Local Distributed Verification

A. Balliu, G. D’Angelo, P. Fraigniaud, and D. Olivetti

CNRS and University Paris Diderot
GSSI L’Aquila
Classify problems according to their difficulty, i.e., build a complexity theory in the distributed setting.

Build a hierarchy of complexity classes in the context of the LOCAL model.
The distributed network is represented by a graph.
Local Model

- The distributed network is represented by a graph.
- Synchronous model.
Local Model

- The distributed network is represented by a graph.
- Synchronous model.
Local Model

- The distributed network is represented by a graph.
- Synchronous model.
The distributed network is represented by a graph.

Synchronous model.

Equivalent to a model where each node sees the network up to distance \( t \).
Local Model

- The distributed network is represented by a graph.
- Synchronous model.
- Equivalent to a model where each node sees the network up to distance $t$.
- The time complexity of a local algorithm $A$ is determined by the range $t$ that it needs to explore.
Local Model

- The distributed network is represented by a graph.
- Synchronous model.
- Equivalent to a model where each node sees the network up to distance $t$.
- The time complexity of a local algorithm $\mathcal{A}$ is determined by the range $t$ that it needs to explore.
- We want $t$ to be constant.
Decision Problems

- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.
Decision Problems

- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.
- Each node:
  - gathers its local information from the network;
Decision Problems

- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.
- Each node:
  - gathers its local information from the network;
  - perform some local computation;
Decision Problems

- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.

- Each node:
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision:
Decision Problems

- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.
- Each node:
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision: "accept"
Decision Problems

- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.

- Each node:
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision: "accept" or "reject".
Decision Problems

- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.

- Each node:
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision: "accept" or "reject".

\[ \text{global\_output} = \bigwedge_{v \in V} \text{local\_output}(v). \]
Decision Problems

- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.

- Each node:
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision: "accept" or "reject".

- \( \text{global\_output} = \bigwedge_{v \in V} \text{local\_output}(v). \)
Decision Problems

- Decision Problems: the aim is to decide whether a global input instance satisfies some specific property.

- Each node:
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision: "accept" or "reject".

\[ \text{global\_output} = \bigwedge_{v \in V} \text{local\_output}(v). \]
Example: Proper Coloring

- Node input: a color.
- Each node checks the colors of its neighbors.
Example: Proper Coloring

- Node input: a color.
- Each node checks the colors of its neighbors.

Local Decision (LD) is the class of distributed languages that can be locally decided [NS ’95].
LD Class

LD is the class of all distributed languages $\mathcal{L}$ for which there exists a local algorithm $A$ satisfying the following: for every input instance $(G, x)$,

$$(G, x) \in \mathcal{L} \Rightarrow \forall id \in ID(G), \forall u \in V(G), A(G, x, id, u) = \text{accept}$$

$$(G, x) \notin \mathcal{L} \Rightarrow \forall id \in ID(G), \exists u \in V(G), A(G, x, id, u) = \text{reject}$$
Verification Problems

- Verification problem: the aim is to **verify** whether a global input instance satisfies some specific property.
Verification Problems

- Verification problem: the aim is to verify whether a global input instance satisfies some specific property.
- Each node:
  - has a certificate, unbounded size and independent from the id assignment;
Verification Problems

- Verification problem: the aim is to **verify** whether a global input instance satisfies some specific property.
- Each node:
  - **has a certificate**, unbounded size and independent from the id assignment;
  - gathers its local information from the network;
Verification Problems

- Verification problem: the aim is to **verify** whether a global input instance satisfies some specific property.
- Each node:
  - **has a certificate**, unbounded size and independent from the id assignment;
  - gathers its local information from the network;
  - perform some local computation;

\[ \text{global}_{-output} = \bigwedge_{v \in V} \text{local}_{-output}(v) \]
Verification Problems

- Verification problem: the aim is to **verify** whether a global input instance satisfies some specific property.

- Each node:
  - **has a certificate**, unbounded size and independent from the id assignment;
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision, that is ether "accept" or "reject".
Verification Problems

- Verification problem: the aim is to **verify** whether a global input instance satisfies some specific property.
- Each node:
  - **has a certificate**, unbounded size and independent from the id assignment;
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision, that is either "accept" or "reject".
- $\text{global\_output} = \bigwedge_{v \in V} \text{local\_output}(v)$. 

Similar to PLS, but with id-independent certificates.
Verification Problems

- Verification problem: the aim is to **verify** whether a global input instance satisfies some specific property.
- Each node:
  - **has a certificate**, unbounded size and independent from the id assignment;
  - gathers its local information from the network;
  - perform some local computation;
  - output its local decision, that is either ”accept” or ”reject”.

\[
\text{global}\_\text{output} = \bigwedge_{v \in V} \text{local}\_\text{output}(v).
\]

- Similar to PLS, but with id-independent certificates.
Example: is the given graph a tree?

- Not locally decidable, but locally verifiable.
Example: is the given graph a tree?

- Not locally decidable, but locally verifiable.
- Choose a node to be the root.
Example: is the given graph a tree?

- Not locally decidable, but locally verifiable.
- Choose a node to be the root.
- Certificate of a node $v$: its hop-distance from the chosen root.
Example: is the given graph a tree?

- Not locally decidable, but locally verifiable.
- Choose a node to be the root.
- Certificate of a node $v$: its hop-distance from the chosen root.
Example: is the given graph a tree?

- Not locally decidable, but locally verifiable.
- Choose a node to be the root.
- Certificate of a node $v$: its hop-distance from the chosen root.

\[ \begin{array}{cccc}
0 & 1 & 1 & 2 \\
2 & 3 & 3 & 3 \\
3 & 4 & & \\
\end{array} \]

- *Nondeterministic LD (NLD)* is the class of distributed languages that can be locally verified [FKP ’11].
NLD Class

NLD is the class of all distributed languages $\mathcal{L}$ for which there exists a local algorithm $A$ satisfying the following: for every input instance $(G, x)$,

1. $(G, x) \in \mathcal{L} \Rightarrow \exists c \in C(G), \forall \text{id} \in \text{ID}(G), \forall u \in V(G), \quad A(G, x, c, \text{id}, u) = \text{accepts}$

2. $(G, x) \notin \mathcal{L} \Rightarrow \forall c \in C(G), \forall \text{id} \in \text{ID}(G), \exists u \in V(G), \quad A(G, x, c, \text{id}, u) = \text{rejects}$
NLD is the class of all distributed languages \( \mathcal{L} \) for which there exists a local algorithm \( A \) satisfying the following: for every input instance \((G, x)\),

- \((G, x) \in \mathcal{L} \Rightarrow \exists c \in \mathcal{C}(G), \forall \text{id} \in \text{ID}(G), \forall u \in \text{V}(G),\ A(G, x, c, \text{id}, u) = \text{accepts}\)
- \((G, x) \notin \mathcal{L} \Rightarrow \forall c \in \mathcal{C}(G), \forall \text{id} \in \text{ID}(G), \exists u \in \text{V}(G),\ A(G, x, c, \text{id}, u) = \text{rejects}\)

\( L \in \text{NP} \) if there is a polynomial time algorithm \( A \) such that,

\[ x \in L \iff \exists c \text{ s.t. } A \text{ accepts } x \text{ with } c. \]
More about NLD

NLD is the class of all problems closed under lift [FKP ’11].

- Let \((G, x)\) and \((G', x')\) be two input instances.
- \((G', x')\) is a lift of \((G, x)\) if there exists a function \(f\) such that:
  \[ f : V(G') \rightarrow V(G) \]
  preserving the local view of each node.
Let $\mathcal{L}$ be a language in NLD.

If $(G, x) \in \mathcal{L} \land (G', x')$ is a lift of $(G, x)$, then $(G', x') \in \mathcal{L}$. 

\[ G \quad G' \]

\[
\begin{array}{c}
C_1 \\
C_3 \\
C_2
\end{array}
\]

\[
\begin{array}{c}
C_1 \\
C_3 \\
C_2
\end{array}
\]
Goal

- Build a hierarchy of complexity classes in the distributed setting.
- Distributed hierarchies in other setting:
  - [Reiter ’14] in the context of automata;
  - [FFH ’16] in a model inspired by the CONGEST one.
Complexity Classes

- LD = $\Sigma^0_{loc} = \Pi^0_{loc}$ (similar to P in the sequential setting).
Complexity Classes

- LD = $\Sigma_{0}^{loc} = \Pi_{0}^{loc}$ (similar to P in the sequential setting).
- NLD = $\Sigma_{1}^{loc}$ (similar to NP in the sequential setting).
Complexity Classes

- \( \text{LD} = \Sigma^\text{loc}_0 = \Pi^\text{loc}_0 \) (similar to P in the sequential setting).
- \( \text{NLD} = \Sigma^\text{loc}_1 \) (similar to NP in the sequential setting).
- \( \Sigma^\text{loc}_k \): An input instance satisfies a certain property in \( \Sigma^\text{loc}_k \) iff
  \[ \exists c_1, \forall c_2, \ldots, Qc_k, \text{ all nodes accept.} \]
Complexity Classes

- LD = $\Sigma^l_{0} = \Pi^l_{0}$ (similar to P in the sequential setting).
- NLD = $\Sigma^l_{1}$ (similar to NP in the sequential setting).
- $\Sigma^l_k$: An input instance satisfies a certain property in $\Sigma^l_k$ iff
  $$\exists c_1, \forall c_2, \ldots, Qc_k, \text{ all nodes accept}.$$
- $\Pi^l_k$: An input instance satisfies a certain property in $\Pi^l_k$ iff
  $$\forall c_1, \exists c_2, \ldots, Qc_k, \text{ all nodes accept}.$$
Complementary Classes

In a class:

A globally accepted input instance.  A globally rejected input instance.

In a complementary class:

A globally accepted input instance.  A globally rejected input instance.
Lever 0 of the Hierarchy

- **AND**: $|\{u \in V(G) : x(u) = 1\}| = 0$
- **OR**: $|\{u \in V(G) : x(u) = 1\}| \geq 1$
\( \Pi^1_{loc} \): The Role of the Last Universal Quantifier

- \( \Pi^1_{loc} \):
  \( (G, x) \in \mathcal{L} \iff \forall c \text{ all nodes accept} \).

- LD:
  \( (G, x) \in \mathcal{L} \iff \text{all nodes accept} \)
$\Pi_{1}^{loc}$: The Role of the Last Universal Quantifier

- $\Pi_{1}^{loc}$:
  $$(G, x) \in L \iff \forall c \text{ all nodes accept}.$$ 

- LD:
  $$(G, x) \in L \iff \text{all nodes accept}$$

- Problems that can be solved only if a specific node knows (an upper bound of) the size of the network!
Let $f$ be a function and $a$ and $b$ two non-negative integers.
Let $f$ be a function and $a$ and $b$ two non-negative integers.

A configuration in ITER consists in a path $P = L v R$ with a special node $v$ (pivot).
Let $f$ be a function and $a$ and $b$ two non-negative integers.

- A configuration in ITER consists in a path $P = LvR$ with a special node $v$ (pivot).
- Nodes in $L$ (resp., in $R$) are given as input $f, f^i(a)$ (resp., $f, f^i(b)$); to $v$ is given in input $f, a, b$. 
Let $f$ be a function and $a$ and $b$ two non-negative integers.

- A configuration in ITER consists in a path $P = LvR$ with a special node $v$ (pivot).
- Nodes in $L$ (resp., in $R$) are given as input $f, f^i(a)$ (resp., $f, f^i(b)$); to $v$ is given in input $f, a, b$.
- $f$ is s.t. $f(0) = 0$ and $f(1) = 1$
Let $f$ be a function and $a$ and $b$ two non-negative integers.

A configuration in ITER consists in a path $P = LvR$ with a special node $v$ (pivot).

Nodes in $L$ (resp., in $R$) are given as input $f, f^i(a)$ (resp., $f, f^i(b)$); to $v$ is given in input $f, a, b$.

$f$ is s.t. $f(0) = 0$ and $f(1) = 1$

An input instance is in ITER if and only if:
- $f^{|L|}(a) \in \{0, 1\}$ and $f^{|R|}(b) \in \{0, 1\}$
- $f^{|L|}(a) = 0$ or $f^{|R|}(b) = 0$
An endpoint node rejects only if it has in input something different from 1 or 0; otherwise accepts.

In this case, the left endpoint node rejects.
Nodes reject if they notice local inconsistencies.
\( (G, x) \notin \mathcal{L} \Rightarrow \exists c \text { s.t. at least one node rejects.} \)

\( v \) rejects only if \( f|_L(a) = f|_R(b) = 1 \); otherwise accepts.

Certificate of node \( v \): un upper bound of the size of the network.
$(G, x) \in \mathcal{L} \Rightarrow \forall c \text{ s.t. all nodes accept.}$

Whatever certificate $v$ has, it will never compute $f^{|L|}(a) = f^{|R|}(b) = 1$. 
Local Hierarchy

\[ \Pi_1^{\text{loc}} \]

\[ \text{ITER} \]

\[ \text{LD} \]

\[ \Pi_1^{\text{co-loc}} \]

\[ \text{ITER} \]

\[ \text{co-LD} \]

\[ \text{AND} \]

\[ \text{DIAM}_k \]

\[ \text{OR} \]
Local Hierarchy

\[ NLD = \Sigma_{2}^{\text{loc}} \]

\[ \text{co-NLD} \]

\[ \Pi_{1}^{\text{loc}} \]

\[ \text{co-} \Pi_{1}^{\text{loc}} \]

\[ \text{MISS} \]

\[ \text{ALTS} \]

\[ \text{TREE} \]

\[ \text{ITER} \]

\[ \text{LD} \]

\[ \text{co-LD} \]

\[ \text{AND} \]

\[ \text{DIA}_k \]

\[ \text{OR} \]

\[ \text{AMOS} \]
\( \Pi_2^{loc} \) Class

- \( \Pi_2 \) class: An input instance satisfies a certain property in \( \Pi_2 \) iff
  \[ \forall c_1, \exists c_2, \text{ all nodes accept.} \]

- Two party game between a *disprover* and a *prover*. 
Exactly Two Selected
Exactly Two Selected
Local Hierarchy

\[
LD \subset \Pi_1^{\text{loc}} \subset NLD = \Sigma_2^{\text{loc}} \subset \Pi_2^{\text{loc}} = \text{All} \quad (\text{all inclusions are strict}).
\]
**MISS: a $\Pi^2_{loc}$-complete Problem**

- Every node $u$ of $(G, x)$ is given a family $\mathcal{F}(u)$ of input instances, each described by
  - An adjacency matrix representing a graph;
  - array representing the inputs to the nodes of that graph.
MISS: a $\Pi^2_{loc}$-complete Problem

- Every node $u$ of $(G, x)$ is given a family $F(u)$ of input instances, each described by
  - An adjacency matrix representing a graph;
  - array representing the inputs to the nodes of that graph.
- Every node $u$ has an input string $x'(u) \in \{0, 1\}^*$ (notice that $(G, x')$ is also an input instance).
**MISS: a $\Pi_{2}^{loc}$-complete Problem**

- Every node $u$ of $(G, x)$ is given a family $\mathcal{F}(u)$ of input instances, each described by
  - An adjacency matrix representing a graph;
  - Array representing the inputs to the nodes of that graph.

- Every node $u$ has an input string $x'(u) \in \{0, 1\}^*$ (notice that $(G, x')$ is also an input instance).

- The current $(G, x)$ is legal if $(G, x')$ is missing in all families $\mathcal{F}(u)$ for every $u \in V(G)$.

$$\text{MISS} = \{(G, x) : \forall u \in V(G), x(u) = (\mathcal{F}(u), x'(u)) \text{ and } (G, x') \notin \mathcal{F}\}$$
MISS: a $\Pi^2_{loc}$-complete Problem
Each node $u$ with identity $\text{id}(u)$ and input $\text{x}(u)$ computes its width $\omega(u) = 2|\text{id}(u)| + |\text{x}(u)|$. 
Reduction to \textsc{miss}

- Each node $u$ with identity $\text{id}(u)$ and input $x(u)$ computes its \textit{width} $\omega(u) = 2|\text{id}(u)| + |x(u)|$.

- Each node $u$ generates $\mathcal{F}(u)$, i.e., all $(H, y) \notin \mathcal{L}$
  - At most $\omega(u)$ nodes;
  - $y(v)$ has value at most $\omega(u)$. 
Each node $u$ with identity $\text{id}(u)$ and input $x(u)$ computes its width $\omega(u) = 2|\text{id}(u)| + |x(u)|$.

Each node $u$ generates $\mathcal{F}(u)$, i.e., all $(H, y) \notin \mathcal{L}$
- At most $\omega(u)$ nodes;
- $y(v)$ has value at most $\omega(u)$.

If $(G, x) \in \mathcal{L}$
- $(G, x) \notin \mathcal{F}$ since only illegal instances are in $\mathcal{F}$;
- all nodes will accept.
Reduction to \textsc{miss}

- Each node $u$ with identity $\text{id}(u)$ and input $x(u)$ computes its \textit{width} $\omega(u) = 2|\text{id}(u)| + |x(u)|$.
- Each node $u$ generates $\mathcal{F}(u)$, i.e., all $(H, y) \notin \mathcal{L}$
  - At most $\omega(u)$ nodes;
  - $y(v)$ has value at most $\omega(u)$.
- If $(G, x) \in \mathcal{L}$
  - $(G, x) \notin \mathcal{F}$ since only illegal instances are in $\mathcal{F}$;
  - all nodes will accept.
- If $(G, x) \notin \mathcal{L}$
  - There exists $u$ with $\text{id}(u)$ or $x(u)$ big enough, which guarantees that $u$ generates the graph $G$, i.e., $(G, x) \in \mathcal{F}(u)$;
  - at least one node will reject.
Open Problems

- **Unbounded size id-independent certificates:**
  - find a complete problem for $\Pi_1^{\text{loc}}$ and $\text{co-}\Pi_1^{\text{loc}}$;
  - find a problem in the intersection between the classes $\Pi_1^{\text{loc}}$ and $\text{co-}\Pi_1^{\text{loc}}$.

- **Bounded size ($O(\log n)$) id-dependent certificates**
  - we don’t know if the hierarchy collapses;
  - there are no separating problems for $\Sigma_2^{\text{loc}}$ and $\Sigma_3^{\text{loc}}$ (neither for classes higher in the hierarchy).
Thank you!