Rendezvous and Election of Mobile Agents: Impact of Sense of Direction*

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

Abstract

Consider a collection of r identical asynchronous mobile agents dispersed on an arbitrary anonymous network of size n. The agents all execute the same protocol and move from node to neighbouring node. At each node there is a whiteboard where the agents can write and read from. The topology of the network is unknown to the agents. We examine the problems of rendezvous (i.e., having the agents gather in the same node) and election (i.e., selecting a leader among those agents). These two problems are computationally equivalent in the context examined here. We study conditions for the existence of deterministic generic solutions, i.e., algorithms that solve the two problems regardless of the network topology and the initial placement of the agents. In particular, we study the impact of edge-labeling on the existence of such solutions. Rendezvous and election are unsolvable (i.e., there are no deterministic generic solutions) if $\gcd(r, n) > 1$, regardless of whether or not the edge-labeling has sense of direction. On the other hand, if $\gcd(r, n) = 1$ then the initial placement of the robots in the network creates topological asymmetries that could be exploited to solve the problems. We prove that these asymmetries can be exploited if the edge labeling has sense of direction, but can not if the edge-labeling is arbitrary. The possibility proof is constructive: we present a solution protocol and prove its correctness. The protocol, among other features, uses a dynamic naming mechanism based on sense of direction to overcome the complete anonymity of the system.

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

*A preliminary version of this paper has been presented at the $10th$ Colloquium on Structural Information and Communication Complexity (SIROCCO) [7].

†Universitat Politècnica de Catalunya, Spain. lali@mat.upc.es

‡*Corresponding author.* SITE, University of Ottawa, 800 King Edward Avenue, Ottawa, ON, K1N 6N5 Canada. flocchin@site.uottawa.ca

§CNRS, Université Paris-Sud, France. pierre@lri.fr

¶Carleton University, Canada. santoro@scs.carleton.ca

1 Introduction

We are interested in the computational issues arising in environments that support autonomous mobile entities. At an abstract level, these environments, which we shall call *distributed mobile systems*, can be described as a collection of autonomous mobile entities located in a spatial universe $\mathcal{U}$. The entities have bounded storage and processing capabilities, exhibit the same behavior (i.e., execute the same protocol), can move in $\mathcal{U}$ (their movement is constrained by the nature of $\mathcal{U}$), and are asynchronous in all their actions (e.g., computation, movement, etc). Depending on the context, the entities are sometimes called *agents*, other times *robots*; in the following, we use the former.

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 to be solved. In this paper, we focus on two basic problems: rendezvous and election. Rendezvous (also called gathering, point formation, or homing), is the process by which the autonomous mobile entities meet in the same spatial location in $\mathcal{U}$. There is no a priori restriction on which location should be the meeting point. The election (or symmetry breaking) problem consists in having the entities unanimously select one of them as the leader. There is no a priori restriction on which entity should be elected leader. We are interested in both election and rendezvous problems when the universe $\mathcal{U}$ is a simple graph G. This setting describes environments such as that of software mobile agents in a network. In those environments, an agent can move from node to neighbouring node. Each node of the network is provided with a *whiteboard* (i.e., a local storage where agents can write and read and erase information), and access to a whiteboard is done in mutual exclusion (e.g., see [10, 16]). Note that, in this setting, the two problems are computationally equivalent. That is, any solution protocol for one can be easily modified to solve the other. In fact, if a leader is elected, the leader can easily make all the other agents gather in a node of its choice. This can be achieved, for example, if the leader traverses the network and breaks the anonymity by assigning unique labels to the nodes; at this point, all the other agents can traverse the network as well, selecting as rendez-vous point the home base of the leader. Conversely, if the agents gather in a node, a leader can easily be elected by exploiting the mutual-exclusion access to the whiteboard of that node; e.g., the first agent to access it becomes the leader. Hence, the answer to any computational question for either problem is valid for both.

In this paper, we examine rendezvous and election in the least powerful settings for distributed mobile computing, that is when the network is anonymous, and the agents are identical and their operations and movements asynchronous. The network is unknown to the agents, i.e., we assume that the network, its size n, as well as the number r of agents are not known a priori. In anonymous networks, edges are locally labeled, i.e., every edge receives two labels, one for each extremity, and all edges incident to a node receive pairwise distinct labels at this node. Our interest is in *generic* solutions, that is, solutions that work independently of the structure of G and of the number of agents, and do not rely on such knowledge. The question we ask is under what conditions, if any, can the two problems

be solved by such weak agents in such a difficult environment. We provide some definite answers. A key role will be played not only by the relationship between the size n of the network and the number r of robots (two quantities unknown to the agents), but also and especially by the presence or lack of *sense of direction* in the network. Sense of direction is a property of edge-labeled graphs (e.g., see [9, 17, 22]). Roughly speaking, it provides to graphs similar properties as the usual notion of North, South, East, West in a mesh. Let us remark that any graph can be endowed with a sense of direction, thus sense of direction restricts the class of edge-labelings, not the class of graphs. In the context of distributed computing, where the network is a graph and the port numbers are the edge labels, the existence of such a property can be usefully employed to reduce the complexity of solution protocols in a variety of situations (for a recent account, see [18]). The computational power of sense of direction in anonymous networks has been thoroughly investigated and characterized [21]. In the context of mobile agents, the impact (if any) of sense of direction on computability and complexity is an unexplored subject. The only known results are for the problem of black hole search, where indeed the presence of sense of direction allows the problem to be solved by fewer agents [14].

1.1 Our Contributions

Our results on the solvability of election and rendezvous are summarized in Table 1.

It is rather obvious that if $\gcd(r, n) > 1$ these two problems are *unsolvable*: there is no generic solution protocol that always correctly terminates in finite time. This negative result holds even if the network has sense of direction known to the agents. This impossibility derives from the fact that, if $\gcd(r, n) > 1$, then the agents can be placed equidistant on a ring, so that they will never be able to break the symmetry (e.g., meet or elect a leader) under a synchronous scheduler.

If however r and n are co-prime, the argument above is no longer possible. So one might reasonably expect that $\gcd(r, n) = 1$ suffices for solvability. Surprisingly, this is not the case. In fact, we show that, with arbitrary edge labelings, no generic solution exists which allows the agents to always correctly terminate in finite time for every input such that $\gcd(r, n) = 1$ (cf. Theorem 2). On the other hand, we prove that, if the system has a sense of direction, then the election problem is *solvable* when $\gcd(r, n) = 1$ (cf. Theorem 3). In other words, we show that sense of direction overcomes anonymity if $\gcd(r, n) = 1$. The proof of this latter result is constructive, i.e., we present an election protocol for fully anonymous system with sense of direction where $\gcd(r, n) = 1$, and prove its correctness. The protocol uses a novel mechanism, called *dynamic name mutation*, based on sense of direction, that allows the anonymous agents in the anonymous network to distinguish agents and nodes in spite of anonymity. The protocol is also shown to be efficient both in terms of total number of movements by the agents and in terms of total amount of time to termination.

This is the first evidence that sense of direction has a positive computational impact also in distributed *mobile* computing.

	$\gcd(n,r) > 1$	$\gcd(n,r) = 1$
Without sense of direction	unsolvable (Theo. 1)	unsolvable (Theo. 2)
With sense of direction	unsolvable (Theo. 1)	**solvable** (Theo. 3)

Table 1: Summary of our results: solvability of the election and the rendezvous problems.

1.2 Previous Work

The rendezvous problem is a fundamental one in distributed mobile computing both with robots in the plane and with agents in a graph. In particular, the gathering problem has been extensively studied (both experimentally and analytically) by investigators in AI, robotics, and more recently algorithmics: the setting is a set of points in the plane and the entities are identical (i.e., anonymous) robots capable of moving in the plane. Results have been established both in the synchronous and asynchronous settings (e.g., see [1, 11, 20, 27]). The problem has also been extensively studied, mostly by investigators in operations research, assuming a geometrical space, such as the line, a circle, or polygonal region (e.g., [4, 23, 26]). In common, these investigations have that the universe where the robot's movement is continuous. See [3] for a detailed review.

In graphs, the rendezvous problem consists of devising a distributed protocol allowing mobile agents scattered in a network to meet at a single node. Hence the setting differs drastically from the previous ones, since the setting here is discrete. The rendezvous problem has been and continues to be investigated extensively in this universe (e.g., see [2, 3, 12, 19, 24, 30]). The investigations differ widely in their assumptions on the capabilities of the agents. For example, in some (e.g., [12, 30]), the robots have distinct identifiers (and thus are not anonymous) but can not leave messages on the whiteboards. In others (e.g., [19, 24]), the agents are anonymous but each has available a single marker that can be left on a single whiteboard. In most investigations, the protocols are topology-dependent. In some, the network is considered unknown but it is assumed that the same spanning-tree is somehow available to the agents [30]. For yet other models and assumptions, see, e.g., [2, 3, 15]. Despite their differences, these investigations in graphs all assume the agents to be fully synchronous: there is a global clock, and at each clock tick, all agents perform their computations and movements (if any) simultaneously. Very little is known about deterministic asynchronous rendezvous of anonymous agents in graphs. Although some solutions for specific network topologies do work also if the agents are asynchronous (e.g., [19]), the only explicit investigation is limited to ring networks [15].

The election problem is, as rendezvous, a fundamental one. In the context of distributed mobile computing, the existing results for election are only for non-anonymous systems, where the agents have *distinct* names; the only difference is whether these names are comparable values [13], or incomparable labels [6]. In distributed computing it has been extensively investigated in asynchronous anonymous networks (e.g., see [5, 8, 21, 25]).

In particular, Yamashita and Kameda [28, 29] have thoroughly investigated the election problem in anonymous networks. As part of this investigation, they established that, in bicolored anonymous networks the election problem is always solvable if $\gcd(n, r) = 1$, where r is the number of nodes with one of the two colours, provided that n is known, but it is sometimes impossible if $gcd(n, r) \neq 1$. Since sense of direction has the same computational power as complete topological knowledge [21], the same results as the one in Table 1 hold in the standard distributed model for anonymous bicolored networks with sense of direction. Obviously, bicolored graphs can describe which nodes are the homebases in the mobile agents setting. Therefore, there is equivalence in terms of computability between the mobile agents setting and the standard distributed setting, then our results could be derived from [28, 29] via [21]. The reduction has been proved in one direction [6]: the mobile agents model can be simulated by the standard model. To establish a reduction in the other direction (the one needed to give a reduction-based proof to our results) appears to be considerably more difficult and, to date, this task has not yet been accomplished.

1.3 Organization of the paper

The paper is organized as follows. In the next Section we introduce terminology and definitions. Then we prove in Section 3 the simple impossibility of election and rendezvous when $\gcd(r, n) > 1$. The unsolvability, without sense of direction, of those problems even when $\gcd(r, n) = 1$ is proven in Section 4. The election algorithm for $\gcd(r, n) = 1$ when there is sense of direction is presented in Section 5, where also its correctness is proven and its efficiency analyzed.

2 Definitions

2.1 The Network and the Agents

Let $G = (V, E)$ be a simple undirected n-node 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 that associates to each incident edge a distinct label, sometimes called *port number*, from a 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.

Scattered in (G, λ) are r autonomous mobile entities called *agents*. The graph G as well as r and n are not known to the agents. Let $p : \mathcal{E} \to V(G)$ be the injection describing the initial placement of the agents in G. The node $p(a)$ will be called the *homebase* of agent $a \in \mathcal{E}$. The agents have computing capabilities and limited storage, execute the same protocol, and can move from node to neighbouring node in G. After moving from u to v, an agent has available the label $\lambda_u(u, v)$ of the edge from which it departed, as well as the label $\lambda_v(v, u)$ of the edge from which it arrived. The agents are anonymous in the sense that they do not have distinct names or labels; they are asynchronous, in the sense that every action they perform (computing, moving, etc.) takes a finite but

otherwise unpredictable amount of time. The network is anonymous, that is, the nodes do not have distinguished names or labels. Each node of the network is provided with a *whiteboard*, i.e., a local storage where agents can write and read (and erase) information. Access to a whiteboard is done in mutual exclusion. Initially, the homebases of the agents are marked. (Note that since both nodes and agents are anonymous this marker denotes that the node is the homebase of some agent, but cannot be used to break symmetry).

The agents execute a deterministic protocol (the same for all agents) that specifies the computational and navigational steps. In each step, an agent will compute based on (1) its current state, (2) the content read on the whiteboard of the node currently visited, (3) the label of the edge from which it arrived, and (4) the current node degree. 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).

Initially, each agent has a predefined state variable set to *available*. The rendezvous problem consists in having all the agents gather at the same node; upon arriving there, each agent terminally sets its variable to *arrived*; when all the agents have arrived, within a finite time, they all enter a final state *gathered*; there is no a priori restriction on which node will become the rendezvous point. The election problem consists in having the agents unanimously select one of them that will terminally set its state variable to *leader*, while all the others will terminally set theirs to *follower*; there is no a priori restriction on which agent will be elected.

We are interested in generic solution protocols for these two problems, that is, solutions that work independently of the structure of G and of the number of agents. In our setting, election and rendezvous are computationally equivalent. That is, any solution protocol for one can be easily modified to solve the other. In fact, if a leader is elected, the leader can easily make all the other agents gather in a node of its choice. Conversely, if the agents gather in a node, a leader can easily be elected by exploiting the mutual-exclusion access to the whiteboard of that node; e.g., the first agent to access it becomes the leader. Hence, the answer to any computational question for either problem is valid for both.

2.2 Sense of Direction

Sense of direction is a property of edge-labeled graphs [17]. Roughly speaking, having sense of direction is the simple ability

1. to tell, by looking at sequences of labels corresponding to different paths starting from the same node, whether they end up in the same node or not; and

2. to translate from neighbor to neighbor the encoded information about paths in the system.

For some graphs, a sense of direction can be given by very natural labelings. This is the case, for example, of the mesh consistently labeled with North, South, East, West. (This labeling is called compass labeling.) Formally, sense of direction is based on the

notions of *coding* and *decoding* functions. Given an edge-labeled graph (G, λ), 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 the set of strings with alphabet $\mathcal{L}$ not including the empty string).

A coding function for λ is any function

$$\mathbf{c} : \mathcal{L}^+ \to \mathcal{N}$$

where $\mathcal{N}$ is a finite set, such that walks originating from the same node are mapped to the same element of $\mathcal{N}$ if and only if they end in the same node. More precisely,

$$\forall u, v, w \in V, \forall \pi_1 \in P[u, v], \forall \pi_2 \in P[u, w], \Big(\mathbf{c}(\Lambda_u(\pi_1)) = \mathbf{c}(\Lambda_u(\pi_2)) \Leftrightarrow v = w\Big).$$

In the example of the mesh with compass labeling, the set of labels $\mathcal{L}$ is $\{N, S, E, W\}$ (for North, South, East, West), and the coding function $\mathbf{c} : \mathcal{L}^+ \to \mathcal{L}^*$ is defined by: given a sequence of labels, $\mathbf{c}$ returns the sequence obtained by: (1) simplifying every occurrence of N (resp., E) with an occurrence of S (resp., W), and (2) lexicographically sorting the resulting sequence. For example $\mathbf{c}(NESSWNNE) = EN$. This simplification process allows one to check if different sequences, starting from the same node, actually lead to the same end node.

A decoding function for $\mathbf{c}$ is any function

$$\mathbf{d} : \mathcal{L} \times \mathcal{N} \to \mathcal{N}$$

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 walk from u to w obtained by traversing $\{u, v\}$, and then following π. More precisely,

$$\forall \{u, v\} \in E, \forall \pi \in P[v], \Big(\mathbf{d}(\lambda_u(\{u, v\}), \mathbf{c}(\Lambda_v(\pi))) = \mathbf{c}(\lambda_u(\{u, v\}) \circ \Lambda_v(\pi))\Big)$$

where $\circ$ denotes concatenation of strings of labels. In other words, while in general to know the coding of the labels of a walk we need to know all the labels, with the decoding function it is sufficient to know only the first label and just the coding of the rest.

In the example of the mesh with compass labeling, the decoding function $\mathbf{d} : \mathcal{L} \times \mathcal{L}^* \to \mathcal{L}^*$ is defined by $\mathbf{d}(l, \beta) = \mathbf{c}(l \circ \beta)$. Consider a walk π starting from x, and let $\Lambda_x(\pi) = (WEWSENNES)$. Knowing the first label of $\Lambda_x(\pi)$ and the coding (EE) of the rest of the sequence $(EWSENNES)$ obtained by application of $\mathbf{c}$, allows to know the coding of the entire sequence. In fact $\mathbf{d}(W, EE) = E = \mathbf{c}(WEWSENNES) = \mathbf{c}(\Lambda_x(\pi))$.

Definition 1 (cf. [17]) *An edge-labeled graph (G, λ) has* sense of direction *if and only if there exist a coding function $\mathbf{c}$ for (G, λ), and a decoding function $\mathbf{d}$ for $\mathbf{c}$. (We will also say that the pair $(\mathbf{c}, \mathbf{d})$ is a sense of direction for (G, λ).)*

Notice that sense of direction could be very powerful when performing rendezvous (or election). In the case of a mesh with compass labeling, a natural asymmetry created by the labels could be exploited. In fact, the rendezvous point could be agreed in advance to be the Northmost-Eastmost node of the mesh and the rendezvous algorithm would simply require the agents to move there. The election would be accomplished by choosing as a leader the first agent that writes on the whiteboard of the meeting point. Both tasks can be easily accomplished by the agents without any additional information (size of the mesh, number of the agents, etc.). However, without the compass sense of direction, i.e., if edges are labeled inconsistently, then this simple protocol would not work.

Remark. Any graph can be endowed with a sense of direction. Indeed, 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 $j - i$. With this labeling there is a simple coding function: two paths from the same node terminate in the same node iff the sum of the corresponding labels is the same, that is

$$\forall u \in V, \forall \pi \in P[u], \text{ if } \Lambda_u(\pi) = [l_0, \ldots, l_k] \text{ then } \mathbf{c}(\Lambda_u(\pi)) = \sum_{i=0}^{k} l_i.$$

The decoding function is defined as follows:

$$\forall \{u, v\} \in E, \forall \pi \in P[v], \mathbf{d}(\lambda_u(\{u, v\}), \mathbf{c}(\Lambda_v(\pi))) = \lambda_u(\{u, v\}) + \mathbf{c}(\Lambda_v(\pi)).$$

It is easy to verify that $\lambda_u(\{u, v\}) + \mathbf{c}(\Lambda_v(\pi)) = \mathbf{c}(\lambda_u(\{u, v\}) \circ \Lambda_v(\pi))$. This sense of direction is one of many that can be constructed in an arbitrary graph.
A feature that makes this sense of direction particularly appealing is that both $\mathbf{c}$ and $\mathbf{d}$ are very simple and require constant memory to be stored.

Terminology: In the following, when we say that "a network (G, λ) has sense of direction $(\mathbf{c}, \mathbf{d})$", we mean that both $\mathbf{c}$ and $\mathbf{d}$ are known to the agents. Notice that, as mentioned above, in any graph there always exists a sense of direction where the coding and the decoding functions require $O(1)$ bits to be represented.

3 A Basic Impossibility Result

If $\gcd(n, r) > 1$, then there are networks where any election (resp. rendezvous) protocol will be unable to elect a leader (resp. gather the agents) in every execution, and this fact is true even if the system is endowed with sense of direction. Indeed, we have:

Theorem 1 *Assume* $\gcd(r, n) > 1$. *Then, the election and the rendezvous problems are deterministically unsolvable in anonymous systems, even if there is sense of direction.*

Proof. Consider a ring network $R = (x_0, x_1, \ldots, x_{n-1})$ with the classical "left/right" sense of direction. Let the set of labels be $\mathcal{L} = \{L, R\}$. Let L^i, (resp., R^i) denote i consecutive occurrences of label L (resp., R). A coding function $\mathbf{c}$ in this case simplifies occurrences of L with occurrences of R and maps R^i and L^{n-i} into the same value.

1. $\forall \alpha, \beta \in \mathcal{L}^*$, $\mathbf{c}(\alpha \circ R^i \circ \beta) = \mathbf{c}(\alpha \circ L^{n-i} \circ \beta)$
2. $\mathbf{c}(L^i) = L^{i \bmod n}$

where $L^0 = \epsilon$ is the empty string.

In an equidistant initial placement of the agents along the ring, the views of all agents are identical at any distance. (Recall that, informally, the *view* of an edge-labeled graph G from a node v is the infinite labeled rooted tree defined as the union of the edge-labeled paths originated at v in G, cf. [28]). A 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. In other words, election (and thus rendezvous) is impossible. ■

4 Unsolvability with arbitrary labelings

We have just seen that, if r divides n, there are some networks in which some initial placements of the agents create an unbreakable symmetry, making the rendezvous and election problems unsolvable. On the other hand, if r and n are coprime, any initial placement of the robots in any network creates a topological asymmetry, that could be exploited to solve the problems. In this section we will prove the perhaps surprising fact that, with arbitrary labelings, these asymmetries can not be exploited and, in fact, the rendezvous and the election problems are unsolvable even if r and n are coprime.

Theorem 2 *Assume* $\gcd(n, r) = 1$. *Then, the election and the rendezvous problems are deterministically unsolvable in an anonymous system with arbitrary labeling.*

Proof. Assume, by contradiction, that there exists a generic election protocol $\mathcal{P}$ for all inputs G where $\gcd(n, r) = 1$. We will focus on the executions of $\mathcal{P}$ by a synchronous scheduler, i.e., where all computations are instantaneous, movements require a unit of time, and all agents start simultaneously. We will consider oriented rings, i.e., consistently labeled with "left" (L) and "right" (R). (Note that orientation alone does not imply sense of direction, cf. [17]).

Consider 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 the execution of $\mathcal{P}$ under a synchronous scheduler. 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 and let $d = \lceil \frac{T(\mathcal{A})}{3} \rceil$.

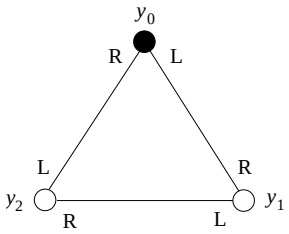

Figure 1: System $\mathcal{A}$.

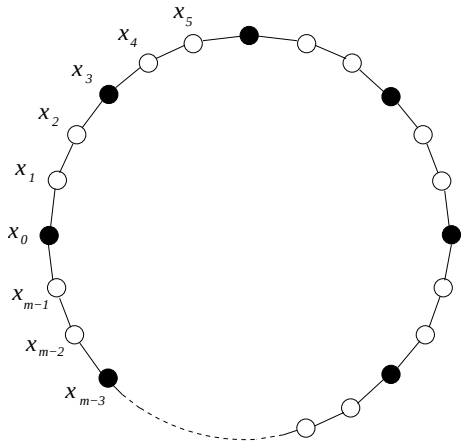

Figure 2: System $\mathcal{B}$ (the orientation of the edge labeling is not shown).

Consider now a system $\mathcal{B}$ composed of an oriented ring $(x_0, x_1, \ldots, x_{4k+8})$ of $m = 4k + 9$ nodes, where $k = 3d$. For convenience, we shall denote node x_{m-i} simply by x_{-i}, $1 \leq i \leq 2k + 4$. An agent is placed in each of locations x_{3j} and x_{-3j}, $0 \leq j \leq 2d + 1$ (see Figure 2). Since the number of agents is $r = 4d + 3$ and the number of nodes is $m = 4k+9 = 3(4d+3) = 3r$, we have $\gcd(m, r) > 1$. Clearly the initial system symmetry in $\mathcal{B}$ implies that the views of all agents are identical at any distance. Furthermore this view is undistinguishable from that of the only agent in $\mathcal{A}$. It is not difficult to see that the relationship between the two systems is time-invariant under a synchronous scheduler. That is, at time $T(\mathcal{A})$, all agents will terminate entering state leader. The fact that $\mathcal{P}$ fails in $\mathcal{B}$ is not surprising since, in $\mathcal{B}$, $\gcd(m, r) > 1$.

Consider now a system $\mathcal{C}$ of $n = m + 1 = 4k + 10$ nodes, obtained from $\mathcal{B}$ by adding a node z, initially without an agent, between x_{2k+4} and $x_{-(2k+4)}$ (see Figure 3). Thus the number of agents is still $r = 4d + 3$, but $\gcd(n, r) = 1$. Denote by b_j the agent with homebase x_j. 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 $\mathcal{C}$. Start a synchronous simultaneous execution of $\mathcal{P}$ by all agents in $\mathcal{C}$. Let us now compare the first k steps of this execution in $\mathcal{C}$ with the one in $\mathcal{B}$. The agents

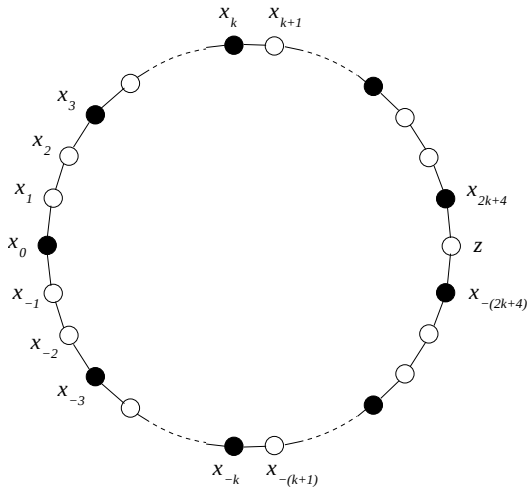

Figure 3: System $\mathcal{C}$ (the orientation of the edge labeling is not shown).

b_{2k+4} and $b_{-(2k+4)}$ have a different 3-neighbourhood from all other agents. One of these two agents, say b_{2k+4}, will be the first to have a different state from all others agents. To do so, it must pass by node z. This implies that, at time k, at most the segment $(x_{k+5}, \ldots, x_{2k+4}, z, x_{-(2k+4)}, \ldots, x_{-(k+5)})$ will be affected by the change. Thus, for the first k steps, the segment $Z = (x_{k+4}, \ldots, x_0, \ldots, x_{-(k+4)})$ in system $\mathcal{C}$ will be undistinguishable from the same segment in system $\mathcal{B}$. Thus, all the agents in $\mathcal{C}$ that, during this time, do not leave Z will all have the same state at each time step, which will be equal to that of the single agent in $\mathcal{A}$ at that time. Since, for $t \leq k$, the three agents b_0, b_3 and b_{-3} are always within Z and, since $k < T(\mathcal{A})$, they will enter state *leader* at time $T(\mathcal{A})$, contradicting the correctness of $\mathcal{P}$ when $\gcd(n, r) = 1$. ■

Note that Theorem 2 holds even if the agents are allowed unbounded amount of local processing and of memory, and the whiteboards have unbounded capacity.

5 Solvability with sense of direction

When r and n are coprime, any initial placement of the agents in any network creates asymmetries in the system. However, in general, without sense of direction, the anonymity of the system does not allow these asymmetries to be exploited: Theorem 2 says that the rendezvous and election problems are unsolvable with arbitrary edge-labeling, even when restricted to those instances where n and r are coprime. We will now show in this section, that sense of direction empowers the agents to effectively exploit the asymmetry created by having $\gcd(r, n) = 1$, and to overcome anonymity in those instances. Indeed, we have

Theorem 3 *Assume* $\gcd(n, r) = 1$. *Then the election and the rendezvous problems are deterministically solvable in anonymous systems with sense of direction.*

1. To determine its initial name, agent a, with homebase u, chooses an arbitrary neighboring node v and determines the label $\lambda_v(\{u,v\})$ (e.g., by moving to v and coming back). Then it sets $Myname := \mathbf{c}(\lambda_u(\{u,v\}) \circ \lambda_v(\{u,v\}))$.

2. When agent a with name $Myname$ at node x moves to the neighboring node y, it modifies its name as follows: $Myname := \mathbf{d}(\lambda_y(\{y,x\}), Myname)$.

Figure 4: Dynamic name mutation of agent a

The proof is constructive, i.e., we will describe and analyze an election protocol that works for every instance with $\gcd(n,r) = 1$. The description requires some preliminaries.

5.1 Dynamic Name Mutation

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 *dynamic name mutation* (DNM), which, exploiting the presence of sense of direction, allows us to operate in spite of these limitations, including homonimity. This mechanism is described Figure 4. 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. Another 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.

Let a be an agent. We will denote by $name(a,v)$ its name at node v. In the following lemma we show that the name of an agent at a node solely depend on its starting point and not on the path followed to reach the node.

Lemma 1 *The name of an agent a at some node v is independent of the path followed to reach v from its homebase u.*

Proof. To prove the lemma we will prove first that $name(a,v) = \mathbf{c}(\alpha)$ where α is a sequence of labels corresponding to any path from v to the homebase u of agent a. Since,

by definition, the coding is the same for different paths provided they start from the same node (in this case v) and they terminate in the same node (in this case u), this claim proves the lemma. Consider thus the path $\pi = (v_0, v_1, v_2 \ldots, v_k)$ traversed by a from $u = v_0$ to $v = v_k$. We will prove our claim by induction on i ($1 \leq i \leq k$).

Basis. The property is true at node u. In fact, by definition of Dynamic Name Mutation, $name(a, u) = \mathbf{c}(\alpha \circ \beta)$ where α is the label $\lambda_u(\{u, v\})$ of any edge $\{u, v\}$ and $\beta = \lambda_v(\{u, v\})$. By definition of coding function, any other path from u to itself would have given the same coding.

Induction. Assume the property holds at v_i, and consider now v_{i+1}. The name of a at v_{i+1} is

$$name(a, v_{i+1}) = \mathbf{d}(\lambda_{v_{i+1}}(\{v_{i+1}, v_i\}), name(a, v_i)) = \mathbf{d}(\lambda_{v_{i+1}}(\{v_{i+1}, v_i\}), \mathbf{c}(\alpha)),$$

where α is a sequence of labels corresponding to a path from v_i to u. In particular such a path could be $(v_i, \ldots, v_1, u)$. Then it follows that

$$name(a, v_{i+1}) = \mathbf{d}(\lambda_{v_{i+1}}(\{v_{i+1}, v_i\}), \mathbf{c}(\Lambda_{v_i}(\alpha))) = \mathbf{c}(\lambda_{v_{i+1}}(\{v_{i+1}, v_i\}) \circ \Lambda_{v_i}(\alpha)))$$

which, by definition of coding function, is equal to $\mathbf{c}(\beta)$, where β is a sequence of labels corresponding to any path from v_{i+1} to u. This completes the proof. ■

We now show that, if two names are different at a node, then they belong to different agents, while, if they are identical, then they necessarily belong to the same agent.

Lemma 2 *Let a and b be any two agents, and let v be any node. We have:*

$$name(a, v) = name(b, v) \quad \Leftrightarrow \quad a = b.$$

Proof. We have to prove 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) is proven in lemma 1. Let us consider point (ii). Let a and b be two different agents with respective homebases u and u', $u \neq u'$. At node v, $name(a, v) = \mathbf{c}(\alpha)$ and $name(b, v) = \mathbf{c}(\beta)$ where α corresponds to any path from v to u, and β corresponds to any path from v to u'. By definition of coding function $\mathbf{c}(\alpha) \neq \mathbf{c}(\beta)$ because $u \neq u'$. ■

Because of Lemma 2, we are guaranteed that whenever an agent leaves a signed information on a whiteboard (i.e., writes some information on the whiteboard and signs it with its local name), 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. In our election protocol we will heavily make use of this properties. In fact, whenever an agent writes an information on the whiteboard of a node, it will sign it with its own name at that node.

5.2 The Election Protocol

We now present a protocol which will elect one of the agents as the leader, provided $\gcd(r, n) = 1$. The high level description of the algorithm is given in Figure 5. The proposed protocol operates in a sequence of *phases*. Each phase starts with two operations that the *active* agents must perform: (1) *territory acquisition*, and (2) *set-up*, followed by (3) a sequence of *pairing* and *partitioning* 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 that will start the next phase is still co-prime with n. The algorithm will terminate when there is only one active agent left, the leader.

```
while (number of active agents > 1) do    /* new phase */
        Territory Acquisition;
        Set-up; /* S and W are created */
        while (|S| ≠ 0 ) do    /* new round */
                Pairing; /* S is paired with a subset of W */
                Partition; /* S and W are updated */
        endwhile
endwhile
```

Figure 5: Overall View of Algorithm Elect.

Notation. In the following we will denote by r_i the number of active agents in Phase $i \geq 1$.

We describe now the actions of the r_i agents entering Phase i, for $i \geq 1$. Initially $r_1 = r$.

5.2.1 Territory Acquisition and Set-up

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*. Every active agent starts a depth-first traversal of G marking as *taken* any available node it visits. The marking is done by writing an appropriate signed information on the whiteboard. During the traversal, an 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 (Lemma 2). A detailed description of Territory Acquisition is given in Figure 6. Notice that an agent a performing territory acquisition in phase i might end up in the homebase of an agent b that is still finishing the previous phase $i - 1$. To decide whether to acquire this node or not, a has to wait for the return of b and verify whether b has become passive or not. Agent a will acquire this node only if b is passive in phase i. As we will prove later, it will never happen that a in phase i waits for another agent in a phase previous to $i - 1$.

1. Active agent a in Phase i records in its homebase the current phase number and starts a (depth-first) traversal of the nodes of G carrying a list L of names, and a list C of counters (one for each entry in L). Initially, L contains only a's name, and its counter in C is set to 1.

2. During the traversal, when a visits a node v from link $\{u, v\}$ carrying the lists $L = \{m_1, \ldots, m_s\}$ and $C = \{n_1, \ldots n_s\}$, it does the following:

   (a) It updates its name by $Myname := \mathbf{d}(\lambda_v(\{v, u\}), Myname)$, and all the names in its list L, i.e., for all $m \in L$, $m := \mathbf{d}(\lambda_v(\{v, u\}), m)$;

   (b) It checks whether or not v has been marked in this phase i.

   - If v has been marked, say by agent b, then a updates L and C, i.e., a creates an entry for b in L (if not already there) and a corresponding counter in C (initially set to 0), and increases the counter.
   - If v has not been marked but it is the homebase of an active agent b (with phase number $i - 1$), then a waits until b returns.
     - If upon its return b becomes passive, then a marks v with $Myname$ and i, and increases its counter in C;
     - Otherwise (i.e., when it returns, b is still active), a updates L and C, i.e., a creates an entry for b in L (if not already there) and a corresponding counter in C (initially set to 0), and increases the counter.
   - Otherwise, a marks v with $Myname$ and i, and increases its counter in C.

   (c) It proceeds with the traversal.

Figure 6: Territory Acquisition (phase i).

After the territory acquisition, the active agents have acquired enough information to proceed with the set-up operation. During the set-up, the active agents locally partition themselves into two sets, W (for *waiting*) and S (for *searching*) as described in Figure 7. All other agents (if any) are passive.

5.2.2 Pairing and Partition Rounds

After Territory Acquisition and setup, a sequence of rounds starts with the objective of reducing the number of active agents. The agents execute different rules depending on the set S or W they are in, and they perform a pairing between the two sets. At the end of the pairing some active agents become passive. Depending on the result of the pairing, either a new round is started with new sets of waiting and searching agents (partition), or the current round 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.

When an agent has completed its territory acquisition, and returned to its homebase, it knows a sequence of integers $n_1, \ldots, n_k$, and a partition $A_1, \ldots, A_k$ of the set of agents, such that, for every ℓ, A_ℓ is the set of agents which have a territory of size n_ℓ. These two sequences satisfy:

- $|A_\ell| \neq 0$ and $1 \leq n_\ell < n$, for all ℓ,
- $n_\ell \neq n_{\ell'}$ for all $\ell \neq \ell'$, and
- $\sum_{\ell=1}^{k} |A_\ell| \cdot n_\ell = n$.

Based on this knowledge, every agent determines the two following sets:

- A: the set of the agents whose territory is of size greater than n/r_i, and
- B: the set of the agents whose territory is of size less than n/r_i. /* since $\gcd(r_i, n) = 1$, n/r_i is not integral */

Let W be the largest of these two sets, and S be the smallest of these two sets. In case of a tie, the agents set $W = A$.

Figure 7: Set up

Basically, during the pairing operation, the agents executes Euclid's algorithm based on the recursion $\gcd(a, b) = \gcd(a, b - a)$ for $b \geq a$. The searching agents (i.e., those in S) aim at pairing with waiting agents (i.e., those in W) so as to make $|S|$ waiting agents becoming passive. Every round of a phase is the analogue of $\gcd(|S|, |W|) = \gcd(|S|, |W| - |S|)$. The pairing operation is described in Figures 8. At the end of the pairing all searching agents and some of waiting agents will still be active. All these agents *locally* perform the partition operation, described in Figure 9 from the point of view of an active agent a.

A new round starts when $S \neq \emptyset$. If $S = \emptyset$, then two cases are considered. If $|W| = 1$ then the only active agent (i.e., the one in W) becomes the leader. Otherwise, a new phase starts, involving active agents (i.e., those currently in W). That is, territory acquisition and setup are performed so as to start a new phase of Euclid's algorithm.

5.3 Correctness

In this section we prove the correctness of Algorithm ELECT. We first show that, although searching agents can wait at homebases of other agents, no deadlocks occur.

Lemma 3 *A searcher executing pairing in round j can only wait for an agent that is executing pairing (as a searcher) in round $j - 1$, and it will complete round j in finite time.*

Proof. By induction on j. It is clearly true in round $j = 1$. Let it be true for round $j - 1 \geq 1$ and consider round j. Let a be a searcher starting round j. We first show that,

Let $W^{(1)} := W$ and $S^{(1)} := S$. Round j involves two sets of agents, $S^{(j)}$ and $W^{(j)}$, constructed during Round $j-1$. We consider separately waiting and searching agents at round j.

Waiting agent a

(1) Agent $a \in W^{(j)}$ is initially *single*, writes the current round number in its homebase u, and waits for the arrival of all the agents $\in S^{(j)}$.

(2) When a searching agent s arrives to the homebase u of a:

- a stores $name(s, u)$;
- If s pairs with a according to the rule below, then a becomes *paired*.

(3) When a has been visited by all the searching agents, if it is *paired* it becomes *passive*, otherwise it starts the partition operation.

Searching agent s.

(1) Agent $s \in S^{(j)}$ performs a traversal of the nodes of G looking for the agents in $W^{(j)}$.

(2) During the traversal, when s enters the homebase u of agent $a \in W^{(j)}$

- If a is not in its homebase (and its round number is $j-1$), then s waits for its return.
- If a is in its homebase (or when it arrives at its homebase),
  - s notifies a of its visit;
  - if both s and a are *single* then s pairs with a — 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 whiteboard, and access to whiteboard is in mutual exclusion */
  - s continues its traversal.

(3) The traversal is completed when s is back to its homebase. At this point s starts the partition operation.

Passive agents. A passive agent, i.e., a paired waiting agent, remains at its homebase, waiting to be notified of termination.

Figure 8: Pairing (round j)

when a searcher waits for another agent, there are no agents still executing round $j-2$. Indeed, to start round j, a must have finished (in round $j-1$) the traversal of the graph of the last pairing. During that traversal, by inductive hypothesis, it might have waited only for agents still in round $j-2$. But all these agents must have returned (otherwise a would have not finished that round), thus entering round $j-1$. In other words, when a starts round j, there are no agents still executing round $j-2$. Therefore, a can only wait in round j for agents that are still in round $j-1$. These agents, by inductive hypothesis, will enter round j in finite time returning to their homebases. Hence, a will also complete its pairing traversal of round j within finite time. ■

Similarly,

Lemma 4 *An agent executing territory acquisition in phase i can only wait at the homebase of an agent that is executing pairing in phase $i-1$ and it will complete territory acquisition in finite time.*

Then we show:

The set of agents which became passive during Round j is denoted by $P^{(j)}$.
(1) Agent a computes the cardinalities of $S^{(j)}$ and $W^{(j)}$, as well as which of these two sets it belongs to, as follows:

- If $|W^{(j)}| - |S^{(j)}| \geq |S^{(j)}|$ then $S^{(j+1)} = S^{(j)}$ and $W^{(j+1)} = W^{(j)} \setminus P^{(j)}$.
- If $|W^{(j)}| - |S^{(j)}| < |S^{(j)}|$ then $W^{(j+1)} = S^{(j)}$ and $S^{(j+1)} = W^{(j)} \setminus P^{(j)}$.

(2) If $|S^{(j+1)}| \neq 0$, a enters Round $j+1$.
(3) Otherwise, let $r_{i+1} = |W^{(j+1)}|$ be the number of still active agents.

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

Figure 9: Partition (round j)

Lemma 5 *At the beginning of Phase i,* $\gcd(r_i, n) = 1$ *and, if $i > 1$, then* $r_i \leq \frac{r_{i-1}}{2}$.

Proof. By induction on i. It is clearly true at the beginning because, by definition we have that $\gcd(r, n) = 1$. Let it be true at Phase i (i.e., $\gcd(r_i, n) = 1$). After the set up of phase i, and, thus, at the beginning of partition and pairing, $|S^{(1)}| + |W^{(1)}| = r_i$. During partition and pairing, by construction of the sets $S^{(j)}$ and $W^{(j)}$, the sequence of pairs $\{|S^{(j)}|, |W^{(j)}|\}$, of rounds $j \geq 1$, is the sequence of pairs of integers obtained by computing $\gcd(|S^{(1)}|, |W^{(1)}|)$ using Euclid's algorithm. Then, by the properties of Euclid's Algorithm, the next set of searching agents for phase $i+1$ will contain $r_{i+1} = \gcd(|S^{(1)}|, |W^{(1)}|)$ agents and $r_{i+1} \leq \frac{r_i}{2}$. Since $|S^{(1)}| + |W^{(1)}| = r_i$ and $\gcd(r_i, n) = 1$, we have that $\gcd(r_{i+1}, n) = 1$. ■

Proof of Theorem 3. By simple application of Lemmas 3, 4, and 5. ■

5.4 Observations on Costs

Although not the primary objective of our study, we observe that our protocol is quite efficient with respect to the traditional cost measures for mobile agents: the number of agent moves and the amount of time. An agent executing the proposed protocol performs several traversals of the network. To traverse the network, an agent can use as a guideline any standard message-optimal distributed depth-first traversal algorithm: the agent will simulate and follow the behavior of the traversing token. This would result in $O(|E|)$ moves per traversal. However, since the system has sense of direction, distributed depth-first traversal can be actually performed using only $2n$ messages [18], resulting in a traversal by an agent in only $2n$ moves.

Property 1 *Assume* $\gcd(n, r) = 1$. *Then, in an n-node anonymous network with sense of direction, r anonymous asynchronous agents that do not know n nor r nor the network*

topology can rendezvous and elect a leader with $O(nr)$ moves in time $O(n \log r)$ in the worst case.

Proof. During the Territory Acquisition of Phase i, each of the r_i agents performs a traversal of the network to acquire territories. Each traversal can be performed by an agent in only $2n$ moves. Thus, the Territory Acquisition of Phase i will cost a total of $2r_i n$ moves. During the remaining rounds of Phase i, traversals are executed by the agents to perform pairings. Starting from r_i, active agents split into two sets $S^{(1)}$ and $W^{(1)}$, the total number of pairings performed to decrease the number of active agents to $r_{i+1} = \gcd(S^{(1)}, W^{(1)})$ is $r_i - r_{i+1}$. Therefore, assuming that there are k phases, the total number of moves performed by the agents for traversals is $\sum_{i=1}^{k} 2(r_i n + (r_i - r_{i+1})n) = 2n \sum_{i=1}^{k} (2r_i - r_{i+1})$. Since, by Lemma 5, $r_{i+1} \leq r_i/2$, k is equal to $\log r$, thus, the total number of moves is at most $O(rn)$.

As for the time costs, the total number of phases is at most $\log r$ and in each phase an active agent performs a constant number of traversals. As observed above, using sense of direction each traversal can be done with $2n$ moves and, thus, the same amount of time. ■

Finally, let us observe that the amount of memory required by an agent is related to the size of the names used by the coding function. In fact, the traversal protocol exploiting sense of direction [18] requires the token (and thus the agent) to carry the list of names of the nodes traversed so far. Therefore, in the specific case when names can be coded by $O(\log n)$ bits, this implies an $O(n \log n)$ bits memory requirement for the agents. Observe that $O(n \log n)$ bits suffice to store the map of the DFS-tree (necessary for traversal), as well as the names of the currently active agents and the size of their territory required by the algorithm.

6 Concluding Remarks

It is well known that the presence of sense of direction has a positive impact both on the complexity and the computability of distributed problems (for detailed account, see [18]). Recently, it has been shown that sense of direction can have a positive impact on the efficiency of solutions also in the mobile agents setting [14]. The results presented here provide the first evidence that, in systems of mobile agents, sense of direction has a positive influence also on computability.

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 [28] to the mobile setting.

Another open problem would be when the initial placement of the agents is not injective; i.e., more than one agent can be initially located in the same node. Note that our results would still hold when $\gcd(n, r) = 1$, where r is the number of distinct homebases.

Acknowledgments. This research was carried out in part 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), NATO, the Research Grant from the Ministry of Science and Technology (Spain), and the European Regional Development Fund (Project TIC2002-00155).

References

[1] Agmon, N., and Peleg, D. Fault tolerant gathering algorithms for autonomous mobile robots. In *Proc. Symposium on Discrete Algorithms (SODA 2004)*, 2004.

[2] Alpern, S., Baston, V., and Essegaier, S. Rendezvous search on a graph. *Journal of Applied Probability 36*, 1 1999, 223–231.

[3] Alpern, S., and Gal, S. *The Theory of Search Games and Rendezvous*. Kluwer Academic Publishers, Norwell, Massachusetts, 2003.

[4] Anderson, E.J., and Essegaier S. Rendezvous search on the line with indistinguishable players. *SIAM Journal of Control and Optimization 33*, 1995, pp. 1637-1642.

[5] Angluin, D. Local and global properties in networks of processors. In *Proc. of 12th A.C.M. Symposium on Theory of Computing*, 1980, pp. 82-93.

[6] Barrière, L., Flocchini, P., Fraigniaud, P., and Santoro, N. Can we elect if we cannot compare ? In *Proc. 15th ACM Symp. on Parallel Algorithms and Architectures (SPAA '03)*, 2003, pp. 324-332.

[7] Barrière, L., Flocchini, P., Fraigniaud, P., and Santoro, N. Election and rendezvous in fully anonymous systems with sense of direction. In *Proc. 10th Colloquium on Structural Information and Communication Complexity (SIROCCO '03)*, 2003, pp. 17-32.

[8] Boldi, P., Codenotti, B., Gemmell, P., Shammah, S., Simon, J., Vigna, S. Symmetry breaking in anonymous networks: characterizations. In *Proc. of 4th Israeli Symposium on Theory of Computing and Systems*, 1996, pp. 16–26.

[9] Boldi, P., and Vigna, S. On the complexity of deciding sense of direction. *SIAM Journal on Computing 29*, 3, 2000, pp 779-78.

[10] Cardelli, L., and Gordon, A. Mobile ambients.. *Foundations of Software Science and Computational Structures*, LNCS 1478, Springer Verlag, 1998.

[11] Cieliebak, M., Flocchini, P., Prencipe, G., and Santoro, N. Solving the gathering problem. In 30^{th} *International Colloquium on Automata, Languages and Programming (ICALP '03)*, 2003.

[12] Dessmark A., Fraigniaud P., and Pelc. A. Deterministic rendezvous in graphs. In *11th Annual European Symposium on Algorithms (ESA'03)* 2003, pp. 184-195.

[13] Deugo, D. Mobile agents for electing a leader. In *4th Int. Symposium on Autonomous Decentralized System* 1999, pp. 324–324.

[14] Dobrev, S., Flocchini, P., Prencipe, G., and Santoro, N. Searching for a black hole in arbitrary networks. In *21st ACM Symposium on Principles of Distributed Computing (PODC '02)* 2002, pp. 153–162.

[15] Dobrev, S., Flocchini, P., Prencipe, G., and Santoro, N. Multiple agents rendezvous in a ring in spite of a black hole In *7th Symposium on Principles of Distributed Systems (OPODIS '03)*, LNCS, 2003.

[16] Domel P., Ligniau A., and Drobnik., O. Mobile agent interaction in heterogeneous environments. In *Mobile Agents '97*, LNCS 1219, Springer Verlag, 1997, pp. 136-148.

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

[18] Flocchini, P., Mans, B., and Santoro, N. Sense of direction in distributed computing. *Theoretical Computer Science 291*, 2003, 29–53.

[19] Flocchini P., Kranakis E., Krizanc D., Santoro N., and Sawchuk C.. Multiple mobile agent rendezvous in a ring. In *Proc. of Int. Conference on Latin American Theoretical Informatics , (LATIN '04)*, 2004.

[20] Flocchini, P., Prencipe, G., Santoro, N., and Widmayer, P. Gathering of asynchronous mobile robots with limited visibility. *Theoretical Computer Science*, 2005.

[21] Flocchini, P., Roncato, A., and Santoro, N. Computing on anonymous networks with sense of direction. *Theoretical Computer Science*, 301, 2003, 355-379.

[22] Flocchini, P., Roncato, A., and Santoro, N. Backward consistency and sense of direction in advanced distributed systems *SIAM Journal on Computing*, 32:2, 2003, pp.281-306.

[23] Howard, J. Rendezvous search on the interval and circle. *Operations Research 47*, No.4, 1999, pp. 550-558.

[24] Kranakis, E., Krizanc, D., Santoro, N., and Sawchuk, C. Mobile agent rendezvous in a ring. In *23rd Int. Conference on Distributed Computing Systems (ICDCS'03)* 2003, pp. 592-599

[25] Itai, A., and Rodeh, M. Symmetry breaking in distributed networks. *Information and Computation 88*, 1, 1990, pp. 60-87.

[26] Lim, W.S., Beck, A., and Alpern, A. Rendezvous search on the line with more than two players. *Operations Research 45*, pp. 357-364, 1997.

[27] Suzuki, I., and Yamashita, M. Distributed anonymous mobile robots: formation of geometric patterns. *Siam Journal on Computing 28*, 4, 1999, pp 1347–1363.

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

[29] Yamashita, M., and Kameda, T. Leader election problem on networks in which processor identity numbers are not distinct. *IEEE Transaction on Parallel and Distributed Systems 10*, 9 (1999), 878–887.

[30] Yu, X., and Yung, M. Agent rendezvous: A dynamic symmetry-breaking problem. In *International Colloquium on Automata, Languages, and Programming (ICALP '96)*, LNCS 1099, , 1996, pp. 610-621.