Distributed Computing
11 - Complexity Classes

Mikaël Rabie
Université de Paris, IRIF
Complexity Landscapes
Landscape of LCL problems

Randomized time complexity

Deterministic time complexity

Figures from https://jukkasuomela.fi/landscape-of-locality/
Complexities on Graphs with the LOCAL Model

Landscape of LCL problems

$\Theta(\log n)$

deterministic

$\Theta(\log \log n)$

randomized

Figures from https://jukkasuomela.fi/landscape-of-locality/
Complexities on Graphs with the LOCAL Model

Landscape of LCL problems

- Maximal matching
- Maximal independent set
- 3-coloring paths and cycles

Figures from https://jukkasuomela.fi/landscape-of-locality/
Complexities on Graphs with the LOCAL Model

Figures from https://jukkasuomela.fi/landscape-of-locality/
Complexities on Graphs with the LOCAL Model

Figure from https://jukkasuomela.fi/landscape-of-locality/
Path and Cycle Complexities
3-coloring neighborhoods

<table>
<thead>
<tr>
<th></th>
<th>213</th>
<th>121</th>
<th>212</th>
<th>123</th>
</tr>
</thead>
<tbody>
<tr>
<td>321</td>
<td>232</td>
<td>323</td>
<td>231</td>
<td></td>
</tr>
<tr>
<td>132</td>
<td>313</td>
<td>131</td>
<td>312</td>
<td></td>
</tr>
</tbody>
</table>
3-coloring neighborhoods

213

121

212

123

321

232

323

231

132

313

131

312
3-coloring neighborhoods
3-coloring neighborhoods

![Diagram of 3-coloring neighborhoods]
3-coloring neighborhoods
3-coloring neighborhoods
3-coloring neighborhoods
Transition Automata

3-coloring

\[ \begin{array}{ccc}1 & 2 & 3 \\ 2 & 3 & 1 \end{array} \]
Transition Automata

3-coloring  Independent Set

[Diagram of a transition automata with nodes 1, 2, and 3 connected by arrows forming a triangle]
Transition Automata

3-coloring  Independent Set

\begin{figure}
\centering
\begin{tikzpicture}
\node[circle,draw] (1) at (0,0) {1};
\node[circle,draw] (2) at (-1,-1) {2};
\node[circle,draw] (3) at (1,-1) {3};
\node[circle,draw] (0) at (1,1) {0};
\node[circle,draw] (1) at (1,1) {1};
\draw[->] (1) edge (2);
\draw[->] (1) edge (3);
\draw[->] (2) edge (3);
\draw[->] (0) edge (1);
\end{tikzpicture}
\end{figure}
Transition Automata

3-coloring  Independent Set  MIS
Transition Automata

3-coloring  Independent Set  MIS

2-coloring

1  0 00

1 3 01 10
Transition Automata

3-coloring  Independent Set  MIS  2-coloring

1 \rightarrow 2 \rightarrow 3 0 \rightarrow 1 00 \rightarrow 01 \rightarrow 10
Transition Automata

3-coloring  Independent Set  MIS  2-coloring

1

2

3

0

1

00

01

10

2
Naor, Stockmeyer (1995)
If the input graph is an unlabeled path or cycle, the time complexity is decidable.

The different time complexities are $O(1)$, $\Theta(\log^* n)$ and $\Omega(n)$.
How to Decide the Complexity

- $O(1)$
How to Decide the Complexity

- $O(1)$: There exists a self-loop.
How to Decide the Complexity

- \( O(1) \): There exists a self-loop.

**Naor, Stockmeyer (1995)**

If a problem on paths can be computed in \( o(\log^* n) \) rounds, there exists an algorithm with the same complexity stable under relative order.
How to Decide the Complexity

- $O(1)$: There exists a self-loop.

- $\Theta(\log^* n)$
How to Decide the Complexity

- $O(1)$: There exists a self-loop.

- $\Theta(\log^* n)$: There exists a node $v$ and $k > 0$ such as $\forall i \geq k$, there exists a path of length $i$ from $v$ to $v$.

- $\Omega(n)$
How to Decide the Complexity

- $O(1)$: There exists a self-loop.

- $\Theta(\log^* n)$: There exists a node $v$ and $k > 0$ such as $\forall i \geq k$, there exists a path of length $i$ from $v$ to $v$.

- $\Omega(n)$: For all nodes $v$, either $v$ cannot be reached from $v$, or there exists $k > 1$ such as any path of length $i > 0$ from $v$ to $v$ is such as $k$ divides $i$. 
Decidability on Paths with Inputs
Problem on Paths with Inputs

• Color the red nodes.
• Carry the color through the blue nodes.
Problem on Paths with Inputs

• 3-color the red nodes.
• Carry the color through the blue nodes.
• 3-color the red nodes.
• Carry the color through the blue nodes.
• 3-color the red nodes.

• Carry the color through the blue nodes.
For any LCL problem on cycle graphs, its complexity is either $\Omega(n)$ or $O(\log^* n)$. Moreover, there is an algorithm that decides whether the problem has complexity $\Omega(n)$ or $O(\log^* n)$ on cycle graphs; for the case the complexity is $O(\log^* n)$, the algorithm outputs a description of an $O(\log^* n)$-round deterministic LOCAL algorithm that solves it.

For any LCL problem on cycle graphs, its complexity is either $\Omega(\log^* n)$ or $O(1)$. Moreover, there is an algorithm that decides whether the problem has complexity $\Omega(\log^* n)$ or $O(1)$ on cycle graphs; for the case the complexity is $O(1)$, the algorithm outputs a description of an $O(1)$-round deterministic LOCAL algorithm that solves it.
Balliu et. al (2019) It is PSPACE-hard to distinguish whether a given LCL problem with input labels can be solved in $O(1)$ time or needs $\Omega(n)$ time on globally oriented path graphs.
Balliu et. al (2019)

It is PSPACE-hard to distinguish whether a given LCL problem with input labels can be solved in $O(1)$ time or needs $\Omega(n)$ time on globally oriented path graphs.

- **Input**: nodes in states $a$, $b$, or representing a Turing Machine tape on space $k$
- **Output**:
  - Tape nodes carry the $a$ or $b$ closest from the left
  - Tape nodes prove that the tape encodes a bad TM simulation and carry state $E$ after the mistake
Turing Machine Encoding

The diagram illustrates the encoding process of a Turing machine, where states transition based on the input symbols. The states and transitions are detailed in the diagram, showing how the machine moves left (L), right (R), and stays in the same position (0) based on the current state and symbol read.
Balliu et. al (2019)
It is PSPACE-hard to distinguish whether a given LCL problem without input labels can be solved in $O(1)$ time or needs $\Omega(n)$ time on trees with degree $\Delta = 3$. 
Encoding a Number in a Tree

Encoding $6 = 0110_2$
Encoding a Number in a Tree

Encoding $6=0110_2$
Encoding a Number in a Tree

Encoding $6 = 0110_2$
Encoding a Number in a Tree

Encoding $6=0110_2$
Encoding a Number in a Tree

Encoding $6=0110_2$
Encoding a Number in a Tree

Encoding $6 = 0110_2$
Encoding a Number in a Tree

Encoding 6 = 0110₂
Encoding a Number in a Tree

Encoding 6 = 0110₂
Encoding the Input of the Path
Encoding the Input of the Path
Toroidal Grids
Grid Graphs:
- Toroidal $n \times n$ grid
- Consistent orientation N/S/E/W

**Brandt et. al (2017)**
Given any LCL problem $P$ with an algorithm $A$ that solves $P$ in time $T = o(n)$, there exists an algorithm $B$ that solves $P$ and has running time $O(\log^* n)$. 
Algorithm $A$ in time $T(n) = o(n)$:

- Take $k$ such as $T(k) < k/4 - 4$
- Compute Independent set $I$ of $G^{k/2}$ in time $O(\log^* n)$
- Compute new identifiers in $[k^2]$
  - Each node $u$ finds its closest anchor $i_u \in I$
  - $id'_u = (x_{i_u} - x_u, y_{i_u} - y_u)$
  - New identifiers form a $k/2$-distance coloring
- Simulate $A$ with the new identifiers in time $T(k)$
Algorithm A in time $T(n) = o(n)$:

- Take $k$ such as $T(k) < k/4 - 4$
- Compute Independent set $I$ of $G^{k/2}$ in time $O(\log^* n)$
- Compute new identifiers in $[k^2]$
  - Each node $u$ finds its closest anchor $i_u \in I$
  - $id'_u = (x_{i_u} - x_u, y_{i_u} - y_u)$
  - New identifiers form a $k/2$-distance coloring
- Simulate $A$ with the new identifiers in time $T(k)$
Algorithm $A$ in time $T(n) = o(n)$:

- Take $k$ such as $T(k) < k/4 - 4$
- Compute Independent set $I$ of $G^{k/2}$ in time $O(\log^* n)$
- Compute new identifiers in $[k^2]$
  - Each node $u$ finds its closest anchor $i_u \in I$
  - $id'_u = (x_{i_u} - x_u, y_{i_u} - y_u)$
  - New identifiers form a $k/2$-distance coloring
- Simulate $A$ with the new identifiers in time $T(k)$
Algorithm $A$ in time $T(n) = o(n)$:

- Take $k$ such as $T(k) < k/4 - 4$
- Compute Independent set $I$ of $G^{k/2}$ in time $O(\log^* n)$
- Compute new identifiers in $[k^2]$
  - Each node $u$ finds its closest anchor $i_u \in I$
  - $id'_u = (x_{i_u} - x_u, y_{i_u} - y_u)$
  - New identifiers form a $k/2$-distance coloring
- Simulate $A$ with the new identifiers in time $T(k)$
Algorithm $A$ in time $T(n) = o(n)$:

- Take $k$ such as $T(k) < k/4 - 4$
- Compute Independent set $I$ of $G^{k/2}$ in time $O(\log^* n)$
- Compute new identifiers in $[k^2]$
  - Each node $u$ finds its closest anchor $i_u \in I$
  - $id'_u = (x_{i_u} - x_u, y_{i_u} - y_u)$
  - New identifiers form a $k/2$-distance coloring
- Simulate $A$ with the new identifiers in time $T(k)$
Algorithm $A$ in time $T(n) = o(n)$:

- Take $k$ such as $T(k) < k/4 - 4$
- Compute Independent set $I$ of $G^{k/2}$ in time $O(\log^* n)$
- Compute new identifiers in $[k^2]$
  - Each node $u$ finds its closest anchor $i_u \in I$
  - $id'_u = (x_{i_u} - x_u, y_{i_u} - y_u)$
  - New identifiers form a $k/2$-distance coloring
- Simulate $A$ with the new identifiers in time $T(k)$
The problem of deciding whether a given LCL can be solved in time $\Theta(\log^* n)$ or $\Theta(n)$ on grids is undecidable.

From a given Turing Machine:

• Find an area to simulate the execution.
• Put an anchor that starts the computation (line 0).
• Line $i$ corresponds to execution step $i$.
• Possibility to check locally that the execution is right.
• Solve a $\Theta(n)$ problem (3-coloring) otherwise.

If the Turing Machine halts, the first solution takes a constant space, and can be repeated. If the Turing Machine does not halt, the global problem must be solved.
Brandt et al. (2017)
The problem of deciding whether a given LCL can be solved in time $\Theta(\log^* n)$ or $\Theta(n)$ on grids is undecidable.
The problem of deciding whether a given LCL can be solved in time $\Theta(\log^* n)$ or $\Theta(n)$ on grids is undecidable.

From a given Turing Machine:

- Find an area to simulate the execution:
  - Put an anchor that starts the computation (line 0)
  - Line $i$ corresponds to execution step $i$
  - Possibility to check locally that the execution is right
- Solve a $\Theta(n)$ problem (3-coloring) otherwise

\textit{Brandt et. al (2017)}
The problem of deciding whether a given $LCL$ can be solved in time $\Theta(\log^* n)$ or $\Theta(n)$ on grids is undecidable.

From a given Turing Machine:

- Find an area to simulate the execution:
  - Put an anchor that starts the computation (line 0)
  - Line $i$ corresponds to execution step $i$
  - Possibility to check locally that the execution is right
- Solve a $\Theta(n)$ problem (3-coloring) otherwise

If the Turing Machine halts, the first solution takes a constant space, and can be repeated. If the Turing Machine does not halt, the global problem must be solved.
Turing Machine Simulation:

- Consistent labelling around anchors
- Tape simulation
Turing Machine Simulation

- Consistent labelling around anchors
- Tape simulation
Turing Machine Simulation:

- Consistent labelling around anchors
- Tape simulation
Turing Machine Simulation:

- Consistent labelling around anchors
- Tape simulation
Turing Machine Simulation:

- Consistent labelling around anchors
- Tape simulation
Turing Machine Simulation

- Consistent labelling around anchors
- Tape simulation
- Problem?
Turing Machine Simulation

- Consistent labelling around anchors
- Tape simulation
- Problem?
Turing Machine Simulation

- Consistent labelling around anchors
- Tape simulation
- 2-coloring along diagonals
Turing Machine Simulation:

- Consistent labelling around anchors
- Tape simulation
- 2-coloring along diagonals
Turing Machine Simulation

- Consistent labelling around anchors
- Tape simulation
- 2-coloring along diagonals
Turing Machine Simulation

- Consistent labelling around anchors
- Tape simulation
- 2-coloring along diagonals
Bibliography

- https://jukkasuomela.fi/landscape-of-locality/