How many oblivious robots can explore a line

Paola Flocchini, David Ilcinkas, Andrzej Pelc, Nicola Santoro

How many oblivious robots can explore a line

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

Abstract

We consider the problem of exploring an anonymous line by a team of k identical, oblivious, asynchronous deterministic mobile robots that can view the environment but cannot communicate. We completely characterize sizes of teams of robots capable of exploring a n-node line. For $k < n$, exploration by k robots turns out to be possible, if and only if either $k = 3$, or $k \geq 5$, or $k = 4$ and n is odd. For all values of k for which exploration is possible, we give an exploration algorithm. For all others, we prove an impossibility result.

Keywords: distributed computing, mobile robots, asynchronous, oblivious, exploration, line.

1 Introduction

1.1 Framework

This study is a part of an ongoing effort to understand computational and complexity issues arising in systems of autonomous mobile entities located in a universe $\mathcal{U}$. The entities, called *robots*, have storage and processing capabilities, are identical (execute the same protocol), and move in an asynchronous way in $\mathcal{U}$.

Depending on the nature of $\mathcal{U}$, there are two settings in which autonomous mobile entities are being investigated. The first setting, called sometimes the *continuous universe*, is when $\mathcal{U}$ is the two-dimensional plane (e.g., [1, 7, 8, 9, 14, 19]). The second setting, sometimes called the *graph world* or the *discrete universe*, is when $\mathcal{U}$ is a simple graph (e.g., [2, 3, 5, 10, 15, 16]).

[*]SITE, University of Ottawa. E-mail: flocchin@site.uottawa.ca

[†]LaBRI, CNRS & Université de Bordeaux. E-mail: david.ilcinkas@labri.fr.

[‡]Département d'informatique, Université du Québec en Outaouais. E-mail: pelc@uqo.ca.

[§]School of Computer Science, Carleton University, Ottawa. E-mail: santoro@scs.carleton.ca

[¶]Partially supported by NSERC Discovery grants. Andrzej Pelc is also partially supported by the Research Chair in Distributed Computing at the Université du Québec en Outaouais, and Paola Flocchini by a University Research Chair at the University of Ottawa.

[‖]Partially supported by the ANR project ALADDIN and the INRIA project CEPAGE.

In both settings, each robot operates in a *Look - Compute - Move* cycle. The robot observes the environment (Look), then, based on this observation it decides to stay idle or to move (Compute), and in the latter case it moves towards its destination (Move).

In the present paper we consider the graph setting, and focus on the problem of exploration with termination: robots start from different nodes of an anonymous graph and each node of the graph has to be visited by at least one robot. Moreover, after finite time, all robots must stay idle. Robots are *asynchronous*: the duration of each operation (Look, Compute, and Move) of each robot is finite but unbounded, and it is decided by an adversary. Robots are also *oblivious*: they do not remember past observations and in each cycle the computation is based only on the result of the Look operation in this cycle. During the Look operation, a robot acquires a snapshot of the graph: it perceives the whole graph and knows which nodes are empty and which are occupied by robots. Moreover, we assume *multiplicity detection*: the robots can perceive, during the Look operation, if there is one or more robots in a given location; however, they do not learn the number of robots located in a node. If more than one robot is located at a node, we say that there is a *tower* in this node. Based on the acquired snapshot, a robot deterministically decides if it should stay idle or move to one of the adjacent nodes. This is decided during the Compute operation. During the Move operation a robot executes its decision. Moves are instantaneous, and hence any robot performing a Look operation sees all other robots at nodes and not on edges. However, a robot $\mathcal{R}$ may perform a Look operation at some time t, perceiving robots at some nodes, then Compute a target neighbour at some time $t' > t$, and Move to this neighbour 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.

One final precision has to be added, concerning the decisions of robots made during the Compute operation. Every such decision is based on the snapshot obtained during the last Look operation. However, it may happen that both edges incident to a node v currently occupied by the deciding robot look identical in this snapshot, i.e., v lies on a symmetry axis of the configuration. In this case, if the robot decides to take one of these edges, it may take any of the two. We assume the worst-case decision in such cases, i.e., that the actual edge among the identically looking ones is chosen by an adversary.

We say that exploration of a n-node graph is possible with k robots, if there exists an algorithm which, starting from any initial configuration of the k robots without towers, allows the robots to explore the entire graph and brings all robots to a *terminal* configuration, that is a configuration in which they all remain idle. Obviously, if $n = k$, the exploration is already accomplished, hence we always assume that $k < n$.

Note that the difficulty of the problem is manifold. First, due to the anonymity of the snapshot, the adversary can prevent one robot from entering one of the two identically-looking parts of the graph. (For example a single robot can only explore one half of a n-node line when n is odd.) Second, if the robots are twice in the same global configuration, they will go in the same direction both times, because they are deterministic and do not remember anything from the past, not even the previous move. This typically prevents two robots

from being able to explore a line. Finally, the requirement that the robots must reach an idle terminal configuration adds to the difficulty of the task. Indeed, the robots are somehow forced to keep track of the progress despite having no persistent memory.

The above scenario has been used, for example, in [12, 13, 17, 18]. In [17, 18] the related problem of *gathering* was investigated on a ring: robots starting from different locations have to meet in one node. In [12, 13] the focus was on exploration, as in the present paper. In [12] the authors investigated the sizes of teams of robots capable of exploring trees: it was proved that there are n-node trees of maximum degree 4 where $\Omega(n)$ robots are necessary for exploration and that all trees of maximum degree 3 can be explored by $O(\frac{\log n}{\log\log n})$ robots. Moreover, the size $\Theta(\frac{\log n}{\log\log n})$ of the team is optimal for some such trees. This left open the analogous problem on trees of maximum degree 2, i.e., the lines, that are the focus of the present paper. In [13] the focus was on sizes of teams capable of exploring a n-node cycle. It was shown that the minimum number of robots that can explore a ring of size n is $O(\log n)$ and that $\Omega(\log n)$ robots are necessary for arbitrarily large n. Finally, in [6] the authors investigated a similar but stronger model with labeled edges and provided necessary and sufficient conditions for explorability of arbitrary graphs for $k = 3, 4$ robots and $k > 4$ odd robots.

In this paper we concentrate on the exploration of the line and solve the following problem: What are the sizes of teams of robots capable of exploring a line of n nodes.

1.2 Our results

We completely characterize sizes of teams of robots capable of exploring a n-node line.

Theorem 1 *Given n and $k < n$, exploration of the n-node line by k robots is possible if and only if $k = 3$, or $k \geq 5$, or $k = 4$ and n is odd.*

For all values of k and n for which exploration is not possible, we prove an impossibility result in Section 2. For all others, we give an exploration algorithm in Section 3.

1.3 Related Work

Algorithms for graph exploration by mobile entities (robots) have been intensly studied in recent literature. Several scenarios have been considered. In particular, exploration of anonymous graphs presents specific 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 [4, 5] 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 [5] the authors compare exploration power of one robot with a constant number of pebbles to that of two cooperating robots, and give an efficient exploration algorithm for the latter scenario. In [4] 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 [5], 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 [16], approximation algorithms are given for the collective exploration problem in arbitrary graphs. In [2, 3] the authors construct approximation algorithms for the collective exploration problem in weighted trees. On the other hand, in [15] 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.

The very weak assumption of asynchronous identical robots that cannot send any messages and that communicate with the environment only by observing it, has been first used in the case of robots moving freely in the plane (e.g., see [1, 7, 8, 9, 14, 19]). The robots were oblivious, i.e., it was assumed that they do not have any memory of past observations. Those robots operated in Look-Compute-Move cycles, similar to those described in our scenario. The differences were in the amount of synchrony assumed in the execution of the cycles. In [11, 19] 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 [7, 8, 9, 14] 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.

As mentioned previously, our scenario for the graph world has been previously used in [17, 18] in the context of gathering and in [12, 13] in the context of exploration.

2 Impossibility of Line Exploration

Lemma 2.1 *Every algorithm that guarantees exploration with termination must include a tower in each of its termination configurations.*

Proof: For any algorithm guaranteeing exploration with termination, an initial configuration cannot be terminal, because the robots would have not yet explored the whole line if started in this configuration (recall that $n > k$). Since all possible configurations without towers are potential initial configurations, the Lemma follows. □

Since with $k = 1$ agent no tower can be formed, we immediately have:

Lemma 2.2 *If $k = 1$, then the exploration of the line is impossible.*

Lemma 2.3 *If $k = 2$, then the exploration of the line is impossible, even if the robots agree on a common orientation of it.*

Proof: Assume, for the purpose of contradiction, that there exists an algorithm A enabling a team of two robots with a common sense of direction to perform exploration of some line. Let $n \geq 3$ be the number of nodes of this line. Suppose that the adversary controlling the asynchrony schedules the two robots alternatively. More precisely, the adversary lets a robot

execute its complete Look-Compute-Move cycle before allowing the other robot to execute its own execution cycle. By Lemma 2.1, any terminal configuration must contain a tower. Besides, by definition of the adversary controlling the asynchrony, if at some point a tower is formed, then either the two robots stay idle forever or the tower is broken after a single robot moves out of it. Consider now any initial configuration. Since Algorithm A is supposed to be correct, there exists a finite sequence of configuration successively reached by the system, starting from this initial configuration. From the previous remarks, the last configuration is terminal and contains a tower, and the penultimate configuration does not contain any tower. Let us choose this configuration as the new (legitimate) initial configuration. Since the robots are oblivious, they behave exactly the same, that is, they immediately form a tower and stop moving. In this case, the line is not explored (only two nodes are explored), which contradicts the correctness of Algorithm A. □

Lemma 2.4 *If $k = 4$ and n is even, then the exploration of the line is impossible.*

Proof: Assume, for the purpose of contradiction, that there exists an algorithm A enabling a team of four robots to perform exploration of some line. Let $n \geq 6$, n even, be the number of nodes of this line. In particular, the algorithm A behaves correctly when the initial configuration is symmetric.

Suppose that the adversary controlling the asynchrony schedules two robots that are symmetric to each other simultaneously but two non-symmetric robots alternatively. More precisely, the adversary lets a pair of symmetric robots execute its complete Look-Compute-Move cycle before allowing the other pair to execute its own execution cycle.

Let Algorithm A' be the algorithm for two robots with a common sense of direction in a line with $n/2 \geq 3$ nodes defined as follows. Given a configuration of the two robots on the half line, consider the associated symmetric configuration of four robots in the n-node line where two half lines are glued together by their common extremity (their "left" one for example). A robot in the half-line move according to Algorithm A' exactly as the corresponding pair of symmetric robots move in the full line according to Algorithm A. Since Algorithm A explores the line and stops, so does Algorithm A'. This contradicts Lemma 2.3, and this concludes the proof. □

3 Line Exploration Algorithms

We first present an algorithm to explore a line $L_n = [v_1 \dots v_n]$ when $k = 3$ or $k \geq 5$. We then consider the case $k = 4$ and n odd.

Case $k = 3$ 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) (*Setup Phase*). When the Setup phase terminates, the second robot from the border of each group moves on the first creating a tower. This indicates the beginning of the *Exploration Phase*: 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).

For any possible configuration, we identify, in Table 1 for k odd and in Table 2 for k even, a set of *players* and corresponding *destinations*. A black triangle denotes a tower. A white, resp. black, circle represents a node hosting no robots, resp. exactly one robot.

consecutive: $(1^k\ 0^{n-k})$ *player*: v_2, *destination*: v_1	
consecutive-with-tower: $(\mathrm{T}\ 0\ 1^{k-2}\ 0^{n-k})$ *player*: v_k, *destination*: v_{k+1}	
exploring: $(\mathrm{T}\ 0\ 1^{k-3}\ 0^{j}\ 1\ 0^{n-j-k})$, with $0 < j < n-k$ *player*: v_{k+j}, *destination*: v_{k+j+1}	
final: $(\mathrm{T}\ 0\ 1^{k-3}\ 0^{n-k}\ 1)$ these configurations are terminal (no players)	
other: different from previous *players* and *destinations* given by Algorithm FIND-PLAYERS-AND-DESTINATIONS-ODD	

FIND-PLAYERS-AND-DESTINATIONS-ODD (for robot r)

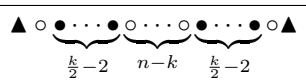
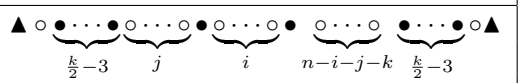
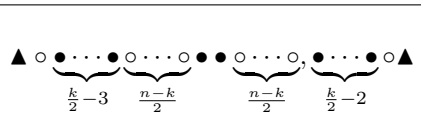
Let h_1 be the number of robots in $[v_1 \dots v_{\lfloor \frac{n}{2} \rfloor}]$ and let h_2 be the number of robots in $[v_{\lceil \frac{n}{2} \rceil + 1} \dots v_n]$.
If $h_1 = h_2$ (thus n is odd)
  player $= v_{\frac{n+1}{2}}$
  destination = any neighbour
If $h_1 \neq h_2$
  let *left-right* be the direction from $min\{h_1, h_2\}$ to $max\{h_1, h_2\}$
  If my right neighbour y is empty
    I am a *player* and my *destination* is y

Table 1: Possible configurations when k is odd.

The exploration algorithm (which contains the rules describing the Compute actions in the robot's cycle) is very simple, once players and destinations are defined. An example of two executions of the algorithm from the same initial configuration is given in Figure 1.

Algorithm LINE EXPLORATION

If I am a *player*
  move one step towards my *destination*

2-consecutive: $(1^{\frac{k}{2}}\ 0^{n-k}\ 1^{\frac{k}{2}})$ *players*: v_2, v_{n-1}, *destinations*: v_1, v_n	
2-consecutive-with-towers: $(\mathrm{T}\ 0\ 1^{\frac{k}{2}-2}\ 0^{n-k}\ 1^{\frac{k}{2}-2}\ \mathrm{T})$ *players*: $v_{\frac{k}{2}}, v_{\frac{k}{2}+n-k+1}$, *destinations*: $v_{\frac{k}{2}+1}, v_{\frac{k}{2}+n-k}$	
2-exploring: $(\mathrm{T}\ 0\ 1^{\frac{k}{2}-3}\ 0^{j}\ 1\ 0^{i}\ 1\ 0^{n-i-j-k}\ 1^{\frac{k}{2}-3}\ 0\ \mathrm{T})$ with $0 < j \leq n-i-j-k$ *player*: $v_{\frac{k}{2}+j}$ *destination*: $v_{\frac{k}{2}+j+1}$	
final: $(\mathrm{T}\ 0\ 1^{\frac{k}{2}-3}\ 0^{\frac{n-k}{2}}\ 1^2\ 0^{\frac{n-k}{2}}\ 1^{\frac{k}{2}-3}\ 0\ \mathrm{T})$ $(\mathrm{T}\ 0\ 1^{\frac{k}{2}-3}\ 0^{\frac{n-k+1}{2}}\ \mathrm{T}\ 0^{\frac{n-k+1}{2}}\ 1^{\frac{k}{2}-3}\ 0\ \mathrm{T})$ these configurations are terminal (no players)	
other: different from previous *players* and *destinations* given by Algorithm FIND-PLAYERS-AND-DESTINATIONS-EVEN	

FIND-PLAYERS-AND-DESTINATIONS-EVEN (for robot r)

Let h_1 and h_2 be the numbers of robots on each side of my position.
Let *left-right* be the direction from $min\{h_1, h_2\}$ to $max\{h_1, h_2\}$.
If my left neighbour y is empty
  I am a *player* and my *destination* is y

Table 2: Possible configurations when k is even.

Correctness

Lemma 3.1 *In the execution of* LINE EXPLORATION *with* $k = 3$ *or* $k \geq 5$*, a* consecutive *or* 2-consecutive *configuration is reached in finite time.*

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 divide themselves evenly among the two halves of the line and follow the same procedure, each half towards their respective border, clearly reaching a *2-consecutive* configuration. □

Lemma 3.2 *In the execution of* LINE EXPLORATION *with* $k = 3$ *or* $k \geq 5$*, from a* consecutive *or* 2-consecutive *configuration, the line is correctly explored.*

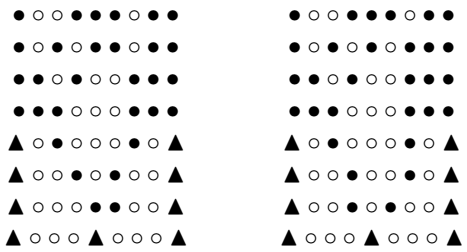


Figure 1: Two possible executions from the same initial configuration, with $k = 6$ and $n = 9$.

Proof: In each consecutive block a tower is unambiguously formed. From now on, all robots know that the exploration is started. If the configuration is *consecutive-with-tower* (i.e. k is odd), then there is a unique robot that can move ($k \geq 3$ guarantees that there is one). By the algorithm, the robot 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 the configuration is *2-consecutive-with-towers* (i.e., k is even), then the two most central robots move progressively towards the center of the line, possibly waiting for each other if one progresses faster than the other, until they form a tower in the middle of the line (if n is odd), or they stop next to each other (if n is even). Two such robots are guaranteed to exist because when k is even, $k \geq 6$. Again such configurations are unambiguously recognized as final because in no other moment there have been two towers on the two extremities and a pair of robots (or a tower) in the center of the line. □

Case $k = 4$ and n is odd. In this case, the following algorithm explores the line. First the 4 robots place themselves so that two are on the neighbours of the central node and two on the extremities of the line. Now the two robots closer to the central node form a tower there. At this point the other two robots move towards the central node.

References

[1] N. Agmon, D. Peleg, Fault-tolerant gathering algorithms for autonomous mobile robots, SIAM J. Comput. 36(1): 56-82 (2006).

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

[3] 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.

[4] 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.

[5] 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 Comp. Science (FOCS 1994), 75-85.

[6] J. Chalopin, P. Flocchini, B. Mans, N. Santoro. Network exploration by silent and oblivious robots, 36th Int. Work. on Graph Theoretic Concepts in Computer Science (WG 2010), LNCS 6410, 208-219.

[7] M. Cieliebak, P. Flocchini, G. Prencipe, N. Santoro, Solving the robots gathering problem, Proc. 30th Int. Col. Automata, Languages and Progr. (ICALP 2003), LNCS 2719, 1181-1196.

[8] R. Cohen, D. Peleg, Convergence properties of the gravitational algorithm in asynchronous robot systems. SIAM J. Comput. 34(6), (2005),1516-1528,

[9] R. Cohen, D. Peleg, Robot convergence via center-of-gravity algorithms, Proc. 11th Int. Col. on Structural Information and Comm. Complexity (SIROCCO 2004), LNCS 3104, 79-88.

[10] S. Das, P. Flocchini, S. Kutten, A. Nayak, N. Santoro, Map construction of unknown graphs by multiple agents, Theoretical Computer Science 385 (2007), 34-48.

[11] X. Défago and S. Souissi. Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity. Theor. Comput. Sci. 396(1-3) (2008), 97-112.

[12] P. Flocchini, D. Ilcinkas, A. Pelc, N. Santoro, Remembering without memory: tree exploration by asynchronous oblivious robots, Theoretical Computer Science 411 (2010), 1583-1598.

[13] P. Flocchini, D. Ilcinkas, A. Pelc, N. Santoro, Computing without communicating: Ring exploration by asynchronous oblivious robots, Proc. 11th International Conference on Principles of Distributed Systems (OPODIS 2007), LNCS 4878, 105-118.

[14] P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer, Arbitrary pattern formation by asynchronous, anonymous, oblivious robots, Theoretical Computer Science 407 (2008), 412-447.

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

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

[17] R. Klasing, A. Kosowski, A. Navarra, Taking advantage of symmetries: gathering of many asynchronous oblivious robots on a ring, Theoretical Computer Science, 411(2010), 3235–3246.

[18] R. Klasing, E. Markou, A. Pelc, Gathering asynchronous oblivious mobile robots in a ring, Theoretical Computer Science, 390 (2008), 27–39.

[19] I. Suzuki, M. Yamashita, Distributed anonymous mobile robots: formation of geometric patterns. SIAM J. Comput. 28(4): 1347-1363 (1999).