To use ID or not to use ID, is that a question?

Pierre Fraigniaud[1]

Laboratoire d'Informatique Algorithmique (LIAFA)
CNRS and University Paris Diderot

Workshop on Advances on Distributed Graph Algorithms, Oct 19, 2012

[1]Joint work with
- Mika Göös (University of Toronto)
- Magnus Halldorsson (Reykjavik University)
- Amos Korman (CNRS and University Paris Diderot)
- Jukka Suomela (University of Helsinki).

Outline

1. **Measuring the impact of IDs on local computation**
2. **Arguments in favor of** $\mathrm{LD} = \mathrm{LD}^*$
3. **Arguments against** $\mathrm{LD} = \mathrm{LD}^*$
4. **Conclusion**

Outline

$\mathcal{LOCAL}$ model

Nodes are labeled by pairwise distinct IDs (i.e., a non-negative integer)

Nodes perform in synchronous rounds.

At each round, a node u of a graph $G = (V, E)$:

1. Sends messages to its neighbors in G;
2. Receives the messages sent by its neighbors;
3. Performs some individual computation.

Construction task

Definition

A language is a decidable collection of pairs (G, x) where

- $G = (V, E)$ is a graph
- $\mathrm{x} = \{\mathrm{x}(u) \in \{0,1\}^*,\ u \in V\}$

Construction tasks for $\mathcal{L}$

Each node u has to compute an output value $\mathrm{x}(u) \in \{0,1\}^*$ such that $(G, \mathrm{x}) \in \mathcal{L}$.

Examples

MIS, dominating set, MST, coloring, leader election, etc.

Challenge: symmetry breaking

Decision task

Decision tasks for $\mathcal{L}$

Each node u gets an input value $x(u) \in \{0,1\}^*$, and all nodes have to collectively decide whether $(G, x) \in \mathcal{L}$.

Application

- Checking the correctness of results produced by a construction algorithm
- Provide a basic framework for a DC complexity theory

Decision task

Decision tasks for $\mathcal{L}$

Each node u gets an input value $x(u) \in \{0,1\}^*$, and all nodes have to collectively decide whether $(G, x) \in \mathcal{L}$.

Application

- Checking the correctness of results produced by a construction algorithm
- Provide a basic framework for a DC complexity theory

Decision rules

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

Remark: symmetry breaking is not much of an issue

$\mathcal{LOCAL}$ model revisited

Equivalence

Any algorithm A running in $t = O(1)$ rounds in the $\mathcal{LOCAL}$ model can be transformed into an algorithm A' in which every node u:

1. Collects the structure of the ball $B(u, t)$ together with all the inputs $\mathrm{x}(v)$ and identities $\mathrm{Id}(v)$ of these nodes
2. Performs some individual computation

$\mathcal{LOCAL}$ model revisited

Equivalence

Any algorithm A running in $t = O(1)$ rounds in the $\mathcal{LOCAL}$ model can be transformed into an algorithm A' in which every node u:

1. Collects the structure of the ball $B(u, t)$ together with all the inputs $\mathrm{x}(v)$ and identities $\mathrm{Id}(v)$ of these nodes
2. Performs some individual computation

Anonymous $\mathcal{LOCAL}$ model

An algorithm A running in $t = O(1)$ rounds in the anonymous $\mathcal{LOCAL}$ model is an algorithm in which every node u:

1. Gets a snapshot of the structure of the ball $B(u, t)$ together with all the inputs $\mathrm{x}(v)$ of the nodes in this ball
2. Performs some individual computation

Local decision classes

Let $t \geq 0$.

$\mathrm{LD}(t)$ is the class of all languages that can be decided in t rounds in the $\mathcal{LOCAL}$ model.

Local decision classes

Let $t \geq 0$.

$\mathrm{LD}(t)$ is the class of all languages that can be decided in t rounds in the $\mathcal{LOCAL}$ model.

$\mathrm{LD}^*(t)$ is the class of all languages that can be decided in t rounds in the anonymous $\mathcal{LOCAL}$ model

Local decision classes

Let $t \geq 0$.

$\mathrm{LD}(t)$ is the class of all languages that can be decided in t rounds in the $\mathcal{LOCAL}$ model.

$\mathrm{LD}^*(t)$ is the class of all languages that can be decided in t rounds in the anonymous $\mathcal{LOCAL}$ model

$$\mathrm{LD} = \bigcup_{t \geq 0} \mathrm{LD}(t) \qquad \mathrm{LD}^* = \bigcup_{t \geq 0} \mathrm{LD}^*(t)$$

LD versus LD^*

By definition, $\mathrm{LD}^* \subseteq \mathrm{LD}$.

Conjecture:

$$\mathrm{LD} = \mathrm{LD}^*$$

LD versus LD^*

By definition, $\mathrm{LD}^* \subseteq \mathrm{LD}$.

Conjecture:

$$\mathrm{LD} = \mathrm{LD}^*$$

Recall that:

1. IDs are arbitrary
2. Each individual algorithm is... computable

Whenever IDs are bounded to be in $\{1, \ldots, n\}$

Whenever IDs are bounded to be in $\{1, \ldots, n\}$

$$\mathcal{L} = \{(G, \mathrm{x}) : G \text{ has at most x nodes}\}$$

Observation

$\mathcal{L} \in \mathrm{LD} \setminus \mathrm{LD}^*$

Algorithm of node v:

If $\mathrm{Id}(v) \leq x$ then output "yes", else output "no".

Proof.

$\mathcal{L} \notin \mathrm{LD}^*$ because nodes cannot locally distinguish C_n from $C_{n'}$

$\mathcal{L} \in \mathrm{LD} \iff n \leq \mathrm{x} \iff \forall i \leq n,$ we have $i \leq \mathrm{x}$ □

Whenever the local “function” is not computable

Whenever the local “function” is not computable

Observation

$\mathrm{LD} = \mathrm{LD}^*$

Proof.

Let A be a LD algorithm for $\mathcal{L}$.

LD^* algorithm at node u:

Return “no” if and only if

$\exists$ ID-assignment to the nodes of $B(u, t)$

for which A returns “no” at u. □

Objective of the talk

Discuss the issue: LD **versus** LD^*

Outline

Hereditary languages

Definition

A hereditary language is a language closed under node deletion.

Examples: k-`Coloring`, Independent set, Planar graphs, Interval graphs, Forests, Chordal graphs, Cographs, Perfect graphs, etc.

Observation

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

Proof

(p, q)-decider

- if $(G, \mathrm{x}) \in \mathcal{L}$, then, with probability $\geq p$, all nodes output "yes";
- if $(G, \mathrm{x}) \notin \mathcal{L}$, then, with probability $\geq q$, some node(s) outputs "no".

Proof

(p, q)-decider

- if $(G, x) \in \mathcal{L}$, then, with probability $\geq p$, all nodes output "yes";
- if $(G, x) \notin \mathcal{L}$, then, with probability $\geq q$, some node(s) outputs "no".

Theorem (F., Korman, Peleg [FOCS 2011])

In the $\mathcal{LOCAL}$ model, if $\mathcal{L}$ is hereditary, and there exists a (p, q)-decider A for $\mathcal{L}$ with $p^2 + q > 1$, running in t rounds, then there exists a deterministic algorithm D for $\mathcal{L}$ running in $O(t)$ rounds.

Proof

(p, q)-decider

- if $(G, x) \in \mathcal{L}$, then, with probability $\geq p$, all nodes output “yes”;
- if $(G, x) \notin \mathcal{L}$, then, with probability $\geq q$, some node(s) outputs “no”.

Theorem (F., Korman, Peleg [FOCS 2011])

In the $\mathcal{LOCAL}$ model, if $\mathcal{L}$ is hereditary, and there exists a (p, q)-decider A for $\mathcal{L}$ with $p^2 + q > 1$, running in t rounds, then there exists a deterministic algorithm D for $\mathcal{L}$ running in $O(t)$ rounds.

Proof.

- A LD algorithm A deciding $\mathcal{L}$ is a $(1, 1)$-decider for $\mathcal{L}$.
- The algorithm D is in fact anonymous.

□

Bounded-degree and bounded-input instances

As a consequence of [F., Korman, Parter, and Peleg, DISC 2012]:

Observation

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

Bounded-degree and bounded-input instances

As a consequence of [F., Korman, Parter, and Peleg, DISC 2012]:

Observation

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

Observation

$\mathrm{LD} = \mathrm{LD}^*$ *for languages defined on bounded degree graphs, with a finite set of input values.*

Proof.

There are finitely many different balls for instances (G, x) with

- $\deg(G) \leq \Delta$
- $|\mathrm{x}(u)| \leq k$ for every node u

□

Oracles

Oracles

Oracle N

For every node u of an n-node graph, $n \leq \mathbf{N}(u)$.

We denote by $\mathrm{LD}^{*\mathbf{N}}$ the class of languages that can be decided by a LD^* algorithm having access to oracle **N**.

Oracles

Oracle N

For every node u of an n-node graph, $n \leq \mathbf{N}(u)$.

We denote by $\mathrm{LD}^{*\mathbf{N}}$ the class of languages that can be decided by a LD^* algorithm having access to oracle **N**.

Observation

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

Proof.

Let A be a LD algorithm deciding $\mathcal{L}$ in t rounds.

$\mathrm{LD}^{*\mathbf{N}}$ algorithm at node u:

Return “no” if and only if there exists an ID-assignment to the nodes of $B(u,t)$ from the range $[1, \mathbf{N}(u)]$ for which A returns “no” at u. □

Local verification class

$$\text{Certificate } y = \{y(u) \in \{0,1\}^*, u \in V\}.$$

Verification rules

- if $(G, \mathrm{x}) \in \mathcal{L}$, then $\exists$ certificate y : every node outputs “yes”;
- if $(G, \mathrm{x}) \notin \mathcal{L}$, then $\forall$ certificate y : at least one node outputs “no”.

Local verification class

$$\text{Certificate } y = \{y(u) \in \{0,1\}^*, u \in V\}.$$

Verification rules

- if $(G, \mathrm{x}) \in \mathcal{L}$, then $\exists$ certificate y : every node outputs “yes”;
- if $(G, \mathrm{x}) \notin \mathcal{L}$, then $\forall$ certificate y : at least one node outputs “no”.

Applications

- Checking the correctness of data structures (e.g., proof-labeling schemes)
- Non-deterministic version of LD (and LD^*)

Local verification class

$$\text{Certificate } y = \{y(u) \in \{0,1\}^*, u \in V\}.$$

Verification rules

- if $(G, \mathrm{x}) \in \mathcal{L}$, then $\exists$ certificate y : every node outputs "yes";
- if $(G, \mathrm{x}) \notin \mathcal{L}$, then $\forall$ certificate y : at least one node outputs "no".

Applications

- Checking the correctness of data structures (e.g., proof-labeling schemes)
- Non-deterministic version of LD (and LD^*)

$\mathrm{NLD}(t)$ (resp., $\mathrm{NLD}^*(t)$) is the class of all languages that can be verified in t rounds in the $\mathcal{LOCAL}$ (resp., anonymous $\mathcal{LOCAL}$) model.

$$\mathrm{NLD} = \bigcup_{t \geq 0} \mathrm{NLD}(t) \qquad \mathrm{NLD}^* = \bigcup_{t \geq 0} \mathrm{NLD}^*(t)$$

Conjecture holds non-deterministically

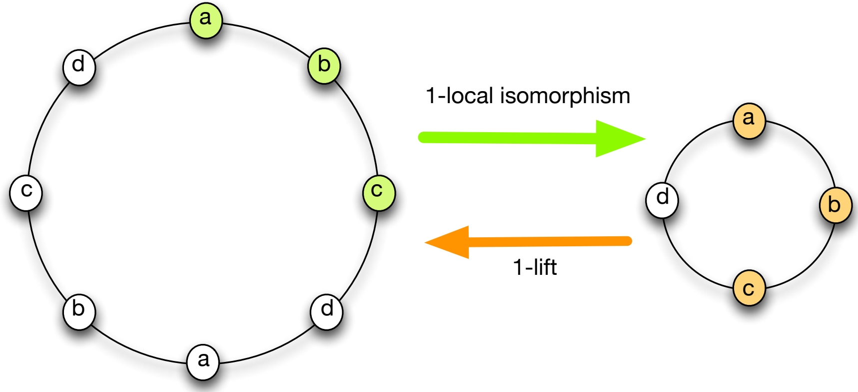
Theorem

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

Conjecture holds non-deterministically

Theorem

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



Proof.

- $\mathcal{L}$ is *t-closed under lift* if, for every two instances I, I' such that I is t-local isomorphic to I', we have:

$$I' \in \mathcal{L} \Rightarrow I \in \mathcal{L}$$

- If there exists $t \geq 1$ such that $\mathcal{L}$ is t-closed under lift, then $\mathcal{L} \in \mathrm{NLD}^*$.
- If $\mathcal{L} \in \mathrm{NLD}$, then there exists $t \geq 1$ such that $\mathcal{L}$ is t-closed under lift.

□

Completeness under anonymous reduction

Definition

$\mathcal{L}_1$ is locally reducible to $\mathcal{L}_2$ if there exists an algorithm $\mathcal{A}$ running in $t = O(1)$ rounds such that, for every instance (G, x), $\mathcal{A}$ produces $\mathrm{out}(u) \in \{0,1\}^*$ at every node $u \in V(G)$, satisfying:

$$(G, \mathrm{x}) \in \mathcal{L}_1 \iff (G, \mathrm{out}) \in \mathcal{L}_2 \ .$$

Completeness under anonymous reduction

Definition

$\mathcal{L}_1$ is locally reducible to $\mathcal{L}_2$ if there exists an algorithm $\mathcal{A}$ running in $t = O(1)$ rounds such that, for every instance (G, x), $\mathcal{A}$ produces $\mathrm{out}(u) \in \{0,1\}^*$ at every node $u \in V(G)$, satisfying:

$$(G, \mathrm{x}) \in \mathcal{L}_1 \iff (G, \mathrm{out}) \in \mathcal{L}_2 \; .$$

$\mathrm{x}(u) = (\mathcal{E}(u), \mathcal{S}(u))$

- $\mathcal{E}(u)$ is an element (say an integer $\mathcal{E}(u) \in \mathbb{N}$)
- $\mathcal{S}(u)$ is a finite collection of sets (say, of subsets of $\mathbb{N}$)

$\mathcal{L}^* = \{(G, (\mathcal{E}, \mathcal{S})) \mid \exists v \in V(G),\ \exists S \in \mathcal{S}(v) \text{ s.t. } S \supseteq \{\mathcal{E}(u) \mid u \in V(G)\}\}.$

Completeness under anonymous reduction

Definition

$\mathcal{L}_1$ is locally reducible to $\mathcal{L}_2$ if there exists an algorithm $\mathcal{A}$ running in $t = O(1)$ rounds such that, for every instance (G, x), $\mathcal{A}$ produces $\mathrm{out}(u) \in \{0,1\}^*$ at every node $u \in V(G)$, satisfying:

$$(G, \mathrm{x}) \in \mathcal{L}_1 \iff (G, \mathrm{out}) \in \mathcal{L}_2 \ .$$

$\mathrm{x}(u) = (\mathcal{E}(u), \mathcal{S}(u))$

- $\mathcal{E}(u)$ is an element (say an integer $\mathcal{E}(u) \in \mathbb{N}$)
- $\mathcal{S}(u)$ is a finite collection of sets (say, of subsets of $\mathbb{N}$)

$\mathcal{L}^* = \{(G, (\mathcal{E}, \mathcal{S})) \mid \exists v \in V(G),\ \exists S \in \mathcal{S}(v) \text{ s.t. } S \supseteq \{\mathcal{E}(u) \mid u \in V(G)\}\}.$

Theorem (F., Korman, Peleg [FOCS 2011])

$\mathcal{L}^*$ *is* NLD*-complete (for non anonymous local reductions).*

Essence of the proof (NLD-hardness)

Let (G, x) be an instance for $\mathcal{L} \in \mathrm{NLD}$, and let Id be an ID-assignment.

- $\mathcal{E}(v) = B_G(v, t)$, together with inputs and IDs,
- Let $\mathrm{width}(v) = 2^{|\mathrm{Id}(v)| + |\mathrm{x}(v)|}$.
- Node v first generates all instances $(G', \mathrm{x}') \in \mathcal{L}$ where
  - G' is a graph with $k \leq \mathrm{width}(v)$ vertices,
  - x' is a collection of k input strings of length at most $\mathrm{width}(v)$,
- For each (G', x'), node v generates all possible ID-assignments Id' to $V(G')$ such that $\forall u \in V(G')$, $|\mathrm{Id}'(u)| \leq \mathrm{width}(v)$.
- $S = \{B_{G'}(u, t)$, for every node u of $(G', \mathrm{x}')\} \in \mathcal{S}(v)$.

Claim

$(G, \mathrm{x}) \in \mathcal{L} \iff (G, \mathrm{out}) \in \mathcal{L}^*$.

Outline

Languages with promise

Instances are of the form (G, M) where

- G is an n-node graph
- M is a Turing machine (the same for all nodes).

The promise:

$\{(G, M) : M$ does not stop, or it stops in at most n steps$\}$.

$$\begin{cases} \mathcal{L}_{yes} = \{(G, M) : M \text{ does not stop}\} \\ \mathcal{L}_{no} = \{(G, M) : M \text{ stops in at most } n \text{ steps}\}. \end{cases}$$

Languages with promise

Instances are of the form (G, M) where

- G is an n-node graph
- M is a Turing machine (the same for all nodes).

The promise:

$\{(G, M) : M$ does not stop, or it stops in at most n steps$\}$.

$$\begin{cases} \mathcal{L}_{yes} & = & \{(G,M) : M \text{ does not stop}\} \\ \mathcal{L}_{no} & = & \{(G,M) : M \text{ stops in at most } n \text{ steps}\}. \end{cases}$$

Observation

$\mathcal{L} \in \mathrm{LD} \setminus \mathrm{LD}^*$

Algorithm of node v:

If M does not stop in Id(v) steps
then output “yes”, else output “no”.

Bounded IDs: Id(v) $\in \{1, \ldots, n^c\}$

$$p\text{-counter:} \quad C(p) = \begin{matrix} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{matrix}$$

p copies of a $C(p)$ vs. 1 copy of a $C(p^2)$ for prime p

000000000		000000000
001001001		000000001
010010010		000000010
011011011		000000011
100100100	versus	000000100
101101101		000000101
110110110		000000110
111111111		000000111
		000001000
		⋮ ⋮ ⋮
		111111100
		111111101
		111111110
		111111111

Separation (rough idea)

$$\mathcal{L} = \{(G, p) : G = p \times C(p).\}$$

$$\mathcal{L}^* = \{(G, p) : G = p \times C(p), \text{ or } G = C(p^2).\}$$

Separation (rough idea)

$$\mathcal{L} = \{(G, p) : G = p \times C(p).\}$$

$$\mathcal{L}^* = \{(G, p) : G = p \times C(p), \text{ or } G = C(p^2).\}$$

One can show that $\mathcal{L}^* \in \mathrm{LD}^*$

Separation (rough idea)

$$\mathcal{L} = \{(G, p) : G = p \times C(p).\}$$

$$\mathcal{L}^* = \{(G, p) : G = p \times C(p), \text{ or } G = C(p^2).\}$$

One can show that $\mathcal{L}^* \in \mathrm{LD}^*$

But one cannot distinguish $p \times C(p)$ from $C(p^2)$ in LD^*

Separation (rough idea)

$$\mathcal{L} = \{(G, p) : G = p \times C(p).\}$$

$$\mathcal{L}^* = \{(G, p) : G = p \times C(p), \text{ or } G = C(p^2).\}$$

One can show that $\mathcal{L}^* \in \mathrm{LD}^*$

But one cannot distinguish $p \times C(p)$ from $C(p^2)$ in LD^*

Observation

If IDs are in $\{1, \ldots, n^c\}$, then $p \times C(p)$ versus $C(p^2)$ is in LD.

Outline

Open problems

- $\mathrm{LD} = \mathrm{LD}^*$?

Open problems

- $\mathrm{LD} = \mathrm{LD}^*$?
- Is $\bigcup_{p^2+q>1} \mathrm{BPLD}(p, q) = \mathrm{LD}$ for non hereditary languages?

Open problems

- $\mathrm{LD} = \mathrm{LD}^*$?
- Is $\bigcup_{p^2+q>1} \mathrm{BPLD}(p, q) = \mathrm{LD}$ for non hereditary languages?
- Is there a NLD^*-complete problem (for anonymous local reductions)?

Open problems

- $\mathrm{LD} = \mathrm{LD}^*$?
- Is $\bigcup_{p^2+q>1} \mathrm{BPLD}(p, q) = \mathrm{LD}$ for non hereditary languages?
- Is there a NLD^*-complete problem (for anonymous local reductions)?

Thank you!