcal{FSTA}^S$, *and* $\mathcal{OBLOT}^{RS} > \mathcal{OBLOT}^S$.

We now show the dominance of $\mathcal{LUMI}^S$ over $\mathcal{FSTA}^{RS}$ and the orthogonality of $\mathcal{FSTA}^{RS}$ with $\mathcal{FCOM}^S$. In order to show these results, we use the following problem.

Definition 1 (Cyclic Circles (CYC)). *Let $n \geq 3$, $k = 2^{n-1}$, and $d : \mathbb{N} \to \mathbb{R}$ is a non-invertible function. The problem is to form a cyclic sequence of patterns $C, C_0, C, C_1, C, C_2, \ldots, C, C_{k-1}$, where C is a pattern of $n-1$ robots occupying vertices of a regular n-gon with the nth robot in its center (see Figure 1 (a)), and C_i (for $0 \leq i \leq k-1$) is a configuration with the $n-1$ n-gon robots in the same position, but the center robot occupying a point at distance $d(i)$ from the center in the direction of the empty vertex (see Figure 1 (b)). In other words, the central robot moves to the designated position at distance $d(i)$ and comes back to the center. The process repeats after all 2^{n-1} configurations C_i have been formed.*

Lemma 4. *Let $n \geq 3$. $\exists R \in \mathcal{R}_n, \mathrm{CYC} \notin \mathcal{FSTA}^F(R)$. This result holds even in presence of chirality and rigid movements.*

Next, we show that $\mathcal{FCOM}$ robots can solve CYC under Ssynch. Intuitively, the $n-1$ robots on the circle act as a distributed counter using their lights to display the binary representation of the index i of the next configuration to be formed (see Figure 1 (c)). The increment of the counter is performed by changing the bits accordingly and maintaining the carry, as in a full adder. Whenever activated, the central robot "reads" the information and understands when it is time to move and to which destination.

Lemma 5. *Let $n \geq 3$. $\forall R \in \mathcal{R}_n$, $\mathrm{CYC} \in \mathcal{FCOM}^S(R)$, assuming chirality.*

The orthogonality of $\mathcal{FSTA}^{RS}$ and $\mathcal{FCOM}^S$ follows from Lemmas 3–5, and the dominance of $\mathcal{LUMI}^S$ over $\mathcal{FSTA}^{RS}$ follows from Lemmas 4 and 5.

Theorem 2. *$\mathcal{FSTA}^{RS} < \mathcal{LUMI}^S$ and $\mathcal{FSTA}^{RS} \perp \mathcal{FCOM}^S$.*

3.2 Power of Rsynch in $\mathcal{LUMI}$

We now show that, in spite being more powerful in $\mathcal{FCOM}$ and $\mathcal{FSTA}$, Rsynch robots with full lights ($\mathcal{LUMI}$) are *not* more powerful than Ssynch robots with full lights. That is, $\mathcal{LUMI}^S$ is computationally equivalent to $\mathcal{LUMI}^{RS}$. To do so we prove the following theorem.

Theorem 3. *$\forall R \in \mathcal{R}$, $\mathcal{LUMI}^{RS}(R) \leq \mathcal{LUMI}^S(R)$.*

The approach is to show that $\mathcal{LUMI}$ robots under Ssynch can simulate any algorithm designed for $\mathcal{LUMI}$ robots under Rsynch (see algorithm `sim-RS-by-S(`A`)` in [2]). Let algorithm A for $\mathcal{LUMI}$ robots in Rsynch use light with ℓ colors: $C = \{c_0, c_1, \ldots, c_{\ell-1}\}$. Ssynch robots run algorithm `sim-RS-by-S(`A`)` simulating the execution of A as follows. Each robot r has the following sets of colors:

- $r.color \in C$ indicating its own light used in algorithm A, initially set to $r.color = c_0$;
- $r.step \in \{1, 2, 3, 4, 5, m\}$ indicating the step of the simulation currently under execution. Initially $r.step$ is set to 1 for all robots, and thus initially the simulation is in Step 1;

$r.executed \in \{True, False\}$ indicating whether r has executed algorithm A in the current mega-cycle (see below). Initially $r.executed$ is set to *False* for all robots;

$r.charged \in \{C, E, M\}$, where C, E, and M stand for "charged", "empty", and "just moved" respectively. The flag is used to ensure the validity of the simulated RSYNCH activation sequence, it indicates whether r is charged and can execute the algorithm A. Initially $r.charged$ is set to C for all robots.

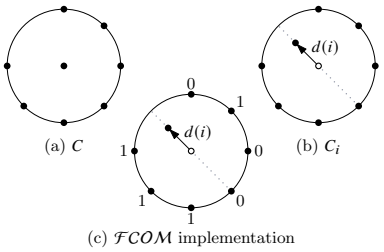

Fig. 1. The configurations of CYC.

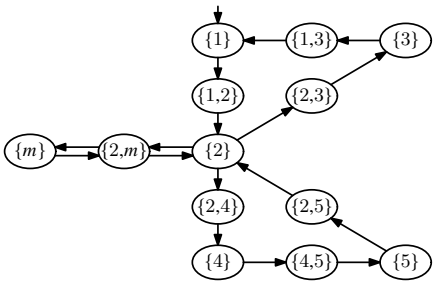

Fig. 2. Transition diagram `sim-RS-by-S(`A`)`.

The simulation proceeds in two phases. The first phase of the simulation corresponds to the first $p \geq 0$ (where p can be ∞) activations in a simulated RSYNCH activation schedule where all robots in R are activated at each round. The second phase corresponds to the remaining activation cycles where a strict subset of R is activated at each round.

The state of the robot system is defined by the set of colors *step* currently set by the robots (see Figure 2). By construction, there can be at most two different *step* colors in the system at each moment. If all robots have the same *step* color i (for $i \in \{1, 2, 3, 4, 5, m\}$), we say the robots execute Step i of the simulation. If not all robots have the same *step*, this corresponds to the system transitioning from one step to another.

The two phases of the simulation consist of executing so called *mega-cycles*. The mega-cycle of the first phase corresponds to one cycle $\{1\} \to \{2\} \to \{3\} \to \{1\}$. The mega-cycle of the second phase consists of several cycles $\{2\} \to \{4\} \to \{5\} \to \{2\}$ (until all robots have executed it) with one cycle $\{2\} \to \{m\} \to \{2\}$ to reset the flags and start the new mega-cycle. The robots execute A in Step 1 in the first phase, and in Step 2 in the second phase. The remaining steps serve for bookkeeping of flags *executed* and *charged*.

Specifically, the states of the simulation are (refer to Figure 2):

- State $\{i, j\}$ are the transition states between Step i and Step j. Activated robots do not execute A nor change the values of the *charged* and *executed* flags, but only set their $r.step$ to either i or to j (depending on the specific case).

- Step 1: activated robot executes one cycle of the simulated algorithm A. Note that when Step 1 begins execution, it holds that $\forall \rho \in R\,(\rho.\mathit{executed} = \mathit{False}$ and $\rho.\mathit{charged} = C)$.
- Step 2: activated robot checks if all robots have their *charged* flag set to E, that is, if all robots were activated in Step 1. If so, we are in the first phase, and the simulation proceeds to Step 3 resetting all the *charged* and *executed* flags to prepare for the next move in Step 1. Otherwise, we are in the second phase, and there are two cases:
  (1) If all robots have their *executed* flag set to *True*, the mega-cycle has finished. The robot proceeds to Step m.
  (2) If some robots have their *executed* flag set to *False*, the mega-cycle has not finished yet. The simulation proceeds executing A. As soon as among the activated robots there is a non-empty subset R' with $\mathit{charged} = C$ and $\mathit{executed} = \mathit{False}$, they execute algorithm A, set *charged* to M and *executed* to *True*, and proceed to Step 4. Afterwards, the remaining robots proceed to Step 4 without executing A or changing their *charged* and *executed* flags.
- Step 3: robot resets the *charged* and *executed* flags to prepare the next move in Step 1. Note that as long as Step 3 is executed, full activation continues.
- Step 4: robot updates the *charged* flag if it did not execute A in preceding Step 2 from E to C (the robots that did not execute are recharged).
- Step 5: robot updates the *charged* flag if it executed A in preceding Step 2 from M to E (the robots that executed are discharged).
- Step m executes when one mega-cycle of phase two is completed. Robots reset their *executed* flags and transition back to Step 2. Note that Step m does not affect the *charged* flag, thus the robots which executed A in the last activation cycle of the preceding mega-cycle are discharged in the first activation cycle of the new mega-cycle and cannot be activated.

The initial configuration of `sim-RS-by-S(`A`)` satisfies $r.\mathit{step} = 1$, $r.\mathit{charged} = C$ and $r.\mathit{executed} = \mathit{False}$ for any robot r. For a configuration K, if the set of values of $r.\mathit{step}$ appearing in K is V, we say that the *step configuration* of K is V.

Observe that at any moment in time of `sim-RS-by-S(`A`)` the step configuration $V \in \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{m\}, \{1,2\}, \{2,3\}, \{3,1\}, \{2,4\}, \{4,5\}, \{2,5\}, \{2,m\}\}$. Indeed, the case when $|V| = 1$ corresponds to a (sub)set of activated robots performing the same action (executing A, or modifying the flags *charged* and *executed*). These robots may update their *step* value leading to a new step configuration V' of a size at most two. In the case when $|V| = 2$ no actions are performed by the robots except of updating their *step* to some value $i \in V$ (remaining robots transition to Step i), eventually leading to the next step configuration $V' = \{i\}$.

The correctness of `sim-RS-by-S(`A`)` is shown by proving properties of the step configurations. The details and their proofs are shown in [2]. Note that, if algorithm A uses ℓ colors, the simulating algorithm `sim-RS-by-S(A)` uses $O(\ell)$ colors.

Lemma 6. *Algorithm* `sim-RS-by-S(A)` *correctly simulates execution of algorithm A run on $\mathcal{LUMI}^R$ robots by $\mathcal{LUMI}^S$ robots.*

Theorem 4. $\mathcal{LUMI}^S \equiv \mathcal{LUMI}^{RS}$.

4 Computational relationship between FSYNCH and RSYNCH

We have seen that RSYNCH is more powerful than SSYNCH in $\mathcal{FCOM}$, $\mathcal{FSTA}$, $\mathcal{OBLOT}$, and it has the same computational power in $\mathcal{LUMI}$. To better understand the power of RSYNCH among the classical synchronous schedulers, we now turn our attention to the relationship between RSYNCH and FSYNCH.

4.1 Dominance of FSYNCH over RSYNCH

The problem CENTER OF GRAVITY EXPANSION (CGE) was used to show dominance of FSYNCH over SSYNCH, that is, CGE is solvable in $\mathcal{FCOM}^F$ and $\mathcal{FSTA}^F$ but is not solvable in $\mathcal{LUMI}^S$ [15]. This problem can also be used to show dominance of FSYNCH over RSYNCH. In fact, we can obtain a stronger result showing that CGE is not solvable in $\mathcal{LUMI}^{F'}$, where F' is any scheduler such that the first activation does not contain all robots. The details can be found in [2].

Lemma 7. *Let $n \geq 2$. $\exists R \in \mathcal{R}_n$:* CGE $\notin \mathcal{LUMI}^{F'}(R)$*, where F' is any scheduler such that the first activation does not contain all robots.*

Since RSYNCH contains patterns in F', we have the following.

Corollary 1. *Let $n \geq 2$. $\exists R \in \mathcal{R}_n$:* CGE $\notin \mathcal{LUMI}^{RS}(R)$.

As a consequence, we obtain dominance of FSYNCH over RSYNCH.

Theorem 5. $\forall \mathcal{X} \in \{\mathcal{OBLOT}, \mathcal{FCOM}, \mathcal{FSTA}, \mathcal{LUMI}\}$ $[\mathcal{X}^F > \mathcal{X}^{RS}]$.

4.2 Orthogonality of FSYNCH with RSYNCH

We now proceed to show incomparability of $\mathcal{FSTA}^F$ with $\mathcal{X}^{RS}$ for $\mathcal{X} \in \{\mathcal{LUMI}, \mathcal{FCOM}\}$, and of $\mathcal{OBLOT}^F$ with $\mathcal{X}^{RS}$ for $\mathcal{X} \in \{\mathcal{LUMI}, \mathcal{FCOM}, \mathcal{FSTA}\}$. The former can be obtained by observing that (1) CYC is not in $\mathcal{FSTA}^F$ (Lemma 4) and is in $\mathcal{FCOM}^S$ (Lemma 5), and thus CYC is in $\mathcal{FCOM}^{RS}$ and $\mathcal{LUMI}^{RS}$, and (2) CGE is in $\mathcal{FSTA}^F$ ([15]) and is not in $\mathcal{LUMI}^{RS}$ (Corollary 1), and thus CGE is not in $\mathcal{FCOM}^{RS}$.

Theorem 6. *$\mathcal{FSTA}^F \perp \mathcal{LUMI}^{RS}$ and $\mathcal{FSTA}^F \perp \mathcal{FCOM}^{RS}$.*

The problems OSP and CGE*[6] were used to show $\mathcal{OBLOT}^F \perp \mathcal{LUMI}^S$ [15]. That is, problem OSP can be solved in $\mathcal{LUMI}^S$, but not in $\mathcal{OBLOT}^F$, and

[6] OSP is OSCILLATING POINTS and CGE* is PERPETUAL CENTER OF GRAVITY EXPANSION.

problem CGE* can be trivially solved in $\mathcal{OBLOT}^F$, but not in $\mathcal{LUMI}^S$. These problems can also be used to show $\mathcal{OBLOT}^F \perp \mathcal{X}^{RS}$ (for $\mathcal{X} \in \{\mathcal{LUMI}, \mathcal{FCOM}, \mathcal{FSTA}\}$) by using the below Lemmas 8–9, Corollary 1, and the fact that CGE* can be solved in $\mathcal{FSTA}^F$ [15].

Lemma 8 ([7]). $\exists R \in \mathcal{R}_2$, OSP $\notin \mathcal{OBLOT}^F(R)$.

Lemma 9. $\forall R \in \mathcal{R}_2$, OSP $\in \mathcal{FSTA}^{RS}(R) \cap \mathcal{FCOM}^{RS}(R)$.

It follows that

Theorem 7. $\mathcal{OBLOT}^F \perp \mathcal{LUMI}^{RS}$, $\mathcal{OBLOT}^F \perp \mathcal{FCOM}^{RS}$, *and* $\mathcal{OBLOT}^F \perp \mathcal{FSTA}^{RS}$.

5 Analysis within RSYNCH

In this section, we study the impact that memory persistence and communication capabilities have on the computational power of energy-constrained systems of robots.

We start by establishing the following theorem. Note that due to Theorem 4, this would imply that $\mathcal{LUMI}^{RS} \equiv \mathcal{FCOM}^{RS}$.

Theorem 8. $\forall R \in \mathcal{R}$, $\mathcal{LUMI}^S(R) \subseteq \mathcal{FCOM}^{RS}(R)$.

We do so, once again, by developing an algorithm `sim-LUMI-by-FCOM(`A`)` for $\mathcal{FCOM}$ robots that simulates a given algorithm A for $\mathcal{LUMI}$ robots (see [2] for details). Let A be an algorithm for $\mathcal{LUMI}$ robots with disorientation and non-rigid movement under SSYNCH, and let A use light with ℓ colors: $C = \{c_0, c_1, \ldots c_{\ell-1}\}$. We now extend the simulation algorithm of $\mathcal{LUMI}$ robots by $\mathcal{FCOM}$ robots in FSYNCH described in [15], designing a more complex simulation algorithm `sim-LUMI-by-FCOM(`A`)` by $\mathcal{FCOM}$ robots in RSYNCH.

The main ideas of the simulation remain the same. Robots first copy the lights of their neighbors. This in turn allows them to look at their neighbors to gain information about the color of their own light. Next, some robots activate and have enough information to execute a step in A. Lastly, the robots reset their states such that they can start copying the lights of their neighbors again. This cycle repeats itself and every time some robots execute algorithm A.

Contrary to the simulation algorithm in FSYNCH, we need to take extra care here to ensure that the resulting schedule is fair and every robot executes a step in A infinitely often. After all, the same subset of robots might execute algorithm A each time the robots perform this cycle. As with `sim-RS-by-S(`A`)`, we consider the concept of *mega-cycles*, in which every robot executes a step in A exactly once. To facilitate this, each robot gets a flag indicating if the robot has already executed A in this mega-cycle. To give a robot information about its own execute status, we let the robots copy this flag in the same way as they copy the other lights of their neighbors. As soon as all robots have executed A in one mega-cycle, these flags are reset and the next mega-cycle begins.

Recall that `suc` and `pred` are the successor and predecessor functions defined on a unique circular order on robots' positions $X(t)$. Figure 3 shows the transition diagram as the robots run the simulation. Each robot r has the following colors of $\mathcal{FCOM}$:

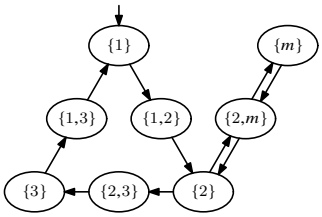

Fig. 3. Transition diagram of `sim-LUMI-by-FCOM(`A`)`.

- $r.color \in C$ indicating its own light used in algorithm A, initially set to $r.color = c_0$;
- $r.\texttt{suc}.color$ indicating a set of colors from 2^C at $\texttt{suc}(x)$, initially set to $r.\texttt{suc}.color = \varnothing$;
- $r.step \in \{1, 2, 3, m\}$ indicating the step of the algorithm. See below for more details on each step. Initially $r.step$ is set to 1;
- $r.executed \in \{True, False\}$ indicating whether r has executed the algorithm A in the current mega-cycle, initially set to $r.executed = False$;
- $r.\texttt{suc}.executed$ indicating a set from $2^{\{True, False\}}$ at $\texttt{suc}(x)$ and having the same role of $r.\texttt{suc}.color$, initially set to $r.\texttt{suc}.executed = \varnothing$;
- $r.me.checked$, $r.\texttt{suc}.checked \in \{True, False\}$ indicating whether all *color* and *executed* flags are correctly set, initially set to $r.me.checked$, $r.\texttt{suc}.checked = False$.

The algorithm is a sequence of mega-cycles, each of which lasts until all robots execute the simulated algorithm exactly once, and the end of a mega-cycle is checked at the beginning of Step 2. During each mega-cycle, this algorithm is composed of the following three steps:

- In Step 1, every robot obtains the neighbor's information so that robots can recognize their own lights.
- In Step 2, activated robots execute algorithm A using the color of their light obtained.
- In Step 3, every robot resets check flags to prepare the next cycle of execution of algorithm A.

Repeating the three steps, the end of the current mega-cycle is checked at the beginning of Step 2 and if the mega-cycle is finished, each robot r proceeds to Step m to reset.

- In Step m, every robot resets flag $r.executed$ indicating whether r has executed in this mega-cycle or not to *False* and the control returns to Step 2.

The details of these steps are as follows:

Step 1: Copy colors and activation ($\forall \rho \neq r\,(\rho.step = 1)$). In the *Look* phase, r checks if it is in Step 1 by detecting $\rho.step = 1$ for all other robots $\rho \neq r$. Every robot r stores (and displays) in variable $r.\texttt{suc}.color$ and $r.\texttt{suc}.executed$, respectively, its successors' sets of colors and also their *executed* flags indicating whether the successor robots have been activated in this mega-cycle. None of

the robots move in this step. Note that every robot can recognize its own color of light and its flags in the configuration after Step 1 is completed. This step ends when all robots store their neighbors' sets of *color* and *executed* flags. To be able to detect this, every robot sets $r.\mathit{checked}$ to *True*, indicating that it has executed Step 1. Moreover, whenever all successors of r have set their *checked* flags to *True*, r also sets $r.\texttt{suc}.\mathit{checked}$ to *True*. Now every robot can determine the end of Step 1 when the two checked flags ($\rho.\mathit{me}.\mathit{checked}$, $\rho.\texttt{suc}.\mathit{checked}$) of every robot ρ become *True*. When this condition is satisfied, r changes $r.\mathit{step}$ to 2.

Step 2: Perform simulation ($\forall \rho \neq r\,(\rho.\mathit{step} = 2)$). In the *Look* phase, r recognizes to be in Step 2 by observing $\rho.\mathit{step} = 2$ of all other robots ρ. Robot r can derive $r.\mathit{color}$ and $r.\mathit{executed}$ by using its predecessor's $\texttt{suc}.\mathit{color}$ and $\texttt{suc}.\mathit{executed}$. The rest of Step 2 consists of two phases: checking the end of a mega-cycle, and executing the simulated algorithm.

***Checking end of a mega-cycle*:** After calculating $r.\mathit{color}$ and $r.\mathit{executed}$, r checks if the current mega-cycle has ended. If all robots have executed algorithm A in this mega-cycle (all robots have $\rho.\mathit{executed}$ set to *True*), then r moves to Step m (resetting all the executed flags) without changing colors of lights.

***Execution of simulation*:** If the mega-cycle has not ended yet, the robots activated in this round will perform a step in A. The activated robots (indicated by set S_2) at this time, actually perform their cycle according to algorithm A if they have not performed algorithm A in this mega-cycle yet ($r.\mathit{executed}$ is *False*). The robots that executed algorithm A update $r.\mathit{executed}$ and furthermore update $r.\mathit{color}$ and move according to algorithm A. After the actual execution of algorithm A, each robot r performing the simulation sets $r.\mathit{step} = 3$ to reset all checking flags (Step 3) to prepare for the next Step 1. If all robots in S_2 have performed algorithm A so far in this mega-cycle, no light is changed, and Step 2 is performed again at the next round. This continues until some robots not having executed algorithm A are activated. This is guaranteed by the fairness of the scheduler.

Step 3: Reset check flags ($\forall \rho \neq r(\rho.\mathit{step} = 3)$). In the *Look* phase, r checks if it is in Step 3 by observing $\rho.\mathit{step} = 3$ for all other robots ρ. In Step 3, all robots reset $r.\mathit{checked}$, $r.\texttt{suc}.\mathit{checked}$ to *False*, and $r.\texttt{suc}.\mathit{executed}$ to $\{\mathit{False}\}$, enabling the robots to perform another round in this cycle. Each robot resetting its flags changes $r.\mathit{step}$ to 1.

Step m: Reset mega-cycle ($\forall \rho \neq r(\rho.\mathit{step} = m)$). In the *Look* phase, r checks if it is in Step m by observing $\rho.\mathit{step} = m$ for all other robots ρ. In Step m, each robot r sets $r.\mathit{executed} = \mathit{False}$ and $r.\texttt{suc}.\mathit{executed} = \{\mathit{False}\}$ and the step returns to 2 to begin a new mega-cycle after all robots reset their activated flags.

We can show that this simulation algorithm works correctly for $\mathcal{FCOM}$ robots in RSYNCH. Considering the specifics of $\mathcal{FCOM}$, the following two schemes are

Algorithm 1 `Scheme-for-modification-flags` for robot r at location x.

Assumptions

Let configurations be $same(step = \alpha)$ or $except1(step = \alpha; \gamma)$

State Compute

```
1: if ∀ρ ≠ r (ρ.step = α) then                                   // step α
2:     if not ∀ρ ≠ r (ρ.flag = True) then
3:         r.step ← α, r.flag ← True
4:     else if ∀ρ ≠ r (ρ.flag = True) then
5:         r.flag ← True, r.step ← β
6: else if ∀ρ ≠ r (ρ.step = β) then r.step ← β                   // step β
7: else if ∀ρ ≠ r ((ρ.step = α) or (ρ.step = β)) then r.step ← β
```

essential for the algorithm to work correctly: (1) transition scheme between step configurations, and (2) recognition of own light color.

(1) transition scheme between step configurations: Since we consider $\mathcal{FCOM}$ robots, when a robot r checks a predicate, for example $\forall\rho\,(\rho.step = \alpha)$, r cannot see its own value $r.step$. Therefore, r only checks $\forall\rho \neq r\,(\rho.step = \alpha)$ and observes that the predicate may be satisfied although $r.step$ is not α. Thus, predicates appearing in the simulation must be of the form $\forall\rho \neq r\,(\cdots)$, and we must consider configurations in which only one robot r observes that some predicate holds while any of the other robots would observe that the predicate does not hold.

To provide some intuition, consider a simple example where every robot ρ has two lights $\rho.step \in \{\alpha, \beta, \gamma\}$ and $\rho.flag \in \{True, False\}$ (Algorithm 1). Let us consider a configuration, denoted $same(step = \alpha)$, where $\rho.step = \alpha$ for any robot ρ, and a configuration, denoted $except1(step = \alpha; \gamma)$, where there exists just one robot r_e with $r_e.step = \gamma$ while $\rho.step = \alpha$ for any other robot ρ. Lemma 10 below shows the following: if a step configuration is $same(step = \alpha)$ or $except1(step = \alpha; \gamma)$ with $\rho.flag = False$ for every robot ρ, Algorithm 1 transforms the step configuration into one of type $same(step = \beta)$ or $except1(step = \beta; \alpha)$ with $\rho.flag$ changed to $True$ for every robot ρ. The idea is that, as long as a robot still observes at least one other robot with $flag = False$, it sets its $flag$ to $True$, while maintaining $step = \alpha$. As soon all robots have their $flag = True$, the simulation needs to progress to $step = \beta$. However, the activated robot r may see that all other robots have their $flag = True$, but r itself may not have been active before and therefore still needs to set its $flag$ to $True$. Now any robot active that sees another robot with $step = \beta$ knows that all values $flag$ are $True$, including its own. The formal proof is shown in [2].

Lemma 10. *Let $C(t_a)$ be a step configuration either of type $same(step = \alpha)$ or $except1(step = \alpha, \gamma \neq \beta)$ at time t_a, and let $\forall\rho\,(\rho.flag = False)$ hold at t_a. If Algorithm 1 is executed on $C(t_a)$, then there is a time $t_b > t_a$ when $C(t_b)$ satisfies the following conditions:*

(a) The step configuration $C(t_b)$ is either same(step = β) or except1(step = $\beta;\alpha$).
(b) $\forall\rho\,(\rho.flag = True)$ holds at t_b.

This scheme of transitions between step configurations is used in our simulation algorithm. Refer to [2] for the correctness of transitions between the steps and of the scheme as a whole.

(2) recognition of own light color: The color *r.color* used in the execution of algorithm A is determined as follows: Assuming all robots have correctly set their $\texttt{suc}.color$, $\rho.\texttt{suc}.color$ for any $\rho \in \texttt{pred}(x)$ contains all colors at location x. Let $r.color.here$ be the set of colors seen at x. Note that, by definition, this does not include $r.color$, since r is on location x and it cannot see its own light. Now r can calculate $r.color$ in the following way:

$$r.color = \texttt{pred}(x).\texttt{suc}.color - x.color.here\,.$$

Since the executed flag of robot r contains two colors, *r.executed* can be determined similarly to the case of determining *r.color* by using $\texttt{pred}(x).\texttt{suc}.executed$ instead of $\texttt{pred}(x).\texttt{suc}.color$.

Note, that we can remove the assumption of chirality by using the method of [15]. The robots copy colors of both predecessors and successors, and deduce their colors by looking at the colors mirrored by the robots in the two-hop neighborhood.

It is fairly straightforward that this algorithm works well in the first phase of RSYNCH (equivalent to FSYNCH). Once a non-full activation occurs, full activation will never happen and thereafter activations of disjoint sets of robots happen alternatingly (see [2]).

Algorithm `sim-LUMI-by-FCOM(`A`)` executes Steps 1–3 and Step m in infinite rounds in RSYNCH and the execution of A obeys SSYNCH. Let E be the sequence of the set of activated robots that execute steps of algorithm A in simulation `sim-LUMI-by-FCOM(`A`)`. Since, we can show that any mega-cycle is completed and every robot executes exactly once in every mega-cycle, E is fair. Then we have obtained Theorem 8. Note that, if algorithm A uses ℓ colors, the simulating algorithm `sim-LUMI-by-FCOM(`A`)` uses $O(\ell 2^\ell)$ colors.

Therefore it holds that $\mathcal{FCOM}^{RS} \geq \mathcal{LUMI}^{S}$. Since $\mathcal{LUMI}^{S} \equiv \mathcal{LUMI}^{RS}$ by Theorem 4, we have that: $\mathcal{FCOM}^{RS} \geq \mathcal{LUMI}^{RS}$. Moreover, since the reverse relation is trivial and $\mathcal{FSTA}^{RS} < \mathcal{LUMI}^{S}$ (Theorem 2), the next theorem follows. The proof is shown in [2].

Theorem 9. *$\mathcal{FCOM}^{RS} \equiv \mathcal{LUMI}^{RS} \equiv \mathcal{LUMI}^{S}$, and $\mathcal{FCOM}^{RS} > \mathcal{FSTA}^{RS} > \mathcal{OBLOT}^{RS}$.*

6 Concluding remarks

In this paper, we have started the investigation of the algorithmic and computational issues arising in distributed systems of autonomous mobile entities in the Euclidean plane where their energy is limited, albeit renewable.

We have studied the difference in computational power caused by the energy restriction and provided a complete characterization of the computational difference between energy-constrained and unrestricted robots in all four models considered in the literature: $\mathcal{OBLOT}, \mathcal{FSTA}, \mathcal{FCOM}$, $\mathcal{LUMI}$. We have also examined the difference with robots with unlimited energy, operating under a fully-synchronous scheduler. Furthermore, we have studied the impact of memory persistence and communication capabilities on the computational power of such energy-constrained systems of robots. Some of these results have been obtained through the design and analysis of novel *simulators*: algorithms that allow a set of robots with a given set of capabilities to execute correctly any protocol designed for robots with more powerful capabilities.

References

1. Agmon, N., Peleg, D.: Fault-tolerant gathering algorithms for autonomous mobile robots. SIAM J. Comput. **36**(1), 56–82 (2006)
2. Buchin, K., Flocchini, P., Kostitsyna, I., Peters, T., Santoro, N., Wada, K.: On the computational power of energy-constrained mobile robots: Algorithms and cross-model analysis. arXiv.org **cs**(ArXiv:2203.06546) (2022)
3. Canepa, D., Potop-Butucaru, M.: Stabilizing flocking via leader election in robot networks. In: Proc. 10th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS). pp. 52–66 (2007)
4. Cicerone, S., Stefano, D., Navarra, A.: Gathering of robots on meeting-points. Distributed Computing **31**(1), 1–50 (2018)
5. Cieliebak, M., Flocchini, P., Prencipe, G., Santoro, N.: Distributed computing by mobile robots: Gathering. SIAM J. Comput. **41**(4), 829–879 (2012)
6. Cohen, R., Peleg, D.: Convergence properties of the gravitational algorithms in asynchronous robot systems. SIAM J. on Computing **34**(15), 1516–1528 (2005)
7. Das, S., Flocchini, P., Prencipe, G., Santoro, N., Yamashita, M.: Autonomous mobile robots with lights. Theoretical Computer Science **609**, 171–184 (2016)
8. Di Luna, G., Flocchini, P., Chaudhuri, S., Poloni, F., Santoro, N., Viglietta, G.: Mutual visibility by luminous robots without collisions. Information and Computation **254**(3), 392–418 (2017)
9. Di Luna, G., Viglietta, G.: Robots with lights. Ch. 11 of *[13]* pp. 252–277 (2019)
10. Flocchini, P., Prencipe, G., (Eds), N.S.: Distributed Computing by Mobile Entities. Springer (2019)
11. Flocchini, P., Prencipe, G., Santoro, N.: Distributed Computing by Oblivious Mobile Robots. Morgan & Claypool (2012)
12. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Gathering of asynchronous robots with limited visibility. Theoretical Computer Science **337**(1–3), 147–169 (2005)
13. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Arbitrary pattern formation by asynchronous oblivious robots. Theor. Comput. Sci. **407**, 412–447 (2008)
14. Flocchini, P., Santoro, N., Viglietta, G., Yamashita, M.: Rendezvous with constant memory. Theor. Comput. Sci. **621**, 57–72 (2016)
15. Flocchini, P., Santoro, N., Wada, K.: On memory, communication, and synchronous schedulers when moving and computing. In: Proc. 23rd Int. Conference on Principles of Distributed Systems (OPODIS). pp. 25:1–25:17 (2019)

16. Fujinaga, N., Yamauchi, Y., Ono, H., Kijima, S., Yamashita, M.: Pattern formation by oblivious asynchronous mobile robots. SIAM J. Comput. **44**(3), 740–785 (2015)
17. Gervasi, V., Prencipe, G.: Coordination without communication: The case of the flocking problem. Discrete Applied Mathematics **144**(3), 324–344 (2004)
18. Hériban, A., Défago, X., Tixeuil, S.: Optimally gathering two robots. In: Proc. 19th Int. Conference on Distributed Computing and Networking (ICDCN). pp. 1–10 (2018)
19. Izumi, T., Souissi, S., Katayama, Y., Inuzuka, N., Défago, X., Wada, K., Yamashita, M.: The gathering problem for two oblivious robots with unreliable compasses. SIAM J. Comput. **41**(1), 26–46 (2012)
20. Okumura, T., Wada, K., Défago, X.: Optimal rendezvous $\mathcal{L}$-algorithms for asynchronous mobile robots with external-lights. In: Proc. 22nd Int. Conference on Principles of Distributed Systems (OPODIS). pp. 24:1–24:16 (2018)
21. Okumura, T., Wada, K., Katayama, Y.: Brief announcement: Optimal asynchronous rendezvous for mobile robots with lights. In: Proc. 19th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS). pp. 484–488 (2017)
22. Sharma, G., Alsaedi, R., Bush, C., Mukhopadyay, S.: The complete visibility problem for fat robots with lights. In: Proc. 19th Int. Conference on Distributed Computing and Networking (ICDCN). pp. 21:1–21:4 (2018)
23. Sharma, G., Vaidyanathan, R., Bush, C., Rai, S., Borzoo, B.: Complete visibility for robots with lights in $o(1)$ time. In: Proc. 18th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS). pp. 327–345 (2016)
24. Sharma, H., Ahteshamul, H., Zainul A., J.: Solar energy harvesting wireless sensor network nodes: A survey. Journal of Renewable and Sustainable Energy **10**(2), 023704 (2018)
25. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: Formation of geometric patterns. SIAM J. Comput. **28**, 1347–1363 (1999)
26. Terai, S., Wada, K., Katayama, Y.: Gathering problems for autonomous mobile robots with lights. arXiv.org **cs**(ArXiv:1811.12068) (2018)
27. Viglietta, G.: Rendezvous of two robots with visible bits. In: 10th Int. Symp. on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS). pp. 291–306 (2013)
28. Yamashita, M., Suzuki, I.: Characterizing geometric patterns formable by oblivious anonymous mobile robots. Theor. Comput. Sci. **411**(26–28), 2433–2453 (2010)
29. Yamauchi, Y., Uehara, T., Kijima, S., Yamashita, M.: Plane formation by synchronous mobile robots in the three-dimensional euclidean space. J. ACM **64:3**(16), 16:1–16:43 (2017)