Local Decision and Verification with Bounded-Size Outputs

Heger Arfaoui[1], Pierre Fraigniaud[1,⋆], and Andrzej Pelc[2,⋆⋆]

[1] CNRS and University Paris Diderot, France
[2] Université du Québec en Outaouais, Canada

Abstract. We are dealing with the design of algorithms using few resources, enabling to decide whether or not any given n-node graph G belongs to some given graph class $\mathcal{C}$. Our model borrows from property testing the way the decision is taken, by an *unconstrained interpretation function* applied to the set of outputs produced by individual queries (instead of an interpretation function limited to the conjunction operator as in local distributed decision). It borrows from local distributed decision the fact that *all nodes are involved* in the decision (instead of $o(n)$ nodes as in property testing). The unique, but severe restriction we impose to the nodes is a limitation on the amount of information they are enabled to output: every node is bounded to output a *constant* number of bits. In this paper, we provide separation results between distributed decision and verification classes, and we analyze the size of the certificates enabling to verify distributed languages.

1 Introduction

Objective. The ability to computationally *decide* a language using few resources is at the core of computer science, where, e.g., P can be interpreted as the class of languages *decidable* using little computation time, L can be interpreted as the class of languages decidable using little memory space, and NP and NL, the non deterministic versions of P and L, can be interpreted as the classes of languages *verifiable* using little computation time and little memory space, respectively, with small certificates. Still in the spirit of minimizing resources, *property testing* [17] is aiming at identifying languages that can be decided without accessing the whole instance. In particular, for any fixed graph class $\mathcal{C}$, property testing on graphs [18] aims at designing algorithms able to decide whether or not any given n-node graph G belongs to $\mathcal{C}$, by querying only $o(n)$ (random) nodes of the graph. In essence, the result of querying a node u is a value $out(u)$ such as, e.g., the degree of u. The decision algorithm acts and

⋆ The first and second authors receive support from the ANR project DISPLEXITY, and from the INRIA project GANG.

⋆⋆ Partially supported by NSERC discovery grant and by the Research Chair in Distributed Computing at the Université du Québec en Outaouais. Additional support from the Foundation Sciences Mathématiques de Paris.

T. Higashino et al. (Eds.): SSS 2013, LNCS 8255, pp. 133–147, 2013.

decides depending on the set of values collected so far. There are no restrictions imposed to this algorithm, except that one aims at designing algorithms deciding in polynomial time. In the distributed setting, the theory of *local distributed decision* [12,26] also aims at understanding the ability to decide graph properties, with or without inputs at nodes, using few resources. In this framework, all nodes are involved, each node u computes and outputs a single bit value b_u by inspecting its local neighborhood, and the result of the distributed decision is obtained by applying the conjunctive operator $\bigwedge_u b_u$ on these 1-bit-per-node outputs. (That is, the input configuration is accepted if and only if all nodes locally accept).

The restriction to the conjunctive operator in the framework of local distributed decision is conceptually elegant, and is well motivated by practical applications. For instance, the output $b_u = 0$ at node u can be interpreted as node u "raising an alarm". This alarm can be used by a central entity collecting data, like in, e.g., sensor networks. It can also be used in a distributed manner. For instance, in the framework of *self-stabilzation*, the alarm at node u may correspond to the detection of an invalid state of the system, and yields the launch of a recovery procedure [4,20]. Also, since the conjunctive operator is idempotent, commutative, and associative, it is easy to conceive a gossip procedure enabling all nodes to become aware of the global decision, by flooding in diameter rounds, where each round involves exchanging a single bit on each link (see, e.g., [5] for decision and verification in the $\mathcal{CONGEST}$ model).

Nevertheless, restricting ourselves to the conjunctive operator in the framework of local distributed decision is not a law carved in stone. In fact, there are several arguments against restricting the distributed decision setting to this specific operator. For instance, this restriction does not fit with elementary algebraic operations on sets. Typically, one may be able to distributedly decide locally two distributed properties $\mathcal{P}$ and $\mathcal{P}'$, and yet be unable to distributedly decide $\mathcal{P} \vee \mathcal{P}'$. For instance, using the conjunctive operator, one cannot decide locally whether nodes are properly k-colored if the set of k colors is not specified (this holds even if this set of colors is specified as being either $\{\text{green}, \text{orange}, \text{red}\}$ or $\{\text{green}, \text{orange}, \text{blue}\}$). Last but not least, recent results in the framework of distributed local decision [11] reveal that restricting ourselves to the conjunction operator prevents us from "boosting" probabilistic decisions. That is, using the conjunction operator, and as opposed to, say, languages in BPP, there are classes of distributed languages that can be decided distributedly with a fixed probabilistic guarantee, but which cannot be decided distributedly with better guarantees. In fact, there are computational models (e.g., distributed quantum computing) in which the exclusive-disjunction operator is known to be far more practical and efficient than the conjunctive operator (see [2], and the references therein).

To sum up the above discussion, while using the conjunctive operator in the framework of local distributed decision is well grounded for some settings, there are no reasons to stick to this specific operator in general. The objective of this

paper is thus to revisit local distributed decision, without restricting ourselves to the conjunctive operator.

Framework. We consider the $\mathcal{LOCAL}$ model [27], which is a standard network computing model capturing the essence of locality. In this model, processors have individual inputs, and arbitrary pairwise distinct identities. The algorithms should work properly for every possible identity assignment. They are woken up simultaneously, and computation proceeds in fault-free synchronous rounds during which every processor exchanges messages of unlimited size with its neighbors, and performs arbitrary computations on its data. Our aim is to decide distributed languages, locally, i.e., in a constant number of rounds. A distributed language is a (TM-decidable) collection $\mathcal{L}$ of pairs $(G, \mathbf{x})$, where G is a (connected) graph, and $\mathbf{x} = \{\mathbf{x}_u, u \in V(G)\}$ denotes the set of inputs given to the nodes (node u receives the binary string $\mathbf{x}_u$ as input). Typical examples of distributed languages are

$$\texttt{IsColored} = \{(G, \mathbf{x}) \text{ s.t. } \forall u \in V(G), \forall v \in N(u), \mathbf{x}_u \neq \mathbf{x}_v\}$$

where $N(u)$ denotes the (open) neighborhood of u, that is, all nodes at distance exactly 1 from u, and

$$\texttt{Tree} = \{(G, \mathbf{x}) \text{ s.t. } G \text{ is a tree}\}.$$

Deciding a distributed language $\mathcal{L}$ relies on two ingredients. One ingredient is a distributed algorithm $\mathcal{A}$ enabling each node u to output some value $out(u)$ in a constant number of rounds t, that is, after inspecting all nodes in the ball of radius t. This includes the structure of that ball, and the input values given to the nodes in this ball. The second ingredient is an interpretation operator $\mathcal{I}$, which applies to the collection $\{out(u), u \in V(G)\}$ of all values output by the nodes. As in classical local distributed decision, the interpretation is taken over all outputs. However, as in property testing, we allow any form of interpretation. Obviously, all distributed languages would be decidable in a single round in this setting if one did not impose restrictions on the values output by the nodes. We restrict every node to output a *constant* number k of bits. Hence, our framework is the extension of the model in [11,12,19,26] where:

1. every node outputs a number of bits bounded by a language-specific constant (rather than only one bit), and
2. the interpretation of these outputs is allowed to be any binary valued function $\mathcal{I}$ whose arguments are the unordered multi-sets of outputs of all nodes (rather than only the conjunction operator applied to these outputs).

Note that, using the interpretation function $\mathcal{I}$, an instance is accepted or rejected on the basis of the number of outputs of each type, regardless by which node a given output is produced.

For a non-negative integer t, we define the class ULD(t) (for *Unrestricted Local Decision*) as the class of distributed languages that can be decided in t

communication rounds in the $\mathcal{LOCAL}$ model, where decision is taken according to the rules specified above. We then define one of two classes of main interest for the purpose of this paper: $\text{ULD} = \cup_{t \geq 0} \text{ULD}(t)$.

We are also concerned with distributed *verification*, which can be seen as the nondeterministic version of decision, in the same way as NP is the nondeterministic version of P. In distributed verification, every node $u \in V(G)$ is given a *certificate* $\mathbf{y}_u$, in addition to the input $\mathbf{x}_u$. Each certificate is an arbitrary binary string. A distributed language $\mathcal{L}$ is locally verifiable if there exists a pair $(\mathcal{A}, \mathcal{I})$, where $\mathcal{A}$ is a distributed algorithm performing in a constant number of rounds, and $\mathcal{I}$ is an interpretation of the outputs produced by $\mathcal{A}$ at all nodes, such that the following holds:

- if $(G, \mathbf{x}) \in \mathcal{L}$ then there exists a collection of certificates $\mathbf{y} = \{\mathbf{y}_u, u \in V(G)\}$ satisfying that $\mathcal{A}$ running in G, with the pair $(\mathbf{x}_u, \mathbf{y}_u)$ given to each node u, returns $out(u)$ at every node u such that $\mathcal{I}$ accepts the multi-set $\{out(u), u \in V(G)\}$;
- if $(G, \mathbf{x}) \notin \mathcal{L}$ then for any collection of certificates $\mathbf{y} = \{\mathbf{y}_u, u \in V(G)\}$, $\mathcal{A}$ running in G, with the pair $(\mathbf{x}_u, \mathbf{y}_u)$ given to each node u, returns $out(u)$ at every node u such that $\mathcal{I}$ rejects the multi-set $\{out(u), u \in V(G)\}$.

For a non-negative integer t, we define the class $\text{UNLD}(t)$ (for *Unrestricted Nondeterministic Local Decision*) as the class of distributed languages that can be verified in t communication rounds in the $\mathcal{LOCAL}$ model, where the verification is performed according to the rules specified above. We then define our second class of interest: $\text{UNLD} = \cup_{t \geq 0} \text{UNLD}(t)$.

Observe that, for both decision and verification, the global outcome should not depend on the identities assigned to the nodes. In particular, the certificate $\mathbf{y}$ for a legal instance $(G, \mathbf{x})$, i.e., for an instance $(G, \mathbf{x}) \in \mathcal{L}$, enabling the interpretation to accept $(G, \mathbf{x})$ should not depend on the identity assignment to the nodes. This is in accordance to distributed verification, as studied in [11,12], but should not be mixed up with *proof-labeling schemes* [19,25] in which the certificates can possibly depend on the identity assignment.

Our Results. We first establish a set of classification and separation results, by placing various languages in their appropriate decision and verification classes. These results are summarized in Figure 1, where the classes LD and NLD are the classes of locally decidable languages, and of non-deterministic locally decidable languages, respectively, defined in [12]. These classes can alternatively be defined as the restriction of ULD and UNLD to the setting in which each node can output a single bit, and the interpretation is the result of the conjunction operator on these outputs.

We then prove that, in our "universal" decision and verification model, as opposed to classical distributed decision, *all* distributed languages can be *verified*. More specifically, all distributed languages on n-node networks with k-bit input per node can be verified using certificates of $O(n^2 + kn)$ bits, by having each node inspecting its neighborhood at distance 1 only, with just 1-bit-per-node outputs.

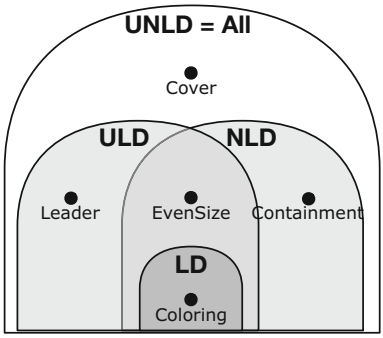

Fig. 1. *Four distributed decision and verification classes, with representatives*

Hence, in other words, UNLD = All. This result is essentially obtained by proving that the problem `Cover`, known to be a "hardest" local decision problem up to local reduction [12], is in UNLD. (Formally, `Cover` is BPNLD-complete[1] [12]).

The above upper bound on the certificate size enabling to put any language in UNLD is tight. Indeed, we prove that there are languages which require $\Omega(n^2 + kn)$ bits to be verified, even if the nodes are allowed to perform an arbitrarily large number of communication rounds, and even if each node can output an arbitrarily large number of bits.

From the fact that all distributed languages can be *verified*, it results that, as for *proof-labeling scheme*, one major issue in our setting is minimizing the size of the certificates. We prove that just enabling two output bits per node instead of just one, and just enabling a slightly more complex interpretation that the conjunction operator, has a tremendous impact on the size of the certificates. For instance, it is known that verifying trees using the logical conjunction operator, on 1-bit-per-node outputs, requires certificates of $\Omega(\log n)$ bits. (This holds even in the proof-labeling setting, i.e., when the certificates can possibly depend on the identity assignment). One of our perhaps most surprising results is a proof that, by simply using the conjunction and the disjunction operators together, on only 2-bit-per-node outputs, one can verify trees using certificates of only $O(1)$ bits.

Importantly, several of our positive results use interpretations of the outputs that have desirable properties. In particular, they are idempotent, commutative and associative. As a consequence, all nodes can become aware of the decision result by a simple gossip protocol performing in $O(\log n)$ time whenever such a mechanism can be implemented on top of the network. Alternatively, the global decision can be computed by all nodes in $O(D)$ time in the $\mathcal{CONGEST}(1)$ model [27], where D denotes the diameter of the network. Our universal verifier, used to establish UNLD = All, does not satisfy the idempotence property. Nevertheless, the global decision can still be computed by all nodes in $O(D)$ time in the $\mathcal{CONGEST}(\log n)$ model.

[1] BPNLD stands for Bounded-error Probabilistic Non-deterministic Local Decision.

Related Work. Locality issues have been thoroughly studied in the literature, via the analysis of various *construction* problems, including coloring and maximal independent set (MIS), minimum-weight spanning tree (MST), matching, dominating set, spanners, etc. We refer to the book [27] for an excellent introduction to local computing, providing pointers to the most relevant techniques for solving construction problems. The question of what can be computed in a constant number of communication rounds was actually posed in the seminal work of Naor and Stockmeyer [26], which considered a subclass of LD, called LCL, which is essentially LD restricted to languages involving graphs of constant maximum degree and processor inputs taken from a set of constant size. In fact, [26] studies the question of how to compute in $O(1)$ rounds the constructive versions of decision problems in LCL.

Recently, several results were established concerning *decision* problems in distributed computing. For example, [5] and [21] study specific decision problems in the $\mathcal{CONGEST}$ model. Specifically, tight bounds are established in [21] for the time and message complexities of the problem of deciding whether a given subgraph is an MST of the network, and time lower bounds for many other subgraph-decision problems (e.g., spanning tree, connectivity) are established in [5]. Decision problems have recently received attention in the asynchronous setting too, in the framework of wait-free computing [15]. In particular, [16] extends the results in [15] by allowing interpretations beyond the simple conjunction operator. Similarly, decision problem have also received attention in the context of computing with mobile agents [14].

The theory of proof labeling schemes [19,23,25] was designed to tackle the issue of locally verifying (with the aid of a "proof", i.e., a certificate, at each node) solutions to problems that cannot be decided locally. Investigations in this framework mostly focus on the minimum size of the certificate necessary so that verification can be performed in a single round [19,23,25], or in t rounds [24]. Hence, the model of proof labeling schemes has some resemblance to our definition of the class UNLD. The notion of proof labeling schemes also has interesting similarities with the notions of local detection [1], local checking [3], or silent stabilization [6], which were introduced in the context of self-stabilization.

The use of oracles that provide information to nodes was studied intensively in the context of distributed construction tasks. In particular, it was studied in the framework of *local computation with advice*. In this framework, MST construction was studied in [10], 3-coloring of cycles in [7], and broadcast and wake up in [8]. Finally, in [22] it is shown that, in the context of local computation, access to the oracle providing the number of nodes is not required for solving efficiently several central problems (e.g., $O(\Delta)$-coloring, MIS, etc.), while previous algorithms in the literature explicitly or implicitly assumed the use of this oracle.

2 Classification and Separation

Recall that the classes LD and NLD, defined in [12], are the respective restrictions of ULD and UNLD to the setting in which each node can output a single

bit, and the interpretation is the result of the conjunction operator on these outputs. Hence, by definition, LD $\subseteq$ ULD, and NLD $\subseteq$ UNLD. Also, by definition, ULD $\subseteq$ UNLD. The purpose of this section is to show that these inclusions are strict (the strict inclusion LD $\subset$ NLD is established in [12]), and to study the relationship between ULD and NLD. The following result is illustrated in Figure 1.

Theorem 1. ULD $\setminus$ NLD $\neq \emptyset$, NLD $\setminus$ ULD $\neq \emptyset$, *and* LD $\subset$ (ULD $\cap$ NLD) $\subset$ (ULD $\cup$ NLD) $\subset$ UNLD = All.

The proof of the above theorem is direct by combining the following four lemmas, including Lemma 2 which, in addition, provides an upper bound on the size of the certificates enabling to place every language in UNLD.

Lemma 1. ULD $\setminus$ NLD $\neq \emptyset$ *and* LD $\subset$ ULD $\cap$ NLD.

Proof. Let $\texttt{Leader} = \{(G, \mathbf{x})$ s.t. $\forall u \in V(G), \mathbf{x}_u \in \{0,1\},$ and $\sum_{u \in V(G)} \mathbf{x}_u = 1\}$. We have $\texttt{Leader} \notin$ NLD because this language is not closed under lift (see [9] for the characterization of NLD in term of lifts). To establish that $\texttt{Leader} \in$ ULD, we describe a local distributed algorithm enabling each node to output a constant number of bits, with the associated interpretation. The algorithm performs in zero rounds: every node u simply returns the single bit $b_u = \mathbf{x}_u$. The decision is then made according to the collection $\{b_i \in \{0,1\}, i \in [n]\}$ of outputs[2], by applying the logical operator $\mathcal{I} = \bigvee_{i=1}^{n} \left(b_i \wedge \bigwedge_{j \neq i} \overline{b_j} \right)$ which is true if and only if there is a unique b_i equal to 1. Hence, the input configuration is accepted if and only if there is a unique node u with $x_u = 1$, as desired. This proves that ULD $\setminus$ NLD $\neq \emptyset$.

Let $\texttt{EvenSize} = \{(G, \mathbf{x})$ s.t. G has an even number of nodes$\}$. This language is in NLD because it is closed under lift (see [9]). To establish that $\texttt{EvenSize} \in$ ULD, consider the algorithm performing in zero rounds consisting, for each node u, in outputting the single bit $b_u = 1$. The decision is then made by applying the operator $\mathcal{I} = 1 - \bigoplus_{i=1}^{n} b_i$ to the collection $\{b_i \in \{0,1\}, i \in [n]\}$ of output bits, where $\oplus$ denotes the exclusive-disjunctive operator. The value of $\mathcal{I}$ is equal to 1 if and only if the graph has an even number of nodes. Now, we also have $\texttt{EvenSize} \notin$ LD. This is because if some node u outputs 0 in an odd cycle C with some identity assignment (there must be such a node for C being rejected by the conjunction operator), then it also outputs 0 in some even cycle, causing this latter legal instance to be wrongly rejected. (Take the same cycle C with the same identity assignment, and insert one node between the two nodes at distance $\lfloor n/2 \rfloor$ from u, with some arbitrary identity distinct from the existing ones: node u still outputs 0 in this cycle). This proves LD $\subset$ ULD $\cap$ NLD, which completes the proof. $\square$

Let us consider the language

$$\texttt{Cover} \;=\; \{(G, (\mathbf{e}, \mathbf{S})) \mid \exists v \in V(G),\ \exists S \in \mathbf{S}_v \text{ s.t. } S = \{\mathbf{e}_u : u \in V(G)\}\}$$

[2] The indexes $i = 1, \ldots, n$ are only for the purpose of notation. The decision is made based on an unordered multiset of outputs.

introduced in [12]. This language is formed by all configurations $(G, \mathbf{x})$ with $\mathbf{x}_u = (\mathbf{e}_u, \mathbf{S}_u)$, where $\mathbf{e}_u$ is an element of some universe U, and $\mathbf{S}_u = \{S_1, \ldots, S_{k_u}\}$ is a collection of sets with elements in U, such that there exists a node v whose collection $\mathbf{S}_v$ contains a set S that is equal to the set formed of all the elements $\mathbf{e}_u$ for all $u \in V(G)$. We have $\texttt{Cover} \in$ UNLD as a consequence of the combined observations that (1) by providing every node with an oracle deciding `Leader`, all distributed languages are in NLD, and (2) $\texttt{Leader} \in$ ULD. The first claim is implicit in [12], and the second has been established in the proof of Lemma 1. In other words, $\texttt{Cover} \in$ UNLD simply because UNLD $=$ NLD$^{\texttt{Leader}} =$ All. We provide a complete proof of UNLD $=$ All below, for the purpose of completeness and further references in the text, and refer to [12] for more details on the impact of using oracles on the theory of local decision.

Lemma 2. *Every TM-decidable distributed language is in* UNLD*. Moreover, the verification of languages on n-node networks with k-bit input per node can be achieved using certificates of $O(n^2 + kn)$ bits, by having each node inspecting its neighborhood at distance* 1*, and with 1-bit-per-node outputs.*

Proof. Let $\mathcal{L}$ be a language. We describe a 1-round nondeterministic verification scheme $(\mathcal{A}, \mathcal{I})$ for $\mathcal{L}$. The certificate $\mathbf{y}$ of an instance $(G, \mathbf{x}) \in \mathcal{L}$ is a $n \times n$ adjacency matrix M of G, with vertices indexed arbitrarily by distinct integers in $[1, n]$, plus a n-dimensional vector I where I_i is the input of vertex $i \in [1, n]$. In addition, every node v receives the index $\lambda(v) \in [1, n]$ corresponding to v in M and I. More formally, the certificate at node v is $\mathbf{y}_v = ((G', \mathbf{x}'), i)$, where G' is an isomorphic copy of G with nodes labeled by λ from 1 to n, $\mathbf{x}'$ is an n-dimensional vector such that $\mathbf{x}'_{\lambda(u)} = \mathbf{x}_u$ for every node u, and $i = \lambda(v)$. In n-node networks with k-bit input per node, such a certificate is on $O(n^2 + kn)$ bits.

The local algorithm $\mathcal{A}$ executed on an instance $(G, \mathbf{x})$ with certificate $\mathbf{y}$ outputs one bit c_u at every node u. Let us first describe an algorithm with two bits a_u and b_u at every node u, and then we will show how to reduce these two bits into just one. Every node u with index $\lambda(u) = 1$ sets $a_u = 1$. The others set $a_u = 0$. For computing b_u, every node performs a single round of communication. First, every node u checks that it has received the input as specified by $\mathbf{x}'$, i.e., u checks whether $\mathbf{x}'_{\lambda(u)} = \mathbf{x}_v$, and set $b_u = 0$ if this does not hold. Second, each node u communicates with its neighbors to check that (1) they all got the same graph G' and the same input vector $\mathbf{x}'$, and (2) they are indexed the way they should be according to the map G'. If some inconsistency is detected by a node, then this node sets $b_u = 0$. At this point, each node u that has not yet set the variable b_u sets it to 1 if $(G', \mathbf{x}') \in \mathcal{L}$, and to 0 otherwise. All nodes u output the pair (a_u, b_u). The decision is then made according to the collection $\{(a_i, b_i) \in \{0,1\}^2, i \in [n]\}$ of outputs, by applying the operator

$$\mathcal{I} = \left(\bigvee_{i=1}^{n} \left(a_i \wedge \bigwedge_{j \neq i} \overline{a_j} \right) \right) \wedge \left(\bigwedge_{i=1}^{n} b_i \right)$$

which is 1 if and only if $(G, \mathbf{x}) \in \mathcal{L}$. To see why, observe that if every node u passes the tests regarding the certificates without setting b_u to 0, then all nodes agree on the graph G' and on the input vector $\mathbf{x}'$. Moreover, they know that their respective neighborhood in G fits with the corresponding one in G'. Therefore, if every node u passes the tests regarding the certificates without setting b_u to 0, then $(G', \mathbf{x}')$ is either identical to $(G, \mathbf{x})$ or to a lift of it[3]. It follows that, if all bits b_u are 1, then $(G', \mathbf{x}') = (G, \mathbf{x})$ if and only if there exists exactly one node $v \in G$, whose index $\lambda(v) = 1$. This is precisely the `Leader` problem, which is decided using the a_us.

Now, we reduce the two bits a_u and b_u into just one bit c_u. This reduction is based on the observation that if any node u detects some inconsistencies, then at least one of it neighbors also detects the same inconsistencies. As a consequence, if some node "raises an alarm" (i.e., set $b_u = 0$), then at least another node does the same. Thus, every node u sets $c_u = a_u \vee \overline{b_u}$ and output c_u. The decision is then made according to the collection $\{c_i \in \{0,1\}, i \in [n]\}$ of outputs, by applying the operator

$$\mathcal{I}' = \bigvee_{i=1}^{n} \left(c_i \wedge \bigwedge_{j \neq i} \overline{c_j} \right)$$

which is 1 if and only if $(G, \mathbf{x}) \in \mathcal{L}$. Indeed, $c_u = 1$ if and only if u detects some inconsistencies (i.e., $b_u = 0$) or $\lambda(u) = 1$ (i.e., $a_u = 1$). However, if u has detected some inconsistencies, then one of its neighbors u' has also detected the same inconsistencies, which guarantees $c_{u'} = 1$ for u' as well. Thus $\mathcal{I}' = 0$ if $(G, \mathbf{x}) \notin \mathcal{L}$. (The case where G is reduced to a single node is an exception: in this case, the unique node u sets $c_u = a_u \wedge b_u$). This completes the proof that UNLD = All. □

Lemma 3. ULD ∪ NLD ⊂ UNLD.

Proof. It is known that `Cover` ∉ NLD [12]. We prove that `Cover` ∉ ULD by contradiction, using arguments from communication complexity. Assume that there exists a local algorithm $\mathcal{A}$ and an interpretation $\mathcal{I}$ of the individual outputs produced by $\mathcal{A}$ enabling to decide `Cover`. In particular, $(\mathcal{A}, \mathcal{I})$ must decide the restricted version of `Cover`, defined on paths $P = (v_1, \ldots, v_n)$ with $U = \{0,1\}^k$, defined as follows. Let $\bar{0}$ denote the k-bit string formed by k consecutive 0s. We set

$$\mathbf{e}_1 = x, \mathbf{e}_n = y, \text{ and } \mathbf{e}_i = \bar{0} \text{ for } 1 < i < n,$$

and $\mathbf{S}_i = \{S_i\}$ for $i = 1, \ldots, n$ with

$$S_1 = \{\bar{0}, x\}, S_n = \{\bar{0}, y\} \text{ and } S_i = \emptyset \text{ for } 1 < i < n \ .$$

Such a configuration is in `Cover` if and only if $x = y$. We show that, using $(\mathcal{A}, \mathcal{I})$, one could solve the communication complexity problem "Equality" between Alice and Bob, by exchanging less than k bits. Assume $\mathcal{A}$ performs in t

[3] A graph H is a lift of a graph G if there exists a homomorphism from H to G preserving the neighborhood of each node.

rounds. Then, given x as input, Alice simulates the algorithm $\mathcal{A}$ applied at the $n-t-1$ nodes $v_1, \ldots, v_{n-t-1}$, while, given y as input, Bob simulates $\mathcal{A}$ applied to the $t+1$ nodes $v_{n-t}, \ldots, v_n$. Assume that $\mathcal{A}$ produces B bits of output at each node. The simulation of $\mathcal{A}$ allows Alice to compute $(n-t+1)B$ bits, i.e., the $n-t-1$ outputs of the nodes $v_1, \ldots, v_{n-t-1}$. Similarly, Bob computes $(t+1)B$ bits. It is thus sufficient for Bob to send these $(t+1)B = O(1)$ bits to Alice so that she can apply $\mathcal{I}$ on these bits together with her own $(n-t+1)B$ bits to determine whether $x = y$ or not. This holds for any $x, y \in \{0,1\}^k$. This is a contradiction, whenever $k > (t+1)B$ because "Equality" requires k bits to be exchanged between Alice and Bob for being solved. Hence $\texttt{Cover} \notin \text{ULD} \cup \text{NLD}$, which completes the proof. □

Lemma 4. $\text{NLD} \setminus \text{ULD} \neq \emptyset$.

Proof. Let us consider the following language, similar to `Cover`:

$$\texttt{Containment} = \{(G, (\mathbf{e}, \mathbf{S})) \mid \exists v \in V(G),\ \exists S \in \mathbf{S}_v \text{ s.t. } S \supseteq \{\mathbf{e}_u : u \in V(G)\}\}$$

The two languages `Cover` and `Containment` differ only in the fact that `Cover` asks for $S = \{\mathbf{e}_u : u \in V(G)\}$ while `Containment` simply asks for $S \supseteq \{\mathbf{e}_u : u \in V(G)\}$. It is known [12] that $\texttt{Containment} \in \text{NLD}$. Now, by the same arguments as for proving $\texttt{Cover} \notin \text{ULD}$, one can show $\texttt{Containment} \notin \text{ULD}$ as well. □

Remark. Lemma 2 states that all distributed languages are verifiable using certificates of $O(n^2 + kn)$ bits, which is the same upper bound as for proof-labeling schemes [25]. However, while proof-labeling schemes allows certificates to depend on the identity assignment, our verification algorithm uses certificates that are independent of the identity assignment.

3 Minimum Certificate Size for Universal Verification

By Lemma 2, we know that every TM-decidable distributed language with k-bit inputs is locally verifiable by providing nodes with certificates of $O(n^2 + kn)$ bits in n-node networks. Moreover, the verification is performed in one round, with 1-bit outputs. The following theorem proves that this bound is tight, in the sense that, for every k, there exist languages with k-bit inputs which require certificates of size $\Omega(n^2 + nk)$ bits to be verified in t rounds for b-bit outputs, for all t and b.

Theorem 2. *There exist languages with k-bit inputs that require certificates of size $\Omega(n^2 + nk)$ bits in n-node networks to be verified locally (i.e., to be placed in* UNLD*).*

Proof. We define the language `Symmetry` as follows. Given a graph G with k-bit input $\mathbf{x}_u$ per node u, an *input-preserving* automorphism ϕ of G is an automorphism satisfying $\mathbf{x}_u = \mathbf{x}_{\phi(u)}$ for every node u. Let

$$\texttt{Symmetry} = \{(G, \mathbf{x}) : \text{ there is a non-trivial input-preserving automorphism for } G\}.$$

The proof that Symmetry requires $\Omega(n^2 + nk)$ bits to be verified in n-node networks with k-bit inputs is based on a construction used in [19] to prove a lower bound on the size of the certificates when using the conjunction operator. We extend the arguments from [19] so that they apply to languages with inputs (and not only to graph properties), and apply to all possible operators for interpreting b-bit outputs (and not only the conjunction operator for 1-bit outputs).

Let $\mathcal{F}_{n,k}$ be the family of configurations $(G, \mathbf{x})$ where G is a non-symmetric graph with n-nodes, and $|\mathbf{x}_u| = k$ for every node u of G. More precisely, by labeling the nodes of G from 1 to n in arbitrary manner, we select a unique (labeled) instance of each non-symmetric graph with n nodes, to be placed in $\mathcal{F}_{n,k}$. It results from the same analysis as in [19] that

$$|\mathcal{F}_{n,k}| = 2^{kn} \ \frac{(1 - o(1))2^{\binom{n}{2}}}{n!}$$

and thus $\log |\mathcal{F}_{n,k}| = \Theta(n^2 + nk)$. Now, for every two configurations $(F_1, \mathbf{x}_1)$ and $(F_2, \mathbf{x}_2)$ in $\mathcal{F}_{n,k}$, let $(G, \mathbf{x}) = (F_1, \mathbf{x}_1) + (F_2, \mathbf{x}_2)$ be the configuration formed by a copy of F_1 together with its inputs $\mathbf{x}_1$, a copy of F_2 together with its inputs $\mathbf{x}_2$, and a path P of $4t + 1$ nodes (without inputs), connecting the node with label 1 in F_1 to the node with label 1 in F_2. The number of nodes in G is $2n + 4t + 1 = \Theta(n)$. Let

$$\mathcal{C} = \{(G, \mathbf{x}) = (F_1, \mathbf{x}_1) + (F_2, \mathbf{x}_2) \ : \ (F_1, \mathbf{x}_1) \in \mathcal{F}_{n,k} \text{ and } (F_2, \mathbf{x}_2) \in \mathcal{F}_{n,k}\}.$$

We show that even verifying Symmetry-membership for configurations in $\mathcal{C}$ requires $\Omega(n^2 + nk)$-bit certificates. Since all graphs in $\mathcal{F}_{n,k}$ are non-symmetric, we get that, for any $(G, \mathbf{x}) \in \mathcal{C}$, we have $(G, \mathbf{x}) \in$ Symmetry if and only if $(F_1, \mathbf{x}_1) = (F_2, \mathbf{x}_2)$. (Recall that the graphs in $\mathcal{F}_{n,k}$ are labeled, and thus equality here means the existence of a label-preserving input-preserving isomorphism between F_1 and F_2). Let $\mathcal{C}_{\text{sym}}$ be the subset of $\mathcal{C}$ consisting of symmetric graphs in $\mathcal{C}$, i.e., $\mathcal{C}_{\text{sym}} = \mathcal{C} \cap$ Symmetry. We have:

$$\mathcal{C}_{\text{sym}} = \{(G, \mathbf{x}) = (F, \mathbf{x}') + (F, \mathbf{x}') \ : \ (F, \mathbf{x}') \in \mathcal{F}_{n,k}\}.$$

Note that $|\mathcal{C}_{\text{sym}}| = |\mathcal{F}_{n,k}| \geq 2^{c(n^2+nk)}$ for some constant $c > 0$ and for big enough values of n. Assume now, for the sake of contradiction, that one can verify Symmetry in t rounds with certificates of size $s = o(n^2 + nk)$ bits per node, using algorithm $\mathcal{A}$ with interpretation $\mathcal{I}$. Then, for every configuration in $\mathcal{C}$, the path P includes $4t+1$ certificates, for a total of $(4t+1)s$ bits, that is still $o(n^2 + nk)$ bits since t is constant. Therefore, there are at least

$$R = 2^{c'(n^2+nk)}$$

graphs in $\mathcal{C}_{\text{sym}}$, that have the same collection of certificates on their respective paths P, for some c', $0 < c' < c$. On the other hand, for an $(n+t)$-node graph with b bits of output per node, the total number of possible multi-sets the verification

algorithm $\mathcal{A}$ can produce on this graph is upper bounded. If $\left(\binom{x}{y}\right)$ denotes the multinomial coefficient "x multichoose y", then this number is:

$$N = \left(\binom{2^b}{n+t}\right) = \binom{2^b+n+t-1}{n+t}.$$

Therefore

$$N = \frac{(n+t+1)(n+t+2)...(n+t+2^b-1)}{(2^b-1)!} = O(n^{2^b}).$$

So, let us assign identities to every graphs $(G,\mathbf{x}) = (F,\mathbf{x}') + (F,\mathbf{x}')$ in $\mathcal{C}_{\text{sym}}$ as follows. One copy of $(F,\mathbf{x}')$ is given identities from 1 to n, while the other copy of $(F,\mathbf{x}')$ is given identities from $n+1$ to $2n$. In both copies, the identity assignment is set with respect to the labeling of F, i.e., node labeled i receive identity i in one copy, and $n+i$ in the other copy. Nodes in the path P are given identities from $2n+1$ to $2n+4t+1$.

Since R is very large compared to N^2, there exist two configurations $(G_1,\mathbf{x}_1) = (F_1,\mathbf{x}'_1) + (F_1,\mathbf{x}'_1)$ and $(G_2,\mathbf{x}_2) = (F_2,\mathbf{x}'_2) + (F_2,\mathbf{x}'_2)$ in $\mathcal{C}_{\text{sym}}$ that receive the same collection of certificates on their respective path P, and for which $\mathcal{A}$ produces the same multi-set M_1 of outputs in the copies of $(F_1,\mathbf{x}'_1)$ and $(F_2,\mathbf{x}'_2)$ connected to the nodes with identities $2n+1,\ldots,2n+t$ on P, and the same multi-set M_2 of outputs in the copies of $(F_1,\mathbf{x}'_1)$ and $(F_2,\mathbf{x}'_2)$ connected to the nodes with identities $2n+3t+1,\ldots,4t+1$ on P. Let us denote by M_0 the multi-set of produced produced by $\mathcal{A}$ on the $2t+1$ nodes at the middle of P in both configuration $(G_1,\mathbf{x}_1)$ and $(G_2,\mathbf{x}_2)$.

Now, consider the following configuration $(G,\mathbf{x})$ formed by "cutting and gluing" $(G_1,\mathbf{x}_1)$ and $(G_2,\mathbf{x}_2)$. More precisely, $(G,\mathbf{x})$ is formed by connecting $(F_1,\mathbf{x}_1)$, $(P,\varnothing)$, and $(F_2,\mathbf{x}_2)$, with identities in $[1,n]$ for F_1, in $[n+1,2n]$ for F_2, and, as usual, in $[2n+1,2n+4t+1]$ for P. Let us provide these nodes with the certificates inherited from these respective copies of $(F_1,\mathbf{x}_1)$, and $(F_2,\mathbf{x}_2)$. Each node with identities $\{1,...,n\}\cup\{2n+1,\ldots,2n+t\}$ (resp., with identities in $\{n+1,\ldots,2n\}\cup\{2n+3t+1,\ldots,2n+4t+1\}$) has the same local view of radius t in $(G,\mathbf{x})$ as in $(F_1,\mathbf{x}_1)$ (resp., $(F_2,\mathbf{x}_2)$). Moreover, nodes in the middle part of the path, with identities in $[2n+t+1,2n+3t]$ have the same view in $(G,\mathbf{x})$ as in $(G_1,\mathbf{x}_1)$ and $(G_2,\mathbf{x}_2)$. Therefore, the verification algorithm $\mathcal{A}$ outputs the same multi-set $M_0\cup M_1\cup M_2$ for the illegal configuration $(G,\mathbf{x})$, as it does for the legal configurations $(G_1,\mathbf{x}_1)$ and $(G_2,\mathbf{x}_2)$, yielding the desired contradiction. □

Remark. By inspecting R and N in the proof of Theorem 2, we can notice that the theorem holds even if the number of output bits per node is up to $c\log(n^2+nk)$, for $c<1$, and, by the construction of the accepted illegal configuration, even for verification algorithms performing in time up to $o(n)$ rounds.

4 Verifying Trees with Constant-Size Certificates

In this section, we show that, for languages in NLD, restricting the interpretation to the use of the conjunctive operator may have a significant cost in terms of

certificate size. For instance, it is known [25] that verifying `Tree` using the conjunction operator requires $\Omega(\log n)$-bit certificates for n-node trees. This holds even if the certificates can depend on the identity assignment, and even if the verification can take an arbitrarily large (but constant) number of rounds. In contrast, we show that using conjointly the conjunction and disjunction operators, on 2-bit outputs, enables to verify `Tree` in one round, using certificates of only $O(1)$ bits. Moreover, as we can see in the proof of this result, the decision is made according to the application of a 2-bit logical operator $\mathcal{I}$ that is idempotent, commutative, and associative, and thus with all the desirable properties to be used in environments supporting gossip protocols, as well as in the $\mathcal{CONGEST}(1)$ model.

Theorem 3. `Tree` *can be verified in one round, with certificates of constant size, and two output bits per node.*

Proof. To establish the theorem, we first describe the collection of $O(1)$-bit certificates assigned to the nodes in the case of a valid instance of `Tree`, i.e., for a tree T. The certificate assigned to node v is a pair $\mathbf{y}_v = (r(v), d(v))$, where $r(v)$ is on one bit, and $d(v)$ is on two bits. Every certificate is thus encoded using three bits. To define the assignment of these bits at node v, let us pick an arbitrary node u_0 of T, and set u_0 as the root of T. Set $r(u_0) = 1$, and $r(v) = 0$ for every node $v \neq u_0$. For every $v \in V(T)$, let $d(v) = dist_T(v, u_0) \bmod 3$, where $dist_T(x, y)$ denotes the distance in T between nodes x and y, i.e., the minimum number of edges of a path from x to y in T.

We now describe the verification algorithm. It performs in just one round, during which every node v sends its certificate $\mathbf{y}_v$ to all its neighbors, and receives all the certificates of its neighbors. Given its own certificate and the certificates of its neighbors, every node v then computes a pair of bits (a_v, b_v) as follows. First, every node v checks whether it has at most one neighbor w with $d(w) = d(v) - 1 \pmod 3$. Node w is called the parent of v. More precisely, if $r(v) = 1$ then there must be no parent for v, and, if $r(v) = 0$ then there must be exactly one parent for v. Similarly, v checks whether all its neighbors w different from its parent satisfy $d(w) = d(v) + 1 \pmod 3$. All such nodes are called the children of v. If any of these tests is not passed, then v aborts, and outputs $(0, 0)$. If node v has not aborted, then it has identified its parent and its children (apart the root which has no parent), and it outputs $(1, r(v))$. This completes the description of the verification algorithm.

We now describe the interpretation of the collection of 2-bit outputs $\{(a_i, b_i), i = 1, \ldots, n\}$. It is the result of the following operator:

$$\mathcal{I} = \Big(\bigwedge_{i=1}^{n} a_i \Big) \wedge \Big(\bigvee_{i=1}^{n} b_i \Big).$$

By construction, if T is a tree, then $\mathcal{I} = 1$. Indeed, all tests are passed successfully, and thus the (unique) node v with $r(v) = 1$ returns $(1, 1)$ while all the other nodes return $(1, 0)$.

Establishing that $\mathcal{I} = 0$ whenever T is not a tree, independently from the certificates given to the nodes, is based on the fact that, if all tests are passed (i.e., if $\bigwedge_{i=1}^{n} a_i = 1$) then there cannot be a node v with $r(v) = 1$, and therefore $\bigvee_{i=1}^{n} b_i = 0$, yielding $\mathcal{I} = 0$. To see why this is indeed the case, assume that the current input (connected) graph G is not a tree. Assume moreover that the verification algorithm returns a set $\{(a_i, b_i), i = 1, \ldots, n\}$ such that $\bigwedge_{i=1}^{n} a_i = 1$. (Note that if this is not the case, then $\mathcal{I} = 0$, and we are done).

Since $\bigwedge_{i=1}^{n} a_i = 1$, every edge of G is given an orientation, from child to parent, and this orientation in locally consistent. That is, every node has exactly one outgoing edge, and a (potentially empty) set of incoming edges, apart from nodes marked $r(v) = 1$, if any, which may have no outgoing edges. Since G is not a tree, there is a cycle C in G. Since $a_i = 1$ for all i, it must be the case that all edges of the cycle are consistently oriented along C. That is, each node in C has exactly one outgoing edge in C and one incoming edge in C. In particular, all edges incident to C are entering C. As a consequence, there is a unique cycle in G. Indeed, if there were two node-disjoint cycles, then one could not guarantee consistency of the edge orientation along a path connecting these two cycles. The same holds if the two cycles would share one or more nodes. So, G is an "octopus". That is, it consists of a cycle C to which are attached a collection of trees, whose edges are all consistently oriented toward the cycle. Therefore, every node has an outgoing edge, and thus there cannot be a root node in G, i.e., a node v with $r(v) = 1$. Thus, $b_i = 0$ for all i, yielding $\mathcal{I} = 0$, which completes the proof of the theorem. □

Acknowledgements. the authors are thankful to Amos Korman for fruitful discussions regarding the subject of this paper.

References

1. Afek, Y., Kutten, S., Yung, M.: The local detection paradigm and its applications to self stabilization. Theoretical Computer Science 186(1-2), 199–230 (1997)
2. Arfaoui, H., Fraigniaud, P.: What Can Be Computed without Communications? In: Even, G., Halldórsson, M.M. (eds.) SIROCCO 2012. LNCS, vol. 7355, pp. 135–146. Springer, Heidelberg (2012)
3. Awerbuch, B., Patt-Shamir, B., Varghese, G.: Self-Stabilization By Local Checking and Correction. In: Proc. IEEE Symp. on the Foundations of Computer Science (FOCS), pp. 268–277 (1991)
4. Awerbuch, B., Patt-Shamir, B., Varghese, G., Dolev, S.: Self-Stabilization by Local Checking and Global Reset. In: Tel, G., Vitányi, P. (eds.) WDAG 1994. LNCS, vol. 857, pp. 326–339. Springer, Heidelberg (1994)
5. Das Sarma, A., Holzer, S., Kor, L., Korman, A., Nanongkai, D., Pandurangan, G., Peleg, D., Wattenhofer, R.: Distributed Verification and Hardness of Distributed Approximation. In: Proc. 43rd ACM Symp. on Theory of Computing, STOC (2011)
6. Dolev, S., Gouda, M., Schneider, M.: Requirements for silent stabilization. Acta Informatica 36(6), 447–462 (1999)
7. Fraigniaud, P., Gavoille, C., Ilcinkas, D., Pelc, A.: Distributed Computing with Advice: Information Sensitivity of Graph Coloring. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 231–242. Springer, Heidelberg (2007)

8. Fraigniaud, P., Ilcinkas, D., Pelc, A.: Communication algorithms with advice. J. Comput. Syst. Sci. 76(3-4), 222–232 (2008)
9. Fraigniaud, P., Halldórsson, M.M., Korman, A.: On the Impact of Identifiers on Local Decision. In: Baldoni, R., Flocchini, P., Binoy, R. (eds.) OPODIS 2012. LNCS, vol. 7702, pp. 224–238. Springer, Heidelberg (2012)
10. Fraigniaud, P., Korman, A., Lebhar, E.: Local MST computation with short advice. In: Proc. 19th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA), pp. 154–160 (2007)
11. Fraigniaud, P., Korman, A., Parter, M., Peleg, D.: Randomized Distributed Decision. In: Aguilera, M.K. (ed.) DISC 2012. LNCS, vol. 7611, pp. 371–385. Springer, Heidelberg (2012)
12. Fraigniaud, P., Korman, A., Peleg, D.: Local Distributed Decision. In: Proc. 52nd Annual IEEE Symp. on Foundations of Computer Science (FOCS), pp. 708–717 (2011)
13. Fraigniaud, P., Göös, M., Korman, A., Suomela, J.: What can be decided locally without identifiers? In: 32nd ACM Symp. on Principles of Distributed Computing, PODC (2013)
14. Fraigniaud, P., Pelc, A.: Decidability Classes for Mobile Agents Computing. In: Fernández-Baca, D. (ed.) LATIN 2012. LNCS, vol. 7256, pp. 362–374. Springer, Heidelberg (2012)
15. Fraigniaud, P., Rajsbaum, S., Travers, C.: Locality and Checkability in Wait-Free Computing. In: Peleg, D. (ed.) DISC 2011. LNCS, vol. 6950, pp. 333–347. Springer, Heidelberg (2011)
16. Fraigniaud, P., Rajsbaum, S., Travers, C.: An Impossibility Result for Run-Time Monitoring (submitted, 2013)
17. Goldreich, O. (ed.): Property Testing. LNCS, vol. 6390. Springer, Heidelberg (2010)
18. Goldreich, O., Ron, D.: Property Testing in Bounded Degree Graphs. Algorithmica 32(2), 302–343 (2002)
19. Göös, M., Suomela, J.: Locally checkable proofs. In: Proc. 30th ACM Symp. on Principles of Distributed Computing, PODC (2011)
20. Katz, S., Perry, K.: Self-stabilizing extensions to for message-passing systems. Distributed Computing 7, 17–26 (1993)
21. Kor, L., Korman, A., Peleg, D.: Tight Bounds For Distributed MST Verification. In: Proc. 28th Int. Symp. on Theoretical Aspects of Computer Science, STACS (2011)
22. Korman, A., Sereni, J.S., Viennot, L.: Toward More Localized Local Algorithms: Removing Assumptions Concerning Global Knowledge. In: Proc. 30th ACM Symp. on Principles of Distributed Computing, PODC, pp. 49–58 (2011)
23. Korman, A., Kutten, S.: Distributed verification of minimum spanning trees. Distributed Computing 20, 253–266 (2007)
24. Korman, A., Kutten, S., Masuzawa, T.: Fast and Compact Self-Stabilizing Verification, Computation, and Fault Detection of an MST. In: Proc. 30th ACM Symp. on Principles of Distributed Computing, PODC (2011)
25. Korman, A., Kutten, S., Peleg, D.: Proof labeling schemes. Distributed Computing 22, 215–233 (2010)
26. Naor, M., Stockmeyer, L.: What can be computed locally? SIAM J. Comput. 24(6), 1259–1277 (1995)
27. Peleg, D.: Distributed Computing: A Locality-Sensitive Approach. SIAM (2000)