Local Distributed Verification

Towards a Better Understanding of Distributed Computation

Alkida Balliu
CNRS and University Paris Diderot
GSSI L'Aquila

Goal and Motivation

It is enough to think of Internet or mobile networks to realize that distributed systems are part of our daily life. Due to their huge size, it is difficult for an entity to have some global knowledge of the network. Thus, one important aspect is the *locality*, that is the ability of an entity to solve a given task just by exploring the network near its neighborhood. Although widely studied and despite the important theoretical results achieved, distributed systems still lack of the ability to classify problems according to their difficulty, i.e., a complexity theory.

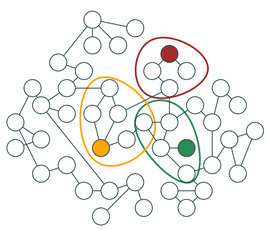
For these reasons, this work aims to improve our understanding of distributed computation, by building a hierarchy of complexity classes in the context of the LOCAL model, as well as defining problems that belong to these classes. To do so, we focus on *decision* and *verification* problems.

LOCAL Model

In the LOCAL model ([1]), a network is represented by an arbitrary connected and simple graph $G = (V, E)$, where V is the set of nodes while E is the set of edges. Each node:

- has a unique identifier;
- has a *local input*;
- provides a *local output*.

The local output is based on the information each node v gathers from its local view, i.e., from the set of nodes at hop-distance t from v, where t is a constant.



Nodes explore the network up to distance 2.

Each node does not know about the existence of nodes outside its ball.

The time complexity of an algorithm $\mathcal{A}$ that solves a problem in the LOCAL model, is determined by the range t that it needs to explore. This means that, in this context, the distributed algorithms run in a constant number of synchronous communication rounds.

Decision Problems

Decision tasks. The aim of decision problems is to *decide* whether a global input instance satisfies some specific property. The strong connection between construction and decision tasks in the LOCAL model, underlines the importance of decision problems.

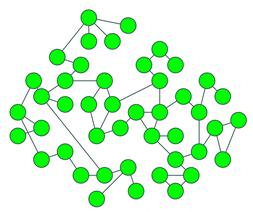
Input instance. We consider as an input instance the pair (G, x), where G is a connected simple graph and x is a function that assigns to each node v the local input $x(v)$.

Distributed language. A distributed language is represented by all the input instances that satisfies a specific property.

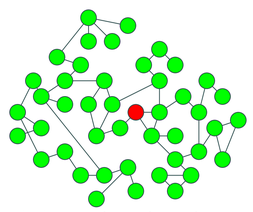
Local decision. To decide whether an input instance satisfies a given property in the LOCAL model, each node v gathers its local information from its local view and outputs its local decision:

- "accept" if the input instance satisfies the desired property;
- "reject" if the input instance does not satisfy the desired property;
- an input instance is globally accepted if and only if *all* nodes have locally accepted, that is

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



A globaly accepted input instance.

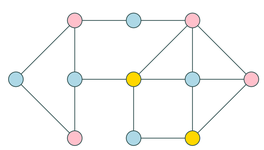


A globaly rejected imput instance.

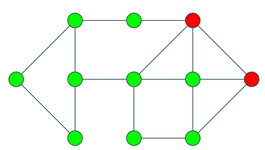
Is the Graph Properly Colored?

- *Input instance*: a graph where each node has in input a color.
- *Decision task*: is the graph properly colored?

In other words, we want to decide if each node has a color that is different from the one of its one-hop neighbors. If it is the case, all nodes will locally accept. Otherwise at least one node of the graph will output "reject", causing a global rejection of the input instance.



A non properly colored input instance.



Output: two nods reject.

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

More in detail, LD, studied in [2], is the class of all distributed languages $\mathcal{L}$ for which there exists a local algorithm $\mathcal{A}$ such that

$$(G, x) \in \mathcal{L} \iff \mathcal{A} \text{ accepts } (G, x).$$

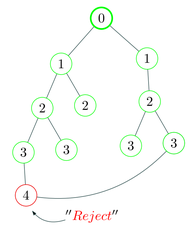
Verification Problems

Verification tasks. The aim of verification problems is to *verify* whether a global input instance satisfies some specific property.

Certificate. Each node has in input a bit string that we call *certificate*, that can be seen as an information given by a third party that we don't trusted a priori. To preserve privacy, we consider these certificates to be independent from the identification number each node has.

Is the Graph a Tree?

- *Task*: is the given graph a tree?
- *Local decidability*: this task is not locally decidable since it requires some global knowledge of the network.
- *Local Verifiability*: this task is locally verifiable and the verification algorithm would perform 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.



All nodes accept.

Notice that there does not exist a certificate assignment capable to full nodes and make them all accept on a graph that is not a tree.

- *Nondeterministic Local Decision (*NLD*)* is the class of distributed languages that are locally verifiable.

More specifically, NLD is the class of all distributed languages $\mathcal{L}$ for which there exists a local algorithm $\mathcal{A}$ such that,

$$(G, x) \in \mathcal{L} \iff \exists\, c \text{ s.t. all nodes output "accept"},$$

where c is a certificate. The NLD class has been studied in [3].

Distributed Complexity Classes

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

- $\mathsf{LD} = \Sigma_0^{loc} = \Pi_0^{loc}$.
- $NLD = \Sigma_1^{loc}$.
- 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),

$$(G, x) \in \mathcal{L} \iff \exists c_1, \forall c_2, \ldots, Q c_k, \text{ such that all nodes accept.}$$

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.

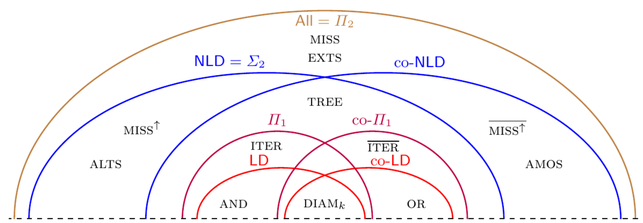
A Relation with the Polynomial Hierarchy in the Sequential Setting. We bring to the attention that the class LD is equivalent to class P of the polynomial hierarchy in the sequential setting, as they are defined similarly. To see this, let us recall that a language $L \in \mathsf{P}$ if there is a polynomial time algorithm A such that for all x $x \in L \iff A$ accepts x. Similarly, it is easy to see that the class NLD is by definition the equivalent of class NP of the polynomial hierarchy.

Results

As a result, 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. Observe that the class Σ_2^{loc} collapses to Σ_1^{loc}, and perhaps the most interesting result is that this hierarchy collapses at the Π_2^{loc} level, showing that this class is enough powerful to solve all decision and verification problems. More in detail, we show that

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

where all inclusions are strict.



References

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

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

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

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