Electing a leader among anonymous mobile agents in anonymous networks with sense-of-direction[*]

Lali Barrière[†] Paola Flocchini[‡] Pierre Fraigniaud[§] Nicola Santoro[¶]

Abstract

We consider a collection of r anonymous asynchronous mobile agents dispersed on an arbitrary anonymous network of size n. Neither r nor n are known a priori by the agents. We examine the problem of electing a leader among those agents and study the conditions for its solvability. We show that, without sense of direction, the problem is unsolvable, even if restricted to instances for which $\gcd(r,n) = 1$. We also show that, with sense of direction, the problem remains unsolvable, but it becomes solvable if restricted to instances for which $\gcd(r,n) = 1$. Since sense of direction can be given to any m-edge graph in $O(m)$ time, our result shows that one can easily label the edges of an anonymous network in order to significantly improve its computational power.

Keywords: Anonymous Mobile Agents, Anonymous Networks, Sense of Direction, Election.

Submission to SIROCCO 2003
Corresponding author: Lali Barrière, *lali@mat.upc.es*

[*]This research was carried out when the first and third authors were visiting Carleton University and the University of Ottawa. This work has been partially supported by: the Natural Sciences and Engineering Research Council (NSERC), Nortel Networks, and NATO.

[†]Dept. de Matemàtica Aplicada IV, *Universitat Politècnica de Catalunya*, `lali@mat.upc.es`.

[‡]School of Information Technology and Engineering, *University of Ottawa*, `flocchin@site.uottawa.ca`.

[§]CNRS, Laboratoire de Recherche en Informatique, *Univ. Paris-Sud*, `pierre@lri.fr`.

[¶]School of Computer Science, *Carleton University*, `santoro@scs.carleton.ca`.

1 Introduction

We are interested in the computational issues arising in networked environments which support autonomous asynchronous mobile agents. At an abstract level, these environments, which we shall call *distributed mobile systems*, can be described as a collection $\mathcal{E}$ of autonomous mobile entities located in a graph G. The entities have the same processing capabilities, execute the same protocol, and move in G from node to neighboring node. Depending on the context, they are sometime called robots or agents. The research concern is on determining what tasks can be performed by such entities, under what conditions, and at what cost. In particular, a central question is to determine what minimal hypotheses allow a given problem $\mathcal{P}$ to be solved. Example of this type of investigations are the studies on topology-reconstruction: $\mathcal{E}$ is typically a single entity, sometimes two, G is unknown to the entity, and $\mathcal{P}$ is the construction of a map of the graph (e.g., see [2, 6, 7]). Other examples are graph-exploration [8, 9, 12], wake-up [1, 16], black-hole search [10, 11], searching for a mobile intruder [18], etc. In this paper, we focus on a fundamental problem in distributed mobile computing: *election*, that is the process by which a group of autonomous asynchronous mobile entities initially in the same state and scattered in G selects one of them as a *leader*.

1.1 Statement of the problem

We focus on the computational problem of electing a leader in *fully anonymous* systems, that is, when both the agents and the nodes are unlabeled, but when edges incident to each node of the network are given distinct labels. (In absence of such edge-labeling, an agent would be unable to explore even the star with three leaves since, coming back from the second explored leaf, it would be unable to recognize the already explored edge from the unexplored one.) The agents have computing capabilities and bounded storage, execute the same protocol, and can move from node to neighboring node in the network. The agents are asynchronous, in the sense that every action they perform, like computing, moving, etc., takes a finite but otherwise unpredictable amount of time. Initially, the agents are placed at r distinct nodes, called *homebases*. Each node is provided with a *whiteboard*, i.e., a local storage where agents can write and read (and erase) information. This capability is motivated by the ability of software agents to let messages in a network. Access to a whiteboard is done in mutual exclusion.

Initially, all agents have a predefined state variable set to *available*. The election problem consists in having the agents unanimously select one of them which will set its state variable to *leader*, while all the others will set their state variable to *follower*. The behavior of an agent will be the following. The agent will compute based on (1) its current state, (2) the content of the witheboard of the node currently visited, and (3) the label of the edge from which it arrived. The computation is indivisible and, upon completion, the agent will change its state and then depart through an exit port determined during the computation. A *null* port can be added to describe a decision by the agent not to move.

We are interested in *generic* solutions, that is, solutions which work independently of the structure of G and of the number of agents. Thus, we assume that the network G, its size n, as well as the number r of agents are not known a priori. In this context, the relationship between n and r plays an important role, and the feasibility of the election is directly related to $\gcd(r, n)$. If $\gcd(r, n) = d > 1$, and if the network is symmetric, then one can identify several inputs of the election problem for which a set of d agents will perpetually perform the same movements, and remain in identical states. One of our first results shows that, in fully anonymous systems with arbitrary edge labeling, the election problem remains unsolvable even if $\gcd(r,n) = 1$. That is no deterministic solution exists which allows the agents to always correctly terminate in finite time for every input such that $\gcd(r,n) = 1$. We therefore slightly strengthen the characteristics of the system, by assuming the existence of a *sense of direction*.

1.2 Sense of direction

Roughly speaking, sense of direction is an edge-labeling which provides to graphs similar properties as the usual notion of North, South, East, West in plane maps (see [13]). More precisely, let $G = (V,E)$ be a simple undirected graph. Let $E(u)$ denote the set of edges incident to node $u \in V$, and let $\lambda_u : E(u) \to \mathcal{L}$ be an injective function which associates to each incident edge a distinct label from the set of labels $\mathcal{L}$. Note that for each edge $e = \{u,v\}$ there are two associated labels, $\lambda_u(e)$ and $\lambda_v(e)$, which are possibly different. The set $\lambda = \{\lambda_u : u \in V\}$ constitutes the labeling of G, and by (G,λ) we shall denote the corresponding edge-labeled graph. Let $P[u]$ denote the set of all the non empty walks starting from $u \in V$. Similarly, let $P[u,v]$ denote the set of walks starting from $u \in V$ and ending in $v \in V$. Let $\Lambda_u : P[u] \to \mathcal{L}^+$ and $\Lambda = \{\Lambda_u : u \in V\}$ denote the extension of λ_u and λ, respectively, from edges to walks ($\mathcal{L}^+$ is $\mathcal{L}^*$ but non including the empty string).

Given (G,λ), a *coding function* $\mathbf{c}$ for λ is any function with domain $\mathcal{L}^+$ such that walks originating from the same node are mapped to the same value if and only if they end in the same node. More precisely, $\forall u,v,w \in V$, $\forall \pi_1 \in P[u,v]$, $\pi_2 \in P[u,w]$, $\mathbf{c}(\Lambda_u(\pi_1)) = \mathbf{c}(\Lambda_u(\pi_2))$ if and only if $v = w$. A *decoding function* $\mathbf{d}$ for $\mathbf{c}$ is a function which, given the label $\lambda_u(u,v)$ of an edge (u,v) and the coding of a walk π from v to another node w, returns the coding of the overall walk from u to w. More precisely, $\forall\{u,v\} \in E$, $\forall \alpha = \Lambda_v(\pi)$, $\mathbf{d}(\lambda_u(\{u,v\}), \mathbf{c}(\alpha)) = \mathbf{c}(\lambda_u(\{u,v\}) \circ \alpha)$, where $\circ$ denotes concatenation of strings. Given a coding function $\mathbf{c}$ of (G,λ) and a decoding function $\mathbf{d}$ for $\mathbf{c}$, the couple $(\mathbf{c},\mathbf{d})$ is called a sense of direction for (G,λ).

An example of a common sense of direction is the dimensional labeling in the hypercube, where an edge is labeled according to the corresponding dimension.

Remark. Any graph can be endowed with a sense of direction. Therefore, sense of direction restricts the class of labelings, not the class of graphs.

For example, consider an arbitrary graph $G = (V,E)$, $V = \{v_0, \ldots, v_{n-1}\}$, where the edge $\{v_i, v_j\}$ is labeled at v_i by the label $(j-i) \bmod n$. With this labeling there is a simple coding function: two paths from the same node terminate in the same node iff the sum modulo n of the corresponding labels is the same, that is $\forall u,v \in V$, $\forall \pi \in P[u,v]$, if $[l_0, \ldots, l_k] = \Lambda_u(\pi)$ then $\mathbf{c}(\Lambda_u(\pi)) = \sum_{i=0}^{k} l_i \bmod n$. The decoding function is defined as follows: $\forall\{u,v\} \in E$, $\forall \pi \in P[v]$, $\mathbf{d}(\lambda_u(\{u,v\}), \mathbf{c}(\alpha)) = \lambda_u(\{u,v\}) + \mathbf{c}(\alpha) \bmod n$, where $\alpha = \Lambda_v(\pi)$. It is easy to verify that $\lambda_u(\{u,v\}) + \mathbf{c}(\alpha) = \mathbf{c}(\lambda_u(\{u,v\}) \circ \alpha)$. This sense of direction is called *chordal* and is one of many that can be constructed in an arbitrary graph.

1.3 Our results

We already mentioned our first result about the unsolvability of election with arbitrary labelings, i.e., in fully anonymous systems with arbitrary edge labelings, the election problem remains unsolvable even restricted to the class of inputs for which $\gcd(r,n) = 1$.

We then consider fully anonymous systems where there is however sense of direction. We first show another negative result: the problem is unsolvable for arbitrary instances. On the other hand, we prove that, if $\gcd(r,n) = 1$, then the election problem is solvable, and the result holds for any sense of direction. In other words, we show that sense of direction overcomes anonymity if $\gcd(r, n) = 1$. This is the first evidence that sense of direction has a positive impact also in distributed *mobile* computing. The proof of our positive result is constructive. The number of movements of the agents is at most $O(rn)$.

2 Unsolvability of election with arbitrary labelings

In this section we will prove that, with arbitrary labelings (thus, without sense of direction), the election problem is unsovable, even if $\gcd(n, r) = 1$.

Theorem 2.1 *In a fully anonymous system with arbitrary labelings, the election problem is deterministically unsolvable, even if restricted to the class of inputs for which* $\gcd(n, r) = 1$.

Proof. Assume, for the purpose of contradiction, that there exists a correct election protocol P. We will consider synchronous executions of P simultaneously started by all agents, i.e., we will assume (1) all computations to be instantaneous, (2) movements to require a unit of time, and (3) agents to start simultaneously. We will consider oriented rings (i.e., consistently labeled with "left" and "right"). Note that orientation does not imply sense of direction. In fact, without knowledge of n, an oriented ring does not have any consistent coding and, thus, there is no sense of direction (cf. [13]). Further, we will assume that a node is visited by only one agent per time unit. Let

$s(a, t)$ = state of agent a at time t;
$p(a, t)$ = location of agent a at time t;
$w(v, t)$ = content of the whiteboard of node v at time t;
$e(a, t)$ = label of the entry port by which a arrives at $p(a, t)$ at time t.

Then, the system will evolve according to the following three facts.

Fact 1. The content of a whiteboard can change only when the corresponding node is visited by an agent. The new content is a function f solely of the current information, and the state and the entry port of the visiting agent (if any). In other words, if no agent is at v at time $t + 1$, $w(v, t + 1) = w(v, t)$; otherwise, $w(v, t + 1) = f(w(v, t), s(a, t), e(a, t))$ where $p(a, t) = v$.

Fact 2. The choice by an agent of a link to traverse is a function g solely of the current state of the agent, the label of the incoming edge, and the content of the whiteboard. In other words, if $p(a, t) = v$, then $\ell(v, t + 1) = g(w(v, t), s(a, t), e(a, t))$.

Fact 3. The new state of an agent is a function γ solely of the content of the whiteboard, the entry port number, and the current state. In other words, if $p(a, t) = v$, $s(a, t + 1) = \gamma(w(v, t), s(a, t), e(a, t))$.

Consider now the system $\mathcal{A}$ composed of an oriented ring of three nodes (y_0, y_1, y_2), with a single agent located in y_0 (see Figure 1). Consider a synchronous execution of P by the agent. After a finite number of moves, the execution must terminate with the agent becoming the leader. Let $T(\mathcal{A})$ be the time elapsed in this execution. Consider then a system $\mathcal{B}$ composed of an oriented ring of $n = 3q$ nodes, $(x_0, x_1, \ldots, x_{n-1})$, with $q \geq \frac{4}{3}(T(\mathcal{A}) + 1)$, and an agent placed in each of locations x_{3j}, $0 \leq j \leq q - 1$ (see Figure 2). We have $\gcd(n, r) = q \neq 1$. Clearly the initial system symmetry in $\mathcal{B}$ is total: the views [19] of all agents are identical at any distance. Furthermore this view is undistinguishable from that of the only agent in $\mathcal{A}$. In a traditional distributed setting, i.e., with *static* entities, this would be enough to guarantee that the relationship between the two systems is time-invariant under a synchronous scheduler (e.g., see [19]), and hence to yield the desired contradiction. We prove that, even in our mobile setting, the initial correspondence between the two systems can be preserved through time. Consider a synchronous execution of P in $\mathcal{B}$ simultaneously started by all agents. This execution has several important properties. Denote by a_j the agent with homebase x_j, and, for simplicity of notation, denote $x(t) = p(a_0, t)$ and $x(t) + h = p(a_h, t)$ the location of a_0 and of a_h at time t, respectively.

Lemma 2.1 *At any time instant $t \geq 0$*
(1) for each $i = 0, 1, 2$ the contents $w(x_{i+3j}, t)$ are identical for all j, $0 \leq j \leq q-1$,
(2) the states $s(a_{3j}, t)$, are identical for all j, $0 \leq j \leq q$,
(3) the entry ports $e(a_{3j}, t)$, are identical for all j, $0 \leq j \leq q$, and
(4) $p(a_{3j}, t) = x(t) + 3j$, for all j, $1 \leq j \leq q-1$, where operations are modulo n.

Proof. The lemma trivially holds at time $t = 0$. Assume that it holds at time $t \geq 0$. Thus, each agent is still at distance three from its neighboring agents on each side, the whiteboards of the nodes where the agents are currently located have the same content, and the state of all agents is the same. Consider first the port through which agent a_0 will exit the current node. By definition, $\ell(x(t), t+1) = g(w(x(t), t), s(a_0, t), e(a_0, t))$. Since, by induction hypothesis, $w(x(t), t) = w(x(t)+3j, t)$, $e(a_0, t) = e(a_{3j}, t)$, and $s(a_0, t) = s(a_{3j}, t)$, then $\ell(x(t), t+1) = \ell(x(t)+3j, t+1)$, for $0 \leq j \leq q-1$. In other words, the agents will all choose the same edge label l (say "left"), and exit their current location through the port with label l. Thus, since the ring is oriented, they will all enter the new node at time $t+1$ from the port labeled "right", i.e., $e(x(t), t+1) = e(x(t)+3j, t+1)$, for $0 \leq j \leq q-1$ proving thesis (3) of the lemma. Furthermore, $p(a_{3j}, t+1) = x(t+1) + 3j$, for $1 \leq j \leq q-1$, proving thesis (4).

Let us now consider for $0 \leq j \leq q-1$ and $0 \leq i \leq 3$, the whiteboards of nodes x_{i+3j} at time t. Since nodes $x(t) + 3j + 1$ will not be visited by an agent at time t, their whiteboard will be unchanged. That is, $w(x(t) + 3j + 1, t + 1) = w(x(t) + 3j + 1, t)$ and, by induction hypothesis, their contents will remain equal. A similar situation occurs for the nodes $x(t+1) + 3j + 2$. The node $x(t)$ is visited by agent a_0 at time t, so we have that $w(x(t), t+1) = f(w(x(t), t), s(a_0, t), e(a_0, t))$. But, by induction hypothesis, $p(a_{3j}, t) = x(t) + 3j$, $w(x(t), t) = w(x(t) + 3j, t)$, $e(a_0, t) = e(a_{3j}, t)$, and $s(a_0, t) = s(a_{3j}, t)$. Thus, $w(x(t), t+1) = w(x(t)+3j, t+1)$. This completes the proof of thesis (1).

Let us finally consider the state of agent a_0. By definition, $s(a_0, t+1) = \gamma(w(x(t), t), s(a_0, t), e(a_0, t))$. But, by induction hypothesis, $p(a_{3j}, t) = x(t) + 3j$, $w(x(t), t) = w(x(t) + 3j, t)$, $e(a_0, t) = e(a_{3j}, t)$, and $s(a_0, t) = s(a_{3j}, t)$, for $0 \leq j \leq q-1$. Thus, also thesis (2) holds. ■

Let us now compare the execution of P in $\mathcal{A}$ with the execution in $\mathcal{B}$. Let a be the sole agent in $\mathcal{A}$.

Lemma 2.2 *At any time instant t,*
(1) $w(x_{i+3j}, t) = w(y_i, t)$ for all i, j, $0 \leq j \leq q-1$ and $0 \leq i \leq 3$,
(2) $s(a_0, t) = s(a, t)$,
(3) $e(a_0, t) = e(a, t)$, and
(4) if $p(a_0, t) = u$ and $p(a, t) = v$, then $\ell(u, t+1) = \ell(v, t+1)$.

Proof. Thesis (1) (2) and (3) clearly hold for $t = 0$. Assume that they hold up to $t \geq 0$. Again, we denote $x(t) = p(a_0, t)$, $x(t) + h = p(a_h, t)$, and $y(t) = p(a, t)$. By definition of ℓ and induction hypothesis, we have $\ell(x(t), t+1) = g(w(x(t), t), s(a_0, t), e(a_0, t)) = g(w(y(t), t), s(a, t), e(a, t)) = \ell(y(t), t+1)$; hence (4) holds. Because of orientation, $e(a_0, t+1) = e(a, t+1)$, and (3) holds. Now, by definition of w, $w(x(t), t+1) = f(w(x(t), t), s(a_0, t), e(a_0, t))$. By induction hypothesis,

$w(x(t),t) = w(y(t),t)$, $e(a_0,t) = e(a,t)$ and $s(a_0,t) = s(a,t)$. Thus, $w(x(t),t+1) = w(y(t),t+1)$. Lemma 2.1 implies that $w(x(t)+3j,t+1) = w(y(t),t+1)$. Since all other nodes in both systems are not visited by any agent at time t, the content of their whiteboards there stays unchanged. Thus, by induction hypothesis, thesis (1) holds. Finally, the state of a_0 at time $t+1$ satisfies $s(a_0,t+1) = \gamma(w(x(t),t), s(a_0,t), e(a_0,t))$. By induction hypothesis, $e(a_0,t) = e(a,t)$, $s(a_0,t) = s(a,t)$, and $w(x(t),t) = w(y(t),t)$; thus, $s(a_0,t+1) = s(a,t+1)$ and (2) holds. ■

Without loss of generality, let $k = T(\mathcal{A})$ be a multiple of 3, i.e., $k = 3d$ (otherwise, in the following discussion, use $k' = 3\lceil k/3 \rceil$ instead of k). Consider now a system $\mathcal{C}$ composed of an oriented ring $(z_0, z_1, \ldots, z_{m-1})$ of size $m = 4k+2$, where there is an agent in node z_0 and in each of locations z_{3j} and z_{-3j}, $1 \leq j \leq 2d$ with all operations on the indices are modulo m (see Figure 3). Thus $r = 4d+1$, and clearly $\gcd(m,r) = 1$. Denote by b_j the agent with homebase z_j. Start a synchronous simultaneous execution of P by all agents in $\mathcal{C}$. Let us now compare the first k steps of the execution of P in $\mathcal{C}$ with the ones in $\mathcal{B}$.

Lemma 2.3 *At any time instant $t \leq k$,*
(1) for each $i = 0,1,2$, $w(z_{\pm(i+3j)},t) = w(x_i,t)$, $0 \leq j \leq d$, and
(2) $s(b_0,t) = s(b_3,t) = s(b_{-3},t) = s(a_0,t)$.

Proof. Initially all witheboards of nodes which are not homebase are identical, and so are the witheboards of the homebases. Similarly, all agents are in the same initial state. In particular, there is no possibility, initially, for the agents to distinguish between system $\mathcal{B}$ and system $\mathcal{C}$. The agents b_{2k}, b_{-2k} have a different 2-neighborhood from all other agents. One of these two agents, say b_{2k}, will be the first to have a different state from all others agents. This will happen either at time $t = 2$, in which case b_{2k} will be at z_{-2k}, or at time $t = 3$, in which case b_{2k} will be at z_{2k+1}. In either case, the distance between the location of b_{2k} when it becomes different and the closest of x_{k+1} and $x_{-(k+1)}$ is at least k. This implies that, at time $t = k$, at most the segment $\langle z_{k+1}, \ldots, z_{3k+2} \rangle$ will be affected by the change. Thus, for the first k steps, the segment $Z = \langle z_{-k}, z_{-k+1} \ldots, z_0 \ldots z_k \rangle$ in system $\mathcal{C}$ will be undistinguishable from the segment $X = \langle x_{-k}, x_{-k+1}, \ldots, x_0 \ldots x_k \rangle$ in system $\mathcal{B}$. Furthermore, all the agents that, in all this time, do not leave Z will all have the same state at each time step, which will equal to that of a_0 at that time. Since, for $t \leq k$, the three agents b_0, b_3 and b_{-3} are always within Z, the lemma follows. ■

From Lemma 2.2 it follows that at time $T(\mathcal{A})$, agent a_0 becomes leader in $\mathcal{B}$. Then, by Lemma 2.1, all agents will become leader in $\mathcal{B}$ at time $T(\mathcal{A})$, contradicting the correctness of protocol P for the case $\gcd(n,r) \neq 1$. Also, by Lemma 2.3, at time $T(\mathcal{A})$, agents b_0, b_3 and b_{-3} become leader in $\mathcal{C}$, contradicting the correctness of P for the case $\gcd(n,r) = 1$. This completes the proof of Theorem 2.1. ■

3 Election with sense of direction

In this section, we consider a fully anonymous systems, where both nodes and agents are anonymous, where however there is a sense of direction $(\mathbf{c}, \mathbf{d})$ available to the agents.

Theorem 3.1 *In a fully anonymous system with sense of direction, the election problem is deterministically unsolvable.*

Proof. Consider a ring network $(x_0, x_1, \ldots, x_{n-1})$ with the classical "left/right" sense of direction (i.e., the ring is oriented and n is known to the agents). Let $\gcd(r, n) = d \neq 1$. Consider an initial equidistant, i.e., separated by $d-1$ empty nodes, placement of the agents in the ring, and a synchronous scheduler. The initial system symmetry is total: the system is homonymous and the views of all agents are identical at any distance. The synchronous scheduler will maintain the symmetry at every step: at each time unit, the agents will be in the same state, react to the same event, and perform the same action, reading and writing the same information on the whiteboard, and selecting the same port label for the next move. If an agent becomes leader, they all simultaneously will. ∎

Note that systems $\mathcal{A}$ and $\mathcal{B}$, used in the impossibility result Theorem 2.1, are no longer undistinguishable by the agents. Indeed, because of sense of direction, an agent reaching a homebase can determine whether it is its own homebase or not (this will appear clear in the next section). In fact, we prove the following.

Theorem 3.2 *In a fully anonymous system with sense of direction, the election problem is deterministically solvable if* $\gcd(n, r) = 1$.

The description of an election protocol for every instance with $\gcd(n, r) = 1$ requires some preliminaries. We first prove that, even in a totally anonymous system, an agent can locally (i.e., privately) assign a unique "name" to itself and to the other agents, as well as to the nodes of the graph. However, since all agents are behaviorally identical and start with the same initial values, there is no guarantee that such a name would be unique. In fact it is possible that they all choose the same name for themselves, creating an *homonymous* universe. We now present a mechanism, called DNM for *dynamic name mutation*, which, exploiting the presence of sense of direction, allows us to operate in spite of these limitations, including homonimity.

In our mechanism, initially every agent chooses its private name based on the labels of the edges incident to its homebase. The private name is then modified whenever the agent moves on the graph. The name will be always relative to the current position of the agent. The main difficulty is to modify the names in such a way that, at any location v, two names will be different if and only if they refer to different agents. This will ensure that messages written on v's whiteboard by different agents will have different signatures. An other related difficulty is to ensure that an agent is capable of recognizing, as its own, any message it has written in previous visits. These difficulties are overcome by the use of the existing sense of direction $(\mathbf{c}, \mathbf{d})$ by the mobile agents.

Strategy DNM
(1) To determine its initial name, an agent a with homebase $p(a) = u$, chooses an arbitrary neighboring node $v \in E(u)$ and determines the label $\lambda_v(\{u, v\})$ (e.g., by moving to v and coming back). Then, its name is $Myname := \mathbf{c}(\lambda_u(\{u, v\}) \circ \lambda_v(\{u, v\}))$.
(2) When an agent with name $Myname$ at node u moves to the neighboring node v, it modifies its name as follows: $Myname := \mathbf{d}(\lambda_v(\{v, u\}), Myname)$.

Note that u and v are used for notation purposes only and are not available to the agents. Further note that if G is regular of degree d and λ is minimal (i.e., it uses only d labels), all agents will choose the same name for themselves.

Lemma 3.1 *At any location v two names will be different if and only if they refer to different agents.*

Proof. To prove the lemma, we first prove the following claim: The name of a at v is $\mathbf{c}(\alpha)$ where α is the sequence of labels corresponding to an arbitrary path from v to the homebase h_a of agent a. We prove that claim by induction. Consider the walk $\pi = (v_0 = h_a, v_1, v_2 \ldots, v_k = v)$ traversed by a to reach a node v_k. The property is true at node h_a. Assume it holds at v_i and that the name is M. Consider now v_{i+1}. The name of a at v_{i+1} is $\mathbf{d}(\lambda_{v_{i+1}}(v_{i+1}, v_i), M) = \mathbf{d}(\lambda_{v_{i+1}}(v_{i+1}, v_i), \mathbf{c}(\delta))$, where δ is an arbitrary path from v_i to h_a. In particular such a path could be $(v_i, \ldots v_1, h_a)$, then it follows that the name of a at v_{i+1} is $\mathbf{c}(\Lambda(v_{i+1}, v_i, \ldots v_1, h_a))$, which, by definition of coding function, is equal to $\mathbf{c}(\alpha)$, where α is the sequence of labels corresponding to any walk from v_{i+1} to h_a. This completes the proof of the claim.

In the remaining part of the proof, we show that if an agent a arriving at a node v "sees" two names, it can tell whether they refer to the same agent or not (and whether they refer to itself). In other words, we have to prove that (i) an agent at v always has the same name regardless the walk it traversed to arrive there, (ii) different agents arriving at node v have different names. Point (i) follows from the claim. Let us consider point (ii). Let a_1 and a_2 be two different agents with respective homebases h_{a_1} and h_{a_2}. At node v the name of a_1 is $\mathbf{c}(\alpha)$ and the name of a_2 is $\mathbf{c}(\beta)$ where α corresponds to an arbitrary path from v to h_{a_1} and β corresponds to an arbitrary path from v to h_{a_2}. By definition of coding function $\mathbf{c}(\alpha) \neq \mathbf{c}(\beta)$ because $h_{a_1} \neq h_{a_2}$. ■

Because of Lemma 3.1, we are guaranteed that whenever an agent writes some information on a whiteboard and signs it, it will be able in subsequent visits to the same node to identify that information as its own. Moreover, it will correctly recognize signatures written by other agents as not its own. There is an additional extremely useful consequence. Let an agent a find a signature of another agent b in the whiteboard of a visited site. If the agent carries with itself that signature and applies to it the dynamic name mutation, a will be able to detect if any of the information written on the whiteboard of another node has been written by b.

We now present a protocol which will elect one of the agents as the leader, provided $\gcd(r, n) = 1$. It is based on roughly the same techniques as in [4] but "colors" are replaced here by the signature. The proposed protocol operates in a sequence of *electoral phases*. Each phase is composed of two operations that the agents *active* in that phase must perform: (1) territory acquisition, and (2) a sequence of partitioning and pairing rounds. At the end of a phase, as we will show, at least half of the active agents which entered the phase become *passive*, and the number of those which will start the next phase is still co-prime with n. In the following we will denote by r_i the number of active agents in Phase i (initially $r_1 = r$). We describe now the actions of the r_i agents entering Phase i. The first operation an active agent performs is to "acquire" as many nodes as possible. Assume that, at the beginning of the phase, all nodes, except the homebases of the agents active in this phase, are available. So, an active agent will start a depth-first traversal of G marking as *taken* any available node it visits. Note that the marking is done by writing the appropriate signed information on the whiteboard. Similarly, the whiteboard is used to write the relevant signed information (e.g., already traversed links, ...) used for the traversal. During the traversal, the agent will keep track of how many nodes are taken and by what other agents. The names of this list of occurrences will be decoded at each move, so they are always consistent.

Operation Territory Acquisition

(1) Active agent a in Phase i records in its homebase $p(a)$ the current phase number and starts a (depth-first) traversal of the nodes of G.

(2) During the traversal, when a enters node u from link $\{v, u\}$ carrying the list $L = m_1, \ldots, m_s$:

- a updates its name: $Myname := \mathbf{d}(\lambda_v(\{v, u\}), Myname)$, and all the names in its list L:

$\forall m_i \in L$, $m_i := \mathbf{d}(\lambda_v(\{v,u\}), m_i)$;

- if u has been already marked in this phase, a updates L and the associated counter;
- otherwise (i.e., u has not been marked in this phase)
  (a) if $u = p(b)$ is the homebase of a non-passive agent b, and the current phase number is not recorded, a waits until the phase is recorded by b; it updates L and the associated counter; it then proceeds with the traversal;
  (b) otherwise, a marks u with $Myname$ and *phase* and proceeds with the traversal.

The other operation in a phase is a sequence of *partition and pairing rounds.* In each Round j, the active agents will partition themselves into two sets, $W^{(j)}$ and $S^{(j)}$ and, executing different rules depending on the set they are in, they perform a pairing between the two sets. At the end of the pairing, some active agents will become passive. Depending on the result of the pairing either a new round is started, or the current phase terminates. In the latter case, if there is only one active agent left, that agent becomes the leader and starts the termination of the protocol, otherwise a new phase is started. Let us now describe each round. We distinguish the first round from the subsequent ones.

First Round ($j = 1$): Partition. When an agent has completed its territory acquisition, and returned to its homebase $p(a)$, it knows a sequence of integers $n_1, \dots, n_k$, and a partition $S_1, \dots, S_k$ of the set of agents, such that, for every ℓ, S_ℓ is the set of agents which have a territory of size n_ℓ. These two sequences satisfy

(1) $|S_\ell| \neq 0$ and $1 \leq n_\ell < n$, for all ℓ,
(2) $n_\ell \neq n_{\ell'}$ for all $\ell \neq \ell'$, and
(3) $\sum_{\ell=1}^{k} |S_\ell| \cdot n_\ell = n$.

Based on this knowledge, every agent determines the two following sets: A = set of the agents whose territory is of size $> n/r_i$, and B = set of the agents whose territory is of size $< n/r_i$. Note that since $\gcd(r_i, n) = 1$, n/r_i is not integral. Let $W^{(1)}$ be the largest of these two sets, and $S^{(1)}$ be the smallest of these two sets. In case of a tie, the agents set $W^{(1)} = A$. The agents in W will be called *waiting*, while those in S will be called *searching*. All other agents (if any) are passive.

First Round ($j = 1$): Pairing. We consider separately waiting and searching agents.

Waiting agents.
(1) Active agent $a \in W^{(1)}$ is initially *single*, writes the current round number in its homebase $p(a)$, and waits for the arrival of all the agents in $S^{(1)}$.
(2) When a searching agent s arrives to the homebase $p(a)$:

- a stores the local name of s
- if s has paired with a, a becomes *paired*

(3) When a is paired and has been visited by all the searching agents, it becomes passive.

Searching agents.
(1) Active agent $s \in S^{(1)}$ performs again a (depth-first) traversal of the nodes of G looking for the agents in $W^{(1)}$.(The traversal can be performed by using the spanning tree constructed during the first traversal in the first phase.)
(2) During the traversal, when s enters the homebase $p(a)$ of a waiting agent $a \in W^{(1)}$, s checks whether or not a has already returned to its homebase.

- If a is not in its homebase, s will wait for its return.

- If (when) a is in its homebase,
  (a) s will notify a of its visit;
  (b) if both s and a are single then s will pair with a;
  (c) s will continue its traversal.

If several single searching agents were waiting for a, only one of them will pair with it (since pairing is done by writing on the board, and access to witheboard is in mutual exclusion).

Passive agents.
A passive agent, i.e., a paired waiting agent, remains at its homebase, waiting to be notified of termination. We denote by $P^{(1)}$ the set of agents becoming passive during Round 1.

Subsequent Rounds ($j \geq 2$): Partition. We denote by $P^{(j-1)}$ the set of agents becoming passive during Round $j-1$. We define the new set of searching agents $S^{(j)}$, and the new set of waiting agents $W^{(j)}$, as follows.

- If $|W^{(j-1)}| - |S^{(j-1)}| \geq |S^{(j-1)}|$ then $S^{(j)} = S^{(j-1)}$ and $W^{(j)} = W^{(j-1)} \setminus P^{(j-1)}$. In other words, the set of searching agents does not change, and passive agents are removed from the set of waiting agents.
- If $|W^{(j-1)}| - |S^{(j-1)}| < |S^{(j-1)}|$ then $W^{(j)} = S^{(j-1)}$ and $S^{(j)} = W^{(j-1)} \setminus P^{(j-1)}$. In other words, we remove the passive agents from the set of waiting agents, and then the role of the two sets is switched.

Operation Partition. Each active waiting agent that has been visited by all the searching agents, and every searching agent that has completed its traversal, computes the cardinalities of $S^{(j)}$ and $W^{(j)}$, as well as which of these two sets it belongs to. Let a be such an agent.
(1) If $|S^{(j)}| \neq 0$, a enters Round $j+1$.
(2) Otherwise, let $r_{i+1} = |W^{(j)}|$ be the number of still active agents.

- If $r_{i+1} = 1$, a becomes leader and starts the termination of the protocol;
- Otherwise ($r_{i+1} > 1$), a enters Phase $i+1$.

Subsequent Rounds ($j \geq 2$): Pairing. Round j involves two sets of agents, $S^{(j)}$ and $W^{(j)}$, constructed during Round $j-1$. By construction, one of them is equal to $S^{(j-1)}$, and the other is the set of agents in $W^{(j-1)}$ which remain active at the end of Round $j-1$.

Waiting agents.
(1) Active agent $a \in W^{(j)}$ is initially single, writes the current round number in its homebase $p(a)$, and waits for the arrival of all the agents in $S^{(j)}$.
(2) When a searching agent s arrives to the homebase $p(a)$:

- a stores the local name of s;
- if s has paired with a, a becomes paired.

(3) When a is paired and has been visited by all the searching agents, it becomes passive.

Searching agents. Each active agent $s \in S^{(j)}$ performs again a (depth-first) traversal of the nodes of G looking for the agents in $W^{(j)}$. (The traversal can be performed by using the spanning tree constructed during the first traversal in the first phase.) There are two cases according to the set up of the sets $S^{(j)}$ and $W^{(j)}$.

Case 1: $S^{(j)} = S^{(j-1)}$ and $W^{(j)} = W^{(j-1)} \setminus P^{(j-1)}$. In this case, the current waiting agents were already waiting agents in the previous round, and therefore they are currently waiting at their homebases. If a searching agent visits an active waiting agent still in the previous round, then the searching agent stays in the homebase of the waiting agent until the latter updates its round number, or becomes passive.

Case 2: $S^{(j)} = W^{(j-1)} \setminus P^{(j-1)}$ and $W^{(j)} = S^{(j-1)}$. Since the current searching agents were waiting agents at the previous round, they have been visited by all the current waiting agents during Round $j - 1$. Therefore, every agent in $S^{(j)}$ knows the local name of the agents in $W^{(j)}$. This is useful because a searching agent may enter a homebase of a waiting agent which has not yet finished the previous round. As in Round 1, when a searching agent enters the homebase of an active agent, it checks whether it is the homebase of a searching agent, or the homebase of a waiting agent which has not yet finished the previous round. If a searching agent is currently in the homebase of a waiting agent which has not yet finished the previous round, then the searching agent waits until the latter updates its round number.

As in Round 1, the first time that a searching agent meets a single waiting agent, it pairs with it, and then completes the traversal in order to visit all the waiting agents. This completes the description of the election protocol.

The details of the proof of correctness of the protocol can be found in [5]. We just mention the following lemma, whose statement and proof contains the elements justifying the condition $\gcd(n, r) = 1$.

Lemma 3.2 *If $\gcd(r, n) = 1$, then for any i, $\gcd(n, r_i) = 1$ and and $r_{i+1} \leq r_i/2$, where r_i is the number of active agents at the ith phase.*

Proof. Consider Phase i with $\gcd(r_i, n) = 1$. One can easily show that $|S^{(1)}| + |W^{(1)}| = r_i$ and that every round of Phase i completes. Thus, by construction of the sets $S^{(j)}$ and $W^{(j)}$, the sequence of pairs $\{|S^{(j)}|, |W^{(j)}|\}$, $j \geq 1$, is the sequence of pairs of integers obtained by computing $\gcd(|S^{(1)}|, |W^{(1)}|)$ using Euclid's algorithm. From the property of the Euclid's algorithm, $r_{i+1} = \gcd(|S^{(1)}|, |W^{(1)}|)$. This implies that $\gcd(r_{i+1}, n) = 1$, and $r_{i+1} \leq r_i/2$. ∎

From Lemma 3.2, if $\gcd(r_1, n) = 1$, then, after at most $\log_2 r$ phases, only one agent will remain active: the leader. Note that our protocol is also quite efficient in terms of the traditional cost measure for mobile agents: the number of agent moves. In fact one can prove (see [5]) that the total number of edge-traversal in the protocol is $O(rn)$ in the worst case.

4 Concluding Remarks

It is known that the presence of sense of direction has a positive impact in distributed computations (for a survey, see [14]). The results presented here provide the first evidence that sense of direction has a positive influence on computability also in systems of mobile agents. An interesting open problem is to determine how to exploit, in addition to sense of direction, any existing asymmetry of the network or of the placement of the agents, so to sidestep the unsolvability result established here, extending the work of [19] to the mobile setting.

References

[1] E. Arkin, M. Bender, S. Fekete, and J. Mitchell. The freeze-tag problem: how to wake up a swarm of robots. In 13th ACM-SIAM Symposium on Discrete Algorithms (SODA '02), pages 568–577, 2002.

[2] B. Awerbuch, M. Betke, and M. Singh. Piecemeal graph learning by a mobile robot. Information and Computation, 152:155–172, 1999.

[3] L. Barrière and S. Dobrev. Leader election in abelian Cayley graphs. In 8th Colloquium on Structural Information and Communication Complexity (SIROCCO '01), Carleton Scientific, pages 5–20, 2001.

[4] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Distributed Mobile Computing with Incomparable Labels. Technical Report LRI-1309, Université Paris-Sud, France, 2002.

[5] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Electing a leader among anonymous mobile agents in anonymous networks with sense-of-direction. Technical Report LRI-1310, Université Paris-Sud, France, 2002.

[6] M. Bender, A. Fernandez, D. Ron, A. Sahai, and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. In 30th ACM Symp. on Theory of Computing (STOC '98), pages 269–278, 1998.

[7] X. Deng, T. Kameda and C. H. Papadimitriou. How to learn an unknown environment I: the rectilinear case. Journal of the ACM, 45:215-245, 1998.

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

[9] K. Diks, P. Fraigniaud, E. Kranakis and A. Pelc. Tree exploration with little memory. In 13th ACM-SIAM Symposium on Discrete Algorithms (SODA '02), pages 588–597, 2002.

[10] S. Dobrev, P. Flocchini, G. Prencipe and N. Santoro. Mobile agents searching for a black hole in an anonymous ring. In 15th Int. Symposium on Distributed Computing (DISC 2001), pages 166–179, 2001.

[11] S. Dobrev, P. Flocchini, G. Prencipe and N. Santoro. Searching for a black hole in arbitrary networks. In 21st ACM Symposium on Principles of Distributed Computing (PODC 2002), to appear.

[12] C. Duncan, S. Kobourov and V. Kumar. Optimal constrained graph exploration. In 12th ACM-SIAM Symp. on Discrete Algorithms (SODA '01) pages 807-814, 2001.

[13] P. Flocchini, B. Mans and N. Santoro. Sense of direction: definition, properties and classes, Networks 32: 165-180, 1998.

[14] P. Flocchini, B. Mans and N. Santoro. Sense of direction in distributed computing, In 12th International Symposium on Distributed Computing (DISC 98), pages 1-15, 1998. To appear in Theoretical Computer Science.

[15] P. Fraigniaud, C. Gavoille and B. Mans. Interval routing schemes allow broadcasting with linear message-complexity. In 19th ACM Symposium on Principles of Distributed Computing (PODC '00), pages 11-20, 2000.

[16] P. Fraigniaud, A. Pelc, D. Peleg and S. Pérennes. Assigning labels in unknown anonymous networks In 19th ACM Symposium on Principles of Distributed Computing (PODC '00), pages 101-112, 2000.

[17] N. Lynch. Distributed Algorithms. Morgan Kaufmann, Inc. San Francisco, California, 1996.

[18] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. SIAM Journal on Computing, 21(5):863–888, 1992.

[19] M. Yamashita and T. Kameda. Computing on anonymous networks, part I: characterizing the solvable cases. IEEE Transaction on Parallel and Distributed Computing, 7(1):69–89, 1996.

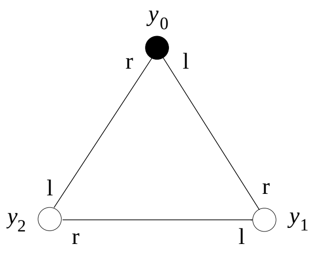

Figure 1: System $\mathcal{A}$.

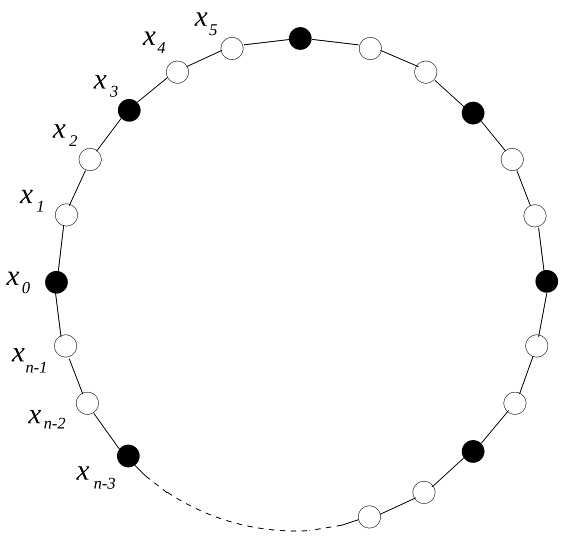

Figure 2: System $\mathcal{B}$ (the orientation of the edge labeling is not shown).

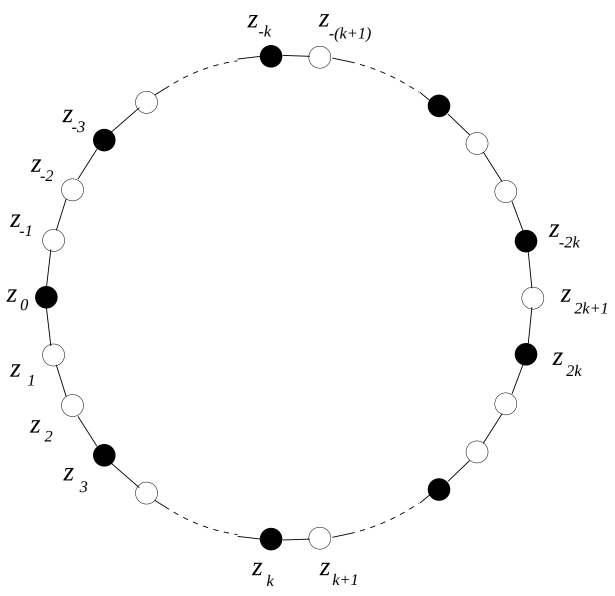

Figure 3: System $\mathcal{C}$ (the orientation of the edge labeling is not shown).