What Can(not) Be Computed in One Round in Interconnection networks?

F. Becker\textsuperscript{1}  A. Kosowski\textsuperscript{2}  M. Matamala\textsuperscript{3}  N. Nisse\textsuperscript{4}  I. Rapaport\textsuperscript{3}  K. Suchan\textsuperscript{5}  I. Todenca\textsuperscript{1}

\textsuperscript{1} LIFO, Univ. Orléans, France
\textsuperscript{2} Inria, LaBRI, Bordeaux, France
\textsuperscript{3} DIM, Universidad de Chile, Santiago, Chile
\textsuperscript{4} COATI, Inria, I3S, CNRS, UNS, Sophia Antipolis, France
\textsuperscript{5} Universidad Adolfo Ibáñez, Santiago, Chile

ADR Network Science
Paris, Feb. 28\textsuperscript{th} 2013

Adding a Referee to an Interconnection Network: What Can(not) Be Computed in One Round. IPDPS 2011
Allowing each node to communicate only once in a distributed system: shared whiteboard models SPAA 2012
Distributed computation of network properties

Problem

Huge networks $\Rightarrow$ generic algorithms (even polynomial) are not efficient

Need to use structural properties
Distributed computation of network properties

Problem

Huge networks \( \Rightarrow \) generic algorithms (even polynomial) are not efficient

Need to use structural properties

Example of compact routing

<table>
<thead>
<tr>
<th>network</th>
<th>mult. stretch</th>
<th>routing table</th>
</tr>
</thead>
<tbody>
<tr>
<td>arbitrary</td>
<td>shortest path</td>
<td>( O(n \log n) )</td>
</tr>
<tr>
<td>((k \geq 2))</td>
<td>( O(k) )</td>
<td>( \Theta(n \log n) ) [folk]</td>
</tr>
<tr>
<td>trees</td>
<td>shortest path</td>
<td>( O(\log n) )</td>
</tr>
<tr>
<td>( 2^k - 1 )</td>
<td>( \Theta(n^{1/k}) ) [Thorup,Zwick]</td>
<td></td>
</tr>
<tr>
<td>doubling-(\alpha) dimension</td>
<td>( O(1) + \epsilon )</td>
<td>( O(\log \Delta) ) [Talwar/Slivkins]</td>
</tr>
<tr>
<td>planar</td>
<td>( 1 + \epsilon )</td>
<td>( O(\log n) )</td>
</tr>
<tr>
<td>( H)-minor free</td>
<td>( 1 + \epsilon )</td>
<td>( O(</td>
</tr>
</tbody>
</table>

BGP is generic \( \Rightarrow \) large Routing Tables :(
but easy to compute and update :) 

other schemes require structural information (e.g., decompositions) on the graph
Problem

Huge networks $\Rightarrow$ generic algorithms (even polynomial) are not efficient

Need to use structural properties

Objectives

- Understand, compute, discover... structural Properties
- Distributed/Local computation
- Use it for algorithmic purposes (not only routing)
- Model/simulate such networks (static/dynamic behavior)

Questions

What hypothesis can we adopt for the computation?
What is feasible in a given model?
Model of distributed computing

Graph with $n$ nodes. Nodes have distinct identifiers in $\{1, \cdots, n\}$.

A node knows: its ID and the IDs of its neighbors
A node has arbitrary computation power.

Goal: **encode its local knowledge in a small message** (typically $O(\log n)$)
Each node sends its (unique) message to a central entity.

**Remark:** If $|message| = n$ bits, then node gives its whole neighborhood.
The referee has arbitrary computation power and use the $n$ messages to...
Model of distributed computing

... answer a question about the graph (typically: "does G has some property?")
To summerize: Model of distributed computing

**Principle**

Does $G$ belongs to $\mathcal{P}$?

- each node encodes its local knowledge

  \[ message : \text{ID of } v \times \text{IDs of } N(v) \rightarrow message(v), \text{ and} \]

  \[ |message(v)| = O(\log n) \text{ bits} \]

- the referee decodes the $n$ messages to answer

  \[ answer : (message(v_i))_{i \leq n} \rightarrow \{\text{true}; \text{false}\} \]

**Hypothesis**

- arbitrary computational power: $message$ and $answer$ are arbitrary functions

- IDs are distinct in $\{1, \cdots, n\}$

**Remark:** if bounded maximum degree: each node may send its full adjacency list.

**Problem**

in total: $O(n \log n)$ bits of local information

What kind of question can be answered?

---

Becker et al. What Can(not) Be Computed in One Round?
Example: Does $G$ is a tree? If yes, compute its adjacency matrix.

Each node sends its ID, its degree and the sum of the IDs of its neighbors.
Example: Does $G$ is a tree? If yes, compute its adjacency matrix.

The referee iteratively “prunes” the one-degree nodes in the whiteboard. In parallel, he re-builds the tree.
Example: Does $G$ is a tree? If yes, compute its adjacency matrix.

The referee iteratively "prunes" the one-degree nodes in the whiteboard. In parallel, he re-builds the tree.
Example: Does $G$ is a tree? If yes, compute its adjacency matrix.

The referee iteratively “prunes” the one-degree nodes in the whiteboard in parallel, he re-builds the tree.
Example: Does $G$ is a tree? If yes, compute its adjacency matrix.

The referee iteratively “prunes” the one-degree nodes in the whiteboard. In parallel, he re-builds the tree.
Possible

Decide if a graph has bounded degeneracy (include planar graphs, bounded genus graphs, bounded treewidth graphs...). If yes, build their adjacency matrix.

*proof*: generalization of the “pruning process” of trees.

Not possible

Decide if the graph contains a triangle, a (induced or not) square.
Decide if the graph has diameter at most 3

*proof*: Kind of reduction.
If possible $\Rightarrow$ Possible to build adjacency matrix of bipartite graphs.
$2^{\Omega(n^2)}$ such graphs $\Rightarrow$ impossible to distinguish all of them with $O(n \log n)$ bits.
$\Rightarrow$ contradiction

We don’t know????

Decide if the graph is connected.
Until now:
All nodes write *simultaneously* on the whiteboard
Don't take advantage of what is written by other nodes.

Now:
Nodes can also read the whiteboard.
Can use previous messages to compute their own message
4 Models

**SimAsync**
- model above
- All nodes write *simultaneously* on the whiteboard

**SimSync**
- Nodes write *sequentially*.
- **Worst ordering**: order chosen by an adversary

**Async**
- model of asynchronicity
- Nodes *rise hand to speak*.
  - If several nodes rise hand, all write *simultaneously*.

**Sync**
- model of synchronicity
- Nodes *rise hand to speak*.
  - If several nodes rise hand, they write *sequentially* in worst ordering.
Results

Hierarchy of models

\[ \text{SimAsync}(\log n) < \text{SimSync}(\log n) < \text{Async}(\log n) \leq \text{Sync}(\log n) \]

<table>
<thead>
<tr>
<th>message: ( O(\log n) ) bits</th>
<th>SimAsync</th>
<th>SimSync</th>
<th>Async</th>
<th>Sync</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUILD k-degenerate</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>rooted MIS</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Square</td>
<td>no</td>
<td>no</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Connectivity</td>
<td>?</td>
<td>?</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Spanning tree</td>
<td>?</td>
<td>?</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Bipartite-BFS</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>BFS</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>yes</td>
</tr>
</tbody>
</table>

Orthogonal criteria

Let \( f(n) = o(n) \) and \( g(n) = o(f(n)) \).

There exist problems solvable in \( \text{SimAsync}(f(n)) \) and not in \( \text{Sync}(g(n)) \).
Further works

Several messages per nodes?

Probabilistic algorithms?

What if graph partially known (only few messages)?

Connectivity?

What is a realistic model?

...