On the Impact of Identifiers on Local Decision*

Pierre Fraigniaud† Magnús M. Halldórsson‡ Amos Korman†

Abstract

The issue of identifiers is crucial in distributed computing. Informally, in deterministic network computing, identities are used for tackling two of the fundamental difficulties that are inherent to distributed computing, namely: (1) *symmetry breaking*, and (2) *topological information gathering*. In the context of *local computation*, i.e., when nodes can gather information only from nodes at bounded distances, some insight regarding the role of identities has been established. For instance, it was shown that, for large classes of *construction* problems, the role of the identities can be rather small. However, for the identities to play no role, some other kinds of mechanisms for breaking symmetry must be employed, like, e.g., edge-labeling or sense of direction. When it comes to local distributed *decision* problems, the specification of the decision task does not seem to involve symmetry breaking. Therefore, it is expected that, assuming nodes can gather sufficient information about their neighborhood, one could get rid of the identities, without employing extra mechanisms for breaking symmetry. We tackle this question in the framework of the $\mathcal{LOCAL}$ model.

Let LD be the class of all problems that can be *decided* in a constant number of rounds in the $\mathcal{LOCAL}$ model. Similarly, let LD^* be the class of all problems that can be decided at constant cost in the anonymous variant of the $\mathcal{LOCAL}$ model, in which nodes have no identities, but each node can get access to the (anonymous) ball of radius t around it, for any t, at a cost of t. It is clear that $\mathrm{LD}^* \subseteq \mathrm{LD}$. We conjecture that $\mathrm{LD}^* = \mathrm{LD}$. In this paper, we give several evidences supporting this conjecture. In particular, we show that it holds for *hereditary* problems, or assuming that nodes know an arbitrary upper bound on the total number of nodes. Moreover, we prove that the conjecture holds in the context of *non-deterministic* local decision, where nodes are given certificates (independent of the identities, if they exist), and the decision consists in verifying these certificates. In short, we prove that $\mathrm{NLD}^* = \mathrm{NLD}$.

Keywords: Distributed complexity; locality; identities; decision problems; symmetry breaking; non-determinism

*This work is supported by the *Jules Verne* Franco-Icelandic bilateral scientific framework.

†CNRS and University Paris Diderot, France. E-mail: {pierre.fraigniaud,amos.korman}@liafa.jussieu.fr. Additional support from the ANR projects DISPLEXITY and PROSE, and from the INRIA project GANG.

‡ICE-TCS, School of Computer Science, Reykjavik University, Iceland. Supported by Iceland Research Foundation grant-of-excellence 90032021. E-mail: mmh@ru.is.

1 Introduction

1.1 Background and Motivation

The issue of identifiers is crucial in distributed computing [2, 32]. Indeed, the correct operation of deterministic protocols often relies on the assumption that each processor u is attached with a unique *identity*, $\mathrm{Id}(u)$ [9]. Informally, in network computing, such an identity assignment is crucial for tackling two of the fundamental difficulties that are inherent to distributed computing, namely: (1) *symmetry breaking*, and (2) *topological information gathering*.

The use of identities for tacking the above two difficulties is illustrated well in the context of *local* algorithms [29, 31]. Indeed, in the $\mathcal{LOCAL}$ model [34], an algorithm that runs in t communication rounds, assuming an identity assignment, can be viewed as composed of two parts: first, collecting at each node u, the ball $B(u,t)$ of radius t around it (together with the inputs of nodes), and second, deciding the output at u based solely on the information in $B(u,t)$. To achieve these two tasks, one should first obtain the ball $B(u,t)$, which may not be possible if the underlying graph is anonymous (i.e., without identities). Moreover, even if obtaining the ball is still be possible, e.g., if the structure of the graph allows it, the absence of unique identities given to the nodes may prevent the algorithm from breaking symmetry. For example, with the absence of unique identities, it is impossible to design a distributed deterministic coloring algorithm, even for the symmetric connected graph composed of two nodes only. In fact, to the best of our knowledge, all algorithms in the $\mathcal{LOCAL}$ model are designed assuming the presence of pairwise distinct identities or some other type of node-labeling or edge-labeling, including, e.g., sense of direction [5, 21, 28, 30, 32, 33].

The seminal paper of Naor and Stockmeyer [32] provides an important insight regarding the role of identities in local computation. Informally, they show that, even though identities are necessary, in many cases the actual values of identities is not crucial, and only their relative order matters. Specifically, [32] shows that for a particular class of problems, called LCL (for *Locally Checkable Languages*), if there exists a local algorithm that, for any identity assignment, constructs an instance of a problem in LCL in constant number of rounds, then there exists an *order invariant*[1] algorithm for that problem that runs in the same number of rounds. LCL restricts its concern to graphs with constant maximum degree, and to problems with a constant number of inputs. The assumption on the size of the inputs of problems in LCL was shown necessary in [20], by exhibiting a natural problem that is locally checkable, has unbounded input size, can be solved in 1 round with identities, but cannot be solved in constant time by any order invariant algorithm. The role of identities can also be measured by comparing their impact to the impact of "orientation mechanisms". For instance, Göös et al. [19] have shown that for a large class of optimization problems, called PO-checkable problems, local algorithms do not benefit from any kind of identifiers: if a PO-checkable optimization problem can be approximated with a local algorithm, the same approximation factor can be achieved in anonymous networks if the network is provided with a port-numbering and an orientation.

In the discussion above, we considered general distributed *construction* tasks, including, e.g., graph coloring [5, 28, 29, 32, 33], maximal independent set [29, 33], maximal matching [21, 30],

[1] Essentially, an order invariant algorithm uses the actual values of the identities only to impose an ordering between the nodes, that is, it behaves the same for any two identity assignments that preserve the total order between the nodes. For more details refer to [32].

etc. When it comes to distributed *decision* tasks [13, 14], symmetry breaking issues do not however seem to play a role. Informally, a decision task requires the nodes to "collectively decide" whether the given instance (i.e., a graph with inputs to the nodes) satisfies some specific properties. For instance, deciding coloring requires, given a colored graph, to check whether this graph is properly colored. The meaning of "collectively decide" is as follows. On a legal instance, all nodes should output "yes", and on an illegal one, at least one node should output "no". Note that it is not really important whether this node is unique or not, hence this specification does not inherently require any symmetry breaking. Therefore, assuming that each node u can obtain the ball $B(u,t)$, it makes sense that the assumption of having an identity assignment may not be crucial for achieving correct decision.

1.2 Model and Objectives

We tackle the question of whether identities play a role in decision problems in the framework of the aforementioned $\mathcal{LOCAL}$ model [34], which is a standard distributed computing model capturing the essence of locality. Recall that, in this model, processors are nodes of a connected network $G = (V(G), E(G))$, have pairwise distinct identities, and have inputs. More formally, a *configuration* is a triplet $(G, \mathbf{x}, \mathrm{Id})$ where G is a connected graph, every node $v \in V(G)$ is assigned as its *local input* a binary string $\mathbf{x}(v) \in \{0,1\}^*$, and $\mathrm{Id}(v)$ denotes the identity of node v. (In some problems, the local input of every node is empty, i.e., $\mathbf{x}(v) = \epsilon$ for every $v \in V(G)$, where ϵ denotes the empty binary string). Processors are woken up simultaneously, and computation proceeds over the input configuration $(G, \mathbf{x}, \mathrm{Id})$ in fault-free synchronous *rounds* during which every processor exchanges messages of unlimited size with its neighbors in the underlying network G, and performs arbitrary individual computations on its data. In many cases, the running time of an algorithm is measured with respect to the size n of G: the running time of an algorithm is defined as the maximum number of rounds it takes to terminate at all nodes, over all possible n-node networks. Similarly to [20, 32], we consider algorithms whose running time is independent of the size of the network, that is they run in constant time. As mentioned before, without loss of generality, any algorithm running in time $t = O(1)$ in the $\mathcal{LOCAL}$ model consists of:

1. collecting (in t rounds) at every node u the structure of the ball $B(u,t)$ consisting of an isomorphic copy of the graph induced by all nodes $v \in B(u,t)$, excluding the edges between two nodes at distance exactly t from u, together with all the inputs $\mathbf{x}(v)$ and identities $\mathrm{Id}(v)$ of these nodes, and,
2. performing some individual computation at every node.

We define the *anonymous* $\mathcal{LOCAL}$ model similarly to the $\mathcal{LOCAL}$ model, except that nodes have no identities. More precisely, an input configuration in the anonymous $\mathcal{LOCAL}$ model is just a pair $(G, \mathbf{x})$. An algorithm running in time $t = O(1)$ in the anonymous $\mathcal{LOCAL}$ model consists of:

1. getting at every node u a snapshot of the structure of the ball $B(u,t)$ together with all the inputs of the nodes in this ball, and,
2. performing some individual computation at every node.

Note that *anonymous* $\mathcal{LOCAL}$ model does not explicitly involve communications between nodes. Instead, it implicitly assumes that the underlying network supports the snapshot operation. Clearly, this model is not stronger than the $\mathcal{LOCAL}$ model, and possibly even strictly weaker, for every node u can no longer base its individual computation on the identities of the nodes in the ball $B(u,t)$.

Recall from [13] that a *distributed language* is a decidable collection $\mathcal{L}$ of configurations. (Since an undecidable collection of configurations remains undecidable in the distributed setting too, we consider only decidable collections of configurations). A typical example of a language is

$$\texttt{Coloring} = \{(G,\mathbf{x}) \mid \forall v \in V(G), \forall w \in N(v), \mathbf{x}(v) \neq \mathbf{x}(w)\} \ ,$$

where $N(v)$ denotes the (open) neighborhood of v, that is, all nodes at distance exactly 1 from v. Still following the terminology from [13], we say that a distributed algorithm A *decides* a distributed language $\mathcal{L}$ if and only if for every configuration $(G,\mathbf{x})$, every node of G eventually terminates and outputs "yes" or "no", satisfying the following decision rules:

- if $(G,\mathbf{x}) \in \mathcal{L}$, then each node outputs "yes";
- if $(G,\mathbf{x}) \notin \mathcal{L}$, then at least one node outputs "no".

In the (non-anonymous) $\mathcal{LOCAL}$ model, these two rules must be satisfied for every identity assignment. That is, all processes must output "yes" on a legal instance, independently from their identities. And, on an illegal instance, at least one node must output "no", for every identity assignment. Note that this node may potentially differ according to the identity assignment. Some languages can be decided in constant time (e.g., `Coloring`), while some others can easily be shown to not be decidable in constant time (e.g., "is the network planar?"). In contrast to the above examples, there are some languages whose status is unclear. To elaborate on this, consider the particular case where it is required to decide whether the network belongs to some specified family $\mathcal{F}$ of graphs. If this question can be decided in a constant number of communication rounds, then this means, informally, that the family $\mathcal{F}$ can somehow be characterized by relatively simple conditions. For example, a family $\mathcal{F}$ of graphs that can be characterized as consisting of all graphs having no subgraph from $\mathcal{C}$, where $\mathcal{C}$ is some specified finite set of graphs, is obviously decidable in constant time. However, the question of whether a family of graphs can be characterized as above is often non-trivial. For example, characterizing cographs as precisely the graphs with no induced P_4, attributed to Seinsche [35], is not easy, and requires nontrivial usage of modular decomposition.

We are now ready to define one of our main subjects of interest, the classes LD and LD*. Specifically, LD (for *local decision*) is the class of all distributed languages that can be decided by a distributed algorithm that runs in a constant number of rounds in the $\mathcal{LOCAL}$ model [13]. Similarly, LD*, the anonymous version of LD, is the class of all distributed languages that can be decided by a distributed algorithm that runs in a constant number of rounds in the anonymous $\mathcal{LOCAL}$ model. By definition, LD$^* \subseteq$ LD. We conjecture that

$$\text{LD}^* = \text{LD}.$$

In this paper, we provide several evidences supporting this conjecture. In addition, we investigate the *non-deterministic* version of these classes, and prove that they coincide. More specifically, a distributed *verification* algorithm is a distributed algorithm A that gets as input, in addition to a configuration $(G,\mathbf{x})$, a global *certificate vector* $\mathbf{y}$, i.e., every node v of a graph G gets as input two binary strings, an input $\mathbf{x}(v) \in \{0,1\}^*$ and a certificate $\mathbf{y}(v) \in \{0,1\}^*$. A verification algorithm A verifies $\mathcal{L}$ if and only if for every input configuration $(G,\mathbf{x})$, the following hold:

- if $(G, \mathbf{x}) \in \mathcal{L}$, then there exists a certificate $\mathbf{y}$ such that every node outputs "yes";
- if $(G, \mathbf{x}) \notin \mathcal{L}$, then for every certificate $\mathbf{y}$, at least one node outputs "no".

Again, in the (non-anonymous) $\mathcal{LOCAL}$ model, these two rules must be satisfied for every identity assignment, but the certificates must be the same regardless of the identities. We now recall the class NLD, for *non-deterministic local decision*, as defined in [13]: it is the class of all distributed languages that can be verified in a constant number of rounds in the $\mathcal{LOCAL}$ model. Similarly, we define NLD*, the anonymous version of NLD, as the class of all distributed languages that can be verified in a constant number of rounds in the anonymous $\mathcal{LOCAL}$ model. By definition, NLD$^* \subseteq$ NLD.

1.3 Our Results

In this paper, we give several evidences supporting the conjecture LD* = LD. In particular, we show that it holds for languages defined on paths, with a finite set of input values. More generally, we show that the conjecture holds for *hereditary* languages, that is, languages closed under node deletion. Regarding arbitrary languages, and arbitrary graphs, we prove that the conjecture holds assuming that every node knows an upper bound on the total number of nodes in the input graph. (This upper bound can be arbitrary, and may not be the same for all nodes).

Moreover, we prove that equality between non-anonymous decision and anonymous decision holds in the context of *non-deterministic* local decision, where nodes are given certificates (independent of the identities, if they exist), and the decision consists in verifying these certificates. More precisely, we prove that NLD* = NLD. This latter result is obtained by characterizing both NLD and NLD*.

1.4 Related Work

The question of how to locally decide (or verify) languages has received quite a lot of attention recently. Inspired by classical computation complexity theory [13] suggested that the study of decision problems may lead to new structural insights also in the more complex distributed computing setting. Indeed, following that paper, which focused on the $\mathcal{LOCAL}$ model, efforts were made to form a fundamental computational complexity theory for distributed decision problems in various other aspects of distributed computing [13, 15, 16, 17].

The classes LD, NLD and BPLD defined in [13] are the distributed analogues of the classes P, NP and BPP, respectively. The paper provides structural results, developing a notion of local reduction and establishing completeness results. One of the main results is the existence of a sharp threshold for randomization, above which randomization does not help (at least for hereditary languages). More precisely, the BPLD classes were classified into two: below and above the randomization threshold. In [14], the authors show that the hereditary assumption can be lifted if we restrict our attention to languages on path topologies. These two results from [13, 14] are used in the current paper in a rather surprising manner. The authors in [14] then "zoom" into the spectrum of classes below the randomization threshold, and defines a hierarchy of an infinite set of BPLD classes, each of which is separated from the class above it in the hierarchy.

The precise knowledge of the number of nodes n was shown in [13] to be of large impact on non-deterministic decision. Indeed, with such a knowledge every language can be decides non-

deterministically in the model of NLD. We note, however, that the knowledge of an arbitrary upper bound on n (as assumed here in one of our results) seems to be a much weaker assumption, and, in particular, will not suffice for non-deterministically deciding all languages. In the context of construction problems, it was shown in [27] that in many case, the knowledge of n (or an upper bound on n) is not essential.

The original theoretical basis for non-determinism in local computation was laid by the theory of *proof-labeling schemes* (PLS) [18, 23, 24, 25] originally defined in [25]. As mentioned, this notion resembles the notion of NLD, but differs in the role identities play. Specifically, in PLS the designer of the algorithm may base the certificates' (called labels in the terminology of PLS) construction on the given identity assignment. In contrast, in the model of NLD, the certificates must be the same regardless of the identities of nodes. Indeed, this difference is significant: while every language can be verified by a proof labeling scheme, not every language belongs to NLD [13]. These notions also bear some similarities to the notions of *local computation with advice* [7, 10, 11, 12], *local detection* [1], *local checking* [4], or *silent stabilization* [8]. In addition, as shown later on, the notion of NLD is related also to the theory of *lifts* or *covers* [2, 3].

Finally, the classification of decision problems in distributed computing has been studied in several other models. For example, [6] and [22] study specific decision problems in the $\mathcal{CONGEST}$ model. In [24], the authors study MST verification in the PLS sense but under the $\mathcal{CONGEST}$ model of communication. In addition, decision problems have been studied in the asynchrony discipline too, specifically in the framework of *wait-free computation* [16, 17] and *mobile agents computing* [15]. In the wait-free model, the main issues are not spatial constraints but timing constraints (asynchronism and faults). The main focus of [17] is deterministic protocols aiming at studying the power of the "decoder", i.e., the interpretation of the results. While this paper essentially considers the AND-checker, (as a global "yes" corresponds to all processes saying "yes"), [17] deals with other interpretations, including more values (not only "yes" and "no"), with the objective of designing checkers that use the smallest number of values.

2 Deterministic Decision

We conjecture that $\mathrm{LD} = \mathrm{LD}^*$. A support to this conjecture is that it holds for a large class of languages, namely for all *hereditary* languages, that is languages closed under node deletion. For instance, `Coloring` and `MIS` are hereditary, as well as all languages corresponding to hereditary graph families, such as planar graphs, interval graphs, forests, chordal graphs, cographs, perfect graphs, etc.

Theorem 1 $\mathrm{LD}^* = \mathrm{LD}$ *for hereditary languages.*

To prove the theorem, it is sufficient to show that $\mathrm{LD} \subseteq \mathrm{LD}^*$ for hereditary languages.This immediately follows from the statement and proof of Theorem 3.3 in [13]. Indeed, let A be a non-anonymous local algorithm deciding $\mathcal{L}$. This deterministic algorithm is in particular a randomized algorithm, with success probabilities $p = 1$ for legal instances, and $q = 1$ for illegal instance. That is, algorithm A is a $(1, 1)$-decider for $\mathcal{L}$, according to the definition in [13]. Since $\mathcal{L}$ is hereditary, and since $p^2 + q > 1$, the existence of A implies the existence of a specific deterministic anonymous local algorithm D for $\mathcal{L}$. Indeed, the algorithm D described in the proof of Theorem 3.3 in [13] is in fact anonymous: it simply collects the ball $B(u, t)$ of radius t around each node u for some constant t,

and u decides "yes" or "no" according to whether or not $B(u,t) \in \mathcal{L}$, regardless of the identities.

A similar proof, based on Theorem 4.1 in [14], enables to establish the following:

Theorem 2 $\mathrm{LD}^* = \mathrm{LD}$ *for languages defined on the set of paths, with a finite set of input values.*

Another evidence supporting the conjecture $\mathrm{LD} = \mathrm{LD}^*$ is that it holds assuming that nodes have access to a seemingly weak oracle. Specifically, this oracle, denoted $\mathbf{N}$, simply provides each node with an arbitrarily large upper bound on the total number of nodes in the actual instance. (It is not assumed that all upper bounds provided to nodes are the same). We denote by $\mathrm{LD}^{*\mathbf{N}}$ the class of languages that can be decided by an anonymous local algorithm having access to oracle $\mathbf{N}$, and we prove the following:

Theorem 3 $\mathrm{LD}^* \subseteq \mathrm{LD} \subseteq \mathrm{LD}^{*\mathbf{N}}$.

Proof. We just need to prove that $\mathrm{LD} \subseteq \mathrm{LD}^{*\mathbf{N}}$. Let $\mathcal{L} \in \mathrm{LD}$, and let A be a local (non-anonymous) algorithm deciding $\mathcal{L}$. Assume that the running time of A is t. We transform A into an anonymous algorithm A' deciding $\mathcal{L}$ in time t, assuming each node u in a given input G has an access to the oracle $\mathbf{N}$, i.e., it knows an arbitrary upper bound n_u on the number of nodes in G. (Note that it is not necessarily the case that all n_us are equal for all nodes u). Algorithm A' works as follows. Each node u collects the ball $B(u,t)$ of radius t around it. Then, for every possible assignment of identities to the nodes of $B(u,t)$ taken from the range $[1, n_u]$, node u simulates the behavior of the non-anonymous algorithm A on the ball $B(u,t)$ with the corresponding identities. If, in one of these simulations, algorithm A decides "no", then A' decides "no". Otherwise, A' decides "yes".

We now prove the correctness of A'. If the input $(G, \mathbf{x}) \in \mathcal{L}$ then A accepts it for every identity assignment to the nodes of G. Therefore, since, for every node u, every possible identity assignment to the nodes of the ball $B(u,t)$ can be extended to an identity assignment to all the nodes of G, all the simulations of A by u return "yes", and hence A' accepts $\mathcal{L}$ as well. On the other hand, if $(G, \mathbf{x}) \notin \mathcal{L}$ then A rejects it for every identity assignment to the nodes of G. That is, for every identity assignment to the nodes of G, at least one node u decides "no". (Note that, this node u may be different for two different identity assignments). Let us fix one identity assignment Id to the nodes of G, in the range $[1, n]$, and let u be one node that decides "no" for Id. Let $B_{\mathrm{Id}}(u,t)$ be the ball $B(u,t)$ with the identities of the nodes given by Id. In A', since u tries all possible identity assignments of the ball $B(u,t)$ in the range $[1, n_u]$ with $n \leq n_u$, in one of its simulations of A, node u will simulate A on $B_{\mathrm{Id}}(u,t)$. In this simulation, node u decides "no", hence algorithm A' rejects $\mathcal{L}$ as well. □

Note that the inclusion $\mathrm{LD} \subseteq \mathrm{LD}^{*\mathbf{N}}$ holds when one imposes no restrictions on the individual sequential running time. However, the transformation of a (non-anonymous) local algorithm into an anonymous local algorithm as described in the proof of Theorem 3 is very expensive in terms of individual computation. Indeed, the number of simulations of the original local algorithm A by each node u can be as large as $\binom{n_u}{n_B}$ where n_u is the upper bound on n given by the oracle $\mathbf{N}$, and n_B is the number of nodes in the ball $B(u,t)$. This bound can be exponential in n even if the oracle provides a good approximation of n (even if it gives precisely n). It would be nice to establish $\mathrm{LD} \subseteq \mathrm{LD}^{*\mathbf{N}}$ by using a transformation not involving a huge increase in the individual sequential computation time.

3 Non-deterministic Decision

In the previous section, we have seen several evidences supporting the conjecture that $\mathrm{LD}^* = \mathrm{LD}$, but whether it holds or not remains to be proved. In this section, we turn our attention to the non-deterministic variants of these two classes, and show that they coincide. More formally, we have:

Theorem 4 $\mathrm{NLD}^* = \mathrm{NLD}$.

Proof. To prove $\mathrm{NLD}^* = \mathrm{NLD}$, it is sufficient to prove $\mathrm{NLD} \subseteq \mathrm{NLD}^*$. To establish this inclusion, we provide a sufficient condition for NLD^*-membership, and prove that it is a necessary condition for NLD-membership. Let $I = (G, \mathbf{x})$ and $I' = (G', \mathbf{x}')$ be two input instances. An *homomorphism* from I to I' is a function $f : V(G) \to V(G')$ that preserves the edges of G as well as the inputs to the nodes. Specifically, f maps every edge $\{u, v\} \in E(G)$ to an edge $\{f(u), f(v)\} \in E(G')$, and f maps every node $u \in V(G)$ to a node $f(u) \in V(G')$ satisfying $\mathbf{x}'(f(u)) = \mathbf{x}(u)$. Let t be a positive integer. We say that I is *t-homomorphic* to I' if and only if there exists an homomorphism f from I to I' such that, for every node $v \in V(G)$, f restricted to $B_G(v, t)$ is an isomorphism from $B_G(v, t)$ to $B_{G'}(f(v), t)$. We call such an homomorphism f a t-homomorphism. Note that a t-homomorphism is onto (because if a node has no pre-image then neither do its neighbors have a pre-image, since homomorphisms preserve edges, and so forth). We say that $\mathcal{L}$ is closed under t-homomorphisms if, for every two instances I, I' such that I is t-homomorphic to I', we have: $I' \in \mathcal{L} \Rightarrow I \in \mathcal{L}$. The following claim gives a sufficient condition for NLD^*-membership.

Claim 1 *Let $\mathcal{L}$ be a language. If there exists $t \geq 1$ such that $\mathcal{L}$ is closed under t-homomorphisms then $\mathcal{L} \in \mathrm{NLD}^*$.*

Proof. Let $\mathcal{L}$ be a language, and assume that there exists $t \geq 1$ such that $\mathcal{L}$ is closed under t-homomorphisms. We describe an anonymous non-deterministic local algorithm A deciding $\mathcal{L}$, and performing in t rounds. The certificate of each node v is a triple $\mathbf{y}(v) = (i, G', \mathbf{x}')$ where G' is an n-node graph with nodes labeled by distinct integers in $[1, n] = \{1, \ldots, n\}$, $i \in [1, n]$, and $\mathbf{x}'$ is an n-dimensional vector. Informally, the certificates are interpreted by A as follows. The graph G' is supposed to be a "map" of G, that is, G' is interpreted as an isomorphic copy of G. The integer i is the label of the node in G' corresponding to node v in G. Finally, $\mathbf{x}'$ is interpreted as the input of the nodes in G'.

The algorithm A performs as follows. Every node v gets $B_G(v, t)$, the ball of radius t around it, hence in particular, it collects all the certificates of all the nodes at distance at most t from it. Then, by comparing its own certificate with the ones of its neighbors, it checks that the graph G', and the input $\mathbf{x}'$ in its certificate, are identical to the ones in the certificates of its neighbors. It also verifies consistency between the labels and the nodes in its ball of radius t. That is, it checks whether the labels and inputs in the certificate of the nodes in $B_G(v, t)$ are as described by its certificate. Whenever a node fails to pass any of these tests, it outputs "no". Otherwise it output "yes" or "no" according to whether $(G', \mathbf{x}') \in \mathcal{L}$ or not, respectively. (This is doable because we are considering languages that are decidable in the usual sense of sequential computation).

We show that A performs correctly. If $(G, \mathbf{x}) \in \mathcal{L}$, then by labeling the nodes in G by distinct integers from 1 to $|V(G)|$, and by providing the node v labeled i with $\mathbf{y}(v) = (i, G, \mathbf{x})$, the algorithm A output "yes" at all nodes, as desired. Consider now a instance $I = (G, \mathbf{x}) \notin \mathcal{L}$. Assume, for the

purpose of contradiction that there exists a certificate $\mathbf{y}$ leading all nodes to output "yes". Let $f : V(G) \to V(G')$ be defined by $f(v) = i$ where i is the label of v in its certificate. Since $\mathbf{y}$ passes all tests of A, it means that (1) $\mathbf{y}(v) = (i, G', \mathbf{x}')$ where the instance $I' = (G', \mathbf{x}')$ is the same for all nodes, (2) f restricted to $B_G(v,t)$ is an isomorphism from $B_G(v,t)$ to $B_{G'}(f(v),t)$, for every node v, and (3) $(G', \mathbf{x}') \in \mathcal{L}$. In view of (2), I is t-homomorphic to I'. Therefore, (3) implies that $I = (G, \mathbf{x}) \in \mathcal{L}$, because $\mathcal{L}$ is closed under t-homomorphisms. This is in contradiction with the actual hypothesis $(G, \mathbf{x}) \notin \mathcal{L}$. Thus, for each certificate $\mathbf{y}$, there must exist at least one node that outputs "no". As a consequence, A is a non-deterministic algorithm deciding $\mathcal{L}$, and thus $\mathcal{L} \in \text{NLD}^*$. $\diamond$

The following claim shows that the aforementioned sufficient condition for NLD*-membership is a necessary condition for NLD-membership.

Claim 2 *Let $\mathcal{L}$ be a language. If $\mathcal{L} \in$ NLD then there exists $t \geq 1$ such that $\mathcal{L}$ is closed under t-homomorphisms.*

Proof. Let $\mathcal{L}$ be a language in NLD, and let A be a non-deterministic (non-anonymous) local algorithm deciding $\mathcal{L}$. Assume, for the purpose of contradiction that, for any integer $t \geq 1$, $\mathcal{L}$ is *not* closed under t-homomorphisms. That is, for any t, there exist two input instances I, I' such that I is t-homomorphic to I', with $I \notin \mathcal{L}$ and $I' \in \mathcal{L}$. Assume that A runs in t rounds. Without loss of generality, we can assume that $t \geq 1$. Let $I = (G, \mathbf{x}) \notin \mathcal{L}$ and $I' = (G', \mathbf{x}') \in \mathcal{L}$ satisfying I is t-homomorphic to I'. Since $I' \in \mathcal{L}$, there exists a certificate $\mathbf{y}'$ such that when A is running on I' with certificate $\mathbf{y}'$, every node output "yes" for every identity assignment. Since I is t-homomorphic to I', there exists an homomorphism $f : I \to I'$ such that, for every node $v \in G$, f restricted to $B_G(v,t)$ is an isomorphism from $B_G(v,t)$ to $B_{G'}(f(v),t)$. Let $\mathbf{y}$ be the certificate for I defined by $\mathbf{y}(v) = \mathbf{y}'(f(v))$. Consider the execution of A running on I with certificate $\mathbf{y}$, and some arbitrary identity assignment Id.

Since A performs in t rounds, the decision at each node v is taken according to the inputs, certificates, and identities in the ball $B_G(v,t)$, as well as the structure of this ball. By the nature of the homomorphism f, and by the definition of certificate $\mathbf{y}$, the structure, inputs and certificates of the ball $B_G(v,t)$, are identical to the corresponding structure, inputs and certificates of the ball $B_{G'}(f(v),t)$. Balls may however differ in the identities of their nodes. So, let v_0 be the node in G deciding "no" for $(G, \mathbf{x})$ with certificate $\mathbf{y}$. There exists such a node since $I \notin \mathcal{L}$. Let $v'_0 = f(v_0)$, and assign the same identities to the nodes in $B_{G'}(v'_0, t)$ as their corresponding nodes in $B_G(v_0, t)$. Arbitrarily extend this identities to an identity assignment Id$'$ to the whole graph G'. By doing so, the two balls are not only isomorphic, but every node in $B_G(v_0, t)$ has the same input, certificate and identity as its image in $B_{G'}(v'_0, t)$. Therefore, the decision taken by A at $v_0 \in G$ under Id is the same as its decision at $v'_0 \in G'$ under Id$'$. This is in contradiction to the fact that v_0 decides "no" while v'_0 decides "yes". $\diamond$

Claims 1 and 2 together establish the theorem. $\square$

The proof of Claim 1 also provides an upper bound on the size of the certificates for *graph languages* in NLD, that is, for languages in NLD with no input. (This includes, e.g., recognition of interval graphs, and recognition of chordal graphs). Indeed, given $\mathcal{L} \in$ NLD, Algorithm A in the proof of Claim 1 verifies $\mathcal{L}$ using a certificate at each node which is essentially an isomorphic copy of the input instance $(G, \mathbf{x})$, with nodes labeled by consecutive integers in $[1, n]$. If $\mathcal{L}$ is a graph

language, then there is no input $\mathbf{x}$, and thus the size of the certificates depends only on the size of the graph. More precisely, we have:

Corollary 1 *Let $\mathcal{L} \in$ NLD be a graph language. There exists an algorithm verifying $\mathcal{L}$ using certificates of size $O(n^2)$ bits at each node of every n-node graph in $\mathcal{L}$.*

We now argue that the above bound is tight, that is, we prove the following.

Proposition 1 *There exists a graph language $\mathcal{L} \in$ NLD such that every algorithm verifying $\mathcal{L}$ requires certificates of size $\Omega(n^2)$ bits.*

Proof. Recall that [25] showed that there exists a graph language for which every proof labeling scheme (PLS) requires labels of size $\Omega(n^2)$ bits (the proof of this latter result appears in a detailed version [26]). Still in the context of PLS, [18] showed that this lower bound holds for two *natural* graph families: specifically, [18] showed that verifying symmetric graphs requires labels of size $\Omega(n^2)$ bits, and verifying non-3 colorable graphs requires almost the same size of labels, specifically, $\Omega(n^2/\log n)$ bits. Note that the certificate size required for verifying a language in NLD is at least as large as the minimum label size required for verifying the language via a proof labeling scheme. Unfortunately, however, one cannot obtain our claim directly from the aforementioned results since it turns out that neither of the two graph languages (namely, symmetric graphs and non-3 colorable graphs) belongs to NLD.

We therefore employ an indirect approach. Specifically, consider a graph G. We say that H is a *seed* of G if there exists a 1-homomorphism from G to H. Suppose $\mathcal{F}$ is a family graph. Let Seed-$\mathcal{F}$ denote the family of graphs G, for which there exists a seed of G that belongs to $\mathcal{F}$. Then, Seed-$\mathcal{F}$ is clearly closed under 1-homomorphisms, and is therefore in NLD. Now, in the proof of corollary 2.2 in [26], the authors construct, for every integer n, a family $\mathcal{F}_n$ of n-node graphs that requires proof labels of size $\Omega(n^2)$. Note that for every prime integer n', a graph G of size n' belongs to $\mathcal{F}_{n'}$ if and only if it belongs to Seed-$\mathcal{F}_{n'}$. Therefore, there exists a graph language, namely, Seed-$\mathcal{F}_n$, that requires certificates of size $\Omega(n^2)$ bits (at least for prime n's). □

References

[1] Y. Afek, S. Kutten, and M. Yung. The local detection paradigm and its applications to self stabilization. *Theoretical Computer Science*, 186(1-2):199–230, 1997.

[2] D. Angluin. Local and Global Properties in Networks of Processors. STOC 1980: 82-93.

[3] A. Amit, N. Linial, J. Matousek, and E. Rozenman. Random lifts of graphs. In *Proc. 12th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 883–894, 2001.

[4] B. Awerbuch, B. Patt-Shamir, and G. Varghese. Self-Stabilization By Local Checking and Correction. *Proc. IEEE Symp. on the Foundations of Computer Science (FOCS)*, 1991, 268-277.

[5] L. Barenboim and M. Elkin. Distributed $(\Delta + 1)$-coloring in linear (in delta) time. *Proc. 41st ACM Symp. on Theory of computing (STOC)*, 111–120, 2009.

[6] A. Das Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg and R. Wattenhofer. Distributed Verification and Hardness of Distributed Approximation. *Proc. 43rd ACM Symp. on Theory of Computing (STOC)*, 2011.

[7] D Dereniowski and A. Pelc. Drawing maps with advice. *Journal of Parallel and Distributed Computing*,72:132-143, 2012.

[8] S. Dolev, M. Gouda, and M. Schneider. Requirements for silent stabilization. *Acta Informatica*, 36(6), 447-462, 1999.

[9] R.G. Gallager, P.A. Humblet, P.M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Trans. on Programming Languages and Systems*, 5 (1983) 66-77.

[10] P. Fraigniaud, C. Gavoille, D. Ilcinkas and A. Pelc. Distributed Computing with Advice: Information Sensitivity of Graph Coloring. *Proc. 34th Colloq. on Automata, Languages and Programming (ICALP)*, 231-242, 2007.

[11] P. Fraigniaud, D Ilcinkas, and A. Pelc. Communication algorithms with advice. *J. Comput. Syst. Sci.*, 76(3-4):222–232, 2008.

[12] P. Fraigniaud, A Korman, and E. Lebhar. Local MST computation with short advice. *Proc. 19th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, 154–160, 2007.

[13] P. Fraigniaud, A. Korman, and D. Peleg. Local Distributed Decision. *Proc. 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 708-717, 2011.

[14] P. Fraigniaud, A. Korman, M. Parter and D. Peleg. Randomized Distributed Decision. http://arxiv.org/abs/1207.0252

[15] P. Fraigniaud and A. Pelc. Decidability Classes for Mobile Agents Computing. Proc. LATIN 2012: Theoretical Informatics - 10th Latin American Symposium, 2012.

[16] P. Fraigniaud, S. Rajsbaum, and C. Travers. Locality and Checkability in Wait-free Computing. *Proc. 25th International Symposium on Distributed Computing (DISC)*, 2011.

[17] P. Fraigniaud, S. Rajsbaum, and C. Travers. Universal Distributed Checkers and Orientation-Detection Tasks. Submitted, 2012.

[18] M. Göös and J. Suomela. Locally checkable proofs. *Proc. 30th ACM Symp. on Principles of Distributed Computing (PODC)*, 2011.

[19] M. Göös, J. Hirvonen, and J. Suomela. Lower bounds for local approximation. In *Proc. 31st Symposium on Principles of Distributed Computing (PODC)*, 2012.

[20] H. Hasemann, J. Hirvonen, J. Rybicki and J. Suomela. Deterministic Local Algorithms, Unique Identifiers, and Fractional Graph Colouring. In Proc. *19th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, 2012.

[21] M. Hanckowiak, M. Karonski, and A. Panconesi. On the Distributed Complexity of Computing Maximal Matchings. *SIAM J. Discrete Math.* 15(1): 41-57 (2001).

[22] L. Kor, A. Korman and D. Peleg. Tight Bounds For Distributed MST Verification. *Proc. 28th Int. Symp. on Theoretical Aspects of Computer Science (STACS)*, 2011.

[23] A. Korman and S. Kutten. Distributed verification of minimum spanning trees. *Distributed Computing*, 20:253–266, 2007.

[24] A. Korman, S. Kutten, and T. Masuzawa. Fast and Compact Self-Stabilizing Verification, Computation, and Fault Detection of an MST. *Proc. 30th ACM Symp. on Principles of Distributed Computing (PODC)*, 2011.

[25] A. Korman, S. Kutten, and D Peleg. Proof labeling schemes. *Distributed Computing*, 22:215–233, 2010.

[26] A. Korman, S. Kutten, and D Peleg. Proof labeling schemes. Detailed version. See: http://ie.technion.ac.il/ kutten/ps-links/ProofLabelingSchemes.ps

[27] A. Korman, J.S. Sereni, and L. Viennot. Toward More Localized Local Algorithms: Removing Assumptions Concerning Global Knowledge. *Proc. 30th ACM Symp. on Principles of Distributed Computing (PODC)*, 49-58, 2011.

[28] F. Kuhn. Weak graph colorings: distributed algorithms and applications. *Proc. 21st ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, 138–144, 2009.

[29] N. Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.

[30] Z. Lotker, B. Patt-Shamir and A. Rosen. Distributed Approximate Matching. *SIAM J. Comput.* 39(2): 445-460, (2009).

[31] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15:1036–1053, 1986.

[32] M. Naor and L. Stockmeyer. What can be computed locally? *SIAM J. Comput.* 24(6): 1259-1277 (1995).

[33] A. Panconesi and A. Srinivasan. On the Complexity of Distributed Network Decomposition. *J. Algorithms* 20(2): 356-374, (1996).

[34] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach.* SIAM, 2000.

[35] D. Seinsche. On a property of the class of n-colorable graphs. *J. Combinatorial Theory*, Ser. B, 16, pages 191–193, 1974.