Ring exploration by a team of asynchronous oblivious robots

Paola Flocchini [*] David Ilcinkas [†] Andrzej Pelc [†]
Nicola Santoro [‡]

Abstract

We consider the problem of exploring an anonymous unoriented ring by a team of identical, oblivious, mobile robots. Robots start from different nodes of the ring and they operate in Look-Compute-Move cycles. At the end, every node must be visited by at least one robot, and all robots must stop. In one cycle, a robot takes a snapshot of the current configuration (Look), makes a decision to stay idle or to move to one of its adjacent nodes (Compute), and in the latter case makes an instantaneous move to this neighbor (Move). Cycles are performed asynchronously for each robot.

We show that the minimum number $\rho(n)$ of robots that can explore a ring of size n is $O(\log n)$ and that $\rho(n) = \Omega(\log n)$ for arbitrarily large n. On one hand we give an algorithm that explores the ring starting from any initial configuration, provided that n and k are co-prime, and we show that in this case k is $O(\log n)$. On the other hand we show that when $O(\log n)$ agents are necessary. Notice that, when k and n are not co-prime the problem is unsolvable (i.e., there are initial configurations for which the exploration cannot be done).

Finally, we study and characterize the exploration problem for the *line* where we describe an algorithm that performs the exploration from an arbitrary configuration for all values of k for which the exploration is possible.

Keywords: mobile robot, ring, line, exploration

[*]SITE, University of Ottawa, Ottawa, ON K1N 6N5, Canada. E-mail: flocchin@site.uottawa.ca

[†]Département d'informatique, Université du Québec en Outaouais, Gatineau, Québec J8X 3X7, Canada. E-mail: ilcinkas@lri.fr, pelc@uqo.ca.

[‡]School of Computer Science, Carleton University, Ottawa, Ontario, K1S 5B6, Canada. E-mail: santoro@scs.carleton.ca

1 Introduction

1.1 Framework

Mobile entities (robots), initially located at different nodes of the ring, have to explore it, by collectively visiting all nodes of the ring. At the end, every node must be visited by at least one robot and all robots must stop. We study the exploration problem in a very weak scenario, which, while simple to describe, makes coordination of robots' actions particularly hard, as robots cannot communicate directly but have to make decisions about their moves only by observing the environment. Moreover, they operate asynchronously and do not have memory of past observations.

Consider an unoriented anonymous ring of stations (nodes). Neither nodes nor links of the ring have any labels. Initially, some nodes of the ring are occupied by robots and there is at most one robot in each node. Robots operate in Look-Compute-Move cycles. In one cycle, a robot takes a snapshot of the current configuration (Look), then, based on the perceived configuration, makes a decision to stay idle or to move to one of its adjacent nodes (Compute), and in the latter case makes an instantaneous move to this neighbor (Move). Cycles are performed asynchronously for each robot. This means that the time between Look, Compute, and Move operations is finite but unbounded, and is decided by the adversary for each action of each robot. The only constraint is that moves are instantaneous, and hence any robot performing a Look operation sees all other robots at nodes of the ring and not on edges, while performing a move. However a robot $\mathcal{R}$ may perform a Look operation at some time t, perceiving robots at some nodes, then Compute a target neighbor at some time $t' > t$, and Move to this neighbor at some later time $t'' > t'$ in which some robots are in different nodes from those previously perceived by $\mathcal{R}$ because in the meantime they performed their Move operations. Hence robots may move based on significantly outdated perceptions, which adds to the difficulty of exploration. It should be stressed that robots are memoryless (oblivious), i.e., they do not have any memory of past observations. Thus the target node (which is either the current position of the robot or one of its neighbors) is decided by the robot during a Compute operation solely on the basis of the location of other robots perceived in the previous Look operation. Robots are anonymous and execute the same deterministic algorithm. They cannot leave any marks at visited nodes, nor send any messages to other robots.

This very weak scenario, introduced in [21] and similar to that considered in [1, 3, 11, 12, 19, 23, 24], is justified by the fact that robots may be very small, cheap and mass-produced devices. Adding distinct labels, memory, or communication capabilities would make such robots larger and more expensive, which is not desirable. Thus it is interesting to consider such a scenario from the point of view of applications. On the theoretical side, this weak scenario increases the difficulty of collective exploration by making the problem of coordinating actions of robots particularly hard, and thus provides an interesting insight to the general problem of organizing collective actions of mobile entities in a distributed

environment.

An important and well studied capability is the *multiplicity detection* [19, 21, 24]. This is the ability of the robots to perceive, during the Look operation, if there is one or more robots in a given location. In our case, it is easy to see that without this capability, exploration is always impossible. Indeed, there must be a configuration in which robots stop. If the initial configuration occupied precisely these nodes, only with one robot per node, there would be no way to perceive any difference from the stopping configuration, and thus robots would stop immediately, without exploring any node. Thus we assume the capability of multiplicity detection in our further considerations. It should be stressed that, during a Look operation, a robot can only tell if at some node there are no robots, there is one robot, or there are more than one robots: a robot does not see a difference between a node occupied by a or b robots, for distinct $a, b > 1$.

1.2 Related Work

Algorithms for graph exploration by mobile robots (robots) have been intensly studied in recent literature. Several scenarios have been considered. Most of the research is concerned with the case of a single robot exploring the graph. In [2, 8, 14] the robot explores strongly connected directed graphs and it can move only in the direction from head to tail of an edge, not vice-versa. In particular, [14] investigates the minimum time of exploration of directed graphs, and [2] gives improved algorithms for this problem in terms of the deficiency of the graph (i.e., the minimum number of edges to be added to make the graph Eulerian). Many papers, e.g., [15, 16, 17, 22] study the scenario where the explored graph is undirected and the robot can traverse edges in both directions. In [15] the authors investigate the problem of how the availability of a map influences the efficiency of exploration. In [22] it is shown that a graph with n nodes and e edges can be explored in time $e + O(n)$. In some papers, additional restrictions on the moves of the robot are imposed. It is assumed that the robot has either a restricted tank [6, 9], forcing it to periodically return to the base for refueling, or that it is tethered, i.e., attached to the base by a rope or cable of restricted length [17].

Exploration of anonymous graphs presents different difficulties. In this case it is impossible to explore arbitrary graphs by a single robot if no marking of nodes is allowed. Hence the scenario adopted in [7, 8] allows the use of *pebbles* which the robot can drop on nodes to recognize already visited ones, and then remove them and drop in other places. The authors concentrate attention on the minimum number of pebbles allowing efficient exploration and mapping of arbitrary directed n-node graphs. (In the case of undirected graphs, one pebble suffices for efficient exploration.) In [8] the authors compare exploration power of one robot to that of two cooperating robots with a constant number of pebbles. In [7] it is shown that one pebble is enough if the robot knows an upper bound on the size of the graph, and $\Theta(\log \log n)$ pebbles are necessary and sufficient otherwise.

In all the above papers, except [8], exploration is performed by a single robot. Exploration

by many robots has been investigated mostly in the context when moves of the robots are centrally coordinated. In [18], approximation algorithms are given for the collective exploration problem in arbitrary graphs. In [4, 5] the authors construct approximation algorithms for the collective exploration problem in weighted trees. On the other hand, in [20] the authors study the problem of distributed collective exploration of trees of unknown topology. However, the robots performing exploration have memory and can directly communicate with each other.

To the best of our knowledge, the very weak assumption of asynchronous identical robots that cannot send any messages and communicate with the environment only by observing it, has not been previously used in the context of graph exploration. It has been used, however, to study another problem, that of gathering robots in one location. Most of the research in this area concerned the case of robots moving freely in the plane [1, 3, 10, 11, 12, 13, 19, 23, 24, 26]. The scenario was further precised in various ways. In [10] it was assumed that robots have memory, while in [1, 3, 11, 12, 13, 19, 23, 24, 26] robots were oblivious, i.e., it was assumed that they do not have any memory of past observations. Oblivious robots operate in Look-Compute-Move cycles, similar to those described in our scenario. The differences are in the amount of synchrony assumed in the execution of the cycles. In [3, 26] cycles were executed synchronously in rounds by all active robots, and the adversary could only decide which robots are active in a given cycle. In [10, 11, 12, 13, 19, 23, 24, 26] they were executed asynchronously: the adversary could interleave operations arbitrarily, stop robots during the move, and schedule Look operations of some robots while others were moving.

Our scenario has been recently introduced in [21] to study the gathering problem in the ring. This scenario is very similar to the asynchronous model used in [19, 24]. The only difference with respect to [19, 24] is in the execution of Move operations. This has been adapted to the context of graphs: moves of the robots are executed instantaneously from a node to its neighbor, and hence robots always see other robots at nodes. All possibilities of the adversary concerning interleaving operations performed by various robots are the same as in the model from [19, 24], and the characteristics of the robots (anonymity, obliviousness, multiplicity detection) are also the same.

1.3 Our results

We consider the problem of exploring an anonymous ring of size n by k oblivious anonymous asynchronous robots scattered in the ring: within finite time and regardless of the initial placement of the robots, each node must have been visited by a robot and the robots must be in a configuration from which no further move by any robot is allowed. We study this problem and characterize the conditions for its solvability.

We first prove that this problem is unsolvable if $k|n$. We the prove that this condition is tight. In fact, we show that, whenever $gcd(n, k) = 1$, the robots can explore the ring terminating within finite time. The proof is constructive: we present a terminating

exploration protocol and prove its correctness.

We then consider the minimum number $\rho(n)$ of robots that can explore a ring of size n. We prove that $\rho(n)$ is $O(\log n)$ and that $\rho(n) = \Omega(\log n)$ for arbitrarily large n. More precisely, there exists a constant c such that, for arbitrarily large n, we have $\rho(n) \geq c \log n$.)

As an application of our main ideas in a simpler context, we also characterize sizes of teams of robots capable of exploring a line of length n.

2 Terminology and preliminaries

In a ring of n nodes, initially, some nodes are occupied by robots and there is at most one robot in each node. The number of robots is denoted by k. Two robots are called *neighboring*, if at least one of the two segments of the ring between them does not contain any robots. A segment of the ring between two neighboring robots is called *free* if there is no robot in this segment. If more than one robots are in a node, we say that they form a *tower*. In the following, clockwise/counterclockwise directions are introduced only for the purpose of definition, robots do not have this notion, as the ring is not required to be oriented.

During the exploration robots move, and at any time they occupy some nodes of the ring, forming a *configuration*. In an anonymous unoriented ring of size n, a configuration is defined for various possible starting nodes. In fact, let u be an arbitrary node occupied by at least one robot, then a configuration with respect to u is denoted by a pair of sequences $((a_1, \ldots, a_r), (a_r, \ldots, a_1))$, where the integers a_i a_i for $i < r$, denote the distance (number of links) between neighbouring agents in clockwise direction. Notice that $a_i = 0$ means that two agents are located on the same node (i.e., there is a tower). In a line a configuration is defined in the same way, except that clockwise is replaced with left/right direction, distance a_1 corresponds to the distance from the left border to the first agent and a_r the distance from the last agent to the right border.

It should be clear that in a ring different choices of the starting node give rise to different pairs of sequences. Respective sequences in these pairs are cyclic shifts of each other and correspond to the same positioning of robots. Whenever we do not specify the starting point of a configuration, we consider the lexicographically minimum shift σ_{min} of all cyclic shifts of the configurations.

Consider a configuration $C = (a_1, \ldots, a_r)$ without towers. C is called *periodic* if the sequence $(a_1, \ldots, a_r)$ is a concatenation of at least two copies of a subsequence p.

Since the ring is unoriented and the robots cannot distinguish between a configuration and its symmetric image, the view of a robot $\mathcal{R}$ from a node v is the the pair of configurations starting from v; the *View* of $\mathcal{R}$ is the set of its view from every node. the *view* of a robot *from a node* v is the the pair of configurations starting from v; the view For example,

consider a 9-node ring with nodes $v_1, \ldots, v_9$ where there is a robot at each of v_1 and v_2 , and two robots (i.e. forming a tower) at node v_4; then the view from node x_1 is the set $\{(1, 2, 0, 6), (6, 0, 2, 1)\}$.

A configuration without multiplicities is called *rigid* if the views of all robots are distinct.

We say that exploration of a n-node ring is possible with k robots, if there exists an algorithm which, starting from any configuration of k robots without towers, explores the entire ring and brings all robots to a configuration in which they all remain idle. We say that, in such a configuration, the robots have reached a terminal state.

In the case of the line, the terminology is similar except that we talk about left/right instead of clockwise/counterclockwise.

Obviously, if $n = k$, the exploration is already accomplished, hence we always assume that $n > k$.

3 Exploration of a ring

3.1 Basic restriction

Lemma 3.1 *Let $k < n$. If $k|n$ then exploration of a n nodes ring with k is not possible.*

Proof: By contradiction, let P be a generic solution protocol. Choose as initial configuration an equidistant placement of the k robots in the ring (it exists since $k|n$). Thus, initially the states of all robots are identical, say $\sigma(0)$. Clearly this state is not a terminal state otherwise, since $k < n$, P would terminate without exploring the ring contradicting the correctness of P. Consider now an adversary that uses a synchronous scheduler and a consistent orientation of the ring. Then, at each time step t, the states of all robots continue to be identical, say $\sigma(t)$, and furthermore they are indistinguishable from those of previous steps; i.e., $\sigma(t) = \sigma(0)$ for all t. Hence the the robots will never enter a terminal state, contradicting the fact that P leads within finite time all robots to a configuration in which they all remain idle. □

Hence, in the following we will consider the case when $gcd(n, k) = 1$.

3.2 Ring Exploration

In this section we present an algorithm that allows $k \geq 10$ robots to explore n-node ring whenever $gcd(n, k) = 1$.

First of all notice that when $gcd(n, k) = 1$ any configuration of the robots is aperiodic. Let $\sigma_{min}(t)$ denote the lexicographically minimum string at time t.

Property 3.1 *If $\sigma_{min}(t)$ is symmetric there are two occurrences of $\sigma_{min}(t)$ in σ, at time t and they are in opposite direction. If $\sigma_{min}(t)$ is not symmetric, it is unique.*

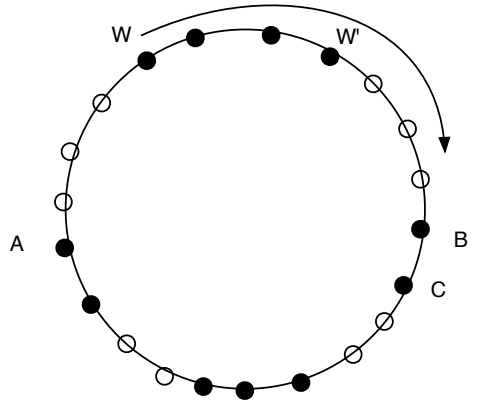


Figure 1: A symmetric configuration. There are two occurrences of the lex-min string, one clock-wise starting from W, the other counter-clock-wise starting from W'.

The idea of the algorithm is the following. When a robot observes a configuration $\sigma = (a_1, \ldots, a_r)$ that does not contain towers, it identifies the smallest lexicographic string σ_{min} among all the cyclic shifts of σ. There could be a unique occurrence of such a string (if the agent are asymmetrically placed) or two occurrences (if there is a symmetry axis). Each string identifies a leader and the idea is for the agents to appropriately move towards the unique leader (or one of the two leaders) so to form a single group of consecutive agents or two such groups. When such consecutive configurations are reached, one or two towers will be formed inside each group. The creation of the towers indicate the beginning of the actual exploration. At this point the extremal agent of each group moves to explore the free part between the groups until precise final configurations are reached. Final configurations contain one or two groups of consecutive agents with a tower, plus two agents in the middle of each interval of free agents (either next to each other, if the interval has an even number of nodes; or forming a tower, if it has an odd number of nodes).

Algorithm RING EXPLORATION for agent a

Case *configuration* **of:**
final
    DONE
consecutive or *2-consecutive* or *some-towers*
    FORM TOWER
consecutive-with-tower(s) or *exploring*
    If I am-the-explorer
      move to exploration direction (* start or continue exploring *)
other
    MAKE CONSECUTIVE

For a robot a, I-AM-THE-EXPLORER is true iff there is a group with a tower, a is isolated and has not reached the "middle" of the free segment.

Notice that the "middle" can always be identified since we are assuming that there are enough agents to unambiguously form two groups with a tower each.

FORM TOWER

If I belong to consecutive-group $x_1 \dots x_k$
  If k is odd and I am on $x_{\frac{k+1}{2}}$
    move to a neighbour to form a tower.
  If k is even and I am on $x_{\frac{k}{2}} - 1$
    move to $x_{\frac{k}{2}} - 2$
  If k is even and I am on $x_{\frac{k}{2}} + 1$
    move to $x_{\frac{k}{2}} + 2$

If there are two occurrences of the minimum string, an agent a observes its position in both strings with respect to the leaders. Let $L_1(a)$ be the closest of the two leaders; for example, in Figure 3.2, $L_1(A) = W'$, while $L_1(B) = L_1(C) = W$.

MAKE CONSECUTIVE

Compute the lexicographically minimum string σ_{min}.
If σ_{min} is *asymmetric*
  identify unique leader L
    Let σ_{min} starting from L identify the right direction
    If I am the closest agent to L with an empty left neighbour x
      move to x
If σ_{min} is *symmetric*
  Identify my-leader $L_1(a)$
    Let σ_{min} starting from $L_1(a)$ identify the *left-right* direction
    If among all the a' s.t. $L_1(a') = L_1(a)$, I am the closest to $L_1(a)$ with an empty *left* neighbour x
      move to x

Possible Configurations.

- *consecutive*: a group of k consecutive robots;
- *2-consecutive*: two groups of $\frac{k}{2}$ consecutive robots; k must be even, and the groups are separated by two free segments of different length (because the configuration cannot be periodic).
- *consecutive-with -tower*: an almost consecutive configuration, except that in the middle there are either two consecutive empty nodes next to two towers (k is even) or a central empty node with a tower in one of its neighbours (k is odd).
- *2- consecutive-with tower*: an almost 2-consecutive configuration, except that in the middle of each group there are either two consecutive empty nodes next to two towers (k is even), or a central empty node with a tower in one of its neighbours (k is odd).
- *exploring:* an almost 2-consecutive-with -tower configuration, except that two robots are isolated in the two free segments (they are exploring), or one is isolated and the other is still part of the group.
- *2-exploring:* same as exploring but with 2 pairs of robots exploring the two free segments.
- *Final*: a consecutive-with -tower configuration with two robots in the middle of the free segment (either forming a tower, if the segment has an odd number of nodes, or next to each other otherwise), or a 2-consecutive-with -tower configuration with two pairs of robots in the middle of the two segments (each of them either forming a tower, if the corresponding segment has an odd number of nodes, or next to each other otherwise.

3.3 Correctness and Complexity

Let us now prove the correctness of algorithm RING EXPLORATION and analyze its complexity.

Property 3.2 *At any point in time in the execution of* Ring Exploration*, at most two robots are allowed to move following an observation.*

Lemma 3.2 *Let a and b be the two agents that, in the execution of* Ring Exploration*, could potentially become active observing a configuration at time t. Let agent a move at time $t' > t$ on the basis of the observation of $\sigma_{min}(t)$, the minimum string $\sigma_{min}(t'+1)$ immediately after the movement of a is smaller than $\sigma_{min}(t)$.*

Proof:

- Let $a = b$. This means that there is a unique leader L and the configuration is asymmetric at time t. Let $\sigma_{min}(t) = (1^q, s_2, \dots s_m)$, (where with 1^q, $q \geq 0$ we denote q occurrences of distance one). After agent a moves, the new configuration starting from L has one of the the following forms: $\sigma(t'+1) = (1^{q+1}, s_2 - 1, \dots, \dots, s_m)$ (if robot a joins the group of consecutive robots following L), or $\sigma(t'+1) = (1^q, s_2 - 1, s_3 + 1, \dots, s_m)$ (otherwise). In both cases, $\sigma_{min}(t'+1) = \sigma(t'+1) < \sigma_{min}(t)$.

- Let $a \neq b$. This means that the configuration is symmetric and thus there are two occurrences of $\sigma_{min}(t) = (1^q, s_2, \dots s_m)$, let us call $\sigma_{(min,a)}(t)$ the one starting from $L(a)$ and $\sigma_{(min,b)}(t)$ the one starting from the other leader $L(b)$ (this is the case, for example, of Figure 1 where a is in A and b is in B) . By definition of the algorithm only a and b can become active. After a moves at time t', we have various cases:
1) in the meantime there has been a single movement by b: in this case the new configurations from $L(a)$ and $L(b)$ immediately after a's movement are still symmetric of the form: $\sigma_{(min,a)}(t'+1) = \sigma_{(min,b)}(t'+1) = (s_1, \dots, s_i - 1, s_{i+1} + 1, \dots, s_w + 1, s_{w+1} - 1, \dots, s_m) = \sigma_{min}(t'+1)$, which is clearly smaller than $\sigma_{min}(t)$.
2) in the meantime there have been several movements: by definition of the algorithm the movements can be performed (either by b only or by different robots after b) only in the direction of the leader of b (for example, in Figure 1 robot b could move 3 times and then robot c in C could start moving). Moreover, a single robot (besides a) can be active at a time. Let us assume first that all the movements (say f movements) have been performed by b to get closer to its leader. The new strings seen from $L(a)$ and from $L(b)$ are:
$\sigma_a(t'+1) = (s_1, \dots s_i - 1, s_{i+1} + 1, s_{i+2}, \dots, s_x + f, s_{x+1} - f, \dots, s_m)$, and
$\sigma_b(t'+1) = (s_1, \dots s_i - f, s_{i+1} + f, s_{i+2}, \dots s_x + 1, s_{x+1} - 1, \dots, s_m)$. Clearly $\sigma_b(t'+1) < \sigma_a(t'+1)$. Moreover, $\sigma_{min}(t'+1) = \sigma_b(t'+1) < \sigma_{min}(t)$.

If some other robots c has moved after b, it means that b has joined the consecutive group of its leader and c has started to get closer as well, the new minimum string is even smaller. Further movements by other robots keep decreasing the string. □

Lemma 3.3 *In the execution of* Ring Exploration*, within finite time, a consecutive or 2-consecutive configuration is reached.*

Proof: It follows from the fact that, by Lemma 3.2, the lexicographically minimum string decreases at each movement until, by definition of movements, a consecutive or 2-consecutive configuration is reached. □

Theorem 3.1 *For any $n > k \geq 10$, Algorithm* Ring Exploration *performs exploration of a n-node ring by any team of k robots, if $gcd(n, k) = 1$.*

Proof:

In each consecutive block a tower is unambiguously formed. From now on all robots know that the exploration is started. If the consecutive configuration is one, there are two robots that can move. By the algorithm, the robots move until they meet in the middle of the free segment (either next to each other, if the number of nodes in the segment in even, or forming a tower if it is odd). Notice that this is unambiguously recognized as a final configuration because in no other moment there has been a group with a tower on one side of the line and a pair of robots or a tower in the middle of the free segment. If there are two consecutive configurations, two pairs of robots can move and the same argument works in the two free segments. □

3.4 Size of the minimum team

In this section we show that the minimum number of robots that can explore a ring regardless of their initial position is logarithmic in n. More precisely, we have the following result.

Theorem 3.2 *The minimum number $\rho(n)$ of robots that can explore a n-node ring has the following properties:*

1. *$\rho(n) \in O(\log n)$;*
2. *there exists a constant c such that, for arbitrarily large n, we have $\rho(n) \geq c \log n$.*

Proof: Let p_j denote the j-th prime, and let $p_j\#$ denote p_j-*primorial*, that is

$$p_j\# = \Pi_{i=1}^{j} \quad p_i \tag{1}$$

An important property of the primorial is the following [25]:

Property 3.3 $\lim_{j\to\infty} \ (p_j\#)^{\frac{1}{p_j}} \ = \ e.$

We will now prove each part of the theorem separately.

Part 1.
Let $f(n)$ be the smallest integer coprime with n. By definition $gcd(n, f(n)) = 1$; thus, by Theorem 3.1, exploration is possible with $f(n)$ agents. Hence, $\rho(n) \leq f(n)$.

Observe that $f(p_j\#) = p_{j+1}$; furthermore, if $p_{j-1}\# < n \leq p_j\#$, then $f(n) = f(p_j\#)$. Let $f(n) = f(p_j\#)$ for some integer j.

By definition of primorial we have

$$p_{j+1} = \frac{p_{j+1}\#}{p_j\#}$$

By Property 3.3 it follows that, for large enough j, $p_{j+1} \approx e^{p_{j+1}-p_j}$, that is, $p_{j+1} \approx p_j + \log_e p_{j+1}$; thus, by Property 3.3, $p_j \in O(\log p_j\#)$. In other words,

$$\rho(n) \leq f(n) = f(p_j\#) = p_{j+1} \in O(\log p_j\#) = O(\log n),$$

completing the proof of Part 1.

Part 2.
The proof of Part 2 is carried out through a series of properties. Let $x \in \mathcal{R}^+$. We denote by $p_{s(x)}$ the largest prime not greater than x, and by $i_j(x)$ the integer such that $p_j^{i_j} \leq x < p_j^{i_j+1}$.

Property 3.4 $i_1(x) \geq i_2(x) \geq i_3(x) \geq \ldots \geq i_{s(x)}(x)$

Proof: By contradiction, let $i_j(x) \geq i_{j+1}(x)$. Then, $p_j^{i_j(x)+1} \leq p_j^{i_{j+1}(x)}$; however $p_{j+1}^{i_{j+1}(x)} < p_j^{i_j(x)+1}$ by definition, a contradiction. □

Define now

$$x\& = \Pi_{j=1}^{s(x)} \quad p_j^{i_j(x)} \tag{2}$$

It is easy to verify the following:

Property 3.5 *For all integers $i \leq x$, $i|x\&$*

As a consequence, the smallest i that does not divide $x\&$ must be greater than x.

The notion of primorial can be extended to the reals in a natural way. For $x \in \mathcal{R}^+$, we define *x-primorial*, denoted by $x\#$, as follows:

$$x\# \ = \Pi_{j=1}^{s(x)} \ \ p_j \tag{3}$$

Note that by definition

$$x\# \ = p_{s(x)}\# \tag{4}$$

From definitions 2 and 3, we have

Property 3.6 $x\& = \ \Pi_{j=1}^{\lceil \log x \rceil} \ x^{\frac{1}{j}}\#$

We thus have the following

Property 3.7 *There exists a constant c such that for every* $x \in \mathcal{R}^+$ *we have* $\log_c(x\&) \leq \sum_{j=1}^{\lceil \log x \rceil} \ x^{\frac{1}{j}}$

Proof: By Property 3.3 it follows that there exists a constant c such that $(p\#)^{\frac{1}{p}} \ \leq c$ for any prime p. That is, for any prime p we have $p\# \leq c^p$. Thus for every $x \in \mathcal{R}^+$ we have

$$x\# = p_{s(x)}\# \leq c^{p_s(x)} \leq c^x.$$

Hence, by Property 3.6, we have

$$\log_c(x\&) = \ \sum_{j=1}^{\lceil \log x \rceil} \ \log_c(x^{\frac{1}{j}}\#) \leq \sum_{j=1}^{\lceil \log x \rceil} \ x^{\frac{1}{j}}$$

□

Property 3.8 $\sum_{j=1}^{\lceil \log x \rceil} \ x^{\frac{1}{j}} \leq 2x$

Proof: $\sum_{j=1}^{\lceil \log x \rceil} \ x^{\frac{1}{j}} = x + \sum_{j=2}^{\lceil \log x \rceil} \ x^{\frac{1}{j}} \leq x + x^{\frac{1}{2}}(\lceil \log x \rceil - 1) \leq 2x$ □

Summarizing, by Lemma 3.1, for exploration to be possible, it must be $gcd(k, n) = 1$. Let $n = j\&$ for some integer j; by Property 3.5, the smallest k that does not divide $n = j\&$ must be greater than j. However, by Properties 3.7 and 3.8, it follows that

$$j \geq \tfrac{1}{2} \log_c(j\&) = \tfrac{1}{2} \log_c n$$

completing the proof of Part 2 of Theorem 3.2. □

It should be noted that for some specific values of n, the number $\rho(n)$ is constant. For example, if n is prime, then $\rho(n) = 5$.

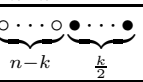
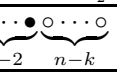
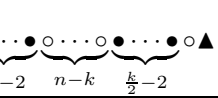
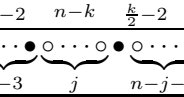
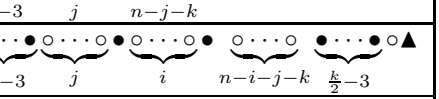
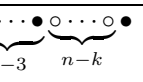
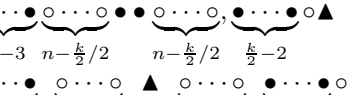
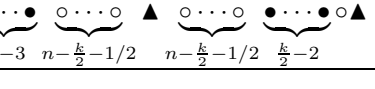
4 Exploration of the line

In this section we characterize sizes of teams of robots capable of exploring a line of length n.

Theorem 4.1 *Consider k robots in a n-node line, where $n > k$. Exploration of a n-node line by $k < n$ robots is possible, if and only if, $k = 4$ and n is odd, or $k = 3$, or $k \geq 5$.*

4.1 Exploration with 3 or $k \geq 5$ Robots

We first present an algorithm to explore a line when $k = 3$ or $k \geq 5$. The idea of the algorithm is to have the k robots occupy k consecutive locations at one extreme of the line (if k is odd) or $\frac{k}{2}$ consecutive locations on both sides of the line (if k is even). When one of these configurations occur, the second robot of each group moves on the first creating a tower. This indicates the beginning of the actual exploration: now the extremal robots move along the line. The algorithm terminates when the exploring robot reaches the other end of the line (k odd), or two exploring robots are next to each other in the middle of the line (k even, n even), or when they form a tower in the middle of the line (k even, n odd).

consecutive: $(1^{k-1}, n-k+1)$	
2-consecutive: $(1^{\frac{k}{2}-1}, n-k, 1^{\frac{k}{2}-1})$	
consecutive-with -tower: $(0, 2, 1^{k-2}, n-k+1)$	
2- consecutive-with tower: $(0, 2, 1^{\frac{k}{2}-3}, n-k+1, 1^{\frac{k}{2}-3}, 0, 2)$	
exploring $(0, 2, 1^{k-4}, j, n-j-k+1)$	
2-exploring $(0, 2, 1^{k-4}, j, i, n-i-j-k+1)$	
Final: (k odd) $(0, 2, k-3, n-k+1)(n)$	
(k even, n even)	
(k even, n odd)	

Algorithm LINE EXPLORATION for robot a.

If *final*
  DONE
If *consecutive* or *2-consecutive*
  FORM TOWER
If *consecutive(s)-with-tower(s)*, *exploring*, *2- exploring*
  If I-AM-THE-EXPLORER
    "MyRight" is the direction from the tower in my group
    move to "MyRight"
If *other*
  MAKE-CONSECUTIVE (* create consecutive or 2-consecutive *)

Given a line $L_n = x_1 \dots x_n$, let h_1 be the number of robots in $[x_1 \dots x_{\lfloor \frac{n}{2} \rfloor}]$ and h_2 the number of robots in $[x_{\lceil \frac{n}{2} \rceil} \dots x_n]$

MAKE-CONSECUTIVE

If k odd
  If n is odd, $h_1 = h_2$, and I am a robot on $x_{\frac{n+1}{2}}$
    move anywhere
  If $h_1 \neq h_2$
    let *left-right* be the direction from $min\{h_1, h_2\}$ to $max\{h_1, h_2\}$
    If my right neighbour y is empty
      move to y
If k even
  DIVIDE HALF-HALF (* $\frac{k}{2}$ robots in h_1 and $\frac{k}{2}$ in h_2 *)
  If my neighbour x towards the closest border is empty
    move to x

I-AM-THE-EXPLORER : If I am the extremal of my consecutive group or I am isolated.

FORM TOWER

If *consecutive* (in $x_1 \dots x_k$)
  If I am on x_2
    move to x_1
  If k is even and I am on $x_k - 1$
    move to x_k

Lemma 4.1 *In the execution of* Line Exploration*, in finite time, a consecutive or 2-consecutive configuration is reached.*

Proof: If k is odd, a left-right direction of the line is determined (from the half of the line containing less robots to the half containing more). Notice that, if the line has an odd number of nodes, and the central node is initially occupied by a robot, this robot is the first that moves (arbitrarily) from there. Once the direction is identified any robot with a right empty neighbour moves (i.e., towards the half that contains more robots), thus maintaining a consistent left-right direction. Eventually none of the robots can move and a consecutive configuration is reached. If k is even, the robots first divide themselves evenly among the two halves of the line and follow the same procedure, each half towards their closest borders, clearly reaching a 2-consecutive configuration. □

Lemma 4.2 *In the execution of* Line Exploration*, from a consecutive or 2-consecutive configuration the line is correctly explored if $k = 3$, or $k \geq 5$.*

Proof: In each consecutive block a tower is unambiguously formed. From now on all robots know that the exploration is started. If the consecutive configuration is one (i.e k is odd), there is a unique robot that can move ($k \geq 3$ guarantees that there is one). By the algorithm, the robots moves until it reaches the other side of the line. Notice that this is unambiguously recognized as a final configuration because in no other moment there has been a tower on one side of the line and a single robot on the other extremity. If there are two consecutive configurations (i.e., k is even), two robots can move until they form a tower in the middle of the line (if it has an odd number of nodes), or they stop next to each other. Two such robots are guaranteed to exist because when k is even, $k \geq 6$

Also these configurations are unambiguously recognized as final because in no other moment there have been two towers on the two extremities and a pair of robot (or a tower) in the center of the line. □

4.2 Exploration of the line when $k = 4$ and n is odd

Let $k = 4$ and n odd. In this case, the following algorithm would explore the line. First the 4 robots place themselves so to have two on one side of the central node, and two on the other side. Now the two robots closer to the central node form a tower there. At this point the other two robots move towards the respective extremities.

4.3 Impossibility of Line Exploration

To prove that exploration of the line is impossible for $k < 3$ and $k = 4$ with n even, we use the following meta-property, which is a general property that holds for the exploration of any graph, not just the line.

Property 4.1 *Let P be a correct exploration protocol. Then, in any execution of P, the robots can never be in the same non terminal configuration $\mathcal{C}$ in two different moments: t and $t' > t$.*

Proof: It follows from the fact that the robots are oblivious, so, if they are in the same configuration the adversary can make them perform exactly the same actions.

Consider an execution $\mathcal{E}$ and consider the first time t' when a configuration $\mathcal{C}$, which already occurred at time t, re-occurs. Let $(C_1, \ldots, C_s)$ be the sequence of configurations of $\mathcal{E}$ up to time t' and let $C_i = C_s = \mathcal{C}$ for some $i < s$. Consider now another execution $\mathcal{E}'$ whose first s configurations coincide with the first s of $\mathcal{E}$ $(C_1, \ldots, C_i, C_{i+1}, \ldots, C_s)$. Since $C_i = C_s$ the adversary can now make the robot behave exactly has in the sequence of configurations $(C_{i+1} \ldots C_s)$ (i.e., $C_{s+x} = C_{i+x}$) creating a periodic sequence. Thus, during execution $\mathcal{E}'$ it is impossible that the robots enter a configuration in which all of them decides to remain idle.

□

Lemma 4.3 *If $k = 1$ the exploration of the line is impossible.*

Proof: Let P be any correct algorithm. Let the single robot r be located in an arbitrary node x_i of the line $[x_1, \ldots x_n]$. Without loss of generality, let r move following algorithm P in the direction of x_1 (the notion of direction is used only for clarity of description, it not known to the robots). Notice that, in any correct algorithm, robot r cannot reverse its direction during its walk, otherwise it would create a configuration that occurred in the past and, by Property 4.1 the algorithm wouldn't be correct. So, r will necessarily reach node x_1. At this point, however, the only possible movement of r creates a configuration that already occurred and by Property 4.1 the algorithm cannot be correct correct. □

Lemma 4.4 *If $k = 2$ the exploration of the line is impossible.*

Proof: If $k = 2$ there are initial configurations from which it is impossible to explore the line. Consider, for example a configuration $\mathcal{C}$ where the two robots r_1 and r_2 are symmetrically placed each at distance d from their closest borders. There are several situations to consider:
1) let robot r_1 move towards its closest border. In this case, the adversary makes also robot r_2 move towards its closest border. From this moment on the adversary keeps making them move simultaneously. Notice that they have to keep moving in the same direction otherwise they would end up in a configuration that already occurred in the past and by Property 4.1 the line would not be explored. Thus they reach the borders. At this point the only possible move would make tham return to a previous configuration and by Property 4.1 the line is not explored.

2) let robot r_1 move towards the center of the line. In this case, the adversary makes also robot r_2 move towards the center of the line; moreover, from this moment it keeps making them move simultaneously. Following the same reasoning as above, r_1 and r_2 will keep moving in the same direction otherwise they will not explore the line by Property 4.1. Let the line contain an even number of nodes. In this case r_1 and r_2 will eventually be next to each other and at the next movement they will necessarily form a configuration already formed in the past (remember that the robots are undistinguishable) and by Property 4.1 the line would not be explored. Let the line contain an odd number of nodes. In this case r_1 and r_2 will eventually form a tower in the central node. At the next movement, the adversary will split the tower and the robots will form a configuration already formed in the past (by Lemma 4.1 the line would not be explored). □

Lemma 4.5 *If $k = 4$ and n is even the exploration of the line is impossible.*

Proof: The argument is similar to the one of Lemma 4.4, where the initial configuration for which there exist no correct exploration algorithms is when the four robots are consecutively placed in the center of the line. □

Acknowledgment This work was done during the stay of David Ilcinkas at the Research Chair in Distributed Computing at the Université du Québec en Outaouais and at the University of Ottawa, as a postdoctoral fellow. Andrzej Pelc was partially supported by the Research Chair in Distributed Computing at the Université du Québec en Outaouais, Paola Flocchini was partially supported by the University Research Chair of the University of Ottawa. This work was supported in part by the Natural Sciences and Engineering Research Council under Discovery grants.

References

[1] N. Agmon, D. Peleg: Fault-Tolerant Gathering Algorithms for Autonomous Mobile Robots. SIAM J. Comput. 36(1): 56-82 (2006).

[2] S. Albers and M. R. Henzinger, Exploring unknown environments, SIAM Journal on Computing 29 (2000), 1164-1188.

[3] H. Ando, Y. Oasa, I. Suzuki, M. Yamashita: Distributed Memoryless Point Convergence Algorithm for Mobile Robots with Limited Visibility. IEEE Trans. on Robotics and Automation 15(5): 818-828 (1999).

[4] I. Averbakh and O. Berman, A heuristic with worst-case analysis for minimax routing of two traveling salesmen on a tree, Discr. Appl. Mathematics 68 (1996), 17-32.

[5] I. Averbakh and O. Berman, $(p-1)/(p+1)$-approximate algorithms for p-traveling salesmen problems on a tree with minmax objective, Discr. Appl. Mathematics 75 (1997), 201-216.

[6] B. Awerbuch, M. Betke, R. Rivest and M. Singh, Piecemeal graph learning by a mobile robot, Proc. 8th Conf. on Comput. Learning Theory (1995), 321-328.

[7] M.A. Bender, A. Fernandez, D. Ron, A. Sahai and S. Vadhan, The power of a pebble: Exploring and mapping directed graphs, Proc. 30th Ann. Symp. on Theory of Computing (STOC 1998), 269-278.

[8] M.A. Bender and D. Slonim, The power of team exploration: Two robots can learn unlabeled directed graphs, Proc. 35th Ann. Symp. on Foundations of Computer Science (FOCS 1994), 75-85.

[9] M. Betke, R. Rivest and M. Singh, Piecemeal learning of an unknown environment, Machine Learning 18 (1995), 231-254.

[10] M. Cieliebak, Gathering Non-oblivious Mobile Robots, Proc. 6th Latin American Symposium on Theoretical Informatics (LATIN 2004): 577-588.

[11] M. Cieliebak, P. Flocchini, G. Prencipe, N. Santoro, Solving the Robots Gathering Problem, Proc. 30th International Colloquium on Automata, Languages and Programming (ICALP 2003), LNCS 2719: 1181-1196.

[12] R. Cohen, D. Peleg, Robot Convergence via Center-of-Gravity Algorithms, Proc. 11th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2004), LNCS 3104: 79-88.

[13] J. Czyzowicz, L. Gasieniec, A. Pelc, Gathering few fat mobile robots in the plane, Proc. 10th International Conference on Principles of Distributed Systems (OPODIS'2006), LNCS 4288, 744-753.

[14] X. Deng and C. H. Papadimitriou, Exploring an unknown graph, Journal of Graph Theory 32 (1999), 265-297.

[15] A. Dessmark and A. Pelc, Optimal graph exploration without good maps, Proc. 10th European Symposium on Algorithms (ESA 2002), LNCS 2461, 374-386.

[16] K. Diks, P. Fraigniaud, E. Kranakis and A. Pelc, Tree exploration with little memory, Proc. 13th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA 2002), 588-597.

[17] C.A. Duncan, S.G. Kobourov and V.S.A. Kumar, Optimal constrained graph exploration, Proc. 12th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA 2001), 807-814.

[18] G. N. Frederickson, M. S. Hecht and C. E. Kim, Approximation algorithms for some routing problems. SIAM J. on Computing 7 (1978), 178-193.

[19] P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer: Gathering of Asynchronous Robots with Limited Visibility. Theoretical Computer Science 337(1-3): 147-168 (2005).

[20] P. Fraigniaud, L. Gasieniec, D. Kowalski, A. Pelc, Collective tree exploration, Proc. Latin American Theoretical Informatics (LATIN'2004), LNCS 2976, 141-151.

[21] R. Klasing, E. Markou, A. Pelc, Gathering asynchronous oblivious mobile robots in a ring, Proc. 17th International Symposium on Algorithms and Computation (ISAAC 2006).

[22] P. Panaite and A. Pelc, Exploring unknown undirected graphs, Journal of Algorithms 33 (1999), 281-295.

[23] G. Prencipe: CORDA: Distributed Coordination of a Set of Autonomous Mobile Robots. Proc. ERSADS 2001: 185-190.

[24] G. Prencipe: On the Feasibility of Gathering by Autonomous Mobile Robots. Proc. 12th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2005), LNCS 3499: 246-261.

[25] S.M. Ruiz: A Result on Prime Numbers. Math. Gaz. 81 (1997), 269.

[26] I. Suzuki, M. Yamashita: Distributed Anonymous Mobile Robots: Formation of Geometric Patterns. SIAM J. Comput. 28(4): 1347-1363 (1999).