Symmetry Breaking in Anonymous Networks: Characterizations

Paolo Boldi
Shella Shammah
Sebastiano Vigna
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
Milano (Italy)

Bruno Codenotti*
Istituto di Matematica
Computazionale del CNR
Pisa (Italy)

Peter Gemmell†
Sandia National Laboratories
Albuquerque NM (USA)

Janos Simon‡
Department of Computer Science
The University of Chicago
Chicago IL (USA)

Abstract

We characterize exactly the cases in which it is possible to elect a leader in an anonymous network of processors by a deterministic algorithm, and we show that for every network there is a weak election algorithm (i.e., if election is impossible all processors detect this fact in a distributed way).

1 Introduction

We consider the problem of electing a leader in an anonymous network of processors. More precisely our model is that of a directed graph, with vertices corresponding to processors, and arcs to communication links (we freely interchange symmetric digraphs and undirected graphs). We make no assumption on the structure of the network: self-loops and parallel arcs are allowed. In particular, processors are *anonymous*: they do not have unique identifiers.

We consider both synchronous and asynchronous processor activation models, and models with and without "port awareness" (local names for outgoing and/or for incoming arcs). We consider both unidirectional and bidirectional links. Our models will be defined precisely in the sequel.

It is well known that once a leader is found, many other properties of the network can be computed easily. For example, the leader can grow a breadth-first tree, count the number of vertices in the network, assign unique identifiers to them, etc. Conversely, if each processor has a unique identifier, one can elect the processor with smallest (or largest) identifier to be the leader. When dealing with randomized algorithms, it is easy to verify that $O(\log n)$ random bits per processor will almost surely result in a unique processor with minimum name in an n-processor network.

Leader election is a basic *symmetry breaking* operation in distributed computing. Intuitively, it is possible to elect a leader in networks with "built-in" asymmetry, and impossible to do so in "symmetric" networks. Our paper gives a precise characterization of this intuition.

It turns out that the kind of asymmetry that can be detected in a distributed fashion depends on the particular model of computation used. For example, the standard definitions of asynchronous models (defined below) let us assume that at any given time only one processor is active. This provides enough asymmetry to ensure that a leader can be elected in K_2 (a single edge connecting two vertices, a very symmetric graph) [1]. On the other hand, it is not hard to prove that no synchronous algorithm can elect a leader in this graph. Similarly, *port awareness*, the existence, at each vertex, of a local numbering of the arcs incident at the vertex (their position in the adjacency list) may introduce some non obvious asymmetry. For example, the complete bipartite graph $K_{2,3}$ has nontrivial automorphisms, so we would expect that no leader election is possible, at least by synchronous algorithms. This is not the case: there is an easy algorithm to elect a leader in *any* labelling of $K_{2,3}$ (see Section 8).

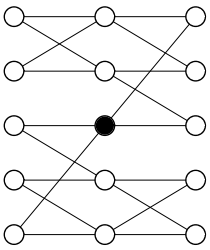
A natural conjecture is that leader election is impossible if and only if for any vertex the graph has an automorphism that moves the vertex (respecting port labels if such labels

*Partially supported by ESPRIT Basic Research Action, Project 9072 "GEPPCOM".

†Portions of this work were done while visiting IMC-CNR in Pisa, partially supported by GNIM-CNR.

‡Portions of this work were done while visiting IMC-CNR in Pisa, sponsored by a grant from CNR.

exist). The condition is clearly necessary, since if we start all processors in the same state, the processors in an orbit of the automorphism will remain in the same state: in particular, all processors (or none) will be elected. Unfortunately, the converse does not hold: the following graph has a distinguished vertex that is not moved by any automorphism, yet we can prove, using our characterizations, that no distributed algorithm can elect a leader in the graph.



These examples should suggest that the problem is in fact interesting. While from a practical standpoint one may argue that assigning unique identifiers to processors is a simple task, and one can easily spare the relatively few bits needed to do so, there are both theoretical and practical reasons to study anonymous networks. In practice, it is unclear how to assign unique identifiers in a distributed fashion, and there may be reliability reasons to assume anonymity (identifiers may get lost, corrupted, etc.). Still, we do not claim that our results are practical. Rather, we believe that distributed symmetry breaking is a worthwhile case study of characterizing the extent to which local information can be used to establish global properties in distributed computations.

2 History

The seminal paper in the area is due to Angluin [1]. She defined the problem and used the notion of *covering* from topological graph theory to capture sufficient conditions for impossibility of leader election on certain anonymous networks [1]. The model of computation used in the paper is rather unorthodox, and is based on Milne and Milner's theory of communicating sequential processes [13], but the proof works for all reasonable models of parallel computation, including the ones we use. She proves that if a network G has the property that there is a smaller network H and a morphism of G into H which is locally an isomorphism (in which case G is said to be a *covering* of H), then there cannot be a deterministic leader election algorithm in G. This implies, for example, that it is impossible to elect a leader in a cycle of composite length.

Unfortunately, Angluin's condition is necessary, but not sufficient. The need for sufficient conditions is illustrated by the fact that until the appearance of the work of Burns and Pachl [6] it was not clear that in rings of prime size it *is* possible to elect a leader in the asynchronous model (in fact with processors that are finite automata [10]).

Further developments are less well known. An interesting paper by Johnson and Schneider [11] introduced port labellings (as part of the tools the algorithm designer could use, although this was not the formalism used in the paper), defined "local views" that processors could obtain, and observed that the Dining Philosophers Problem *could* be solved in some cases (for example when $n = 6$).

Yamashita and Kameda [17], building on the paper above, characterized the solvable instances of the leader election problem in the case of synchronous networks of anonymous processors, by graph-theoretic terms. (The model used in [17] is called asynchronous, but in fact all asynchrony is in the communication mechanism, and corresponds to a synchronous operation.)

In this paper we consider both synchronous and asynchronous networks using a number of communication models. In two of the eighteen cases we study our results correspond to the ones of [17] and [18].

Self-stabilization has been introduced by Dijkstra in his classic 1974 paper [8], and since then has been viewed as a convenient property of distributed computations. Subsequently, there has been a considerable effort towards characterizing network operations that admit self-stabilizing algorithms (see, e.g., [16, 9]). In the case of leader election, an algorithm is self-stabilizing if a leader is eventually elected, even though an adversary can set the initial configuration of the network to an arbitrary state. Our leader election algorithms can be made self-stabilizing, although in this abstract we will only sketch the necessary ideas.

Our results, in a sense, go back to the roots of Angluin's original paper. We establish a very simple notion of *fibration* of graphs (a topological notion weaker than covering) based on the corresponding categorical definition, and we show that the existence of certain fibrations characterizes exactly the (im)possibility of election.

3 Graph-theoretical definitions

A *(directed) (multi)graph* G is defined by a nonempty set V_G of vertices and a set A_G of arcs, and by two functions $s_G, t_G : A_G \to V_G$ which specify the source and the target of each arc (we shall drop the subscript whenever no confusion is possible). A *self-loop* is an arc with the same source and target. A *symmetric graph* is a graph endowed with a symmetry, i.e., a self-inverse bijection $\overline{\ } : A \to A$ such that $s(a) = t(\overline{a})$ and $t(a) = s(\overline{a})$ for all arcs $a \in A$. An *(arc-)coloured graph* (with set of colours $\mathscr{C}$) is a graph endowed with a colouring function $\gamma : A \to \mathscr{C}$. A *symmetrically coloured graph* is a symmetric graph coloured on $\mathscr{C}$ and endowed with a self-inverse bijection $\overline{\ } : \mathscr{C} \to \mathscr{C}$ such that $\gamma(\overline{a}) = \overline{\gamma(a)}$. A coloured graph is *deterministic* iff all arcs outgoing from a vertex have distinct colours.

4 The model

We consider a number of models, corresponding to the choice of the model of activation (synchronous vs. asynchronous) and to the way a processor interacts with its links. In all models there is an underlying graph, where each arc corresponds to a link between processors (parallel links and self-loops are allowed).

In synchronous models we assume the existence of a global clock. At each clock cycle, every processor executes one step of its computation. The step may depend on the state of adjacent processors. (Although this model, inspired by cellular automata, is not very realistic for distributed systems, it can be easily simulated by an asynchronous message-passing network [18].)

Our model of asynchronous distributed computation is somewhat artificial: it is not inspired by actual practice, but it is a model that has been the most extensively used in more theoretical papers. It was suggested by Dijkstra, and it is inspired in his formalism of "guarded commands": it makes possible to reason about distributed systems in exacty the same way as one does about ordinary programs.

One assumes that at any moment there are several *enabled processors* that could take a step, and a *central demon* [6] chooses one of them. As a result, the set of enabled processors changes. The process is then repeated.

The technique essentially serializes a distributed computation. Since speeds of processors are variable and unknown, and, as we know from the work of Lamport, the notion of time in a distributed system should be defined not as a fixed global time, but as an order that is the consequence of exchanges of messages, the definition seems good and elegant. Since the selection of processor to be run is done by an adversary, the intuition is that there is no possible advantage/knowledge to be gained by the serialization.

The model has been used in the classic papers of Dijkstra on self-stabilization, the Burns-Pachl election algorithm for prime size rings, and in most of the theoretical papers on self-stabilization.

In fact, the adversary does break symmetry, choosing only one processor to run. As a result, it is possible to elect a leader in K_2 in this model, while it is impossible to do so in a synchronous model. Note that if we assume the model to be asynchronous, and deal with continuous time, the probability that two events are exactly simultaneous will be 0, if processors have random speeds. Thus, one can argue, that the only behavior excluded by the model is a low probability one.

We consider then several conditions for the links. Processors may have *output port awareness*, i.e., the arcs of the graph G may have a *local output labelling*. If the vertex v has outdegree d, then it uses the numbers $\{1, 2, \ldots, d\}$ for the labelling, a distinct number for each outgoing arc.

We can think of this as the processors being aware of which output port is associated with a given link; in the case there is no output port awareness, processors can only transmit by *broadcast* [18].

Analogously, processors may have *input port awareness*, i.e., the arcs of the graph G may have a *local input labelling*. If the vertex v has indegree d, then it uses the numbers $\{1, 2, \ldots, d\}$ for the labelling. If there is no input port awareness, the processors receive messages in a *mailbox* [18].

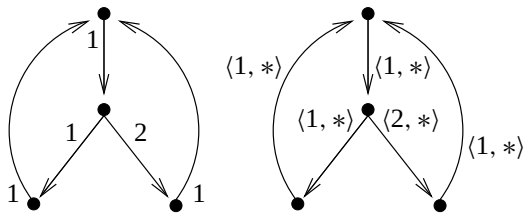
When designing our algorithms, we assume not to have control of this local labelling: our algorithms must work no matter how an adversary chooses to associate links with numbers.

Depending on the graph being symmetric or not, and on the combination of input and output port awareness, we obtain eight models. A ninth model is given by *(complete) port awareness*, in which case not only the graph is symmetric and has input/output port awareness, but also each processor uses the same number for each pair of symmetric input/output arcs (note that in general this does not happen in a symmetric network with input and output port awareness). We can think of this as the processors being aware of which input and output arcs connect to the same neighbour.

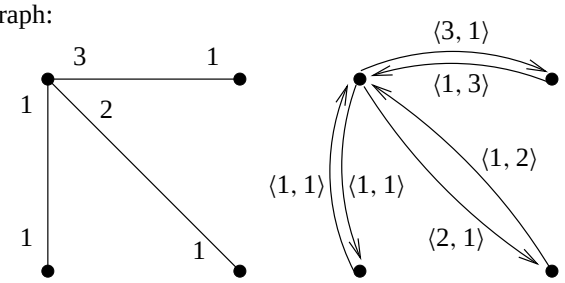
Combining our nine communication models with the possibility of synchronous or asynchronous activation we obtain eighteen models.

We can describe compactly and uniformly all of our link models as follows: the local labellings induce a standard arc-colouring of G on the set $\mathscr{C} = (\mathbf{N} + \{*\})^2$. Each of the vertices adjacent to an arc contributes to the colouring with a number, or with "$*$" in the case of port non-awareness. Namely, an arc labelled as $\alpha \in \mathbf{N} + \{*\}$ by its source processor and by $\beta \in \mathbf{N} + \{*\}$ by its target processor will get the colour $\langle \alpha, \beta \rangle$.

In the following picture, we show an example of a graph with output (but no input) port awareness and of the related coloured graph:



Here instead we show an undirected (i.e., symmetric) graph with complete port awareness and the related coloured graph:



Formally, a *network* is a graph with a colouring induced by a local labelling. Each processor (node) has state space X and transition function $f : X \times (\mathscr{C} \times X)^{\oplus} \to X$, where $(\mathscr{C} \times X)^{\oplus}$ is the set of multisets over $\mathscr{C} \times X$. All processors start from the same initial state. At each step of computation, a processor computes its new state on the basis of the (coloured) neighbourhood relation. Namely, if a processor p in state x has k incoming arcs, with colours $c_1, c_2, \ldots, c_k$ (not necessarily distinct) and sources given by processors $p_1, p_2, \ldots, p_k$ (which do not need to be distinct, or different from p) in state $x_1, x_2, \ldots, x_k$, then the next state of p is $f(x, \{\langle c_1, x_1\rangle, \langle c_2, x_2\rangle, \ldots, \langle c_k, x_k\rangle\})$. The *orbit* of a processor is the sequence of states of X through which the processor passes during the computation of the network.

5 The election problem

An *anonymous election algorithm* for a class of networks is a transition function f inducing a computation in which all processors eventually reach one of two distinguished states denoting election and non-election, and exactly one reaches the election state (note again that this must happen for all networks of the given class). f is supposed to have the given states as fixed points (no matter which are the states of the in-neighbours).

An *anonymous weak election algorithm* for a class of networks is a transition function f inducing a computation in which all processors eventually reach one of three distinguished states denoting election, non-election and impossibility of election. When all processors reach their final states, exactly one of the following two statements holds:

- all processors are in the third state, and there is no anonymous election algorithm for the network on which the weak election algorithm is running;
- all processors but one are in the non-election state, and the remaining one is in the election state.

Previous literature has overlapped the election and the weak election problem. The anonymous election problem is known to be unsolvable in general since [1]; as a result, a weaker problem was introduced (using, unfortunately, the same name), in which a distributed detection of unsolvability is allowed. There is however an important difference between our definition and the one given, for instance, in [18]: our algorithms must detect impossibility of election when election is impossible *for the specific network on which the algorithm is running*, meaning that election must be impossible with the given local labelling. In the abovementioned reference, it is admissible for an algorithm to give up even if election is possible on the actual labelling chosen by the adversary, provided that the adversary *could* find a labelling making election impossible. On the contrary, we require our algorithm to exploit the mistakes of the adversary, and to perform election whenever possible. This fact makes the problem more difficult, and it will give rise to some interesting pathologies described in Section 8.

Note that each processor may possess more or less information about the network it belongs to. Whenever we say that election is possible given the knowledge of certain data, we mean that there exists an algorithm which performs election on all networks satisfying the constraints.

6 Graph fibrations

The fundamental idea behind this paper is that processors which are connected by the same colours to processors behaving in the same way (with respect to the colours) will behave alike. This idea, which was first formalized in [11], and was used subsequently in [17, 18], is exactly captured by the notion of *fibration*, which originated from homotopy theory. The elementary definition we shall use [4] is derived from the categorical abstraction of fibrations between topological spaces.

Recall that a *graph morphism* $G \to H$ is given by a pair of functions $f_V : V_G \to V_H$ and $f_A : A_G \to A_H$ which commute with the source and target functions, i.e., $s_H \circ f_A = f_V \circ s_G$ and $t_H \circ f_A = f_V \circ t_G$. In other words, a morphism maps nodes to nodes and arcs to arcs in such a way to preserve the incidence relation, because the two commuting conditions are to be read "the source (target) of the image of an arc is the image of the source (target) of the arc". In the coloured case, we require also that $\gamma_G = \gamma_H \circ f_A$, i.e., that the map on the arcs preserves colours.

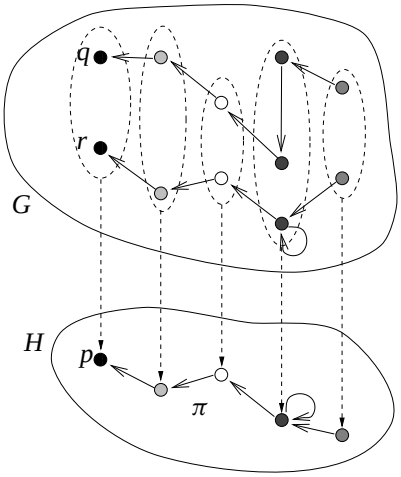
Definition 1 A *fibration*[1] between (coloured) graphs G and B is a morphism $\varphi : G \to B$ such that for each arc $a \in A_B$ and for each $p \in V_G$ such that $\varphi(p) = t(a)$ there is a unique $\widetilde{a}^p \in A_G$ such that $\varphi(\widetilde{a}^p) = a$ and $t(\widetilde{a}^p) = p$.

We recall some topological terminology. If $\varphi : G \to B$ is a fibration, B is called the *base* of the fibration. We shall also say that G is *fibred (over B)*. The *fibre* over a vertex $p \in V_B$ is the set of vertices of G which are mapped to p, and will be denoted by $\varphi^{-1}(p)$. A fibre is *trivial* if it is a singleton, i.e., if $|\varphi^{-1}(p)| = 1$. A fibration is *proper* if every fibre is nontrivial; it is *discrete* if the subgraph induced by each fibre is totally disconnected in the strong sense (i.e., iff it is acyclic).

The geometric meaning of the definition of fibration is that given a vertex $p \in V_B$ and path π terminating at p,

[1] The name we have given is justified by the following fact: a morphism between two graphs induces a functor between the free categories generated by the graphs; the functor is a fibration in the categorical sense (see [5]) exactly when the morphism satisfies our definition. Note also that the notion of *divisibility of graphs* (see [7]), known from the sixties in the algebraic theory of graphs, is related to the theory of graph fibrations because there is a fibration $G \to B$ if and only if B is a *rear divisor* of G. However, in that theory the main interest is in the arithmetic notion of divisibility, rather than in the geometric notion of fibration.

for each vertex q in the fibre over p there is a unique path terminating at q which is mapped by the fibration on π; this path is called the *lifting of* π *at* q. In the following picture, fibres are represented by dotted ovals (not all nodes of a fibre are shown, though), and we indicate how a path can be lifted at two different points of a fibre. Note that self-loops are not necessarily lifted to self-loops.

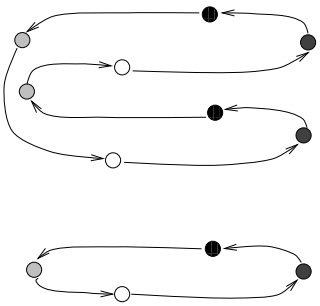


There is a very intuitive characterization of fibrations based on the concept of local isomorphism. A fibration $\varphi : G \to B$ induces an equivalence relation between the vertices of G, whose classes are precisely the fibres of φ. When two vertices p and q are equivalent (i.e., they are in the same fibre), there is a one-to-one correspondence between arcs ending in p and arcs ending in q which preserves colours, and such that the sources of any two related arcs are equivalent.

A covering is a special kind of fibration, where each arc can also be lifted uniquely from its tail; more formally:

Definition 2 A fibration $\varphi : G \to B$ is a *covering* if, for every arc $a \in A_B$ and every vertex $p \in V_G$ such that $\varphi(p) = s(a)$, there is a unique ${}^p\widetilde{a} \in A_G$ such that $\varphi({}^p\widetilde{a}) = a$ and $s({}^p\widetilde{a}) = p$.

In the case of coverings, the local isomorphism property gives a bijective correspondence between the whole (disjoint) neighbourhoods of two nodes in the same fibre. In the following picture, we show graphically how an eight-cycle covers a four-cycle:



A covering $\varphi : G \to B$ between two symmetric graphs is a *symmetric covering* if and only if it commutes with the symmetries, i.e., for all $a \in A_G$ we have $\varphi(\overline{a}) = \overline{\varphi(a)}$.

We will now state some properties and constructions of fibrations. Recall that all our graphs are strongly connected.

Let G be a graph and p a vertex of G. We define a (possibly infinite) in-directed rooted arc-coloured tree $\widetilde{G}^p$ as follows:

- the nodes of $\widetilde{G}^p$ are the (finite) paths of G ending in p, the root of $\widetilde{G}^p$ being the empty path;
- there is an arc from the node π to the node π' if π is obtained by adding an arc a at the beginning of π' (the arc will have the same colour as a).

We can define a graph morphism υ_G^p from $\widetilde{G}^p$ to G, by mapping each node π of $\widetilde{G}^p$ (i.e., each path of G ending in p) to its starting vertex, and each arc of $\widetilde{G}^p$ to the corresponding arc of G.

The following important property holds:

Lemma 1 For every vertex p of a graph G, the morphism $\upsilon_G^p : \widetilde{G}^p \to G$ is a fibration, called the *universal fibration of* G *at* p.

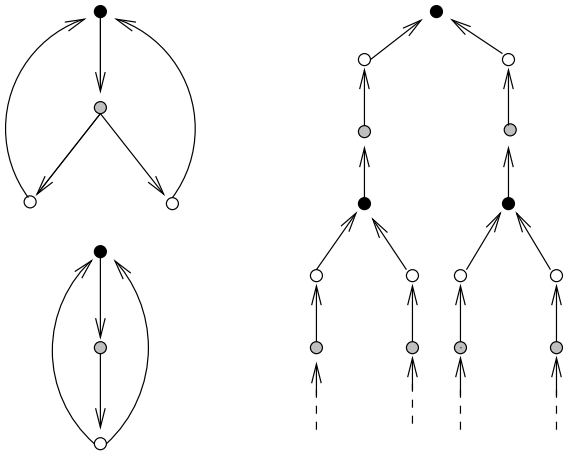
The universal fibration at p is a tree representing intuitively "everything processor p can learn from interaction with its neighbours" (in fact, the *local view* of [11] is essentially its universal fibration).

Now, we define $\widehat{G}$ as the graph obtained from $\widetilde{G}^p$ by identifying isomorphic subtrees; it is easy to verify that this construction does *not* depend on the choice of p. Clearly, one can construct a morphism $\mu_G : G \to \widehat{G}$ mapping each vertex $p \in V_G$ to (the equivalence class of) $\widetilde{G}^p$, and each arc of G to the corresponding arc in $\widehat{G}$.

Lemma 2 For each graph G, the morphism $\mu_G : G \to \widehat{G}$ is a fibration; moreover, if $\varphi : G \to B$ is a fibration of G, there is a fibration $\psi : B \to \widehat{G}$ s.t. $\psi \circ \varphi = \mu_G$ (i.e., μ_G factors through any fibration of G). As a consequence, G is properly fibred over some graph iff μ_G is proper.

In the following picture, we illustrate this notions by

showing a graph G, $\widehat{G}$ and $\widetilde{G}^p$:



Now, we have to face the problem of how a node can build $\widehat{G}$ in an effective, distributed way; the following result (which extends the one given in [14]) is a step in this direction:

Theorem 1 If G has n nodes, for all processors p, q, $\widetilde{G}^p \cong \widetilde{G}^q$ iff there is an isomorphism between the first n levels of the two trees.

Thus, we can build $\widehat{G}$ as follows. Each processor p starts to build $\widetilde{G}^p$; at the beginning of the computation, $\widetilde{G}^p$ is the empty tree. Then, knowing $\widetilde{G}^q$ truncated at height k for all its in-neighbours q, the processor p can build $\widetilde{G}^p$ truncated at height $k+1$, and so on. By Theorem 1, after $2n$ iterations (in fact, after $n+\delta$ iterations, where δ is the diameter of G) the truncated tree so far obtained has the same subtree isomorphism classes as the whole $\widetilde{G}^p$, and so $\widehat{G}$ can be built (appropriate counter-based synchronization is necessary in case of asynchronous activation).

Note that only the knowledge of n is required: the construction of $\widehat{G}$ does not depend on the knowledge of G. Moreover, each processor p knows which fibre of μ_G it belongs to, because—by construction—the universal fibrations of $\widehat{G}$ coincides with those of G, in the sense that $p \in V_G$ and $q \in V_{\widehat{G}}$ have the same universal fibration iff $q = \mu_G(p)$.

7 Characterizing election for fixed labellings

In this section we characterize the networks (with a given local labelling) for which election is possible. This will give us necessary conditions for the characterization in the case an evil adversary can choose the local labellings. In fact, we shall see that the conditions we derive will be also sufficient.

The cornerstone of our theory is the following

Theorem 2 For each (labelled) network G there is a weak election algorithm. Moreover, there is an election algorithm exactly when G is not properly fibred (in the asynchronous activation model, only discrete fibrations are considered).

Proof. If G is properly fibred (discretely, in the asynchronous case), all processors terminate immediately in the "impossibility of election" state.

Otherwise, we have already described how each processor can determine the fibre of μ_G it belongs to. In the synchronous model, if G is not properly fibred a leader will be elected among those processors with trivial fibre, for instance by ordering lexicographically the trees $\widetilde{G}^p$.

In the asynchronous model, μ_G must necessarily be nonproper or nondiscrete. In the first case, we can again elect a leader among those with trivial fibre. In the other case, the processors in the same fibre label themselves either A or B according to the following rule:

- if the processor wakes up and all in-neighbours in its fibre are unlabelled, it labels itself A;
- if the processor wakes up and any in-neighbours in its fibre is labelled, it labels itself B.

At the end of this "A-B game", the processors compute again the fibres of μ_G, this time taking into account the A-B identifier they possess. This process can be repeated, and can only terminate when μ_G is nonproper; this happens because we assumed that no proper discrete fibration existed, and at each step every nontrivial, nondiscrete fibre breaks at least into two pieces (in such a fibre, the first activated processor labels itself A, and among the processors belonging to an oriented cycle the least activated processor necessarily labels itself B).

On the other hand, in the synchronous case the orbits of processors in the same fibre are exactly the same; this happens because the local isomorphism property says that for any two processors p and q in the same fibre, and any c-coloured arc terminating at p and starting from p', there is a c-coloured arc terminating at q and starting from a processor in the same fibre as p'; moreover, this association is a colour preserving bijection between the arcs entering in p and q. Thus, two processors in the same fibre are either elected or not elected at the same time; consequently, when no fibre is trivial it is impossible to terminate with exactly one leader.

In the asynchronous case, the adversary schedules the processors fibrewise. The scheduling starts from a sink of the fibre (i.e., from a node with no outcoming edges), and continues inductively on the subgraph induced by the remaining processors. This is possible because the fibres are discrete, and it guarantees that all processors in the same fibre remain in the same state; being the fibration proper, it is impossible to terminate with exactly one leader. ∎

It is clear that too much knowledge is required in this theorem (in fact, complete information about G). It is not

advisable to assume that complete knowledge of the network topology is at the processors' disposal. However, the theorem is relevant because in a number of special cases we can derive complete knowledge of the network from much less data.

The following theorems show that under certain assumptions on the local labellings, only certain types of fibration can really exist.

Theorem 3 If G has output port awareness, then all fibrations $\varphi : G \to B$ are coverings.

Proof. Note that under the given assumptions G has an induced colouring which is deterministic. Let p and q be any pair of vertices of B. One can easily build an injection from the fibre of q to the one of p by lifting a path connecting p to q at each element of the fibre of q and taking the starting vertex of the resulting path. This association is necessarily injective, for otherwise two arcs with the same label should exit from a node along the path. This implies $|\varphi^{-1}(q)| \leq |\varphi^{-1}(p)|$ for all p and q.

Consider now an arc a of B which goes from p to q. There are exactly $|\varphi^{-1}(q)|$ liftings of a going from the fibre over p to the fibre over q, and any two of them cannot start from the same node because of determinism. Thus, by pigeonholing, a can be tail-lifted uniquely from every node in the fibre of p, and φ is a covering. ∎

Theorem 4 If G has complete port awareness, then for all fibrations $\varphi : G \to B$ we have that B is coloured symmetrically, and φ is a symmetric covering.

Proof. Note that under the given assumptions G is coloured deterministically and symmetrically (the symmetry is given by $\overline{\langle j,k\rangle} = \langle k,j\rangle$). By Theorem 3, φ is certainly a covering. We have to show that B is symmetrically coloured, and that φ commutes with the symmetries of G and B.

Consider an arc a of B going from p to q. Let r be an element of the fibre over q, and $\widetilde{a}^r$ the corresponding lifting of a. Then we define $\overline{a} = \varphi(\overline{\widetilde{a}^r})$; in other words, we lift a, we take the symmetric in G, and we map it with φ in B. This association is independent of the choice of r, because there cannot be two parallel arcs with the same colour; for the same reason, it is a self-inverse bijection and it satisfies the commutation condition.

Finally, we show that $\overline{(\ )}$ commutes with the symmetry on the colours. This is easy, as for any arc a of B

$$\gamma(\overline{a}) = \gamma(\varphi(\overline{\widetilde{a}^r})) = \gamma(\overline{\widetilde{a}^r}) = \overline{\gamma(\widetilde{a}^r)} = \overline{\gamma(\varphi(\widetilde{a}^r))} = \overline{\gamma(a)}.$$ ∎

8 Election in spite of an evil adversary

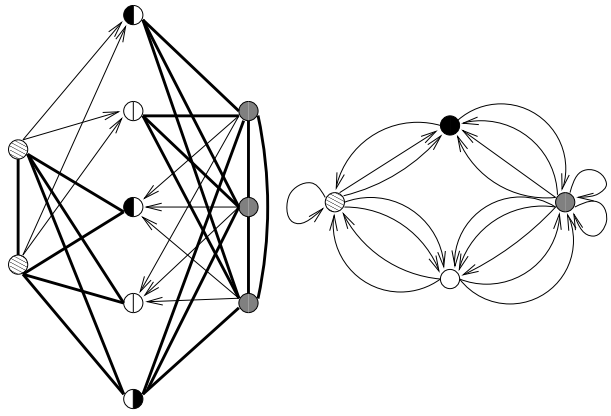
In this section we characterize those networks in which election is possible, given that an evil adversary has the possibility of choosing the local labellings. In other words, given an uncoloured graph H, we must provide an algorithm which will weakly elect a leader on all networks using a certain communication/activation model and having H as an underlying graph (said otherwise, the processors have knowledge of H, but not of its colouring). In order to mark the distinction with the previous discussion, we shall use the letter H to denote an uncoloured graph.

We shall use several time the idea of *lifting a colouring*: whenever we have a morphism $\varphi : H \to B$, and B has a colouring, the arcs of H can be coloured using the same colour as their image, i.e., $\gamma(a) = \gamma(\varphi(a))$. This guarantees that φ is still a morphism between H and B seen as coloured graphs.

In the next section, we shall provide weak election algorithms, working in the presence of output port awareness (Theorem 10) or symmetry (Theorem 9), which only require the knowledge of the number of processors; in the case of no awareness only one colouring is possible, and we can apply Theorem 2. Thus, we have *a fortiori* the following

Theorem 5 There is a weak election algorithm for every graph H, except in the case H is not symmetric, and the model has input port awareness but no output port awareness.

The (perhaps surprising) fact that in the case of non-symmetric graphs with input (but no output) port awareness weak election is not possible is witnessed by the following counterexample:



The graph H on the left of the picture is strongly connected (we used thick lines to represent pairs of arcs with opposite direction), and can be fibred onto the graph B on the right with two different fibrations φ and ψ (represented by the colours of each half of the nodes). Note that ψ is proper, while φ is not. It is easy to give local input labellings to B in such a way that the resulting coloured graph has no non-trivial fibrations; by lifting its colouring to H using φ and ψ we obtain two different networks, with the same underlying

graph, such that only the second one admits election[2]. Thus, in the case of input port awareness only, no weak election algorithm exists.

We conclude this section by classifying the graphs in which election is possible.

Theorem 6 In all models without output port awareness, a graph H has an election algorithm exactly when it is not properly fibred (in the asynchronous activation model, only discrete fibrations are considered).

Proof. The election algorithm is essentially the same as in Theorem 2, but colours are ignored: the hypotheses guarantee that a leader can be elected (there is at least a trivial fibre).

In order to show that no election algorithm exists if H is not properly fibred, we prove that, given a proper fibration $\varphi : H \to B$, there is a way of assigning local labellings in such a way to make election impossible.

First of all, we label locally B using $*$ for all outcoming arcs, and using $*$ or distinct numbers for the incoming arcs, depending on the presence of input port awareness. This local labelling induces a colouring on B, which we lift to H. Now the lifting property ensures that this colouring of H induces a local labelling. ∎

Note that in the following two cases the weak algorithm of Theorem 5 applies.

Theorem 7 In all models with output (but not complete) port awareness, a graph H has an election algorithm exactly when it does not cover properly another graph (in the asynchronous activation model, only discrete coverings are considered).

Proof. We just have to prove that a proper coloured fibration of H exists for some local labelling if and only if H properly covers some other graph. The proof is analogous to Theorem 6. The only difference is that we must label locally B using distinct numbers for the outcoming arcs. The lifting and tail-lifting properties ensure that this induces a local labelling on H. ∎

It should be noted that the previous two theorems are independent both of the presence of input port awareness and of the presence of symmetry. This fact, together with the following theorem, shows that bidirectionality does not help in breaking symmetry unless complete knowledge of the incoming and outcoming ports and of their association is available.

Theorem 8 In all models with complete port awareness, a graph H has an election algorithm exactly when it does not cover properly and symmetrically a graph (in the asynchronous activation model, only discrete coverings are considered).

Proof. If the adversary has a way of choosing the labelling so that election is impossible, then H *a fortiori* covers properly and symmetrically. We just have to show that given a proper symmetric covering $\varphi : H \to B$ there is a way of assigning local labellings in such a way to make election impossible.

The labelling is obtained by lifting the colouring of B defined as follows: first of all, we label locally all arcs which are not self-loops, using the same number for incoming and outcoming arcs related by the symmetry. Then we divide the self-loops at each node into the ones fixed by the symmetry, which are labelled locally using the same number for input and output, and the remaining ones, which are labelled in such a way that if a has input label j then $\overline{a}$ has output label j.

We have to show that the lifting of this colouring induces a local labelling on H which gives complete port awareness. Let a be an arc of H with label $\langle j, k\rangle$. Then, since φ is symmetric,

$$\gamma(\overline{a}) = \gamma(\varphi(\overline{a})) = \gamma(\overline{\varphi(a)}) = \overline{\gamma(\varphi(a))} = \overline{\gamma(a)} = \langle k, j\rangle,$$

so that a and $\overline{a}$ are labelled locally in the same way. ∎

9 Election knowing the number of nodes

Up to this point, we characterized the graphs in which election is possible, and we provided weak election algorithms for single networks. Now we start showing that under topological assumptions weak election is possible knowing much less information, i.e., the number of nodes. In other words, our evil adversary has now the possibility of completely configuring the network, provided it is strongly connected.

Theorem 9 If H is symmetric and all processors know the number n of nodes, then weak election is possible.

This theorem is an immediate consequence of the following graph-theoretical lemma:

Lemma 3 Let $\varphi : H \to B$ be a fibration, where H is a symmetric graph. For all vertices p and q of B let $d_{pq} = |\{a \in A_B \mid s(a) = p \text{ and } t(a) = q\}|$. Then $|\varphi^{-1}(p)|d_{qp} = |\varphi^{-1}(q)|d_{pq}$ holds for all $p, q \in V_B$.

Let now k be the number of nodes of $\widehat{H}$, and m_1, $m_2, \ldots, m_k$ the cardinality of the fibres of $\mu_H : H \to \widehat{H}$. Since $\widehat{H}$ has at least $k-1$ distinct (unordered) pairs of connected nodes (by strong connection), the previous lemma

[2] In order to extend the counterexample to the asynchronous activation case, one just needs to remove the self-loops and the relative liftings from B and H, connect the left and right nodes of H in such a way to form $K_{2,3}$, and update B correspondingly.

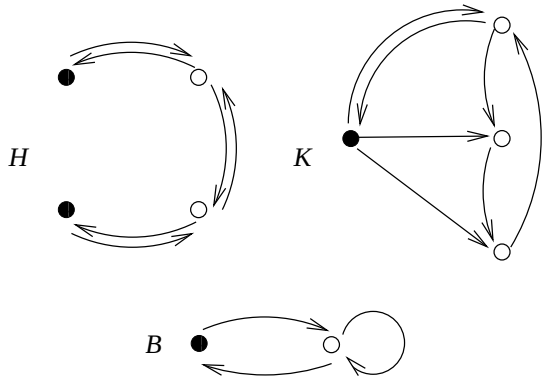
gives us at least $k-1$ independent homogeneous linear constraint on the m_i's. Then the equation $m_1+m_2+\cdots+m_k = n$ forces the system to have at most one solution. Since all processors can build $\widehat{H}$, knowing just the number of nodes they can also solve the system; then we just have to apply Theorem 2.

Note that the symmetry condition does *not* imply that the graph is symmetrically coloured.

Finally, using the fact that in a covering all fibres have the same cardinality, it is easy to prove the following theorem:

Theorem 10 If H has output port awareness and all processors know the number of nodes, then weak election is possible.

The following three graphs provide an evidence that the knowledge of the number of nodes is not always sufficient to guarantee that weak election is possible:



One can easily see that B is a base of both H and K. Thus, a processor cannot know if it belongs to H or K, and consequently it cannot decide whether election is possible or not (because the black fibre is trivial for H but not for K).

Our theorem has some simple corollaries, providing possibility and impossibility results for specific graphs.

Corollary 1 For every $n > 1$ it is impossible to elect a leader in K_n in the synchronous model, even in the case of port awareness.

Corollary 2 With port awareness, there exists a leader election algorithm that successfully elects a leader in $K_{i,j}$ iff i,j are relatively prime.

Proof. With port awareness, all fibres have the same cardinality, because of Theorem 4. An immediate corollary of this is that with port awareness, there exists a leader election algorithm that successfully elects a leader in $K_{i,j}$ iff i,j are relatively prime.

Suppose C_1 and C_2 were incident fibres such that $|C_1| > |C_2|$. Then, letting E_α be the set of all edges with label α on the C_2 side, connected to C_1, there have to be vertices p,q in C_1 such that p is adjacent to E_α, but q is not. ∎

10 Further results and conclusions

In this section we just outline some additional results. A complete treatment of these aspects will appear in the full paper.

10.1 Election without knowledge

The algorithms sketched above make essential use of the knowledge of n, the number of vertices in the graph. However, it is possible to elect a leader on nonproperly fibred networks with complete port awareness even if n is not known.

First, what we mean in this case by "leader election" is that *if* a leader can be elected in G (in spite of adversary labeling of edges, and in spite of the schedule given by the central daemon), we will eventually succeed in electing a leader. Otherwise, we may not elect any processor, or may mistakenly designate several processors as leaders. Note that even when a leader is eventually elected, there may be a period during which several processors (mistakenly) believe themselves leaders: all we require is that a single processor be elected "in the limit".

Our general strategy is as follows. For $n = 1, 2, \ldots$ we run the algorithm above. For any n for which the algorithm succeeds, the leader imposes a breadth-first tree on the network, counts the number of vertices, and checks that it is n. This will succeed for the correct value of n (and also for some larger values.) Assuming that a leader can be elected in the network, we will find it as soon as n becomes the actual size of the network. In the synchronous case, the same leader will be elected for all greater values of n. In the asynchronous case, because of different schedules, we may get different leaders. We overcome this difficulty by checking the network for leaders elected in previous rounds, say when we had $n = p$. If there are any, we check whether it is consistent to assume that the network has size p. If so, the leader "abdicates" and lets the previous leader remain the leader. It is only if an error is detected that the previous leader is killed, and the new leader candidate actually becomes a leader if this happens for all smaller values of n.

10.2 Self-stabilization

We can strengthen our algorithms (when they exist) to be self-stabilizing. The self-stabilizing algorithms are far from practical. Due to space limitations, we give only a very rough sketch of the techniques needed to accomplish this.

The basic idea is to continuously restart the election algorithm: eventually it will correctly elect a leader. Also, the size of the network is not known (the adversary may have set the variables to arbitrary values), so we have to also try all possible values for the size of the network. Since we want to eventually elect always the same processor to be

the leader, we must rerun the algorithm so that if a processor was correctly elected by a previous process, we defer to it. To be able to establish what a "previous process" is, we first obtain a global self-stabilizing clock, using the techniques of [2]. We compose two self-stabilizing algorithms using the methodology of [15], so we may assume the existence of a correct clock. Each process carries with it the time it begun. The messy details are omitted from the abstract.

Note that this is not a realistic algorithm: the number of processes (and, therefore, message length) increases without bound.

10.3 Randomized leader election algorithms

Our characterizations for the possibility of leader election by deterministic algorithms allows us to obtain simple *probabilistic* algorithms for leader election in anonymous networks. These algorithms use independent coin tosses at each processor, and correctly elect a leader with high probability. As discussed before, it is easy to obtain such algorithms, either by tossing $O(\log n)$ coins at each processor to obtain distinct identifiers, or to play an elimination tournament among processors where individual games are decided by random coin tosses. Again in this case, we expect that a leader will be elected after $O(\log n)$ rounds, and thus $O(\log n)$ random bits will be used per processor. These bounds depend on n, the number of processors, but are independent of the underlying graph.

Our algorithms use a very small number of random bits. There are several reasons to use algorithms that use few random bits. It is difficult in practice to generate independent random strings in a distributed environment. Thus, random bits are a possibly scarce resource in distributed computations, and it is interesting to study the "randomness requirements" of distributed algorithms.

Consider the following randomized algorithm, for both the synchronous and the asynchronous model. Given a network with underlying graph G, for every node $v \in V_G$ assign independently at random the labels $1, 2, \ldots, d$ (where d is the degree of v) to the edges incident on v. (Note that this corresponds to simply associating each link with an arbitrary port.) From now on, our algorithm becomes the deterministic leader election algorithm described in our proofs.

Theorem 1 The probability that a unique leader will be elected is $1 - 1/\operatorname{poly}(n)$.

The proof is an estimate of the probability that there is an appropriate homomorphism after the random labelling of the edges, and is omitted from this abstract.

Our techniques also yield lower bounds on the number of random bits required for leader election.

Theorem 2 There are n-vertex graphs that require $\Omega(\log n)$ random bits per processor in order to elect a leader.

Proof. (Sketch) Consider K_n, the complete graph on n vertices in the model with no port awareness. We may assume that every probabilistic algorithm starts by tossing the appropriate number of coins at processors, and becomes a deterministic algorithm with access to these bits. If there are less than n distinct bit strings, there will be two processors with identical views, and thus we cannot elect a unique leader. ∎

10.4 Conclusions

We believe that graph morphisms are an elegant and powerful tool in the theory of distributed algorithms. Our fibration-based characterizations not only enabled us to settle the leader election question in several different models of computation, but provides upper and lower bounds for probabilistic algorithms. A graph morphism based technique was recently used to prove impossibility of self-stabilizing phase clocks [12], while fibrations were used to construct networks with optimal sense of direction [3]. We believe that there are many more potential applications.

References

[1] D. Angluin. Global and local properties in networks of processors. In *Proc. 12th Symposium on the Theory of Computing*, pages 82–93, 1980.

[2] A. Arora, S. Dolev, and M. Gouda. Maintaining digital clocks in step. *Parallel Processing Letters*, 2(1):11–18, 1991.

[3] P. Boldi and S. Vigna. Good fibrations and other constructions which preserve sense of direction. Submitted for publication, 1996.

[4] P. Boldi and S. Vigna. Graph fibrations. Preprint, 1996.

[5] F. Borceux. *Handbook of Categorical Algebra 2*, volume 51 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, 1994.

[6] J. Burns and J. Pachl. Uniform self-stabilizing rings. *ACM Trans. Progr. Lang. Syst.*, 11:330–344, 1989.

[7] D. M. Cvetković, M. Doob, and H. Sachs. *Spectra of Graphs*. Academic Press, 1978.

[8] E. Dijkstra. Self-stabilizing systems in spite of distributed control. *CACM*, 17(11):643–644, 1974.

[9] M. Gouda. The triumph and tribulation of system stabilization. In M. Helary and M. Raynal, editors, *Proc. of the 9th International Workshop on Distributed Algorithms*, number 972 in LNCS. Springer-Verlag, 1995.

[10] G. Itkis, C. Lin, and J. Simon. Deterministic, constant space, self-stabilizing leader election on uniform rings. In M. Helary and M. Raynal, editors, *Proc. of the 9th International Workshop on Distributed Algorithms*, number 972 in LNCS. Springer-Verlag, 1995.

[11] R. E. Johnson and F. B. Schneider. Symmetry and similarity in distributed systems. In *Proc. 4th conference on Principles of Distributed Computing*, pages 13–22, 1985.

[12] C. Lin and J. Simon. Possibility and impossibility results for self-stabilizing phase clocks on synchronous rings. In *Proc. of the 2nd WSS*, Las Vegas, 1995. University of Nevada.

[13] G. Milne and R. Milner. Concurrent processes and their syntax. *Journal of the ACM*, 26:302–321, 1979.

[14] N. Norris. Universal covers of graphs: Isomorphism to depth $n - 1$ implies isomorphism to all depth. *Discrete Applied Mathematics*, 56:61–74, 1995.

[15] K. P. S. Katz. Self-stabilizing extensions for message-passing systems. *Distributed Computing*, 7:17–26, 1993.

[16] M. Schneider. Self-stabilization. *ACM Computing Surveys*, 25(1):45–67, 1993.

[17] M. Yamashita and T. Kameda. Computing on anonymous networks. In *Proc. of the 4th PODC*, pages 13–22, 1985.

[18] M. Yamashita and T. Kameda. Electing a leader when processor identity numbers are not distinct. In *Proc. of the 3rd International Workshop on Distributed Algorithms*, number 392 in LNCS. Springer-Verlag, 1989.