Randomized Distributed Decision

Pierre Fraigniaud[1], Amos Korman[1⋆], Merav Parter[2], and David Peleg[2⋆⋆]

[1] CNRS and University Paris Diderot, France .
{pierre.fraigniaud,amos.korman}@liafa.jussieu.fr
[2] The Weizmann Institute of Science, Rehovot, Israel.
{merav.parter,david.peleg}@weizmann.ac.il.

Abstract. The paper tackles the power of randomization in the context of locality by analyzing the ability to "boost" the success probability of deciding a distributed language. The main outcome of this analysis is that the distributed computing setting contrasts significantly with the sequential one as far as randomization is concerned. Indeed, we prove that in some cases, the ability to increase the success probability for deciding distributed languages is rather limited.

We focus on the notion of a (p, q)-*decider* for a language $\mathcal{L}$, which is a distributed randomized algorithm that *accepts* instances in $\mathcal{L}$ with probability at least p and *rejects* instances outside of $\mathcal{L}$ with probability at least q. It is known that every hereditary language that can be decided in t rounds by a (p, q)-decider, where $p^2 + q > 1$, can be decided *deterministically* in $O(t)$ rounds. One of our results gives evidence supporting the conjecture that the above statement holds for all distributed languages and not only for hereditary ones, by proving the conjecture for the restricted case of path topologies.

For the range below the aforementioned threshold, namely, $p^2 + q \leq 1$, we study the class $B_k(t)$ (for $k \in \mathbb{N}^* \cup \{\infty\}$) of all languages decidable in at most t rounds by a (p, q)-decider, where $p^{1+\frac{1}{k}} + q > 1$. Since every language is decidable (in zero rounds) by a (p, q)-decider satisfying $p + q = 1$, the hierarchy B_k provides a spectrum of complexity classes between determinism ($k = 1$, under the above conjecture) and complete randomization ($k = \infty$). We prove that all these classes are separated, in a strong sense: for every integer $k \geq 1$, there exists a language $\mathcal{L}$ satisfying $\mathcal{L} \in B_{k+1}(0)$ but $\mathcal{L} \notin B_k(t)$ for any $t = o(n)$. In addition, we show that $B_\infty(t)$ does not contain all languages, for any $t = o(n)$. In other words, we obtain the hierarchy $B_1(t) \subset B_2(t) \subset \cdots \subset B_\infty(t) \subset \text{All}$.

Finally, we show that if the inputs can be restricted in certain ways, then the ability to boost the success probability becomes almost null, and in particular, derandomization is not possible even beyond the threshold $p^2 + q = 1$.

Keywords: Local distributed algorithms, local decision, randomized algorithms.

⋆ Supported by the ANR projects DISPLEXITY and PROSE, and by the INRIA project GANG.

⋆⋆ Supported in part by the Israel Science Foundation (grant 894/09), the US-Israel Binational Science Foundation (grant 2008348), the Israel Ministry of Science and Technology (infrastructures grant), and the Citi Foundation.

1 Introduction

Background and Motivation. The impact of randomization on computation is one of the most central questions in computer science. In particular, in the context of distributed computing, the question of whether randomization helps in improving locality for construction problems has been studied extensively. While most of these studies were problem-specific, several attempts have been made for tackling this question from a more general and unified perspective. For example, Naor and Stockmeyer [26] focus on a class of problems called LCL (essentially a subclass of the class LD discussed below), and show that if there exists a randomized algorithm that constructs a solution for a problem in LCL in a constant number of rounds, then there is also a constant time deterministic algorithm constructing a solution for that problem.

Recently, this question has been studied in the context of *local decision*, where one aims at deciding locally whether a given global input instance belongs to some specified language [13]. The localities of deterministic algorithms and randomized Monte Carlo algorithms are compared in [13], in the $\mathcal{LOCAL}$ model (cf. [28]). One of the main results of [13] is that randomization does not help for locally deciding *hereditary* languages if the success probability is beyond a certain guarantee threshold. More specifically, a (p,q)-*decider* for a language $\mathcal{L}$ is a distributed randomized Monte Carlo algorithm that *accepts* instances in $\mathcal{L}$ with probability at least p and *rejects* instances outside of $\mathcal{L}$ with probability at least q. It was shown in [13] that every hereditary language that can be decided in t rounds by a (p,q)-decider, where $p^2+q>1$, can actually be decided *deterministically* in $O(t)$ rounds. On the other hand, [13] showed that the aforementioned threshold is sharp, at least when hereditary languages are concerned. In particular, for every p and q, where $p^2+q\leq 1$, there exists an hereditary language that cannot be decided deterministically in $o(n)$ rounds, but can be decided in zero rounds by a (p,q)-decider.

In one of our results we provide evidence supporting the conjecture that the above statement holds for all distributed languages and not only for hereditary ones. This is achieved by considering the restricted case of path topologies. In addition, we present a more refined analysis for the family of languages that can be decided randomly but not deterministically. That is, we focus on the family of languages that can be decided locally by a (p,q)-decider, where $p^2+q\leq 1$, and introduce an infinite hierarchy of classes within this family, characterized by the specific relationships between the parameters p and q. As we shall see, our results imply that the distributed computing setting contrasts significantly with the sequential one as far as randomization is concerned. Indeed, we prove that in some cases, the ability to increase the success probability for deciding distributed languages is very limited.

Model. We consider the $\mathcal{LOCAL}$ model (cf. [28]), which is a standard distributed computing model capturing the essence of spatial locality. In this model, processors 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. It is

important to stress that all the algorithmic constructions that we employ in our positive results use messages of constant size (some of which do not use any communication at all). Hence, all our results apply not only to the $\mathcal{LOCAL}$ model of computation but also to more restricted models, for example, the $\mathcal{CONGEST}(B)$ model[3], where $B = O(1)$.

A distributed algorithm $\mathcal{A}$ that runs on a graph G operates separately on each connected component, and nodes of a component C of G cannot distinguish the underlying graph G from C. Therefore, we consider connected graphs only.

We focus on *distributed decision tasks*. Such a task is characterized by a finite or infinite set Σ of symbols (e.g., $\Sigma = \{0, 1\}$, or $\Sigma = \{0, 1\}^*$), and by a *distributed language* $\mathcal{L}$ defined on this set of symbols (see below). An *instance* of a distributed decision task is a pair $(G, \mathbf{x})$ where G is an n-node connected graph, and $\mathbf{x} \in \Sigma^n$, that is, every node $v \in V(G)$ is assigned as its *local input* a value $\mathbf{x}(v) \in \Sigma$. (In some cases, the local input of every node is empty, i.e., $\Sigma = \{\epsilon\}$, where ϵ denotes the empty binary string.) We define a *distributed language* as a decidable collection $\mathcal{L}$ of instances[4].

In the context of distributed computing, each processor must produce a boolean output, and the decision is defined by the conjunction of the processors outputs, i.e., if the instance belongs to the language, then all processors must output "yes", and otherwise, at least one processor must output "no". Formally, for a distributed language $\mathcal{L}$, we say that a distributed algorithm $\mathcal{A}$ *decides* $\mathcal{L}$ if and only if for every instance $(G, \mathbf{x})$ and id-assignment Id, every node v of G eventually terminates and produces an output denoted $\text{out}_{\mathcal{A}}(G, \mathbf{x}, \text{Id}, v)$, which is either "yes" or "no", satisfying the following decision rules:

- If $(G, \mathbf{x}) \in \mathcal{L}$, then $\text{out}_{\mathcal{A}}(G, \mathbf{x}, \text{Id}, v) =$ "yes" for every node $v \in V(G)$;
- If $(G, \mathbf{x}) \notin \mathcal{L}$, then $\text{out}_{\mathcal{A}}(G, \mathbf{x}, \text{Id}, v) =$ "no" for at least one node $v \in V(G)$.

Decision problems provide a natural framework for tackling fault-tolerance: the processors have to collectively check if the network is fault-free, and a node detecting a fault raises an alarm. Many natural problems can be phrased as decision problems, for example: "is the network planar?" or "is there a unique leader in the network?". Moreover, decision problems occur naturally when one aims at checking the validity of the output of a computational task, such as "is the produced coloring legal?", or "is the constructed subgraph an MST?".

The class of decision problems that can be solved in at most t communication rounds is denoted by $\text{LD}(t)$, for *local decision*. More precisely, let t be a function of triplets $(G, \mathbf{x}, \text{Id})$, where Id denotes the identity assignment to the nodes of G. Then $\text{LD}(t)$ is the class of all distributed languages that can be decided by a distributed algorithm that runs in at most t communication rounds. The randomized (Monte Carlo 2-sided error) version of the class $\text{LD}(t)$ is denoted $\text{BPLD}(t, p, q)$, which stands for *bounded-error probabilistic local decision*, and provides an analog of BPP for distributed computing, where p and q respectively

[3] The $\mathcal{CONGEST}(B)$ model is similar to the $\mathcal{LOCAL}$ model, except that message size is assumed to be bounded by B (cf. [28]).

[4] Note that an undecidable collection of instances remains undecidable in the distributed setting too.

denote the yes-error and the no-error guarantees. More precisely, a *randomized* distributed algorithm is a distributed algorithm $\mathcal{A}$ that enables every node v, at any round r during its execution, to generate a certain number of random bits. For constants $p, q \in (0, 1]$, we say that a randomized distributed algorithm $\mathcal{A}$ is a (p, q)-*decider* for $\mathcal{L}$, or, that it decides $\mathcal{L}$ with "yes" success probability p and "no" success probability q, if and only if for every instance $(G, \mathbf{x})$ and id-assignment Id, every node of G eventually terminates and outputs "yes" or "no", and the following properties are satisfied:

- If $(G, \mathbf{x}) \in \mathcal{L}$ then $\Pr[\forall v \in V(G), \text{out}_{\mathcal{A}}(G, \mathbf{x}, \text{Id}, v) = \text{"yes"}] \geq p$;
- If $(G, \mathbf{x}) \notin \mathcal{L}$ then $\Pr[\exists v \in V(G), \text{out}_{\mathcal{A}}(G, \mathbf{x}, \text{Id}, v) = \text{"no"}] \geq q$.

The probabilities in the above definition are taken over all possible coin tosses performed by the nodes. The running time of a (p, q)-decider executed on a node v depends on the triple $(G, \mathbf{x}, \text{Id})$ and on the results of the coin tosses. In the context of a randomized algorithm, $T_v(G, \mathbf{x}, \text{Id})$ denotes the maximal running time of the algorithm on v over all possible coin tosses, for the instance $(G, \mathbf{x})$ and id-assignment Id. Now, just as in the deterministic case, the running time T of the (p, q)-decider is the maximum running time over all nodes. Note that by definition of the distributed Monte-Carlo algorithm, both T_v and T are deterministic. For constant $p, q \in (0, 1]$ and a function t of triplets $(G, \mathbf{x}, \text{Id})$, $\text{BPLD}(t, p, q)$ is the class of all distributed languages that have a randomized distributed (p, q)-decider running in time at most t (i.e., can be decided in time at most t by a randomized distributed algorithm with "yes" success probability p and "no"success probability q).

Our main interest within this context is in studying the connections between the classes $\text{BPLD}(t, p, q)$. In particular, we are interested in the question of whether one can "boost" the success probabilities of a (p, q)-decider. (Recall that in the *sequential* Monte Carlo setting, such "boosting" can easily be achieved by repeating the execution of the algorithm a large number of times.) Our starting point is the recent result of [13] that, for the class of hereditary languages (i.e., closed under sub-graphs), the relation $p^2+q = 1$ is a sharp threshold for randomization. That is, for hereditary languages, $\bigcup_{p^2+q>1} \text{BPLD}(t, p, q)$ collapses to $\text{LD}(O(t))$, but for any $p, q \in (0, 1]$ such that $p^2 + q \leq 1$ there exists a language $\mathcal{L} \in \text{BPLD}(0, p, q)$, while $\mathcal{L} \notin \text{LD}(t)$ for any $t = o(n)$. We conjecture that the hereditary assumption can be removed and we give some evidence supporting this conjecture. Aiming at analyzing the collection of classes $\bigcup_{p^2+q\leq 1} \text{BPLD}(t, p, q)$, we consider the set of classes

$$B_\infty(t) = \bigcup_{p+q>1} \text{BPLD}(t, p, q), \quad B_k(t) = \bigcup_{p^{1+1/k}+q>1} \text{BPLD}(t, p, q) \text{ for any } k \in \mathbb{Z}^+ .$$

Hence, our conjecture states that $B_1(t) = \text{LD}(O(t))$. Note that the class $\bigcup_{p+q\geq 1} \text{BPLD}(0, p, q)$ contains *all* languages, using a $(1, 0)$-decider that systematically returns "yes" at every node (without any communication). Hence, the classes B_k provide a smooth spectrum of randomized distributed complexity classes, from the class of deterministically decidable languages (under our conjecture) to the class of all languages. The ability of boosting the success prob-

abilities of a (p, q)-decider is directly related to the question of whether these classes are different, and to what extent.

Our Results. One of the main outcomes of this paper is a proof that boosting success probabilities in the distributed setting appears to be quite limited. By definition, $B_k(t) \subseteq B_{k+1}(t)$ for any k and t. We prove that these inclusions are strict. In fact, we prove a stronger separation result: there exists a language in $B_{k+1}(0)$ that is not in $B_k(t)$ for any $t = o(n)$, and moreover, $\texttt{Tree} \notin B_\infty(t)$ for any $t = o(n)$, where $\texttt{Tree} = \{(G, \epsilon) : G \text{ is a tree}\}$. Hence $B_\infty(t)$ does not contain all languages, even for $t = o(n)$. In summary, we obtain the hierarchy

$$B_1(t) \subset B_2(t) \subset \cdots \subset B_\infty(t) \subset \text{All} .$$

These results demonstrate that boosting the probability of success might be doable, but only from a (p, q) pair satisfying $p^{1+1/(k+1)} + q > 1$ to a (p, q) pair satisfying $p^{1+1/k} + q > 1$ (with the extremes excluded). It is an open question whether $B_{k+1}(t)$ actually collapses to $\text{BPLD}(O(t), p, q)$, where $p^{1+1/k} + q = 1$, or whether there exist intermediate classes.

Recall that every hereditary language in $B_1(t)$ is also in $\text{LD}(O(t))$ [13]. We conjecture that this derandomization result holds for all languages and not only for hereditary ones. We give evidence supporting this conjecture by showing that restricted to path topologies, finite input and constant running time t, the statement $B_1(t) \subseteq \text{LD}(O(t))$ holds without assuming the hereditary property. This evidence seems to be quite meaningful especially since all our separation results hold even if we restrict ourselves to decision problems on path topologies.

Finally, we show that the situation changes drastically if the distribution of inputs can be restricted in certain ways. Indeed, we show that for every two reals $0 < r < r'$, there exists a language in $C_{r'}(0)$ that is not in $C_r(t)$ for any $t = o(n)$, where the C-classes are the extension of the B-classes to decision problems in which the inputs can be restricted.

All our results hold not only for the $\mathcal{LOCAL}$ model but also for more restrictive models of computation, such as the $\mathcal{CONGEST}(B)$ model (for $B = O(1)$).

Related Work. The notion of local decision and local verification of languages has received quite a lot of attention recently. In the $\mathcal{LOCAL}$ model, solving a decision problem requires the processors to independently inspect their local neighborhood and collectively decide whether the global instance belongs to some specified language. Inspired by classical computation complexity theory, Fraigniaud et al. [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, efforts were made to form a fundamental computational complexity theory for distributed decision problems in various other aspects of distributed computing [13,14,15,16].

The classes LD, NLD and BPLD defined in [13] are the distributed analogues of the classes P, NP and BPP, respectively. The contribution of [13] is threefold: it establishes the impact of nondeterminism, randomization, and randomization + nondeterminism, on local computation. This is done by proving structural results, developing a notion of local reduction and establishing completeness

results. One of the main results is the existence of a sharp threshold above which randomization does not help (at least for hereditary languages), and the BPLD classes were classified into two: below and above the randomization threshold. The current paper "zooms" into the spectrum of classes below the randomization threshold, and defines a infinite hierarchy of BPLD classes, each of which is separated from the class above it in the hierarchy.

The question of whether randomization helps in improving locality for construction problems has been studied extensively. Naor and Stockmeyer [26] considered a subclass of LD($O(1)$), called LCL[5], and studied the question of how to compute in $O(1)$ rounds the constructive versions of decision problems in LCL. The paper demonstrates that randomization does not help, in the sense that if a problem has a local Monte Carlo randomized algorithm, then it also has a local deterministic algorithm. There are several differences between the setting of [26] and ours. First, [26] considers the power of randomization for *constructing* a solution, whereas we study the power of randomization for *deciding* languages[6]. Second, while [26] deals with constant time computations, our separation results apply to arbitrary time computations, potentially depending on the size of the instance (graph and input). The different settings imply different impacts for randomization: while the current paper and [13] show that randomization can indeed help for improving locality of decision problems, [26] shows that randomization does *not* help in constructing a solution for a problem in LCL in constant time. The question of whether randomization helps in local computations was studied for *specific* problems, such as MIS, $(\Delta+1)$-coloring, and maximal matching [2,5,23,24,25,27,29]. Finally, the classification of decision problems in distributed computing has been studied in several other models. For example, [6] and [18] study specific decision problems in the $\mathcal{CONGEST}$ model. In addition, decision problems have been studied in the asynchrony discipline too, specifically in the framework of *wait-free computation* [15,16] and *mobile agents computing* [14]. In the wait-free model, the main issues are not spatial constraints but timing constraints (asynchronism and faults). The main focus of [16] 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"), [16] 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.

Preliminaries. In the $\mathcal{LOCAL}$ (respectively $\mathcal{CONGEST}(B)$) model, processors perform in synchronous rounds, where in each round, every processor (1) sends messages of arbitrary (resp., $O(B)$ bits) size to its neighbors, (2) receives messages from its neighbors, and (3) performs arbitrary individual computations.

[5] LCL is essentially LD($O(1)$) restricted to languages involving graphs of constant maximum degree and processor inputs taken from a set of constant size.

[6] There is a fundamental difference between such tasks when locality is concerned. Indeed, whereas the validity of constructing a problem in LCL is local (by definition), the validity in our setting is "global", in the sense that in an illegal instance, it is sufficient that at least one vertex in the entire network outputs "no".

After a number of rounds (that may depend on the network G connecting the processors, and may vary among the processors, since nodes have different identities, potentially different inputs, and are typically located at non-isomorphic positions in the network), every processor v terminates and generates its output.

Consider a distributed (p, q)-decider $\mathcal{A}$ running in a network G with input $\mathbf{x}$ and identity assignment Id (assigning distinct integers to the nodes of G). The output of processor v in this scenario is denoted by $\text{out}_{\mathcal{A}}(G, \mathbf{x}, \text{Id}, v)$, or simply $\text{out}(v)$ when the parameters are clear from the context. In the case of decision problem, $\text{out}(v) \in \{\text{“yes”}, \text{“no”}\}$ for every processor v.

An n-node path P is represented as a sequence $P = (1, \ldots, n)$, oriented from left to right. (Node i does not know its position in the path.) Given an instance $(P, \mathbf{x})$ with ID's Id and a subpath $S \subset P$, let $\mathbf{x}_S$ (respectively Id_S) be the restriction of $\mathbf{x}$ (resp., Id) to S. We may refer to subpath $S = (u_i, \ldots, u_j) \subset P$ as $S = [i, j]$. For a set $U \subseteq V(G)$, let $\mathcal{E}(G, \mathbf{x}, \text{Id}, U)$ denote the event that, when running $\mathcal{A}$ on $(G, \mathbf{x})$ with id-assignment Id, all nodes in U output “yes”. Given a language $\mathcal{L}$, an instance $(G, \mathbf{x})$ is called *legal* iff $(G, \mathbf{x}) \in \mathcal{L}$. Given a time bound t, a subpath $S = [i, j]$ is called an *internal* subpath of P if $i \geq t + 2$ and $j \leq n - t - 1$. Note that if the subpath S is internal to P, then when running a t-round algorithm, none of the nodes in S “sees” the endpoints of P.

The following concept is crucial to the proofs of our separation results.

Definition 1. *Let S be a subpath of P. For $\delta \in [0, 1]$, S is said to be a (δ, λ)-*secure *subpath if $|S| \geq \lambda$ and $\Pr[\mathcal{E}(P, \mathbf{x}, \text{Id}, V(S))] \geq 1 - \delta$.*

We typically use (δ, λ)-secure subpaths for values of $\lambda \geq 2t + 1$ where t is the running time of the (p, q)-decider $\mathcal{A}$ on $(P, \mathbf{x})$ for some fixed identity assignment Id. Indeed, it is known [13] that if $(P, \mathbf{x}) \in \mathcal{L}$, then every long enough subpath S of P contains an internal (δ, λ)-secure subpath S'. More precisely, define

$$\ell(\delta, \lambda) = 4(\lambda + 2t)\lceil \log p / \log(1 - \delta) \rceil. \tag{1}$$

Fact 1 ([13]) *Let $(P, \mathbf{x}) \in \mathcal{L}$, $\delta \in [0, 1]$, $\lambda \geq 1$. Then for every $\ell(\delta, \lambda)$-length subpath S there is a subpath S' (internal to S) that is (δ, λ)-secure.*

To avoid cumbersome notation, when $\lambda = 2t + 1$, we may omit it and refer to $(\delta, 2t + 1)$-secure subpaths as δ-secure subpaths. In addition, set

$$\ell(\delta) := \ell(\delta, 2t + 1).$$

Let us next illustrate a typical use of Fact 1. Recall that t denotes the running time of the (p, q)-decider $\mathcal{A}$ on $(P, \mathbf{x}) \in \mathcal{L}$ with IDs Id. Let S be a subpath of P of length $\ell(\delta)$. Denote by L (resp., R) the subpath of P to the “left” (resp., “right”) of S. Informally, if the length of S is larger than $2t + 1$, then S serves as a separator between the two subpaths L and R. This follows since as algorithm $\mathcal{A}$ runs in t rounds, each node in P is affected only by its t neighborhood. As the t neighborhood of every node $u \in L$ and $v \in R$ do not intersect, the events $\mathcal{E}(P, \mathbf{x}, \text{Id}, L)$ and $\mathcal{E}(P, \mathbf{x}, \text{Id}, R)$ are independent.

The security property becomes useful when upper bounding the probability that at least some node in P says "no", by applying a union bound on the events $\mathcal{E}(P, \mathbf{x}, \mathrm{Id}, V(L) \cup V(R))$ and $\mathcal{E}(P, \mathbf{x}, \mathrm{Id}, V(S))$. Denoting the event complementary to $\mathcal{E}(P, \mathbf{x}, \mathrm{Id}, V(P))$, by $\mathcal{E}'$ we have

$$\begin{aligned}\Pr[\mathcal{E}'] &= 1 - \Pr[\mathcal{E}(P, \mathbf{x}, \mathrm{Id}, V(P))] \\ &\leq (1 - \Pr[\mathcal{E}(P, \mathbf{x}, \mathrm{Id}, V(L))] \cdot \Pr[\mathcal{E}(P, \mathbf{x}, \mathrm{Id}, V(R))]) \\ &\quad + (1 - \Pr[\mathcal{E}(P, \mathbf{x}, \mathrm{Id}, V(S))]) \\ &\leq 1 - \Pr[\mathcal{E}(P, \mathbf{x}, \mathrm{Id}, V(L))] \cdot \Pr[\mathcal{E}(P, \mathbf{x}, \mathrm{Id}, V(R))] + \delta \ .\end{aligned}$$

The specific choice of λ and δ depends on the context. Informally, the guiding principle is to set δ small enough so that the role of the central section S can be neglected, while dealing separately with the two extreme sections L and R become manageable for they are sufficiently far apart.

2 The B_k Hierarchy is Strict

In this section we show that the classes B_k, $k \geq 1$, form an infinite hierarchy of distinct classes, thereby proving that the general ability to boost the probability of success for a randomized decision problem is quite limited. In fact, we show separation in a very strong sense: there are decision problems in $B_{k+1}(0)$, i.e., that have a (p, q)-decider running in zero rounds with $p^{1+1/(k+1)} + q > 1$, which cannot be decided by a (p, q)-decider with $p^{1+1/k} + q > 1$, even if the number of rounds of the latter is as large as $n^{1-\varepsilon}$ for every fixed $\varepsilon > 0$.

Theorem 2. *$B_{k+1}(0) \setminus B_k(t) \neq \emptyset$ for every $k \geq 1$ and every $t = o(n)$.*

Proof. Let k be any positive integer. We consider the following distributed language, which is a generalized variant `AMOS-`k of the problem `AMOS` introduced in [13]. As in `AMOS`, the input $\mathbf{x}$ of `AMOS-`k satisfies $\mathbf{x} \in \{0, 1\}^n$, i.e., each node v is given as input a boolean $\mathbf{x}(v)$. The language `AMOS-`k is then defined by:

$$\texttt{At-Most-k-Selected (AMOS-}k\texttt{)} = \{(G, \mathbf{x}) \text{ s.t. } \| \mathbf{x} \|_1 \leq k\}.$$

Namely, `AMOS-`k consists of all instances containing at most k selected nodes (i.e., at most k nodes with input 1), with all other nodes unselected (having input 0). In order to prove Theorem 2, we show that `AMOS-`$k \in B_{k+1}(0) \setminus B_k(t)$ for every $t = o(n)$.

We first establish that `AMOS-`k belongs to $B_{k+1}(0)$. We adapt algorithm $\mathcal{A}$ presented in [13] for `AMOS` to the case of `AMOS-`k. The following simple randomized algorithm runs in 0 time: every node v which is not selected, i.e., such that $\mathbf{x}(v) = 0$, says "yes" with probability 1; and every node which is selected, i.e., such that $\mathbf{x}(v) = 1$, says "yes" with probability $p^{1/k}$, and "no" with probability $1 - p^{1/k}$. If the graph has $s \leq k$ nodes selected, then all nodes say "yes" with probability $p^{s/k} \geq p$, as desired. On the other hand, if there are $s \geq k + 1$ selected nodes, then at least one node says "no" with probability $1 - p^{s/k} \geq$

$1 - p^{(k+1)/k} = 1 - p^{1+1/k}$. We therefore get a (p, q)-decider with $p^{1+1/k} + q \geq 1$, that is, such that $p^{1+1/(k+1)} + q > 1$. Thus $\texttt{AMOS-}k \in B_{k+1}(0)$.

We now consider the harder direction, and prove that $\texttt{AMOS-}k \notin B_k(t)$, for any $t = o(n)$. To prove this separation, it is sufficient to consider $\texttt{AMOS-}k$ restricted to the family of n-node paths. Fix a function $t = o(n)$, and assume, towards contradiction, that there exists a distributed (p, q)-decider $\mathcal{A}$ for $\texttt{AMOS-}k$ that runs in $O(t)$ rounds, with $p^{1+1/k} + q > 1$. Let $\varepsilon \in (0, 1)$ be such that $p^{1+1/k+\varepsilon} + q > 1$. Let P be an n-node path, and let $S \subset P$ be a subpath of P. Let $\delta \in [0, 1]$ be a constant satisfying

$$0 < \delta < p^{1+1/k} \left(1 - p^{\varepsilon}\right) / k \ . \tag{2}$$

Consider a positive instance and a negative instance of $\texttt{AMOS-}k$, respectively denoted by $I = (P, \mathbf{x})$ and $I' = (P, \mathbf{x}')$. Both instances are defined on the same n-node path P, where $n \geq k\left(\ell(\delta) + 1\right) + 1$. Recall that $\ell(\delta) = \ell(\delta, 2t + 1)$ (see Eq. (1)). We consider executions of $\mathcal{A}$ on these two instances, where nodes are given the same id's. Both instances have almost the same input. In particular, the only difference is that instance I contains k selected nodes, whereas I' has the same selected nodes as I plus one additional selected node. Therefore I is legal, while I' is illegal. In I', the path P is composed of $k + 1$ sections, each containing a unique selected node, and where each pair of consecutive sections separated by a δ-secure subpath. More precisely, let us enumerate the nodes of P from 1 to n, with node v adjacent to nodes $v - 1$ and $v + 1$, for every $1 < v < n$. Consider the k subpaths of P defined by: $S_i = [(i - 1)\ell(\delta) + i + 1, i \cdot \ell(\delta) + i]$ for $i = \{1, \ldots, k\}$. Let the selected nodes in I' be positioned as follows. Let $u_1 = 1$ and let $u_i = (i - 1)\ell(\delta) + i$ for $i = 2, \ldots, k + 1$. Then set

$$\mathbf{x}'(v) = \begin{cases} 1, \text{ if } v = u_i \text{ for some } i \in \{1, ..., k + 1\} \\ 0, \text{ otherwise.} \end{cases}$$

See Fig. 1(a) for a schematic representation of I'.

Our next goal is to define the legal instance $I = (P, \mathbf{x})$. To do so, we begin by claiming that each S_i contains a δ-secure internal subpath $S'_i = [a_i, b_i]$. Naturally, we would like to employ Fact 1. However, Fact 1 refers to subpaths of *valid* instances $(P, \mathbf{x}) \in \mathcal{L}$, and I' is illegal. So instead, let us focus on the instance $(S_i, \mathbf{x}'_{S_i})$. Since $(S_i, \mathbf{x}'_{S_i})$ contains no leaders, $\| \mathbf{x}'_{S_i} \|_1 = 0$, it follows that $(S_i, \mathbf{x}'_{S_i}) \in \mathcal{L}$, and Fact 1 can be applied on it. Subsequently, since $|S_i| > \ell(\delta)$ it follows that S_i contains an internal δ-secure subpath $S'_i = [a_i, b_i]$, whose t neighborhood is strictly in S_i. Therefore, when applying algorithm $\mathcal{A}$ on $(S_i, \mathbf{x}'_{S_i}, \mathrm{Id}_{S_i})$ and on $(P, \mathbf{x}', \mathrm{Id})$, the nodes in the $(2t + 1)$-length segment S'_i behave the same, thus $\Pr[\mathcal{E}(P, \mathbf{x}', \mathrm{Id}, V(S'_i))] = \Pr[\mathcal{E}(S_i, \mathbf{x}'_{S_i}, \mathrm{Id}_{S_i}, V(S'_i))]$. Hence, S'_i is a δ-secure subpath in $(P, \mathbf{x}', \mathrm{Id})$ as well, for every $i \in \{1, ..., k\}$, see Fig. 1(b).

The δ-secure subpaths S'_i's are now used to divide P into $2k + 1$ segments. Specifically, there are $k + 1$ segments T_i, $i = 1, \ldots, k + 1$, each with one selected node. The δ-secure subpaths $S'_i = [a_i, b_i]$ separate T_i from T_{i+1}. More precisely, set $T_1 = [1, a_1 - 1]$, $T_i = [b_{i-1} + 1, a_i - 1]$ for $i \in 2, ..., k$, and $T_{k+1} = [b_k + 1, n]$, getting $P = T_1 \circ S'_1 \circ T_2 \circ S'_2 \circ \ldots \circ T_k \circ S'_k \circ T_{k+1}$ where $\circ$ denotes path concatenation. Let $\mathcal{T}_i = \mathcal{E}(P, \mathbf{x}', \mathrm{Id}, V(T_i))$ be the event that all nodes in the subpath T_i say

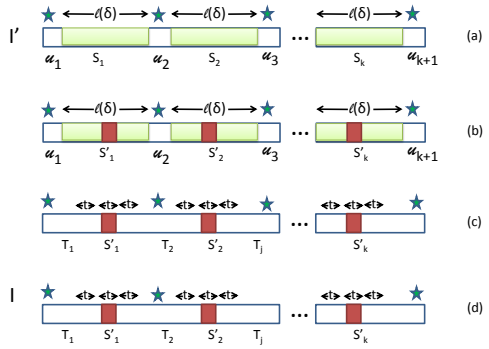

Fig. 1: Illustration of the constructions for Theorem 2. (a) The instance $I' = (P, \mathbf{x}')$ with $k+1$ leaders separated by $\ell(\delta)$-length segements, S_i. (b) The δ-secure subpaths S'_i in each S_i are internal to S_i. (c) The leader-segments T_i interleaving with δ-secure subpaths S'_i. (d) The legal instance $I = (P, \mathbf{x})$, the j^{th} leader of I' is discarded, resulting in a k leader instance.

"yes" in the instance I', for $i \in \{1, ..., k+1\}$ and let $p_i = \Pr[\mathcal{T}_i]$ be its probability. Let j be such that $p_j = \max_i p_i$. We now define the valid instance $I = (P, \mathbf{x})$:

$$\mathbf{x}(v) = \begin{cases} 1, \text{ if } v = u_i \text{ for some } i \in \{1, ..., k+1\},\ i \neq j, \\ 0, \text{ otherwise.} \end{cases}$$

Note that $\| \mathbf{x}' \|_1 = k+1$ and $\| \mathbf{x} \|_1 = k$, thus $I \in$ `AMOS-`k while $I' \notin$ `AMOS-`k. See Fig. 1(c,d) for an illustration of I versus I'.

We now make the following observation (the proof defer to full version).

Claim. $\forall i \neq j,\ \Pr[\mathcal{E}(P, \mathbf{x}, \mathrm{Id}, V(T_i))] = p_i$.

Let $\mathcal{N}$ (resp., $\mathcal{N}'$) be the event that there exists at least one node in I (resp., I') that says "no" when applying algorithm $\mathcal{A}$. Similarly, let $\mathcal{Y}$ (resp., $\mathcal{Y}'$) be the event stating that all nodes in the configuration I (resp., I') say "yes". Let $\mathcal{T} = \bigcup_{i=1}^{k+1} \mathcal{T}_i$ be the event that all nodes in each subpaths T_i, for $i \in \{1, ..., k+1\}$ say "yes" in the instance I'. For every $i \in \{1, ..., k\}$, let $\mathcal{S}_i = \mathcal{E}(P, \mathbf{x}', \mathrm{Id}, V(S'_i))$ be the event that all nodes in the δ-secure subpath S'_i say "yes" in the instance I'. We have $\Pr(\mathcal{Y}) = \Pr[\mathcal{E}(P, \mathbf{x}, \mathrm{Id}, V(P))]$ and $\Pr(\mathcal{Y}') = \Pr[\mathcal{E}(P, \mathbf{x}', \mathrm{Id}, V(P))$, while $\Pr(\mathcal{N}) = 1 - \Pr(\mathcal{Y})$ and $\Pr(\mathcal{N}') = 1 - \Pr(\mathcal{Y}')$.

Since $\mathcal{A}$ is a (p, q)-decider, as we assume by contradiction that `AMOS-`k in B_k, we have $\Pr(\mathcal{N}') \geq q$, and thus $\Pr(\mathcal{N}') > 1 - p^{1+1/k+\varepsilon}$. Therefore, $\Pr(\mathcal{Y}') < p^{1+1/k+\varepsilon}$. Moreover, since $I \in$ `AMOS-`k, we also have that $\Pr(\mathcal{Y}) \geq p$. Therefore, the ratio $\hat{\rho} = \Pr(\mathcal{Y}')/\Pr(\mathcal{Y})$ satisfies

$$\hat{\rho} < p^{1/k+\varepsilon} \ . \tag{3}$$

On the other hand, note that by applying the union bound to the $k+1$ events $\mathcal{T}, \bigcup_{i=1}^{k} \mathcal{S}_i$, we get

$$\Pr(\mathcal{N}') \leq (1 - \Pr[\mathcal{T}]) + \left(\sum_{i=1}^{k} (1 - \Pr[\mathcal{S}_i]) \right) \leq 1 - p_j \cdot \prod_{i \neq j} p_i + k \cdot \delta,$$

where the last inequality follows by the fact that each S'_i is a $(\delta, 2t+1)$-secure subpath, thus the events $\mathcal{T}_{i_1}, \mathcal{T}_{i_2}$ are independent for every $i_1, i_2 \in \{1, ..., k+1\}$ (since the distance between any two nodes $u \in T_{i_1}$ and $v \in T_{i_2}$ is at least $2t+1$). This implies that $\Pr(\mathcal{Y}') \geq p_j \cdot \prod_{i \neq j} p_i - k \cdot \delta$. Since $\Pr(\mathcal{Y}) \leq \prod_{i \neq j} p_i$ (by the independence of the events $\mathcal{T}_{i_1}, \mathcal{T}_{i_2}$, for every $i_1, i_2 \in \{1, ...k+1\}$), it then follows that the ratio $\hat{\rho}$ satisfies

$$\hat{\rho} \geq \frac{p_j \cdot \prod_{i \neq j} p_i - k \cdot \delta}{\prod_{i \neq j} p_i} \geq p_j - \frac{k \cdot \delta}{\prod_{i \neq j} p_i} \geq p_j - k \cdot \delta / p \ , \tag{4}$$

where the last inequality follows by the fact that $I \in \texttt{AMOS-}k$ and thus $\prod_{i \neq j} p_i \geq \Pr(\mathcal{Y}) \geq p$. Finally, note that $p_j \geq p^{1/k}$. This follows since $p_j \geq p_i$ for every $i \in \{1, ..., k+1\}$, so $p_j^k \geq \prod_{i \neq j} p_i \geq p$. By Eq. (4), we then have that $\hat{\rho} \geq p^{1/k} - k \cdot \delta / p$. Combining this with Eq. (3), we get that $p^{1/k} - k \cdot \delta / p < p^{1/k + \varepsilon}$, which is in contradiction to the definition of δ in Eq. (2).

Finally, we show that the $B_k(t)$ hierarchy does not capture all languages even for $k = \infty$ and t as large as $o(n)$. Due to space limitations, the proof of the following theorem is deferred to the full version of this paper.

Theorem 3. *There is a language not in $B_\infty(t)$, for every $t = o(n)$.*

3 A Sharp Determinism - Randomization Threshold

It is known [13] that beyond the threshold $p^2 + q = 1$, randomization does not help. This result however holds only for a particular type of languages, called *hereditary*, i.e., closed under inclusion. In this section, we provide one more evidence supporting our belief that the threshold $p^2 + q = 1$ identified in [13] holds for *all* languages, and not only for hereditary languages. Indeed, we prove that, restricted to path topologies and finite inputs, *every* language $\mathcal{L}$ for which there exists a (p, q)-decider running in constant time, with $p^2 + q > 1$, can actually be decided deterministically in constant time.

Theorem 4. *Let $\mathcal{L}$ be a distributed language restricted to paths, with a finite set of input values. If $\mathcal{L} \in B_1(O(1))$, then $\mathcal{L} \in \mathrm{LD}(O(1))$.*

Proof. Let $\mathcal{L} \in B_1(O(1))$ be a distributed language restricted to paths, and defined on the (finite) input set Σ. Consider a distributed (p, q)-decider $\mathcal{A}$ for $\mathcal{L}$ that runs in $t = O(1)$ rounds, with $p^2 + q > 1$. Fix a constant δ such that $0 < \delta < p^2 + q - 1$.

Given a subpath S of a path P, let us denote by S_l (respectively, S_r) the subpath of P to the left (resp., right) of S, so that $P = S_l \circ S \circ S_r$.

Informally, a collection of three paths P, P', and P'' (of possibly different lengths) is called a *λ-path triplet* if (1) the inputs of those paths agree on some "middle" subpath of size at least λ, (2) paths P and P'' coincide on their corresponding "left" parts, and (3) paths P' and P'' coincide on their "right" parts. See Fig. 2. Formally, a *λ-path triplet* is a triplet $[(P, S, \mathbf{x}), (P', S', \mathbf{x}'), (P'', S'', \mathbf{x}'')]$ such that $|P|, |P'|, |P''| \geq \lambda$, $\mathbf{x}, \mathbf{x}', \mathbf{x}''$ are inputs on these paths, respectively, and $S \subset P$, $S' \subset P'$, $S'' \subset P''$ are three subpaths satisfying (1) $|S| = |S'| = |S''| \geq \lambda$, (2) $\mathbf{x}_S = \mathbf{x}'_{S'} = \mathbf{x}''_{S''}$, and (3) $\mathbf{x}''_{S''_l} = \mathbf{x}_{S_l}$ and $\mathbf{x}''_{S''_r} = \mathbf{x}'_{S'_r}$.

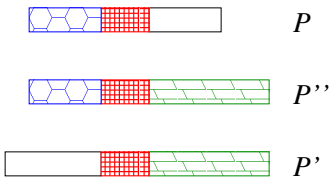

Fig. 2: Example of a λ-path triplet (the red zone is of length at least λ).

Claim. Let $[(P, S, \mathbf{x}), (P', S', \mathbf{x}'), (P'', S'', \mathbf{x}'')]$ be a λ-path triplet. If $\lambda \geq \ell(\delta)$, for ℓ as defined in Eq. (1), then $\Big((P, \mathbf{x}) \in \mathcal{L} \text{ and } (P', \mathbf{x}') \in \mathcal{L}\Big) \Rightarrow (P'', \mathbf{x}'') \in \mathcal{L}$.

Proof. Consider an identity assignment Id'' for $(P'', \mathbf{x}'')$. Let Id and Id' be identity assignments for $(P, \mathbf{x})$, and $(P', \mathbf{x}')$, respectively, which agree with Id'' on the corresponding nodes. That is: (a) assignments Id, Id', and Id'' agree on the nodes in S, S' and S'', respectively; (b) Id and Id'' agree on the nodes in S_ℓ and S''_ℓ, respectively; and (c) Id and Id'' agree on the nodes in S'_r and S''_r, respectively. Since $(P, \mathbf{x}) \in \mathcal{L}$, and since $|S| = \lambda \geq \ell(\delta)$, it follows from Fact 1 that S contains an internal δ-secure subpath H. Then, let H' and H'' be the subpaths of P' and P'' corresponding to H. Since S and S'' coincide in their inputs and identity assignments, then H, H', H'' have the same t-neighborhood in P, P', P'' respectively. Hence, H'' is also a δ-secure (when running algorithm $\mathcal{A}$ in instance $(P'', \mathbf{x}'')$). Since both $(P, \mathbf{x})$ and $(P', \mathbf{x}')$ belong to $\mathcal{L}$, we have

$$\Pr[\mathcal{E}(H''_\ell, \mathrm{Id}'', \mathbf{x}'') = \Pr[\mathcal{E}(H_\ell, \mathrm{Id}, \mathbf{x})])] \geq p$$
$$\Pr[\mathcal{E}(H''_r, \mathrm{Id}'', \mathbf{x}'') = \Pr[\mathcal{E}(H'_r, \mathrm{Id}', \mathbf{x}')])] \geq p.$$

Moreover, as $|H''| \geq 2t+1$, the two events $\mathcal{E}(H''_\ell, \mathrm{Id}'', \mathbf{x}'')$ and $\mathcal{E}(H''_r, \mathrm{Id}'', \mathbf{x}'')$ are independent. Hence $\Pr[\mathcal{E}(H''_\ell \cup H''_r, \mathrm{Id}'', \mathbf{x}'')] \geq p^2$. In other words, the probability that some node in $H''_\ell \cup H''_r$ says "no" is at most $1-p^2$. It follows, by union bound, that the probability that some node in H'' says "no" is at most $1 - p^2 + \delta < q$. Since $\mathcal{A}$ is a (p, q)-decider for $\mathcal{L}$, it cannot be the case that $(H'', \mathbf{x}'') \notin \mathcal{L}$.

We now observe that, w.l.o.g., one can assume that in all instances $(P, \mathbf{x})$ of $\mathcal{L}$, the two extreme vertices of the path P have a special input symbol $\otimes$. To see

why this holds, let $\otimes$ be a symbol not in Σ, and consider the following language $\mathcal{L}'$ defined over $\Sigma \cup \{\otimes\}$. Language $\mathcal{L}'$ consists of instances $(P, \mathbf{x})$ such that (1) the endpoints of P have input $\otimes$, and (2) $(P', \mathbf{x}') \in \mathcal{L}$, where P' is the path resulting from removing the endpoints of P, and where $\mathbf{x}'_v = \mathbf{x}_v$ for every node v of P'. Any (p, q) decider algorithm for $\mathcal{L}$ (resp., $\mathcal{L}'$), can be trivially transformed into a (p, q) decider algorithm for $\mathcal{L}'$ (resp., $\mathcal{L}$) with the same success guarantees and running time. Hence, in the remaining of the proof, we assume that in all instances $(P, \mathbf{x}) \in \mathcal{L}$, the two extreme vertices of the path P have input $\otimes$.

A given instance $(P, \mathbf{x})$ is *extendable* if there exists an extension of it in $\mathcal{L}$, i.e., if there exists an instance $(P', \mathbf{x}') \in \mathcal{L}$ such that $P \subseteq P'$ and $\mathbf{x}'_P = \mathbf{x}$.

Claim. There exists a (centralized) algorithm $\mathcal{X}$ that, given any configuration $(P, \mathbf{x})$ with $|P| \leq 2\ell(\delta) + 1$, decides whether $(P, \mathbf{x})$ is extendable. Moreover, algorithm $\mathcal{X}$ uses messages of constant size.

We may assume, hereafter, that such an algorithm $\mathcal{X}$, as promised by Claim 3, is part of the language specification given to the nodes, each node then can verify by a *local* computation if the instance restricted to its $\ell(\delta)$ is extendable. We show that $\mathcal{L} \in \mathrm{LD}(O(t))$ by proving the existence of a deterministic algorithm $\mathcal{D}$ that recognizes $\mathcal{L}$ in $O(t)$ rounds. Given a path P, an input $\mathbf{x}$ over P, and an identity assignment Id, algorithm $\mathcal{D}$ applied at a node u of P operates as follows. If $\mathbf{x}_u = \otimes$ then u outputs "yes" if and only if u is an endpoint of P. Otherwise, i.e., if $\mathbf{x}_u \neq \{\otimes\}$, then u outputs "yes" if and only if $(\mathtt{b}_u, \mathbf{x}_{\mathtt{b}_u})$ is extendable (using algorithm $\mathcal{X}$), where $\mathtt{b}_u = \mathtt{b}(u, \ell(\delta))$ is the ball centered at u, and of radius $\ell(\delta)$ in P.

Algorithm $\mathcal{D}$ is a deterministic algorithm that runs in $\ell(\delta)$ rounds. We claim that Algorithm $\mathcal{D}$ recognizes $\mathcal{L}$. To establish that claim, consider first an instance $(P, \mathbf{x}) \in \mathcal{L}$. For every node u, $(P, \mathbf{x}) \in \mathcal{L}$ is an extension of $(\mathtt{b}_u, \mathbf{x}_{\mathtt{b}_u})$. Therefore, every node u outputs "yes", as desired. Now consider an instance $(P, \mathbf{x}) \notin \mathcal{L}$. Assume, for the purpose of contradiction, that there exists an identity assignment Id such that, when applying $\mathcal{D}$ on $(P, \mathbf{x}, \mathrm{Id})$, every node u outputs "yes".

Claim. In this case, $|P| > 2\ell(\delta) + 1$.

Let $S \subseteq P$ be the longest subpath of P such that there exists an extension $(P', \mathbf{x}')$ of $(S, \mathbf{x}_S)$, with $(P', \mathbf{x}') \in \mathcal{L}$. Since $|P| > 2\ell(\delta) + 1$, and since the middle node of P outputs "yes", we have $|S| \geq 2\ell(\delta) + 1$. The proof carries on by distinguishing two cases for the length of S. If $S = P$, then $(P, \mathbf{x})$ can be extended to $(P', \mathbf{x}') \in \mathcal{L}$. By the same arguments as above, since each extremity w of P has input $\otimes$, we conclude that $P = P'$, with $\mathbf{x} = \mathbf{x}'$. Contradicting the fact that $(P, \mathbf{x}) \notin \mathcal{L}$. Therefore $2\ell(\delta) + 1 \leq |S| < |P|$. Let a and b be such that $S = [a, b]$. As S is shorter than P, it is impossible for both a and b to be endpoints of P. Without loss of generality, assume that a is not an endpoint of P. Since a outputs "yes", there exists an extension $(P'', \mathbf{x}'') \in \mathcal{L}$ of $(\mathtt{b}_a, \mathbf{x}_{\mathtt{b}_a})$. In fact, $(P'', \mathbf{x}'')$ is also an extension of $\mathbf{x}_{[a, a+\ell(\delta)]}$. Since $\mathbf{x}'$ and $\mathbf{x}''$ agree on $[a, a + \ell(\delta)]$, and since both $(P', \mathbf{x}')$, and $(P'', \mathbf{x}'')$ are in $\mathcal{L}$, we get from Lemma 3 that $\mathbf{x}_{[a-1, b]}$ can be extended to an input $(P''', \mathbf{x}''') \in \mathcal{L}$, which contradicts the choice of S. The theorem follows.

4 On The Impossibility of Boosting

Theorems 2 and 3 demonstrate that boosting the probability of success might be doable, but only from (p, q) satisfying $p^{1+1/(k+1)} + q > 1$ to (p, q) satisfying $p^{1+1/k} + q > 1$ (with the extremes excluded). In this section, we prove that once the inputs may be restricted in certain ways, the ability to boost the success probability become almost null. More precisely, recall that so far we considered languages as collections of pairs $(G, \mathbf{x})$ where G is a (connected) n-node graph and $\mathbf{x} \in \Sigma^n$ is the input vector to the nodes of G, in some finite of infinite alphabet Σ, that is, $\mathbf{x}(v) \in \Sigma$ for all $v \in V(G)$. An instance of an algorithm $\mathcal{A}$ deciding a language $\mathcal{L}$ was defined as *any* such pair $(G, \mathbf{x})$. We now consider the case where the set of instances is restricted to some specific subset of inputs $\mathcal{I} \subset \Sigma^n$. That is, the distributed algorithm $\mathcal{A}$ has now the *promise* that in the instances $(G, \mathbf{x})$ admissible as inputs, the input vector $\mathbf{x}$ is restricted to $\mathbf{x} \in \mathcal{I} \subset \Sigma^n$.

We define the classes $C_r(t)$ in a way identical to the classes $B_k(t)$, but generalized in two ways. First, the parameter r is not bounded to be integral, but can be any positive real. Second, the decision problems under consideration are extended to the ones in which the set of input vectors $\mathbf{x}$ can be restricted. So, in particular, $B_k(t) \subseteq C_k(t)$, for every positive integer k, and every function t. The following theorem proves that boosting can made as limited as desired.

Theorem 5. *Let $r < r'$ be any two positive reals. Then, $C_{r'}(0) \setminus C_r(t) \neq \emptyset$ for every $t = o(n)$.*

Note that Theorem 5 demonstrates not only the (almost) inability of boosting the probability of success when the inputs to the nodes are restricted to specific kinds, but also the inability of derandomizing, even above the threshold $p^2 + q = 1$. Indeed, the following is a direct consequence of Theorem 5.

Corollary 1. *For every positive real r, there is a decision problem in $C_r(0)$ which cannot be decided* deterministically *in $o(n)$ rounds.*

References

1. Y. Afek, S. Kutten, and M. Yung. The local detection paradigm and its applications to self stabilization. *TCS*, 186:199–230, 1997.
2. N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Alg.*, 7:567–583, 1986.
3. A. Amit, N. Linial, J. Matousek, and E. Rozenman. Random lifts of graphs. *12th SODA*, 2001.
4. B. Awerbuch, B. Patt-Shamir, and G. Varghese. Self-Stabilization By Local Checking and Correction. *FOCS*, 1991.
5. L. Barenboim and M. Elkin. Distributed $(\Delta+1)$-coloring in linear (in delta) time. *41st STOC*, 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. *43rd STOC*, 2011.

7. D Dereniowski and A. Pelc. Drawing maps with advice. *JPDC*,72:132-143, 2012.
8. E.W. Dijkstra. Self-stabilization in spite of distributed control. *Comm. ACM*, 17, 643–644, 1974.
9. S. Dolev, M. Gouda, and M. Schneider. Requirements for silent stabilization. *Acta Informatica*, 36(6), 447-462, 1999.
10. P. Fraigniaud, C. Gavoille, D. Ilcinkas and A. Pelc. Distributed Computing with Advice: Information Sensitivity of Graph Coloring. *34th ICALP*, 2007.
11. P. Fraigniaud, D Ilcinkas, and A. Pelc. Communication algorithms with advice. *JCSS*, 76:222–232, 2008.
12. P. Fraigniaud, A Korman, and E. Lebhar. Local MST computation with short advice. *19th SPAA*, 2007.
13. P. Fraigniaud, A. Korman, and D. Peleg. Local Distributed Decision. *52nd FOCS*, 2011.
14. P. Fraigniaud and A. Pelc. Decidability Classes for Mobile Agents Computing. 10th LATIN, 2012.
15. P. Fraigniaud, S. Rajsbaum, and C. Travers. Locality and Checkability in Wait-free Computing. *25th DISC*, 2011.
16. P. Fraigniaud, S. Rajsbaum, and C. Travers. Universal Distributed Checkers and Orientation-Detection Tasks. Submitted, 2012.
17. M. Göös and J. Suomela. Locally checkable proofs. *30th PODC*, 2011.
18. L. Kor, A. Korman and D. Peleg. Tight Bounds For Distributed MST Verification. *28th STACS*, 2011.
19. A. Korman and S. Kutten. Distributed verification of minimum spanning trees. *Distributed Computing*, 20:253–266, 2007.
20. A. Korman, S. Kutten, and T. Masuzawa. Fast and Compact Self-Stabilizing Verification, Computation, and Fault Detection of an MST. *30th PODC*, 2011.
21. A. Korman, S. Kutten, and D Peleg. Proof labeling schemes. *Distributed Computing*, 22:215–233, 2010.
22. A. Korman, J.S. Sereni, and L. Viennot. Toward More Localized Local Algorithms: Removing Assumptions Concerning Global Knowledge. *30th PODC*, 2011.
23. F. Kuhn. Weak graph colorings: distributed algorithms and applications. *21st SPAA*, 2009.
24. M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15:1036–1053, 1986.
25. M. Naor. A Lower Bound on Probabilistic Algorithms for Distributive Ring Coloring. *SIAM J. Discrete Math.*, 4: 409-412 (1991).
26. M. Naor and L. Stockmeyer. What can be computed locally? *SIAM J. Comput.* 24: 1259-1277 (1995).
27. A. Panconesi and A. Srinivasan. On the Complexity of Distributed Network Decomposition. *J. Alg.* 20: 356-374, (1996).
28. D. Peleg. *Distributed Computing: A Locality-Sensitive Approach.* SIAM, 2000.
29. J. Schneider and R. Wattenhofer. A new technique for distributed symmetry breaking. *29th PODC*, 2010.