Local Distributed Verification

Alkida Balliu

CNRS and University Paris Diderot

GSSI L'Aquila

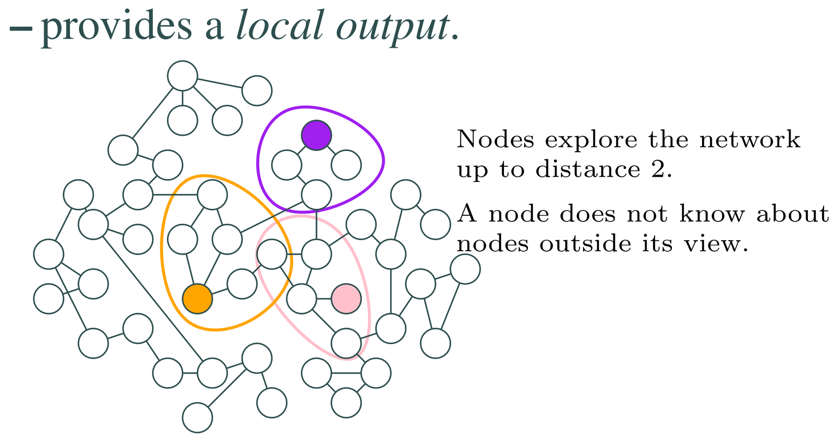
(This is a joint work with G. D'Angelo, P. Fraigniaud, and D. Olivetti)

Goal and Motivation

- 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.

LOCAL Model

- In the LOCAL model ([2]), each node:
  - has a unique identifier;
  - has a *local input*;
  - has a local view (up to constant distance t);
  - provides a *local output*.

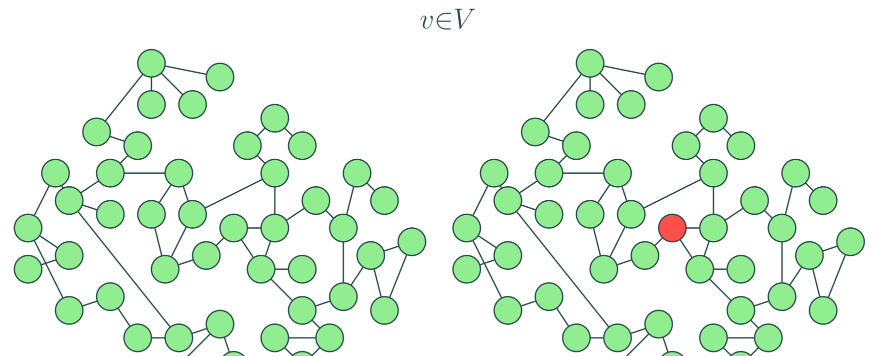


- The time complexity of a local algorithm $\mathcal{A}$ is determined by the range t that it needs to explore.

Decision Problems

- **Objective**: *decide* whether a global input instance satisfies some specific property.
- **Input instance**: we consider as an input instance the pair (G, x), where x is a function that assigns to each node v the local input $x(v)$.
- **Distributed language**: all the input instances that satisfy a specific property.
- **Local decision**: to decide whether an input instance (G, x) satisfies a given property, each node v gathers its local information from its local view and outputs its local decision:
  - "accept" if (G, x) satisfies the desired property;
  - "reject" if (G, x) does not satisfy the property.

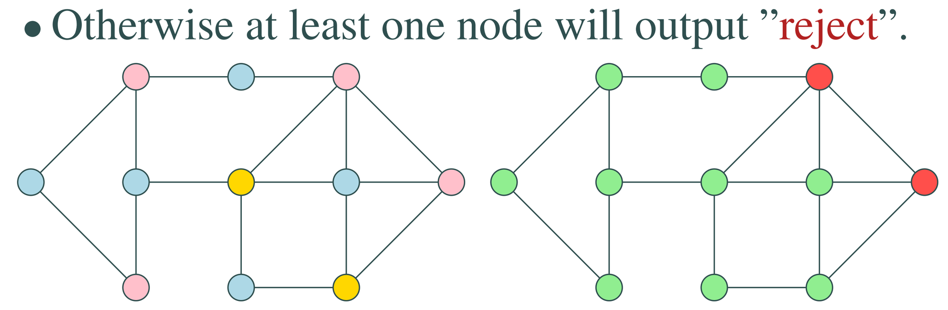
$$global\_output = \bigwedge_{v \in V} local\_output(v).$$



A globaly accepted input instance. A globaly rejected imput instance.

Is the Graph Properly Colored?

- If yes, all nodes will locally accept.
- Otherwise at least one node will output "reject".



A non properly colored input instance. Output: two nods reject.

- *Local Decision (LD)* [3] is the class of distributed languages that are locally decidable.

> LD is the class of all distributed languages $\mathcal{L}$ for which there exists a local algorithm $\mathcal{A}$ such that
>
> $$(G, x) \in \mathcal{L} \iff \mathcal{A} \text{ accepts } (G, x).$$

A Relation with the Polynomial Hierarchy.

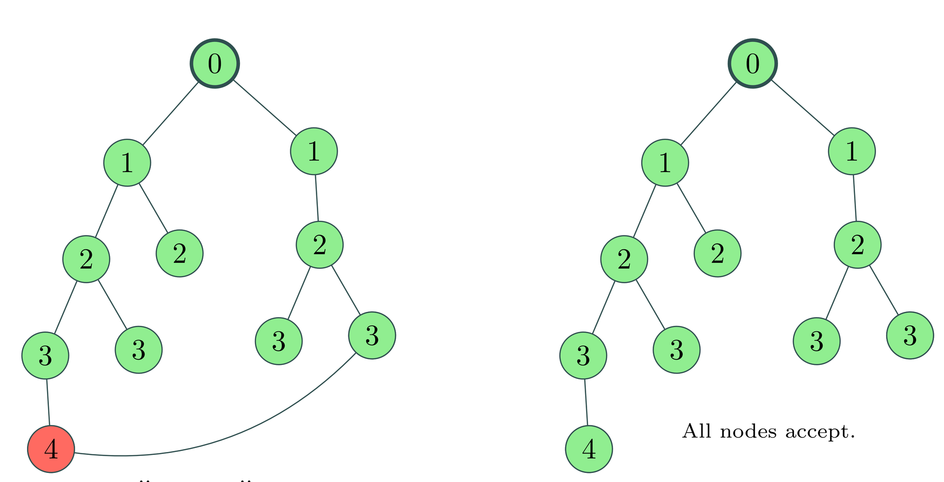
> $L \in$ P if there is a polynomial time algorithm A such that,
>
> $$x \in L \iff A \text{ accepts } x.$$

Verification Problems

- **Objective**: *verify* whether a global input instance satisfies some specific property.
- **Certificate**: information given by a third party that we don't trust a priori. To preserve privacy, each certificate is *independent from the id assignment*.

Is the Graph a Tree?

- *Task*: is the given graph a tree?
- *Local decidability*: not locally decidable since it requires some global knowledge of the network.
- *Local Verifiability*: locally verifiable as follows:
  1. choose a random node in the graph and mark it as "root";
  2. assign to each node v a certificate, that is its hop-distance from the chosen root r, denoted as $d(v, r)$;
  3. each node v accepts if it has *exactly* one neighbor with distance $d(v, r) - 1$ and all the others with distance $d(v, r) + 1$, otherwise it rejects.



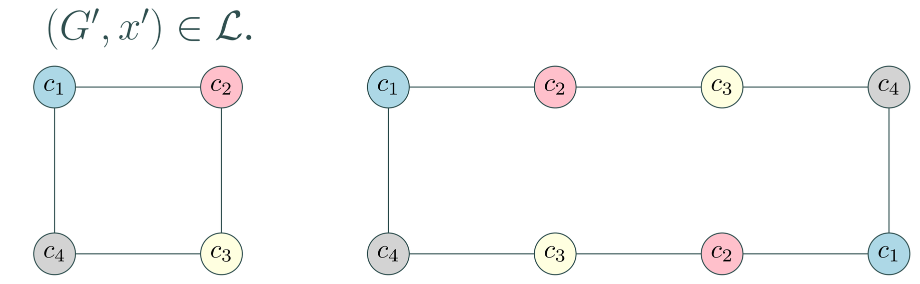
- **Notice** that there does not exist a certificate assignment capable to fool all the nodes and make them all accept on a graph that is not a tree.
- *Nondeterministic Local Decision (NLD)* [4] is the class of distributed languages that are locally verifiable.

> NLD is the class of all distributed languages $\mathcal{L}$ for which there exists a local algorithm $\mathcal{A}$ such that,
>
> $$(G, x) \in \mathcal{L} \iff \exists\, c \text{ s.t. } \mathcal{A} \text{ accepts } (G, x) \text{ with c},$$
>
> where c is a certificate.

A Relation with the Polynomial Hierarchy.

> $L \in$ NP if there is a polynomial time algorithm A such that,
>
> $$x \in L \iff \exists c \text{ s.t. } A \text{ accepts } x \text{ with } c.$$

- NLD is closed under lift [4]:
  if $(G, x) \in \mathcal{L} \ \wedge\ (G', x')$ is a lift of (G, x), then $(G', x') \in \mathcal{L}$.



Distributed Complexity Classes

The classes LD and NLD are the bases of a local hierarchy that we define as follows.

- LD $= \Sigma_0^{loc} = \Pi_0^{loc}$.
- $NLD = \Sigma_1^{loc}$.
- Σ_k^{loc} $(k \geq 1)$ is the class of all distributed languages $\mathcal{L}$ for which there exists a local algorithm $\mathcal{A}$ satisfying that, for every input instance (G, x),

  $$(G, x) \in \mathcal{L} \iff \exists c_1, \forall c_2, \ldots, Qc_k, \text{ s.t. all nodes accept.}$$

  Q is the universal quantifier if k is even and the existential one if k is odd.
- Π_k^{loc} is defined similarly but starting with a universal quantifier instead of an existential one.

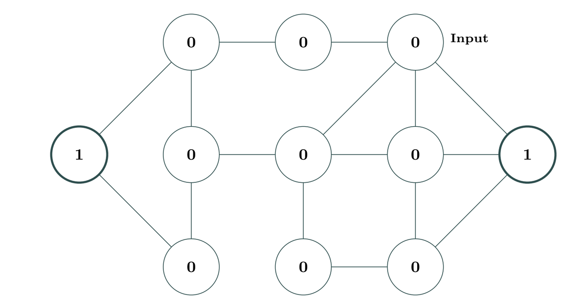
Π_2^{loc} in More Detail

$$(G, x) \in \mathcal{L} \iff \forall c_1 \exists c_2 \text{ s.t. all nodes accept.}$$

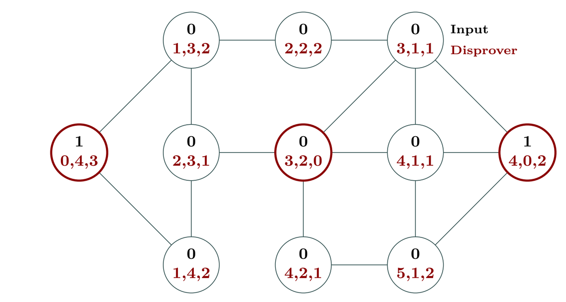
This class can be seen as a two party game between a *disprover* and a *prover*.

- Disprover: tries to make nodes reject the input instance by providing the first certificate.
- Prover: provides the second certificate and tries to make nodes accept.

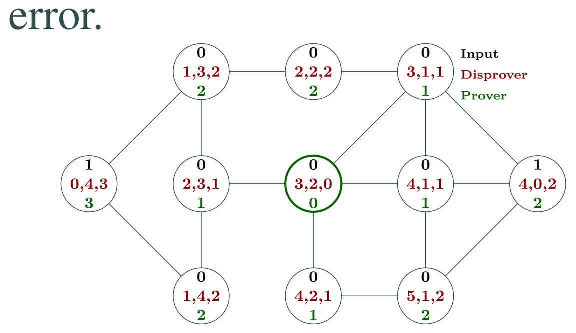
Exactly two selected



The disprover tries to convince the nodes that there are three nodes selected, by giving their distance to each node.



The prover points out to an error in the certificate of the disprover, by giving as a certificate the distance from the error.

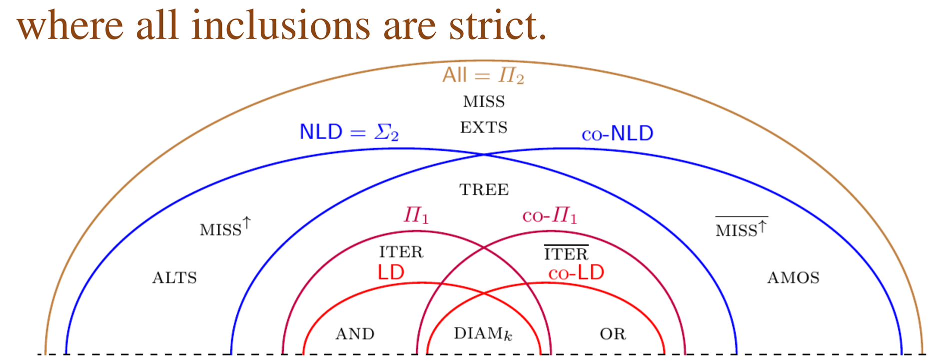


Results

We build a local hierarchy, in which we define complexity classes as well as their complementary ones. For each level, we provide examples of input instances that belong to a specific class, classifying problems by their difficulty.

$$\text{LD} \subset \Pi_1^{\text{loc}} \subset \text{NLD} = \Sigma_2^{\text{loc}} \subset \Pi_2^{\text{loc}} = \text{All},$$

where all inclusions are strict.



References

[1] Balliu A, D'Angelo G, Fraigniaud P, and Olivetti D. Brief announcement: Local distributed verification. *DISC 2016*.

[2] Peleg D. Distributed computing: A locality-sensitive approach. *SIAM 2000*.

[3] Naor M and Stockmeyer L. What can be computed locally? *SIAM J. Comput. '95)*.

[4] Fraigniaud P, Korman A, and Peleg D. Towards a complexity theory for local distributed computing. *J. ACM '13 (FOCS '11)*.