Local Distributed Decision

Based on a paper by Pierre Fraigniaud, Amos Korman and David Peleg

Presented by Mor Weiss
Computational Complexity 101

$P = NP$?
Computational Complexity 101

\[ \begin{align*}
\Sigma_3^P & \quad \Delta_3^P & \quad \Pi_3^P \\
\Sigma_2^P & \quad \Delta_2^P & \quad \Pi_2^P \\
NP & = \Sigma_1^P & Co - NP & = \Pi_1^P \\
\Delta_0^P & = \Sigma_0^P = P & = \Pi_0^P = \Delta_1^P
\end{align*} \]
Computational Complexity 101

**Goal:** do the same for problems in distributed setting

- \( \Delta_0^P = \Sigma_0^P = P = \Pi_0^P = \Delta_1^P \)
- \( NP = \Sigma_1^P \)
- \( Co - NP = \Pi_1^P \)

\( \Delta_2^P \rightarrow \Pi_3^P \)
\( \Sigma_3^P \rightarrow \Delta_2^P \)
How to Sort Problems

• Sorting should be based on parameters of interest
• Parameters of interest in the context of distributed algorithms:
  – locality (# rounds)
  – Determinism\non-determinism\randomization\Mix-and-Match?
  – Time complexity of each node
  – Space complexity of each node
• Goal: hierarchy sorting problems according to “hardness”
• Convention:
  – Less rounds = easier
  – Deterministic = easier than non-deterministic
  – Non randomized = easier than randomized
Ultimate Goal

Easy

Hard
Super Goal
Goal
(Even Better) Goal

[Diagram with various shapes and symbols]
Talk Outline

• Computation model
• Define complexity classes
  – Deterministic computation
  – Non-deterministic computation
  – Randomized computation
  – Non-deterministic and randomized computation
• Show separations and hierarchy
The Model

• We use the **LOCAL model** [Peleg00]
  – Network $G$ of nodes (processors) with unique identifiers $Id(v)$
    • At onset of protocol, $v$ knows identifier $Id(v)$, and input $x(v) \in \{0,1\}^*$ (if exists)
  – Fault-free
  – Synchronous
  – In each round, each $v$:
    • Sends (unlimited size) messages to neighbors
    • Performs arbitrary computation
  – Every node terminates (with output) at some round
    • Different nodes may terminate on different rounds
• Conventions:
  • $n = \#$ nodes
  • Time = $\#$ rounds
Decision Problems

- Complexity classes in **computational complexity** categorize decision problems

- "World" partitioned into "YES" and "NO" instances
  - Set of YES instances is called a "language" \( \mathcal{L}_{YES} \)

- Solution for the problem =
  - Sequential algorithm \( \mathcal{A} \)
  - \( \mathcal{A} \) determines correctly whether instance \( I \in \mathcal{L}_{YES} \) or not

- \( \mathcal{A} \) **decides** \( \mathcal{L}_{YES} \) if \( \mathcal{A} \) solves the decision problem
  - We say that \( \mathcal{L}_{YES} \) is decidable
Distributed Decision Problems

• Also in the distributed setting, we categorize decision problems

• Instance \((G, x)\):
  – \(G\): connected graph over \(n\) nodes
  – \(x = (x_1, \ldots, x_n): x_i \in \{0,1\}^*\) is input string of node \(i\)

• Consider only distributed languages, i.e., decidable collections \(\mathcal{L}\) of instances

• Deterministic distributed algorithm \(A\) decides \(\mathcal{L}\) if:
  – YES instance: \textbf{all} processors output “yes”
    • For every choice of identifiers
  – NO instance: \textbf{at least one} processor outputs “no”
    • For every choice of identifiers
Why Decision Problems?

• Natural “starting point”

• Sufficient for solving “interesting problems”
  – When problem can be phrased as decision problem
  – E.g.: “is network fault-free?”

• For some problems: decision $\Rightarrow$ computation
  – General framework: guess solution, then check
  – E.g., Luby’s MIS algorithm

• Relations to property testing
  – In both cases global decision made based on little information
  – Several graph properties testable by examining neighborhoods of few nodes
Hierarchy of Complexity Classes (1)

• $LD(t)$ (local decision in $t$ time): class of distributed languages deterministically decidable in the $LOCAL$ model in $t$ rounds

• $LD(O(t)) = \bigcup_{c > 0} LD(ct)$
  – Especially interested in $LD := LD(O(1))$

• $BPLD(t, p, q)$ (probabilistic local decision in $t$ time): class of distributed languages probabilistically decidable in the $LOCAL$ model in $t$ rounds
  – Each node tosses coins independently of other nodes
  – YES instances accepted with probability $\geq p$
  – NO instances rejected with probability $\geq q$
  – Probability is over coin tosses at all nodes
  – BP: bounded-error probabilistic
Probabilistic Error Helps

- By allowing probabilistic error, can decrease # rounds from $\Omega(n)$ to constant

- **Thm:** $\forall p, q \in (0,1]$ s.t. $p^2 + q \leq 1$: 
  \[ \exists L \in BPLD(0, p, q) \]
  but for every $t = o(n)$: 
  \[ L \notin LD(t) \]

- We choose $L$="dictatorship or chaos"
  - $(G, x) \in L \iff$ at most one node selected
  - i.e.,
    - $\forall i, x_i \in \{0,1\}$
    - $i$ selected $\iff x_i = 1$
Probabilistic Error Helps

• By allowing probabilistic error, can decrease # rounds from $\Omega(n)$ to constant

• **Thm:** $\forall p, q \in (0,1]$ s.t. $p^2 + q \leq 1$:
  \[
  \exists \mathcal{L} \in BPLD(0, p, q)
  \]
  but for every $t = o(n)$:
  \[
  \mathcal{L} \notin LD(t)
  \]

• We choose $\mathcal{L} =$ “dictatorship or chaos”
  – $(G, x) \in \mathcal{L} \iff$ at most one node selected
  – i.e.,
    • $\forall i, x_i \in \{0,1\}$
    • $i$ selected $\iff x_i = 1$

probabilistic distributed algorithms, no communication
- $\Pr[I \in \mathcal{L} \text{ accepted}] \geq p$
- $\Pr[I \notin \mathcal{L} \text{ rejected}] \geq q$

t-round deterministic distributed algorithms
Probabilistic Error Helps (cont.)

- $\mathcal{L}$ = “dictatorship or chaos”
  - $(G, x) \in \mathcal{L} \iff$ at most one node selected
  - i.e., $\forall i, x_i \in \{0,1\}$, $i$ selected $\iff x_i = 1$

- $\forall t = o(n), \mathcal{L} \notin \text{LD}(t)$:
  - $t$-round deterministic distributed algorithms
  
  ![Graph with nodes and edges showing $n/2$ nodes selected]

\[ n \]
Probabilistic Error Helps (cont.)

- $\mathcal{L}$="dictatorship or chaos"
  - $(G, x) \in \mathcal{L} \iff$ at most one node selected
  - i.e., $\forall i, x_i \in \{0,1\}, i$ selected $\iff x_i = 1$

- $\forall t = o(n), \mathcal{L} \not\in \mathsf{LD}(t)$:
  - $t$-round deterministic distributed algorithms

\[ 0 \ 0 \ 0 \ 0 \ 0 \ \ldots \ \ldots \ 0 \ \ldots \ 0 \ 0 \ 0 \ 0 \ 1 \]

\[ 1 \ 0 \ 0 \ 0 \ 0 \ \ldots \ 0 \ \ldots \ 0 \ 0 \ 0 \ 0 \ 0 \]

\[ 1 \ 0 \ 0 \ 0 \ 0 \ \ldots \ 0 \ \ldots \ 0 \ 0 \ 0 \ 0 \ 1 \]
Probabilistic Error Helps (cont.)

- $\mathcal{L}=$ “dictatorship or chaos”
- $t = o(n)$

\[
\begin{array}{ccccccc}
0 & 0 & 0 & 0 & 0 & \ldots & 0 \\
1 & 0 & 0 & 0 & 0 & \ldots & 0 \\
\end{array}
\]

\[
\begin{array}{ccccccc}
0/2 & & & & & \ldots & \\
1/2 & & & & & \ldots & \\
\end{array}
\]

\[
\begin{array}{ccccccc}
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
Probabilistic Error Helps (cont.)

- $\mathcal{L}=$ “dictatorship or chaos”
- $t = o(n)$

\[ \begin{array}{ccccccc}
0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
\end{array} \quad \begin{array}{ccccccc}
n/2 \\
\end{array} \quad \begin{array}{ccccccc}
0 & \cdots & 0 & 0 & 0 & 1 \\
\end{array} \]

\[ \begin{array}{ccccccc}
1 & 0 & 0 & 0 & 0 & \cdots & 0 \\
\end{array} \quad \begin{array}{ccccccc}
n/4 \\
\end{array} \quad \begin{array}{ccccccc}
n/2 \\
\end{array} \quad \begin{array}{ccccccc}
3n/4 \\
\end{array} \quad \begin{array}{ccccccc}
0 & 0 & 0 & 0 & 0 \\
\end{array} \]

\[ \begin{array}{ccccccc}
1 & 0 & 0 & 0 & 0 & \cdots & 0 \\
\end{array} \quad \begin{array}{ccccccc}
n/4 \\
\end{array} \quad \begin{array}{ccccccc}
n/2 \\
\end{array} \quad \begin{array}{ccccccc}
3n/4 \\
\end{array} \quad \begin{array}{ccccccc}
0 & 0 & 0 & 0 & 1 \\
\end{array} \]
Probabilistic Error Helps (cont.)

- $\mathcal{L} =$ “dictatorship or chaos”
- $t = o(n)$

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & \ldots & 0 & \ldots \\
1 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & \ldots & 0 & \ldots \\
0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]
Probabilistic Error Helps (cont.)

- $\mathcal{L}=$“dictatorship or chaos”
- $t = o(n)$

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]
Probabilistic Error Helps (cont.)

- $\mathcal{L} = \text{"dictatorship or chaos"}$
- $t = o(n)$

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & \ldots & 0 & \text{...} & 0 & 0 & 0 & \text{...} & 0 & 0 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & \text{...} & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & \text{...} & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]
Probabilistic Error Helps (cont.)

- $\mathcal{L} =$ “dictatorship or chaos”

- $t = o(n)$

\[
\begin{array}{cccccccccc}
0 & 0 & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]
Probabilistic Error Helps (cont.)

• $\mathcal{L}=$ “dictatorship or chaos”
• $t = o(n)$

\[
\forall t = o(n) : \text{doc} \notin LD(t)
\]
Probabilistic Error Helps (cont.)

- $\mathcal{L} =$ “dictatorship or chaos”
  
  - $(G, x) \in \mathcal{L} \iff$ at most one node selected
  
  - i.e., $\forall i, x_i \in \{0, 1\}, i$ selected $\iff x_i = 1$

- $p^2 + q \leq 1$: $\mathcal{L} \in BPLD(0, p, q)$
  
  - probabilistic distributed algorithms, no communication

- Node $i$ algorithm:
  
  - If $x_i = 0$ answer “YES” with probability 1
  
  - If $x_i = 1$ answer “YES” with probability $p$

- $(G, x) \in \mathcal{L}$:
  
  - No node selected: $\Pr[out = YES] = 1$
  
  - One node selected: $\Pr[out = YES] = p$

- $(G, x) \notin \mathcal{L}$: $k \geq 2$ nodes selected
  
  - $\Pr[all \ output \ YES] = \prod_i \Pr[v_i \ outputs \ YES] = p^k \leq p^2$
  
  - $\Pr[out = NO] \geq 1 - p^2 \geq q$  \[ p^2 + q \leq 1 \]
$t = o(n)$

$LD(t) \neq BPLD(0, p, q)$
$p^2 + q \leq 1$

$BPLD(t, p, q)$
$p^2 + q \leq 1$
Probabilistic Error Helps (cont.)

- **Thm:** \( \forall p, q \text{ s.t. } p^2 + q \leq 1: \)
  \[
  \exists \mathcal{L} \in BPLD(0, p, q)
  \]
  but for every \( t = o(n) \):
  \[
  \mathcal{L} \notin LD(t)
  \]

- What about other \( p, q \) values?
  - Still open

- **Disclaimer of Liability**

  With respect to the tips provided in this presentation, “Pushing SharePoint’s ‘Go Faster’ Button,” Sean P. McDonough makes no warranties, express or implied, including the warranties of safety and fitness for use in a particular SharePoint environment, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, or represents that its use would not infringe privately owned rights.

  It is further understood and accepted that information presented herein is a derivative of the information on the SharePoint site and is not intended to be used as a substitute for professional judgment, nor does it create a legal or financial obligation. Any suggestions provided are for informational purposes only and should not be relied upon as a substitute for professional advice.

  **TIP #1:** The SharePoint object cache is not available in environments running Windows Server 2003 or SharePoint Foundation 2010. The object cache is backed by memory and can impact server performance when memory allocations are increased or decreased below certain thresholds. Actual thresholds vary from environment to environment, so ensure that the environment being modified is localized and understood from a performance perspective before making changes. Consult the “SharePoint 2010 Object Cache” article on SharePointCertificate.com for further discussion that includes performance counters that can be leveraged. In SharePoint 2010 environments, cache super user and cache super reader accounts must be configured following “SharePoint 2010” system setup to ensure proper operation; consult TechNet for additional details and PowerShell scripting. Enabling the object cache for query results can result in significant performance gains, and it is recommended to disable the object cache for query results.

  - **Bound is tight for specific language family \( \mathcal{F} \)**
    - \( I \in \mathcal{F} \Rightarrow \text{for every sub-instance } I' \text{ of } I, I' \in \mathcal{F} \)
\[ t = o(n) \]

\[ LD(t) \neq BPLD(0, p, q) \]
\[ p^2 + q \leq 1 \]

Goal

\[ BPLD(t, p, q) \]
\[ p^2 + q \leq 1 \]
Hierarchy of Complexity Classes (2)

- **LD**: class of decision problems **deterministically** decidable in the \(\mathcal{LOCAL}\) model in \(O(1)\) rounds

- **NLD\((t)\)** **(non–deterministic local decision in \(t\) time)**: class of distributed languages **non–deterministically** decidable in the \(\mathcal{LOCAL}\) model in \(t\) rounds
  
  - These are verification algorithms: every node \(v\) given auxiliary input \(y(v) \in \{0,1\}^*\) (the certificate)
  
  - YES instances have convincing certificates
    - \(\exists\) certificate s.t. \(\forall\) identifier assignment, all nodes output “yes”
  
  - NO instances do not have any convincing certificates
    - \(\forall\) certificate and \(\forall\) identifier assignment, \(\exists\) node that outputs “no”

\[ NLD := NLD(O(1)) \]
Non-Determinism Helps

- **Thm:** \( \exists \) distributed language \( \mathcal{L} \):
  \[
  \mathcal{L} \in \mathcal{NLD}
  \]
  but for every \( t = o(n) \):
  \[
  \mathcal{L} \notin \mathcal{LD}(t)
  \]

- \( \mathcal{L} = \) language of tree graphs = \{\((G, \epsilon) : G \text{ is a tree}\)\}
  - \( \epsilon \) denotes the empty string, i.e., nodes given no additional input \( x \)
Non-Determinism Helps (cont.)

- $\mathcal{L}$ = language of tree graphs $= \{(G, \epsilon): G \text{ is a tree}\}$
  - $\epsilon$ denotes the empty string, i.e., nodes given no additional input $x$

- $\forall t = o(n)$, $\mathcal{L} \notin LD(t)$: $t$-round deterministic distributed algorithms

\[ 1 \ 2 \ 3 \ 4 \ \ldots \ \ldots \ \ldots \ 4n \]

\[ 2n + 1 \ \ldots \ 4n \ 1 \ \ldots \ \ldots \ 2n \]

\[ 1 \ 2 \ 3 \ 4 \ \ldots \ \ldots \ \ldots \ 4n \]
Non-Determinism Helps (cont.)

- $\mathcal{L} =$ language of tree graphs = \{$(G, \epsilon) : G$ is a tree$\}$
  - $\epsilon$ denotes the empty string, i.e., nodes given no additional input $x$
- $\forall t = o(n)$, $\mathcal{L} \not\subseteq \mathcal{LD}(t)$:
  - $t$-round deterministic distributed algorithms
Non-Determinism Helps (cont.)

- \( \mathcal{L} = \text{language of tree graphs} = \{(G, \epsilon): G \text{ is a tree}\} \)
  - \( \epsilon \) denotes the empty string, i.e., nodes given no additional input \( x \)

- \( \forall t = o(n), \mathcal{L} \notin LD(t); t\)-round deterministic distributed algorithms
Non-Determinism Helps (cont.)

• \( \mathcal{L} = \text{language of tree graphs} = \{(G, \epsilon): G \text{ is a tree}\} \)
  
  – \( \epsilon \) denotes the empty string, i.e., nodes given no additional input \( x \)

• \( \forall t = o(n), \mathcal{L} \notin \mathcal{LD}(t): \) \( t \)-round deterministic distributed algorithms

\[ \forall t = o(n): \mathcal{L} \notin \mathcal{LD}(t) \]
Non-Determinism Helps (cont.)

- $\mathcal{L}$ = language of tree graphs = $\{(G, \epsilon): G$ is a tree$\}$
  - $\epsilon$ denotes the empty string, i.e., nodes given no additional input $x$

- $\mathcal{L} \in \text{NLD}$: constant-round non-deterministic distributed algorithms

- Algorithm:
  - Fix $\hat{v} \in V$
  - Certificate at node $v = \text{dist}(\hat{v}, v)$
    - Denoted $c(v)$
  - Node $v$ verifies (outputs “YES” iff tests pass):
    - $c(v) \geq 0$
    - $v = \hat{v}$
    - If $c(v) = 0$ then $c(w) = 1$ for every neighbor $w$ of $v$
    - If $c(v) > 0$:
      - $\exists$ neighbor $w$ s.t. $c(v) = c(w) + 1$
      - For all other neighbors $w$, $c(w) = c(v) + 1$
Non-Determinism Helps (cont.)

- \( \mathcal{L} = \) language of tree graphs = \( \{(G, \varepsilon): G \text{ is a tree}\} \)
- \( \mathcal{L} \in \mathcal{NLD} \): constant-round non-deterministic distributed algorithms

- Algorithm:
  - Fix \( \hat{v} \in V \), \( c(v) = \text{dist}(\hat{v}, v) \)
  - Node \( v \) verifies (outputs “YES” iff tests pass):
    - \( c(v) \geq 0 \)
    - If \( c(v) = 0 \) then \( c(w) = 1 \) for every neighbor \( w \) of \( v \)
    - If \( c(v) > 0 \):
      - \( \exists \) neighbor \( w \) s.t. \( c(v) = c(w) + 1 \)
      - For all other neighbors \( w \), \( c(w) = c(v) + 1 \)
- \( G \) is a tree \( \Rightarrow \) all output “YES”
Non-Determinism Helps (cont.)

- $\mathcal{L} =$ language of tree graphs $= \{(G, \epsilon): G$ is a tree$\}$
- $\mathcal{L} \in \mathcal{NLD}$: constant-round non-deterministic distributed algorithms

Algorithm:
- Fix $\hat{v} \in V$, $c(v) = \text{dist}(\hat{v}, v)$
- Node $v$ verifies (outputs “YES” iff tests pass):
  - $c(v) \geq 0$
  - If $c(v) = 0$ then $c(w) = 1$ for every neighbor $w$ of $v$
  - If $c(v) > 0$:
    - $\exists$ neighbor $w$ s.t. $c(v) = c(w) + 1$
    - For all other neighbors $w$, $c(w) = c(v) + 1$

- $G$ is a tree $\Rightarrow$ all output “YES”
- $G$ not a tree $\Rightarrow$ at least one outputs “NO”
Non-Determinism Helps (cont.)

- $\mathcal{L}$ = language of tree graphs = $\{(G, \epsilon) : G$ is a tree$\}$
- $\mathcal{L} \in NLD$: constant-round non-deterministic distributed algorithms
- Algorithm:
  - Fix $\hat{v} \in V$, $c(v) = \text{dist}(\hat{v}, v)$
  - Node $v$ verifies (outputs “YES” iff tests pass):
    - $c(v) \geq 0$
    - If $c(v) = 0$ then $c(w) = 1$ for every neighbor $w$ of $v$
    - If $c(v) > 0$:
      - $\exists$ neighbor $w$ s.t. $c(v) = c(w) + 1$
      - For all other neighbors $w$, $c(w) = c(v) + 1$
- $G$ is a tree $\Rightarrow$ all output “YES”
- $G$ not a tree $\Rightarrow$ at least one outputs “NO”
Non-Determinism Helps (cont.)

- $\mathcal{L} =$ language of tree graphs $= \{(G, \epsilon): G$ is a tree\}
- $\mathcal{L} \in \textcolor{red}{\mathcal{NLD}}$: constant-round non-deterministic distributed algorithms

Algorithm:
- Fix $\hat{v} \in V$, $c(v) = \text{dist}(\hat{v}, v)$
- Node $v$ verifies (outputs “YES” iff tests pass):
  - $c(v) \geq 0$
  - If $c(v) = 0$ then $c(w) = 1$ for every neighbor $w$ of $v$
  - If $c(v) > 0$:
    - $\exists$ neighbor $w$ s.t. $c(v) = c(w) + 1$
    - For all other neighbors $w$, $c(w) = c(v) + 1$

- $G$ is a tree $\Rightarrow$ all output “YES”
- $G$ not a tree $\Rightarrow$ at least one outputs “NO”
Non-Determinism Helps (cont.)

- $\mathcal{L} =$ language of tree graphs $= \{(G, \epsilon): G \text{ is a tree}\}$
- $\mathcal{L} \in \textit{NLD}$: constant-round non-deterministic distributed algorithms

Algorithm:
- Fix $\hat{v} \in V$, $c(v) = \text{dist}(\hat{v}, v)$
- Node $v$ verifies (outputs “YES” iff tests pass):
  - $c(v) \geq 0$
  - If $c(v) = 0$ then $c(w) = 1$ for every neighbor $w$ of $v$
  - If $c(v) > 0$:
    - $\exists$ neighbor $w$ s.t. $c(v) = c(w) + 1$
    - For all other neighbors $w$, $c(w) = c(v) + 1$

- $G$ is a tree $\Rightarrow$ all output “YES”
- $G$ not a tree $\Rightarrow$ at least one outputs “NO”
Non-Determinism Helps (cont.)

• \( \mathcal{L} = \) language of tree graphs = \( \{(G, \epsilon): G \text{ is a tree}\} \)

• \( \mathcal{L} \in \text{NLD} \): constant-round non-deterministic distributed algorithms

• Algorithm:
  – Fix \( \hat{v} \in V, c(v) = \text{dist}(\hat{v}, v) \)
  – Node \( v \) verifies (outputs “YES” iff tests pass):
    • \( c(v) \geq 0 \)
    • If \( c(v) = 0 \) then \( c(w) = 1 \) for every neighbor \( w \) of \( v \)
    • If \( c(v) > 0 \):
      – \( \exists \) neighbor \( w \) s.t. \( c(v) = c(w) + 1 \)
      – For all other neighbors \( w \), \( c(w) = c(v) + 1 \)

• \( G \) is a tree \( \Rightarrow \) all output “YES”

• \( G \) not a tree \( \Rightarrow \) at least one outputs “NO”
Non-Determinism Helps (cont.)

- $\mathcal{L} = \text{language of tree graphs} = \{(G, \epsilon) : G \text{ is a tree}\}$
- $\mathcal{L} \in \text{NLD} \rightarrow \text{constant-round non-deterministic distributed algorithms}$

Algorithm:
- Fix $\hat{v} \in V$, $c(v) = \text{dist}(\hat{v}, v)$
- Node $v$ verifies (outputs “YES” iff tests pass):
  - $c(v) \geq 0$
  - If $c(v) = 0$ then $c(w) = 1$ for every neighbor $w$ of $v$
  - If $c(v) > 0$:
    - $\exists$ neighbor $w$ s.t. $c(v) = c(w) + 1$
    - For all other neighbors $w$, $c(w) = c(v) + 1$
- $G$ is a tree $\Rightarrow$ all output “YES”
- $G$ not a tree $\Rightarrow$ at least one outputs “NO”
Non-Determinism Helps (cont.)

- $\mathcal{L} =$ language of tree graphs $= \{(G, \epsilon): G$ is a tree$\}$
- $\mathcal{L} \in \text{NLD}$: constant-round non-deterministic distributed algorithms

- Algorithm:
  - Fix $\hat{v} \in V$, $c(v) = \text{dist}(\hat{v}, v)$
  - Node $v$ verifies (outputs “YES” iff tests pass):
    - $c(v) \geq 0$
    - If $c(v) = 0$ then $c(w) = 1$ for every neighbor $w$ of $v$
    - If $c(v) > 0$:
      - $\exists$ neighbor $w$ s.t. $c(v) = c(w) + 1$
      - For all other neighbors $w$, $c(w) = c(v) + 1$

- $G$ is a tree $\Rightarrow$ all output “YES”
- $G$ not a tree $\Rightarrow$ at least one outputs “NO”
Goal

\[ t = o(n) \]

\[ \subseteq \]

\[ \mathcal{L}D \subseteq \mathcal{N}L\mathcal{D} \]

\[ B\mathcal{P}\mathcal{L}\mathcal{D}(0, p, q) \]

\[ p^2 + q \leq 1 \]

\[ \mathcal{L}D(t) \]

\[ \mathcal{N}L\mathcal{D}(t) \]

\[ \mathcal{L}D(t) \neq \mathcal{N}L\mathcal{D} \]
Non-Determinism Is Not Enough

- \( NLD(t) \) = \( t \)-round non-deterministic distributed algorithms
- \( ALL \) = all (sequentially) decidable distributed languages
- **Thm:** \( \forall t = o(n), NLD(t) \subset ALL \)
- \( \mathcal{L} = \{ (G, x) : \forall v \in V(G), x(v) = |V(G)| \} \)
  - \( \mathcal{L} \subset ALL \)
- We show \( \mathcal{L} \not\in NLD(t) \) for \( t = o(n) \)
Non-Determinism Is Not Enough (cont.)

- \( \mathcal{L} = \{(G,x): \forall v \in V(G), x(v) = |V(G)|\} \)
- **Claim:** \( \mathcal{L} \notin NLD(t) \) for \( t = o(n) \)
- **Proof:** assume \( \mathcal{A} \) decides \( \mathcal{L} \) in \( o(n) \) time

\[ n := \text{smallest such that } t < \left\lfloor \frac{n}{4} \right\rfloor, \text{ where } \mathcal{A} \text{ takes } t \text{ time on:} \]

\[ u_0, u_1, \ldots, v_0, v_1, \ldots, u_{n-1} \]

\[ c_0, c_1, \ldots, c_{n-3}, c_{n-2}, c_{n-1} \]
Non-Determinism Is Not Enough (cont.)

- $\mathcal{L} = \{(G, x) : \forall v \in V(G), x(v) = |V(G)|\}$
- **Claim:** $\mathcal{L} \notin NLD(t)$ for $t = o(n)$
- **Proof:** assume $\mathcal{A}$ decides $\mathcal{L}$ in $o(n)$ time
- $n := \text{smallest such that } t < \left\lceil \frac{n}{4} \right\rceil$, where $\mathcal{A}$ takes $t$ time on:

![Diagram of graphs showing $u_0$ to $u_{n-1}$ and $v_0$ to $v_{2n-1}$ connected with $c_0$ to $c_{n-1}$]}
Non-Determinism Is Not Enough (cont.)

• $\mathcal{L} = \{(G,x) : \forall v \in V(G), x(v) = |V(G)|\}$

• **Claim:** $\mathcal{L} \notin NLD(t)$ for $t = o(n)$

• **Proof:** assume $\mathcal{A}$ decides $\mathcal{L}$ in $o(n)$ time

• $n := \text{smallest such that } t < \left\lfloor \frac{n}{4} \right\rfloor$, where $\mathcal{A}$ takes $t$ time on: 

\[
\begin{align*}
&u_0 \quad u_1 \quad \cdots \quad u_{n-1} \\
&c_0 \quad c_1 \quad \cdots \quad c_{n-1}
\end{align*}
\]

\[
\begin{align*}
&v_0 \quad v_1 \quad \cdots \quad v_{n-1} \quad v_n \quad v_{n+1} \\
&c_0 \quad c_1 \quad \cdots \quad c_{n-1}
\end{align*}
\]
Non-Determinism Is Not Enough (cont.)

• $\mathcal{L} = \{(G, x): \forall v \in V(G), \; x(v) = |V(G)|\}$
• **Claim:** $\mathcal{L} \notin NLD(t)$ for $t = o(n)$
• **Proof:** assume $\mathcal{A}$ decides $\mathcal{L}$ in $o(n)$ time

• $n := \text{smallest such that } t < \left\lceil \frac{n}{4} \right\rceil$, where $\mathcal{A}$ takes $t$ time on:
Non-Determinism Is Not Enough (cont.)

- $\mathcal{L} = \{(G, x): \forall v \in V(G), x(v) = |V(G)|\}$
- **Claim:** $\mathcal{L} \notin NLD(t)$ for $t = o(n)$
- **Proof:** assume $\mathcal{A}$ decides $\mathcal{L}$ in $o(n)$ time

- $n :=$ smallest such that $t < \left\lceil \frac{n}{4} \right\rceil$, where $\mathcal{A}$ takes $t$ time on:

  $\begin{align*}
  &n \quad n \quad n \quad \ldots \quad n \quad n \quad n \\
  &u_0 \quad u_1 \quad \ldots \quad \ldots \quad u_{n-1} \\
  &c_0 \quad c_1 \quad c_2 \quad \ldots \quad c_{n-3} \quad c_{n-2} \quad c_{n-1} \\
  
  &n \quad n \quad n \quad \ldots \quad n \quad n \quad n \\
  &v_0 \quad v_1 \quad \ldots \quad \ldots \quad v_{n-1} \quad v_n \quad v_{n+1} \\
  &c_0 \quad c_1 \quad \ldots \quad c_{n-1} \quad c_0 \quad c_1 \quad \ldots \\
  &i_0 \quad i_1 \quad \ldots \quad i_{n-1} \quad i_n \quad i_{n+1} \quad \ldots \quad i_{2n-2} \quad i_{2n-1}
  \end{align*}$
Non-Determinism Is Not Enough (cont.)

- $\mathcal{L} = \{(G, x) : \forall v \in V(G), \ x(v) = |V(G)|\}$
- $\mathcal{A}$ decides $\mathcal{L}$ in $o(n)$ time

- $n :=$ smallest such that $t < \left\lceil \frac{n}{4} \right\rceil$, where $\mathcal{A}$ takes $t$ time on:

  \[
  j = 0
  \]

  \[
  \text{ } \quad t < \left\lceil \frac{n}{4} \right\rceil \quad \text{ and } \quad t < \left\lceil \frac{n}{4} \right\rceil
  \]

  \[
  C_{\frac{n-\left\lceil \frac{n}{4} \right\rceil}{4}} \quad \ldots \quad C_{\frac{n}{4}}
  \]

  \[
  u_{\frac{n-\left\lceil \frac{n}{4} \right\rceil}{4}} \quad \ldots \quad u_0 \quad u_1 \quad \ldots \quad u_{\frac{n}{4}}
  \]

  \[
  i_{\frac{2n-\left\lceil \frac{n}{4} \right\rceil}{4}} \quad \ldots \quad i_0 \quad i_1 \quad \ldots \quad i_{\frac{n}{4}}
  \]

  \[
  v_{\frac{n}{4}} \quad \ldots \quad v_0 \quad v_1 \quad \ldots \quad v_{2n-1}
  \]

  \[
  c_0 \quad c_1 \quad \ldots \quad c_{\frac{n-1}{4}} \quad c_0 \quad c_1 \quad \ldots \quad c_{\frac{n-2}{4}}
  \]

  \[
  i_0 \quad i_1 \quad \ldots \quad i_{\frac{n-1}{4}} \quad i_0 \quad i_1 \quad \ldots \quad i_{\frac{n-2}{4}}
  \]
Non-Determinism Is Not Enough (cont.)

• $\mathcal{L} = \{(G, x) : \forall v \in V(G), x(v) = |V(G)|\}$

• **Claim:** $\mathcal{L} \notin NLD(t)$ for $t = o(n)$

• **Proof:** assume $\mathcal{A}$ decides $\mathcal{L}$ in $o(n)$ time

• $n := \text{smallest such that } t < \left\lceil \frac{n}{4} \right\rceil$, where $\mathcal{A}$ takes $t$ time on:

  $n$ $n$ $n$ $\ldots$ $n$ $n$ $n$ $n$

  $u_0$ $u_1$ $u_2$ $\ldots$ $u_{n-1}$

  $c_0$ $c_1$ $c_2$ $\ldots$ $c_{n-3}$ $c_{n-2}$ $c_{n-1}$

  $j = 0$

  $i_0$ $i_1$ $i_2$ $\ldots$ $i_{n-1}$ $i_n$ $i_{n+1}$ $i_{2n-2}$ $i_{2n-1}$

  $v_0$ $v_1$ $v_2$ $\ldots$ $v_{n-1}$ $v_n$ $v_{n+1}$ $v_{2n-2}$ $v_{2n-1}$

  $c_0$ $c_1$ $c_2$ $\ldots$ $c_{n-1}$ $c_0$ $c_1$ $\ldots$ $c_{n-2}$ $c_{n-1}$
Non-Determinism Is Not Enough (cont.)

- $\mathcal{L} = \{(G, x): \forall v \in V(G), x(v) = |V(G)|\}$
- $\mathcal{A}$ decides $\mathcal{L}$ in $o(n)$ time
- $n := \text{smallest such that } t < \left\lceil \frac{n}{4} \right\rceil$, where $\mathcal{A}$ takes $t$ time on:

$$
j = 1
$$

$$
C_{2n+1-\left\lceil \frac{n}{4} \right\rceil} \quad \ldots \quad C_0 \quad C_1 \quad C_2 \quad \ldots \quad C_{\left\lceil \frac{n}{4} \right\rceil + 1}
$$

$$
u_{n+1-\left\lceil \frac{n}{4} \right\rceil} \quad \ldots \quad u_0 \quad u_1 \quad u_2 \quad \ldots \quad u_{\left\lceil \frac{n}{4} \right\rceil + 1}
$$

$$
i_{2n+1-\left\lceil \frac{n}{4} \right\rceil} \quad \ldots \quad i_0 \quad i_1 \quad i_2 \quad \ldots \quad i_{\left\lceil \frac{n}{4} \right\rceil + 1}
$$

$$
v_0 \quad v_1 \quad \ldots \quad v_{n-1} \quad v_n \quad v_{n+1} \quad \ldots \quad v_{2n-2} \quad v_{2n-1}
$$

$$
c_0 \quad c_1 \quad \ldots \quad c_{n-1} \quad c_0 \quad c_1 \quad \ldots \quad c_{n-2} \quad c_{n-1}
$$

$$
i_0 \quad i_1 \quad \ldots \quad i_{n-1} \quad i_n \quad i_{n+1} \quad \ldots \quad i_{2n-2} \quad i_{2n-1}
$$
Non-Determinism Is Not Enough (cont.)

- $\mathcal{L} = \{(G,x) : \forall v \in V(G),\ x(v) = |V(G)|\}$
- **Claim:** $\mathcal{L} \notin NLD(t)$ for $t = o(n)$
- **Proof:** assume $\mathcal{A}$ decides $\mathcal{L}$ in $o(n)$ time
  
  $n :=$ smallest such that $t < \left\lceil \frac{n}{4} \right\rceil$, where $\mathcal{A}$ takes $t$ time on:

  \[
  \begin{array}{cccccccccccc}
  & n & n & n & \cdots & n & n & n & n \\
  u_0 & u_1 & & & & \cdots & & & u_{n-1} \\
  c_0 & c_1 & c_2 & \cdots & c_{n-3} & c_{n-2} & c_{n-1} \\
  & & & & & \vdots & & & & \\
  \end{array}
  \]

  \[
  \begin{array}{cccccccccccc}
  & n & n & n & \cdots & n & n & n & n \\
  v_0 & v_1 & & & & \cdots & & & v_{n-1} & v_n & v_{n+1} & \cdots & v_{2n-2} & v_{2n-1} \\
  c_0 & c_1 & c_2 & \cdots & c_{n-1} & c_0 & c_1 & \cdots & c_{n-2} & c_{n-1} & c_0 & c_1 & \cdots & c_{2n-2} & c_{2n-1} \\
  i_0 & i_1 & & & & \cdots & & & i_{n-1} & i_n & i_{n+1} & \cdots & i_{2n-2} & i_{2n-1} \\
  \end{array}
  \]

  \[
  \begin{array}{cccccccccccc}
  & j = 1 \\
  \end{array}
  \]
Non-Determinism Is Not Enough (cont.)

- \( \mathcal{L} = \{(G, x): \forall v \in V(G), \ x(v) = |V(G)|\} \)
- **Claim:** \( \mathcal{L} \not\in NLD(t) \text{ for } t = o(n) \)
- **Proof:** assume \( \mathcal{A} \) decides \( \mathcal{L} \) in \( o(n) \) time

- \( n := \text{smallest such that } t < \left\lfloor \frac{n}{4} \right\rfloor \), where \( \mathcal{A} \) takes \( t \) time on:
Non-Determinism Is Not Enough (cont.)

- \( \mathcal{L} = \{(G, x): \forall v \in V(G), x(v) = |V(G)|\} \)
- **Claim:** \( \mathcal{L} \not\in NLD(t) \) for \( t = o(n) \)
- **Proof:** assume \( \mathcal{A} \) decides \( \mathcal{L} \) in \( o(n) \) time

\[
\forall t = o(n): \mathcal{L} \not\in NLD(t)
\]
\[ t = o(n) \]

\[ \text{Goal} \]

\[ \subseteq \]

\[ BPLD(0, p, q) \]
\[ p^2 + q \leq 1 \]

\[ LD(t) \]

\[ NLD(t) \]

\[ \subseteq \]

\[ \text{ALL} \]
Hierarchy of Complexity Classes (3)

- \( BPNLD(t, p, q) \) (probabilistic and non-deterministic local decision in \( t \) time): class of distributed languages non-deterministically and probabilistically decidable in the \( LOCAL \) model in \( t \) rounds
  - YES instances have certificates that convince with probability \( \geq p \)
    - \( \exists \) certificate s.t. \( \forall \) identifier assignment, all nodes output “yes” with probability \( \geq p \)
  - NO instances do not have certificates that convince with probability \( > 1 - q \)
    - \( \forall \) certificate and \( \forall \) identifier assignment, with probability \( \geq q \) \( \exists \) node that outputs “no”
Combining Non-Determinism and Randomization Is Enough

- **ALL** = all (sequentially) decidable distributed languages
- **Thm:** \( \forall p, q \in (0,1] \) s.t. \( p^2 + q \leq 1 \): 
  \[ BPNLD(1, p, q) = ALL \]

- **Proof:**
  - Certificate contains isomorphic copy \( G' \) of \( G \)
    - Defined by node permutation \( \lambda \)

\[
x = (x_{v_1}, \ldots, x_{v_6})
\]

\[
x' = (x_{\lambda(v_1)}, \ldots, x_{\lambda(v_6)})
\]
Combining Non-Determinism and Randomization Is Enough

- **ALL** = all (sequentially) decidable distributed languages
- **Thm:** $\forall p, q \in (0, 1]$ s.t. $p^2 + q \leq 1$:
  \[ \text{BPNLD}(1, p, q) = \text{ALL} \]
- **Proof:**
  - Certificate contains isomorphic copy $G'$ of $G$
    - Defined by node permutation $\lambda$
  - Certificate at node $v$:
    - Isomorphic copy
    - Permutated input
    - $\lambda(v)$
  - Enough to verify $G'$ isomorphic to $G$
    - Each node sequentially decides $G'$

\[
x = (x_{v_1}, \ldots, x_{v_6})
\]

\[
x' = (x_{\lambda(v_1)}, \ldots, x_{\lambda(v_6)}) \quad \lambda(v)
\]
Combining Non-Determinism and Randomization Is Enough (cont.)

- **Proof (cont.):** we show 1-round algorithm to decide if $G, G'$ are isomorphic
- Every $v$:
  - Checks $x'(\lambda(v)) = x(v)$

![Graph diagram]

\[ x = (x_{v_1}, \ldots, x_{v_6}) \]

\[ x' = (x_{\lambda(v_1)}, \ldots, x_{\lambda(v_6)}), \quad \lambda(v_6) \]
Combining Non-Determinism and Randomization Is Enough (cont.)

• **Proof (cont.):** we show 1-round algorithm to decide if $G, G'$ are isomorphic

• Every $v$:
  - Checks $x'(\lambda(v)) = x(v)$
  - Checks with neighbors:
    • All got same $G', x'$
    • All labeled correctly
  - If fail, output “NO”
  - If pass:
    • $\lambda(v) \neq 1$, output “YES”
    • $\lambda(v) = 1$, output “YES” with probability $p$
Combining Non-Determinism and Randomization Is Enough (cont.)

- **Proof (cont.):** we show 1-round algorithm to decide if $G, G'$ are isomorphic

- Every $v$:
  - Checks $x'(\lambda(v)) = x(v)$ — $x'$ consistent with $x$
  - Checks with neighbors:
    - All got same $G', x'$ — All agree on $G', x'$
    - All labeled correctly — $G'$ locally consistent with $G$
  - If fail, output “NO”
  - If pass:
    - $\lambda(v) \neq 1$, output “YES”
    - $\lambda(v) = 1$, output “YES” with probability $p$

- All checks pass $\Rightarrow$ all agree on $(G', x')$, $x'$ consistent with $x$, $G'$ locally consistent with $G$
Combining Non-Determinism and Randomization Is Enough (cont.)

- **Proof (cont.):**
- If all tests pass:
  - All nodes got same $G'$, $x'$
  - $x'$ consistent with $x$
  - $G'$ locally consistent with $G$
- Is it enough?
- **NO!**

$x' = (1,2,3,1,2,3)$
Combining Non-Determinism and Randomization Is Enough (cont.)

- Proof (cont.):
- If all tests pass:
  - All nodes got same $G'$, $x'$
  - $x'$ consistent with $x$
  - $G'$ locally consistent with $G$
- Is it enough?
- NO!

$x' = (1, 2, 3, 1, 2, 3)$
Combining Non-Determinism and Randomization Is Enough (cont.)

- **Proof (cont.):** we show 1-round algorithm to decide if $G, G'$ are isomorphic
- Every $v$:
  - Checks $x'(\lambda(v)) = x(v)$
  - Checks with neighbors:
    - All got same $G', x'$
    - All labeled correctly
  - If fail, output “NO”
  - If pass:
    - $\lambda(v) \neq 1$, output “YES”
    - $\lambda(v) = 1$, output “YES” with probability $p$
- All checks pass $\Rightarrow$ all agree on $(G', x')$
- $(G', x')$ isomorphic to $(G, x)$ iff $\lambda(v) = 1$ for at most one $v$
  - This is “dictatorship or chaos” – decidable in $BPLD(0, p, q)$
Goal

\[ t = o(n) \]

\[ \text{BPLD}(0, p, q) \quad p^2 + q \leq 1 \]

\[ \text{BPNLD}(O(1), p, q) \quad p^2 + q \leq 1 \]

\[ \text{ALL} \]
Recall: once all vertices locally agree on \((G', x')\), only need to verify \#nodes
  - Instead, nodes “guess”, causing probabilistic error

Can show this is only cause for error:

If \#nodes known, \(\mathcal{L} \in \text{NLD}\)
  - Formalized by using oracle (advice)
Goal

\[ \mathcal{L}_D (t) \quad t = o(n) \]

\[ \mathcal{BPLD} (0, p, q) \quad p^2 + q \leq 1 \]

\[ \mathcal{BPNLD} (O(1), p, q) \quad p^2 + q \leq 1 \]

\[ \mathcal{NLD} \]

\[ \mathcal{ALL} \]
\[
LD(t) \quad t = o(n)
\]
\[
BPLD(0, p, q) \quad p^2 + q \leq 1
\]
\[
BPNLD(O(1), p, q) \quad p^2 + q \leq 1
\]
\[
ALL
\]
Summary

• We sorted decision problems in “local context” according to their “hardness”

• Hardness measured by:
  – # rounds
  – Deterministic or not
  – Probabilistic or not

• Characterization not full
  – E.g., is the bound $p^2 + q \leq 1$ tight for all languages?
  – Relation between $BPLD(t, p, q)$ and $BPLD(t, p', q')$
  – Relation between non-determinism and randomization

• Connections to complexity theory hierarchy?
  – Restrict node resources (time, space)
obrigado  Dank U  Merci  mahalo  Köszi
спацібо  Grazie  Thank  you  mauruuru  Takk
Gracias  Dziękuję  Děkuju  danke  Kiitos